Thesis for the Degree of Doctor of Philosophy

Be More and Be Merry: Enhancing Data and User Authentication in Collaborative Settings

Elena Pagnin

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Be More and Be Merry: Enhancing Data and User Authentication in Collaborative Settings
Elena Pagnin
ISBN: 978-91-7597-774-4
Series number: 4455

Technical report 157D
ISSN0346-718X

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31-772 10 53

Author e-mail: `elenap@chalmers.se, pagnin.elena@gmail.com`

Printed by Chalmers Reproservice
Gothenburg, Sweden 2018

Abstract

Cryptography is the science and art of keeping information secret to un-intended parties. But, how can we determine who is an *intended* party and who is not? Authentication is the branch of cryptography that aims at confirming the source of data or at proving the identity of a person. This Ph.D. thesis is a study of different ways to perform *cryptographic authentication of data and users.*

The main contributions are contained in the six papers included in this thesis and cover the following research areas: (i) *homomorphic authentication*; (ii) *server-aided verification of signatures*; (iii) *distance-bounding authentication*; and (iv) *biometric authentication.* The investigation flow is towards collaborative settings, that is, application scenarios where different and mutually distrustful entities work jointly for a common goal. The results presented in this thesis allow for secure and efficient authentication when more entities are involved, thus the title "*be more and be merry*".

Concretely, the first two papers in the collection are on homomorphic authenticators and provide an in-depth study on how to enhance existing primitives with *multi-key* functionalities. In particular, the papers extend homomorphic signatures and homomorphic message authentication codes to support computations on data authenticated using different secret keys. The third paper explores signer *anonymity* in the area of server-aided verification and provides new secure constructions. The fourth paper is in the area of distance-bounding authentication and describes a generic method to make existing protocols not only authenticate direct-neighbors, but also entities located *two-hop* away. The last two papers investigate the *leakage of information* that affects a special family of biometric authentication systems and how to combine verifiable computation techniques with biometric authentication in order to mitigate known attacks.

Keywords: Homomorphic Signatures, Server-Aided Verification, Verifiable Computation, Distance-Bounding Authentication Protocols, Biometric Authentication.

List of Publications

This Ph.D. thesis comprises a collection of six scientific articles devoted to exploring data and user authentication in different settings. References to these papers will be made using the associated Latin letters. The settings considered in this thesis are: authentication of computations on signed data (**Paper A** and **Paper B**); lightweight verification of data authenticity (**Paper C**); distance-bounding authentication (**Paper D**); and biometric authentication (**Paper E** and **Paper F**). The aforementioned articles are published at the following venues:

Paper A [39] *Multi-Key Homomorphic Authenticators.* D. Fiore, A. Mitrokotsa, L. Nizzardo, and E. Pagnin. In the 22nd International Conference on the Theory and Application of Cryptology and Information Security (asiacrypt), 2016.

Paper B [40] *Matrioska: A Compiler for Multi-Key Homomorphic Signatures.* D. Fiore and E. Pagnin. In the 11th Conference on Security and Cryptography for Networks (scn), 2018.

Paper C [72] *Anonymous Single-Round Server-Aided Verification of Signatures.* E. Pagnin, A. Mitrokotsa, and K. Tanaka. In the 5th International Conference on Cryptology and Information Security (latincrypt), 2017.

Paper D [86] *Two-hop Distance-Bounding Protocols: Keep your Friends Close.* A. Yang, E. Pagnin, A. Mitrokotsa, G. Hancke, and D. S. Wong. In IEEE Transactions on Mobile Computing (17:7), 2018.

Paper E [68] *On the Leakage of Information in Biometric Authentication.* E. Pagnin, C. Dimitrakakis, A. Abidin, and A. Mitrokotsa. In the 15th International Conference on Cryptology in India (indocrypt), 2014.

Paper F [70] *Revisiting Yasuda et al.'s Biometric Authentication Protocol: Are you Private Enough?* E. Pagnin, J. Liu, and A. Mitrokotsa. In the 16th International Conference on Cryptology and Network Security (cans), 2017.

Other articles published during my Ph.D., but not included in this thesis, are:

[67] *HIKE: Walking the Privacy Trail.* E. Pagnin, C. Brunetta, and P. Picazo-Sánchez. In the 17th International Conference on Cryptology and Network Security (cans), 2018.

[74] *HB+DB: Distance-Bounding Meets Human Based Authentication.* E. Pagnin, A. Yang, Q. Hu, G. Hancke, and A. Mitrokotsa. In Future Generation Computer Systems, 2018.

[71] *Privacy-Preserving Biometric Authentication: Challenges and Directions*. E. Pagnin and A. Mitrokotsa. In Security and Communication Networks, 2017.

[69] *Using Distance-Bounding Protocols to Securely Verify the Proximity of Two-hop Neighbours*. E. Pagnin, G. Hancke, and A. Mitrokotsa. In IEEE Communications Letters, 2015.

[73] *HB+DB, Mitigating Man-in-the-Middle Attacks against HB+ with Distance-Bounding*. E. Pagnin, A. Yang, G. Hancke, and A. Mitrokotsa. In ACM Security & Privacy in Wireless and Mobile Networks (wisec), 2015.

[4] *Attacks on Privacy-Preserving Biometric Authentication*. A. Abidin, E. Pagnin, and A. Mitrokotsa. In the 19th Nordic Conference on Secure IT Systems (nordsec), 2014.

Acknowledgements

Der är lätt att vara efterklok.

Elena Pagnin

First and foremost, I want to thank my advisor Andrei Sabelfeld, who took over the supervision of my Ph.D. studies *in media res* and steadily supported me. Your joy and enthusiasm for research lit up my path in its darkest hour and made me regain passion for academic work. I also wish to express my deep gratitude and respect to my co-supervisor, mentor and guide Dario Fiore. It has been an honor to work with you, to learn from you and to have your valuable advice. I could not imagine having a better mentor than you. Besides my supervisors, I would like to thank David Sands, who kindly agreed to become my Ph.D. examiner. Your knowledge and experience were fundamental to set the quality bar of my research.

Next, a spacial thanks goes to Bart Preneel for accepting to be my Ph.D. opponent. You made me rediscover the pleasure of pen-and-paper feedback, including human-based handwriting decryption. I also gratefully acknowledge the grading committee members: Claudio Orlandi, Damien Vergnaud and Martin Hell for their positive and encouraging comments on this thesis.

My Ph.D. studies have been sprinkled with long research visits and several conferences. Adding-up, I have been working away from Sweden for over one year! Nonetheless, in the last period I found two good reasons for doing research within Chalmers: Carlo Brunetta, Pablo Picazo-Sánchez and his little son Óliver (they count as one entity). I will cherish the memories of our morning 'Kaffe?' messages, leading to long 'coffee breaks' that inevitably turned into lively research discussions. I am happy I met both of you. I am deeply grateful for our friendship and for the constructive camaraderie we have when working together. I would also like to mention another co-author and friend: Cristina Onete, who has my gratitude for opening my mind to new, exciting research horizons despite my initial reluctance. I admire your immense knowledge, passion, enthusiasm and helpfulness. It is always a pleasure to hard work with you, even when it leads to long-lasting discussions via Skype! My sincere thanks go also to Aysajan Abidin, Luca Nizzardo, Keisuke

Tanaka and Gerard Hancke for our fruitful collaborations.

Looking back at these last four years, there are still people within Chalmers whom I owe acknowledgement. Olaf Landsiedel, for our open exchange of opinions behind closed doors. Wolfgang Ahrendt, for shedding light on my ethical dilemmas. Agneta Nilsson and Mary Sheeran, for lending me their ears and guiding me to a better life. I can not thank you enough for what you did for me. Tomas Olovsson, for his help, support and especially the countless hours spent together trying to figure out positive solutions to negative situations. And last but not least at Chalmers, I want to say *tack* to all the secretaries of the Computer Science and Engineering Department who every day do an amazing work keeping all the paperwork running. I have never met such a devote, efficient and kind stuff. In particular, my thanks go to Eva Axelsson, Marianne Pleen-Schreiber, Elisabeth Kegel Andreasson, Rebecca Cyren, Anneli Andersson and Tiina Rankanen.

A heartfelt mention goes to my friends, including Cecilia, Guilhem, Inari, Irene, Iulia, Jeff, Marta, Micał, Thomas, Valentina, Wouter. You have been there when needed throughout all of these years, independently of where on Earth I was. Thank you for the many cooking sessions, board game evenings, hikes, saunas, *fika*s, proof-readings, sailings, fermentation parties and traditional Scandinavian activities such as *midsommar* fireplaces and berry picking. Above all, I am glad our paths have crossed and we have walked together along the way.

Loving thanks go to my *sambo* Hedvig Maria Jonsson, for her truthful support, encouragement and endless patient during the last half of my Ph.D. You managed to give me a constant motivation for going back to Sweden and made me start liking this Nordic country. Thank you for being part of my life, and for being stubborn ♡

I reserve the final thanks for the people without whom I most probably would not be where and who I am now: Frédérique Oggier, Mariuccia Paoletti, Marc Stöttinger, Arianna Pagnin, Annamaria Borgato, Lorenzo Pagnin and Aikaterini Mitrokotsa. I hope you can be proud and feel part of my achievement.

Contents

List of Abbreviations

BAP:	Biometric Authentication Protocol.
DBAP:	Distance-Bounding Authentication Protocol.
FHS:	Fully Homomorphic Signature (Scheme).
HA:	Homomorphic Authenticator.
HS:	Homomorphic Signature (Scheme).
MAC:	Message Authentication Code.
MK-HA:	Multi-Key Homomorphic Authenticator.
NP:	Non-deterministic Polynomial time.
Ph.D.:	Doctor of Philosophy (from the Latin *Philosophiae Doctor*).
RFID:	Radio Frequency IDentification.
SAV:	Server-Aided Verification.
SNARK:	Short Non-interactive ARgument of Knowledge.

Part I

Thesis Summary

Introduction

Man is by nature a social animal; an individual who is unsocial naturally and not accidentally is either beneath our notice or more than human.

Aristotle, Politics

The social nature of human beings renders communicating and storing information two essential needs for surviving. Knowing where to go, who people are, asking for clarifications and providing instructions is something we do everyday. In developed countries, the society has taken a *digital* approach: people 'talk' to each other in chats, e-mails or video-calls; and save information they want to 'remember' on smartphones or cloud back-ups. The migration to digital platforms has increased the demand for digital interaction and storage methods that achieve features similar to or better than face-to-face conversations and personal memory. Common concerns are: how can we be sure of the identity of our digital interlocutor, does someone else know what we are talking about; or what guarantees the stored data are always available to us only and not modified without us noticing? Cryptography addresses these and more concerns by keeping information secret to un-intended receivers and allowing secure communication in the presence of untrusted parties [47].

The Cryptographers' World

My parents' generation grew up having face-to-face as the most common way to communicate. For them it was clear who they were talking to and where and when the conversation was taking place. Thus, my parents could easily adjust the content and style of the conversation according to the circumstance. If they had to discuss something private or secret, they would ask to meet in a remote location, or in a place surrounded by people that had no interest in their secret. They would use letters or wired telephones to contact people who were far-away. In the first case, they would not know whether the letter reached its destination until they received a response (and recognized the sender's handwriting); in the second case, they were extremely suspicious on who was listening *inside* the telephone line, but still they were happy they could recognize the interlocutor by hearing their voice. Important information was either learned by heart or written on a piece of paper they would hide somewhere safe to make sure no-one would access it.

My generation is quite different. We were born with modern computers and digital technologies. We are used to asynchronous communications via e-mail and to instant messaging in social networks. Our most common way of communicating is in virtual environments. In particular, we almost never *see* or *hear* our interlocutors

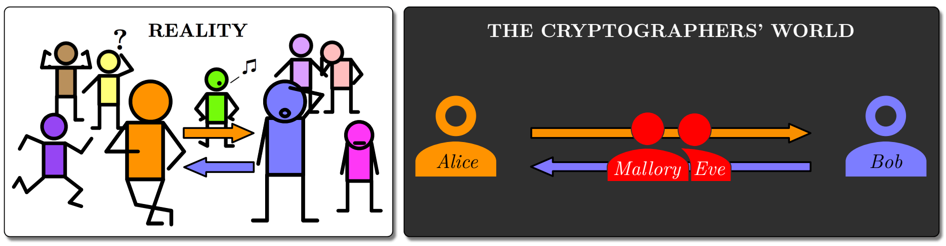


Figure 1: Quirky representation of some differences between the *real*- and the *cryptographers'* (perception of the) world.

in real time and have no way to determine when and where a piece of information is delivered or received. Regarding sensitive data, we may try to learn it by heart, but it is so much easier and handier to store it on our smartphones, computers or directly in the cloud! Therefore, in contrast to my parents, I find it very hard to know for sure who I am writing to, to adjust the content and style of my conversations or to make sure no-one can find my secret data. However, I would still like to have the same guarantees as my parents had. This is what cryptography tries to achieve.

In a nutshell, the cryptographer's world is looking at our digital world with some privacy-paranoid glasses, as figuratively depicted in Figure 1. In cryptography, the *talking* entity magically becomes Alice and has an urge to communicate highly sensitive information to another person, named Bob, who is located far, far away from her. Everyone around them turns into an evil being, Eve or Mallory according to the story, and is suspiciously interested in the content of Alice and Bob's conversation. This setting is formalized in the concept of *communication over an insecure channel.*

Investigating how paranoid the cryptographers' world can be is a Ph.D. on its own and falls outside the scope of this thesis. During my Ph.D., I regarded the cryptographers' view of the world as fascinating and immersed myself in it with the objective to develop some tools that would render it a brighter reality. To this aim, I collect in this thesis new proposals for data and user authentication. Concretely, the presented contributions can be used to ensure Alice that she is talking to Bob and not to Eve, and that her data have not been modified by Mallory.

The Main Security Goals of Cryptography

Cryptographic primitives and protocols are designed to maintain a desired set of security goals even under attempts at making them deviate from the expected functionality. We briefly describe the two most common security goals in the paragraphs below [81] assuming that an entity called Alice wants to communicate with another entity called Bob in the presence of an undesired party called, generically, the adversary.

Confidentiality. This is the main idea people associate to the term "Cryptography". In a nutshell, if a cryptographic scheme or protocol achieves confidentiality it means that Alice is able to send messages to Bob in such a way that only Bob can read the messages and no adversary is able to see the actual content of their communication. Encryption is the queen cryptographic primitive for confidentiality.

Authentication. This property can refer to both data and user authentication. In the case of user authentication, this functionality ensures that a certain person, *e.g.,* Alice, is who she claims to be. For message authentication, the goal is to provide some additional information that guarantees to Bob that the message he received was originated by Alice. In particular, no undesired third party should be able to impersonate Alice. Digital signatures and Message Authentication Codes (MAC) are the two knights cryptographic primitives that grant data authentication. Regarding user authentication, in this thesis we consider the case of distance-bounding and biometric protocols.

Confidentiality and authentication are the two main security goals of cryptography, however, there are other useful functionalities that cryptographic primitives and protocols can guarantee, such as: integrity [9], non-repudiation [27], controlled malleability [44], redactability [24], delegation [61], attribute-based confidentiality [50], proofs of knowledge [76], availability and proofs of work [55], and more. This Ph.D. thesis focuses on authentication and data integrity.

Why Authentication?

More than forty years ago, Diffie and Hellman flagged that authentication was perhaps the main barrier to the universal adoption of digital communications for sensitive data (*e.g.,* business transactions) and that it constituted the heart of any system involving 'contracts and billing' [37]. These statements acted as a spring for the development of (asymmetric) cryptographic tools for user authentication as well as data authentication, integrity and non-repudiation.

My Ph.D. has *authentication* as main topic. The real reason for which I chose to devote these years of my life to studying and (hopefully) contributing to the area of authentication is that I believe that (public-key) encryption looses large part of its usefulness if it is not combined with some sort of authentication. For instance, if I had a sensitive conversation about my health condition, I would *first* make sure that my interlocutor is my doctor –and not some impostor sending fake news to me– and only *secondly* that the conversation is encrypted (thus intelligible only to the doctor and me). Having reliable and secure authentication has become even more relevant thanks to the technological development we have witnessed in the last decades. Nowadays, authentication is a fundamental step in services such as online banking, e-health, e-commerce, automatized border controls and many more. My Ph.D. goal was therefore to get acquainted with known ways to achieve data and user authentication, to propose new solutions and to extend existing ones to collaborative scenarios, where multiple entities want to contribute to a joint cause. The main results of my work are collected into this Ph.D. thesis.

Thesis Overview

This thesis collects the major results I obtained during my Ph.D. at Chalmers University of Technology. The title *be more and be merry* captures the core idea of my works: guaranteeing that certain cryptographic primitives and protocols remain secure even in enhanced environments that involve a number of entities larger than the standard one. This is the case of collaborative scenarios such as team-work activity or sensor networks.

The thesis is organized in two parts. The first part begins with a high-level introduction, some background notions and a brief summary of the results. It concludes with an outlook on directions for future work. The second part of the thesis is a collection of six papers on data and user authentication in collaborative settings including sharing computation on data, taking over specific tasks, or enabling communication. Figure 2 displays connections among the published works I contributed to during my Ph.D. and groups them by topic.

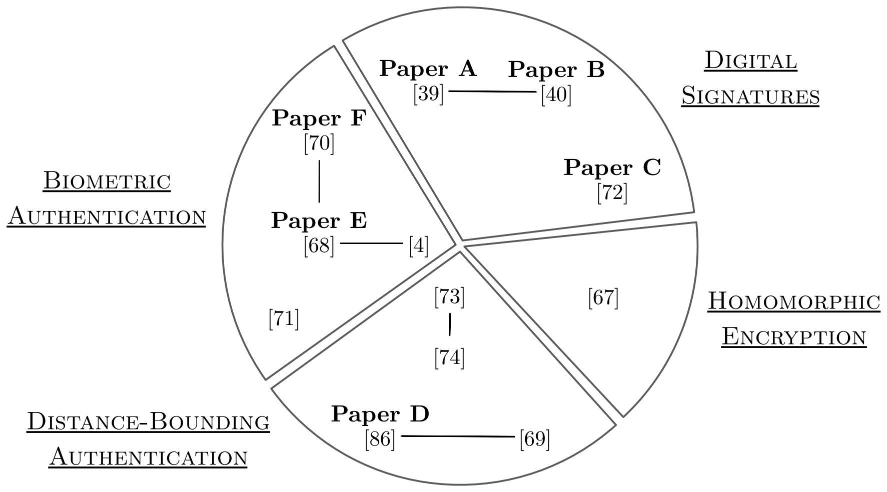


Figure 2: Pie diagram of my publications during the Ph.D. Lines between papers display logical connections among the results contained therein.

In detail, **Paper A** [39] and **Paper B** [40] provide ways for authenticating *computations* on data generated by *multiple users*; **Paper C** [72] investigates how to improve the efficiency and anonymity in settings where the verification of signatures is offloaded to an *untrusted server*. **Paper D** [86] and [69] extend the notion of distance-bounding to a collaborative setting by relying on an *untrusted linker* for authenticating an out-of-reach entity. In the same research area, [73, 74] propose a new authentication protocol that mitigates known attacks against the HB protocol [58]. **Paper E** [68], **Paper F** [70] and [4, 71] address issues in *biometric* authentication protocols. Finally, [67] is my most recent work and falls outside the wide area of authentication. It considers the problem of privacy-preserving processing of outsourced data in the context of user-customised services and develops a new lightweight protocol for private and secure storage, computation and disclosure of users' data.

Background

Cryptography is about communication
in the presence of an adversary.

Goldwasser and Bellare [47]

This section provides high-level and concise introductions to the four main areas of contributions of this thesis, namely: homomorphic signatures, server-aided verification, distance bounding authentication and biometric authentication. The reader is assumed to be familiar with basic concepts of public-key cryptography [47].

Homomorphic Signatures

Digital Signatures [18, 25, 48] enable the holder of a secret key to sign messages in such a way that anyone in possession of the corresponding public verification key can determine the validity of a given message-signature pair. For security, it is required that the signature is unforgeable, *i.e.,* no efficient adversary can forge a valid signature (unless the adversary knows the secret key).

Consider the use case of a school database for students' grades. To prevent students from tampering with their results, each teacher uploads a grade together with a signature (for the student and the grade). The unforgeability property ensures that students cannot arbitrarily change their grades, however, it also limits the utility of the database. For instance, if the school director wants to check the average of the students' grades on a certain subject, she would need to download all the grade-signature pairs related to the subject, check the authenticity of each grade and then compute the average on the (now certified) values. This procedure is quite inconvenient, since the grades need to be checked *before* computing the average, and has a high communication cost, due to the fact that *all* signed data need to be downloaded. A more desirable solution would allow the school director to download directly the average grade together with *one* signature attesting that the returned value is the correct one according to the grades available in the school database, and digitally signed by the legitimate teacher (see Figure 3). Such a scheme would have somehow *malleable* signatures, *i.e.,* signatures that support computation on authenticated data. This kind of schemes are called homomorphic signatures.

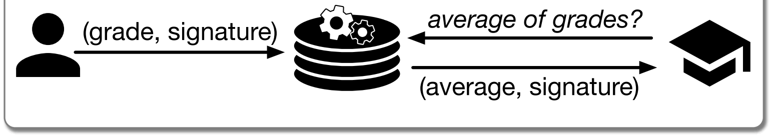


Figure 3: Application scenario for homomorphic signature schemes: a database of signed grades.

Homomorphic signature (HS) schemes [36] enable the holder of a secret key to sign messages $m_1, \dots, m_n$ in such a way that anyone in possession of the corresponding signatures $\sigma_1, \dots, \sigma_n$ and a function f can produce a valid signature σ for the message $y = f(m_1, \dots, m_n)$. The key property of HS is succinctness: the size of the evaluated signature σ should be smaller than the concatenation $(\sigma_1, \dots, \sigma_n)$ and it is usually logarithmic in n, the number of messages. In homomorphic settings the definition of unforgeability depends on the class of functions f supported by the scheme. For schemes that support only linear functions on a vector space, *e.g.,* [16], unforgeability states that the adversary should not be able to derive a correct signature for a message (vector) which cannot be obtained as a linear combination of previously honestly signed messages. If we applied the same reasoning to linearly homomorphic signatures with messages in a field or to Fully Homomorphic Signature schemes (FHS), *e.g.,* [15, 49], we would end up with a useless definition: given a pair (m, σ) it is possible to generate a valid signature σ' for any message $m' = f(m)$. Since f is any polynomial function, from a chosen m and its signature σ one can compute signatures for any message in the whole message space. A meaningful notion of unforgeability for FHS requires that the adversary should not be able to derive a valid signature σ^* for a value y^* that is not the correct output of $f(m_1, \dots, m_n)$ [43, 49]. This notion is achieved thanks to labelled programs [43], as in FHS the signatures, the homomorphically evaluated signatures and the verification procedure all depend on the labels.

The unforgeability intuitions given in this section are approximations of the core meaning of the corresponding security notions. The formal definitions are quite elaborate and include several sub-cases (types of forgeries). We refer the readers to [16, 39, 49] for the details.

In the school database scenario, using FHS to sign the grades solves the problem of computing statistics on the performance of students in each subject. However, FHS does not directly allow to perform computations on grades signed by different teachers, leaving open the following problem:

> *How can we authenticate homomorphic computation of functions that involve data signed by different secret keys?*

To achieve this property we need to make the signature scheme not only homomorphic on the messages, but also 'flexible' enough to accommodate computations on data generated by different signers. The latter property is often referred to as *multi-key.* In **Paper A** [39], we address the above question and formalize the multi-key notion for FHS. Moreover, we provide concrete instantiations of schemes that are multi-key and homomorphic. In **Paper B** [40] we go one step further and investigate connections between single-key and multi-key homomorphic signatures.

Server-Aided Verification of Signatures

In the previous section, we mentioned how digital signature schemes have developed to support more and more advanced *homomorphic* properties. Computing on signed data, however, is not the only line of development for signature schemes. To cover the wide range of applications of this cryptographic primitive, other types of schemes have been proposed such as: ring signatures [10, 21, 62], group signatures [14, 29, 62, 63], blind signatures [2, 11, 28], attribute-based signatures [53, 65, 79], and structure preserving signatures [1, 63]. Despite the different aims, most signature schemes are designed around strong and well-established cryptographic assumptions that guarantee security at the cost of efficiency, especially in the verification process of signatures. There are three possible ways to enjoy both security and efficiency: (i) using a different *hard problem* to design a secure signature scheme, (ii) trying to speed-up inefficient algorithms exploiting clever ways of computing the necessary data, and (iii) off-loading heavy computations to a third party and efficiently verifying the returned result. The latter approach falls into the *server-aided* category of cryptographic schemes. Since in signatures schemes the large bulk of computation is usually in the verification procedure, the main line of research is for Server-Aided signature Verification (SAV) schemes [31, 45, 83, 85]. The aim of such schemes is to reduce the gap between the computational cost of the signing algorithm and the one of the verification algorithm in pairing-based schemes. There exist also work on server-aided signature generation, however in this case the focus is not on efficiency [8, 56].

Relying on a server to carry out expensive computations is a natural solution in applications where resource-constrained devices are required to perform computations above the device capacity. From this point of view, server-aided verification renders computationally heavy signatures accessible to a wide range of resource-limited devices (*e.g.,* smartcards, small-battery smartphones) without affecting the device's performance or battery life. The idea behind this solution is to replace the verification algorithm of a signature scheme with an interactive protocol between the computationally weak verifier and the computationally powerful but untrusted server (see Figure 4).

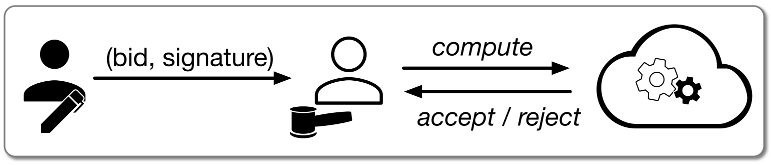


Figure 4: Application scenario for server-aided verification: signed auctions.

A bit more formally, SAV exploits the fact that the verification algorithm of any signature scheme can be split into two parts: a computationally expensive part (that includes most of the operations performed for the verification) and a lightweight equality-check part (see Figure 5). The aim is to replace the computationally expensive part with an interactive protocol that has the same functionality and is more efficient (at least in terms of computational cost for the delegator-verifier). Involving one more entity in the signature verification introduces new privacy and security concerns.

$$\begin{aligned}
&\mathsf{SetUp}(1^\lambda) \to \mathsf{gp} = \mathsf{BilinGroup}\\
&\mathsf{KeyGen}(\mathsf{gp}) \to \mathsf{pk} = g^{\mathsf{sk}},\ \mathsf{sk} \leftarrow \mathbb{Z}_p\\
&\mathsf{Sign}(\mathsf{sk}, m) \to \sigma = \mathsf{Hash}(m)^{\mathsf{sk}}\\
&\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to e(\sigma, g) \overset{?}{=} e(\mathsf{Hash}(m), \mathsf{pk})
\end{aligned}$$

Figure 5: The BLS [17] signature scheme. The expensive computations in the verification algorithm are highlighted with gray background. SAV schemes aim at reducing the gap between the computational cost of Sign and Verify.

There have been some attempts to provide a formal security framework for server-aided verification of signatures [31, 84, 85] and **Paper C** contributes to this line by proposing a more realistic security model and new SAV schemes that achieve stronger notions of security and privacy.

Distance-Bounding Authentication Protocols

Distance-Bounding Authentication Protocols (DBAP) [5, 20] are two-party interactive protocols that allow one entity (called the prover) to authenticate to a verifier under the following conditions: (1) the prover is legitimate and (2) the prover lies within a fixed radius from the verifier. The first condition is checked using a challenge-response approach: the verifier sends a (usually one-bit) challenge c, the prover computes the (usually one-bit) response r using a secret key and some light-weight cryptographic tools. The second condition is checked by equipping the verifier with a clock and measuring the time elapsed between sending c and receiving r. To prove its proximity to the verifier, the prover computes its r immediately after receiving c. To increase accuracy, DBAPs run a series of rapid challenge-response exchanges between the verifier and the prover. Figure 6 depicts the setting of DBAPs. In a nutshell, distance-bounding authentication protocols

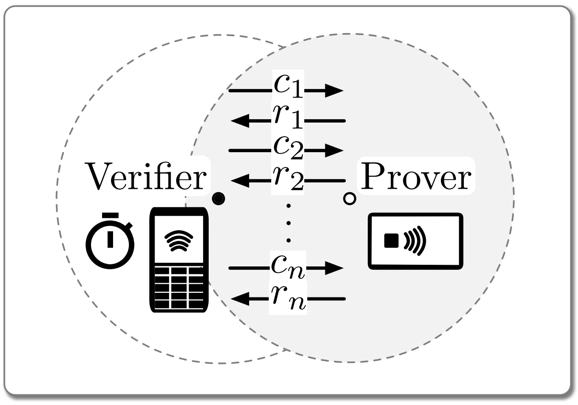


Figure 6: Schematic explanation of distance-bounding authentication. The verifier is a terminal for contact-less payments, the prover is a contact-less smartcard.

blend cryptographic primitives with timing tools to achieve accurate authentication. This dual nature is motivated by real world needs: DBAPs represent the best mitigation against severe attacks such as the ones described below.

Contact-less debit-cards, credit-cards and smartcards in general were designed to bring together security and usability. The chip present in contact-less cards is able to carry out quite sophisticated cryptographic computations once it is brought to life by a magnetic field. In order to authorize the card functionality (*e.g.,* small financial transactions) cardholders need to simply wave the card in front of a terminal machine (*e.g.,* point-of-sale). Within a few seconds the smartcard and the terminal *communicate* with each other and determine whether the functionality (*e.g.,* payment) was successful or not. Unfortunately, the most common contact-less EMV[1] payment protocols (Visa's payWave and MasterCard's PayPass) have flaws and have been shown vulnerable to relay attacks [13, 30, 38] that can be performed even with smartphones [66]. Such attacks may lead to undesirable consequences including changing the amount being charged or the party to be paid. For instance, a businessman seated in a café with his contact-less credit card 'safely' put in his pocket, may be the victim of an attack where an antenna bridges the communication between a contact-less terminal in the jewellery shop next to the café and the businessman's card. By relaying the communication through the antenna, the attacker in the shop may be able to pay the jewellery with the businessman's money! Similar

[1]EMV stands for Europay, MasterCard, and Visa.

attacks have been setup to amplify the communication range of RFID car-keys and unlock cars, while the keys were not in their physical proximity [41].

Relay attacks are a special family of man-in-the-middle attacks where the attacker bridges communications between two parties (the victims). Concretely, the relay-attacker is in communication with both parties and merely relays messages between the victims without manipulating them or even necessarily reading them. What makes relay attacks so dangerous is that in order to tamper with the protocol the adversary does not need to know the details of the protocol or to break the underlying cryptographic functions, it simply relays messages. A quaint example of relay attack is the little girl playing against two chess masters [33]. All the little girl needs to do is to challenge two Grandmasters at postal chess and relay moves between them. Without knowing the rules of the game, the little girl will win (or have a tie) in one of the two games.

The only way to distinguish a response that is being relayed from one that is directly sent by the card to the terminal is to measure how long it takes for the response to reach the terminal. As contact-less communication happens at most at the speed of light, accurate clocks would be able to detect a time difference that corresponds to half a meter space [20]. Therefore, a protocol that combines light-weight cryptographic functions with physical time measurements represents the natural solution against relay attacks. The keyed cryptographic functions are used in a challenge-response framework to authenticate the prover (*e.g.,* a contact-less smartcard) while the recorded round-trip-times of the communication provide an upper-bound on the maximal distance between the prover and the verifier (*e.g.,* contact-less card reader). These are exactly the characteristics of distance-bounding protocols.

Brands and Chaum's seminal work on distance-bounding [20] was followed by a long series of proposals [19, 51, 59, 74]. **Paper D** [86] provides the first formal framework to describe the main classes of existing distance-bounding protocols and also puts forward a general method to extend traditional prover-verifier protocols to the three-participant setting of prover-linker-verifier (two-hop distance estimation).

Biometric Authentication Protocols

While distance-bounding protocols authenticate a user (the prover) via a device she holds, biometric-based authentication relies solely on the user's human features. Biometric Authentication Protocols (BAP) allow quick, accurate and user-friendly authentication of people. In a nutshell, all you need to do is to provide the system with one biometric trait (*e.g.,* your fingerprint or iris scan) and from that point on the system is able to recognize you. In general, biometric traits are distinctive characteristics that are measurable and identify (almost) uniquely each individual. Therefore by measuring a fresh biometric template and comparing it with a reference, the system can recognize people and reject impostors claiming to be someone they are not. Common biometric credentials are: fingerprint [88], iris [35], and face shape [78].

Figure 7 provides a high-level intuition of the main aspects of biometric authentication. To give a concrete example, consider an access gate to a military facility. The gate is equipped with a sensor that scans the soldiers' iris. The iris scan transforms the biometric trait into a digital credential that is compared to a stored biometric template for the soldier. Access will be granted only after the person has been recognized as an authorized soldier in the military facility.

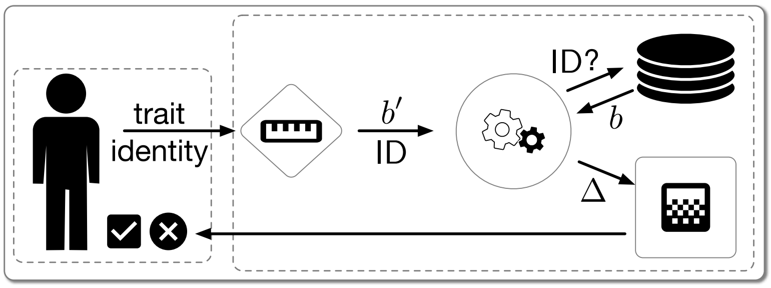


Figure 7: Schematic explanation of how biometric authentication works. The user provides a biometric trait and an identity. The sensor extracts from the trait a biometric template b' for identity ID. The system retrieves the reference template b corresponding to ID and performs a *matching process*. If b is *close enough* to b' (*i.e.,* Δ is small) the user is accepted, otherwise she is rejected.

Biometric authentication has become popular thanks to its usability and user-dependent nature, properties that cannot be achieved with classical authentication methods (*e.g.,* passwords, distance-bounding). In particular, biometric authentication removes the need for users to memorize complicated, long passwords or to carry along special secret tokens. Moreover, biometric credentials are characteristic features naturally bound to the user's body, are hard to steal, reproduce and to spoof [7, 80]. This very same advantageous property, however, raises serious security and privacy concerns in the case of a biometric trait being compromised (cloned, forged).

Unlike passwords or tokens, biometric credentials cannot be kept secret or hidden, and stolen biometrics cannot be revoked as easily [3]. Compromised biometric credentials have an even stronger impact than spoofed passwords or stolen tokens. With a stolen biometric credential attackers can perform crimes such as identity theft and individual profiling and tracking [71, 80]. Moreover, from stolen biometrics traits one can learn sensitive information about the owners, including ethnicity, genetic information [75], medical diseases [12] and can use these data to compromise health records [54].

Motivated by the high sensitivity of biometric data, in the past years several privacy-preserving biometric authentication protocols have been proposed [7, 82, 87]. Such protocols are designed to resist specific attack scenarios including the *biometric reference recovery* attack. In this attack, an unauthorized entity tries to recover the (plaintext) reference biometric template b for a target user ID. A successful reference recovery attack has particularly harmful consequences: the knowledge of the raw credential b gives unauthorized access to any system that uses b as the reference template for user ID and may additionally leak sensitive information about the user's physical characteristics and genetics.

Privacy-preserving biometric authentication protocols make use of advanced cryptographic techniques (such as Oblivious Transfer and Homomorphic Encryption) and are based on a distributed setting, where several entities take part in the protocol. The main reason for this approach is to minimize the amount of information known by each entity.

In **Paper E** [68] we generalize Abidin, Pagnin and Mitrokotsa's biometric reference recovery attack [3] to a wider family of BAPs and investigate the leakage of information that affects biometric authentication. In **Paper F** [70] we show how to mitigate Abidin's attack [3] using Verifiable Computation techniques.

Summary of Papers and Contributions

We hope this will inspire others to work in this fascinating area in which participation has been discouraged in the recent past by a nearly total government monopoly.

Diffie and Hellman, 1976 [37]

This section provides an overview of the main results of the papers included in Part II of this thesis. It also contains descriptions of my contributions to each work.

Multi-Key Homomorphic Authenticators

Problem statement and related work. Homomorphic authenticators enable a client to authenticate a large collection of data in such a way that any third party can generate a short authenticator vouching for the correctness of the output of some computation on the data and the authenticators. Previous works proposed Homomorphic signatures or homomorphic MAC schemes that could support computations of linear functions [16] or of more expressive polynomials [15, 49]. All the aforementioned schemes are however single-key, *i.e.,* computations can only be performed on data generated with a single secret key. This characteristic limits the application range of homomorphic authenticators to non-collaborative settings as it prevents the correct authentication of any computation that requires input from entities with different secret keys.

Consider the earlier example of a school database. Homomorphic signatures enable teachers to upload signed grades and anyone else (*e.g.,* the school director or the students' parents) to check for the authenticity of simple statistics on the grades. Unforgeability ensures that the students cannot upload fake grades. Homomorphic signatures schemes, however, do not directly support authenticated statistics on grades generated with different secret keys. In particular, in our example it would not be possible to authenticate the outcome of computations that involve grades by different teachers. To achieve this property, the signature scheme would need to be homomorphic even among messages signed with different secret keys, in other words, be multi-key and homomorphic.

Contributions and their implications. In this paper, we introduce the notion of Multi-Key Homomorphic Authenticators (MK-HAs), a reasonable security model for this new primitive and two independent constructions. MK-HAs extend the existing notions of Homomorphic Signatures and Homomorphic Message Authentication Codes to support computations on data generated by different secret keys while relying on succinct authenticators, *i.e.,* the size of the authenticators depends at most logarithmically on the total number of inputs to the computation. Our Multi-Key HS scheme is based on standard lattices and supports the evaluation of circuits of bounded polynomial depth. Our construction of a Multi-Key Homomorphic MAC is particularly efficient, it is based on pseudorandom functions and supports the evaluation of low-degree arithmetic circuits.

Statement of contributions. This paper is the result of a collaboration between Dario Fiore, Luca Nizzardo, Aikaterini Mitrokotsa and myself. We developed and formalized the new primitive and its security model during my visit at IMDEA funded by CryptoAction. I mainly worked on the Multi-Key Homomorphic MAC construction and its security proofs. In addition, I proposed adding the $\mathbf{Z}$ component to the signatures of the Multi-Key HS scheme to mitigate a special family of forgeries.

Matrioska: A Compiler for Multi-Key Homomorphic Signatures

Problem statement and related work. This paper is a follow-up of our work on multi-key homomorphic authenticators [39]. Existing multi-key homomorphic signature schemes are ad-hoc adaptations of a single-key homomorphic signature [39] or derived by a generic construction that exploits strong, non-falsifiable cryptographic primitives such as SNARKs [60]. In particular, there is no formal study on the connections between multi-key and single-key HS schemes. This papers fills this gap and provides a generic compiler for constructing a secure multi-key variant of any (sufficiently expressive) single-key homomorphic signature scheme.

Contributions and their implications. In this paper, we establish formal connections between multi-key and single-key homomorphic signatures and *build a (theoretical) bridge* between these two primitives. In more details, we propose Matrioska: the first generic compiler that enhances any (sufficiently expressive) single-key HS with multi-key features under standard falsifiable assumptions only. The existence of this compiler implies that multi-key and single-key homomorphic signatures are equivalent (if they support evaluations of a special class of functions). Moreover, Matrioska can be used to define new multi-key HS schemes from any future proposal of a single-key homomorphic signature. The core of the Matrioska technique is to use the single-key homomorphic evaluation procedure in an original way that allows us to derive t signatures vouching for the authenticity of computations on an arbitrary number of signatures from t different signers. Our approach is completely different from the known ways to obtain multi-key HS schemes [39, 60].

Statement of contributions. This paper is the outcome of a joint work between Dario Fiore and myself. It is a natural follow-up to our paper on multi-key homomorphic authenticators [39] and dives in understanding the relation between single- and multi-key homomorphic signatures. My contribution in this work was to come up with the technical details that made the idea work correctly and securely. All authors contributed equally to the paper.

Anonymous Server-Aided Verification of Signatures

Problem statement and related work. Since the introduction of server-aided verification of signatures [8, 45, 64] there has been a constant development towards more efficient schemes and more realistic security models. The basic security notions for SAV are soundness and existential unforgeability [45]. Wu *et al.* [85] address for the first time attack scenarios where a malicious signer colludes with the server in order to tamper with the outcome of the server-aided verification. Chow *et al.* [31] refine previous definitions and show that the enabler of many attacks to previous SAV schemes is the absence of an integrity check on the results returned by the server. Integrity is not the only concern when outsourcing computations: *how about the signer's privacy?*

Contributions and their implications. In this paper, we provide formal definitions for known and new realistic attack scenarios against server-aided verification of signatures and propose three novel constructions of server-aided verification schemes. Concretely, we present the first compiler that defines a single-round (give-and-take) server-aided verification protocol for any signature given an appropriate verifiable computation scheme. We make use of our compiler to define new SAV schemes that are the first published proposals achieving existential unforgeability and soundness against collusion simultaneously.

In addition, we are the first to consider the notions of signer anonymity and extended existential unforgeability for SAV. To give an idea on the importance of these two attack scenarios consider the case of signed auctions, where bidders sign their bids (messages) to avoid other people impersonating them. In this setting, signer anonymity prevents a malicious server from distinguishing one signer from another. As a consequence, the server cannot 'keep out' target bidders from the auction by making their signatures appear invalid. We also provide an extension to the notion of unforgeability that additionally captures the following attack scenario. Imagine the adversary is a bidder taking part in the auction. In order to steer the price of certain items the adversary could get control over the server used for the aided verification and prevent signatures of higher bids from verifying correctly. Our compiler allows us to determine sufficient requirements on the signature scheme (and/or the verifiable computation scheme) in order to achieve security and anonymity.

Statement of contributions. This paper is the result of a study on server-aided verification of signatures started by Aiketerini Mitrokotsa and myself during a visit at Keisuke Tanaka Sensei's laboratory. Although Dario Fiore is not listed among the authors, he provided me with important technical feedback on the work. I am the main author of this work and developed all the results. This paper is of special importance within my Ph.D. because it represents my 'first step' as an independent researcher on the academic path.

Two-hop Distance-Bounding Protocols: Keep your Friends Close

Problem statement and related work. Traditional distance-bounding authentication protocols aim to authenticate a resource-constrained prover to a (more powerful) verifier [20, 51, 59, 73, 74], assuming that the prover lies within the communication range of the verifier. Albeit most DBAPs are designed for RFID tags, there are works that consider slightly more powerful provers and define public-key privacy-preserving distance-bounding [42, 52] and group distance-bounding [26]. The common factor to all protocols, however, remains that authentication is subjected to the location of the parties: all devices must lie within each others' transmission range. While this requirement represents the main motivation for adopting distance-bounding authentication protocols as a countermeasure against relay attacks, it also limits their application scenarios. In particular, it is hard to directly employ traditional distance-bounding protocols in multiple access control scenarios, in ubiquitous computing environments and even to verify the proximity of a two-hop neighbor. Pagnin *et al.* [69] put forward the idea to extend DBAPs to two-hop neighbors, that is, when the prover and the verifier communicate through an in-between linker. However, a formal framework for constructing two-hop distance-bounding authentication from traditional DBAPs was missing.

Contributions and their implications. In this paper, we extend traditional distance-bounding authentication protocols to also authenticate two-hop neighbors, instead of adjacent devices only. This setting covers environments where the prover is out of the communication range of the verifier, but both parties lie in the proximity of the same untrusted entity, called the linker. We present an intuitive taxonomy of existing DBAPs and provide the first formal framework to extend any register-based protocol to additionally support the two-hop distance-bounding authentication. We also identify connections between attacks against the two-hop and the one-hop settings and implement five two-hop distance-bounding authentication protocols derived from the proposals in [19, 20, 59, 77] using our framework. Our experimental results demonstrate the correctness of our security analysis and the efficiency of our model.

Statement of contributions. This paper is the result of a collaboration started within the objective of a STINT grant awarded to Aikaterini Mitrokotsa and Gerhard Hancke. Anjia Yang is the first author, I am the corresponding author. My contributions in this work include the proposal of the taxonomy of existing distance-bounding authentication protocols, the development of the formalism and the description of the framework for generic extension of register-based DBAP to the two-hop setting. Additionally, I performed the formal security analysis.

On the Leakage of Information in Biometric Authentication

Problem statement and related work. User authentication via biometric credentials has become an increasingly popular way to authenticate people in highly sensitive services such as health care systems [34], but also in everyday tasks such as smartphone unlocking. If not implemented correctly, the wide adoption of these systems might raise severe concerns about the users' privacy and security. Privacy-preserving biometric authentication protocols are designed to mitigate dangerous threats including individual profiling, user tracking and leakage of sensitive information connected to biometric traits (*e.g.,* healthcare data [22, 23, 57]). The current framework for analyzing template security and privacy models distributed biometric authentication systems with internal adversaries [80]. Among the described attacks there is also the so-called center search attack.

Contributions and their implications. In this paper, we provide a formal mathematical framework to analyze the implications of center search attacks against privacy-preserving biometric authentication systems. The standard center search attack is defined on binary strings. In this work, we generalize this efficient hill-climbing technique to vectors with components in $\mathbb{Z}_q$ for $q \geq 2$. As a consequence, certain families of biometric authentication protocols become naturally vulnerable to our biometric template recovery attack. The main implication of our attack is that, if successful, it will let the adversary learn susceptible users' private data that can lead to disclosure of health condition and digital impersonation of the victim. However, not all is lost: one of the starting conditions for the attack to work is the knowledge of a biometric credential that is *close enough* to the target one. We investigated how to get such credentials in a theoretical way and showed that such a problem is equivalent to the set-covering problem which is known to be NP complete [32].

Statement of contributions. This work builds on a previous result by Abidin, Pagnin and Mitrokotsa [4] and has been developed by me, Christos Dimitrakakis, Aysajan Abidin and Aikaterini Mitrokotsa. I am the main author of this paper. I developed the way to generalize Abidin's attack to a larger setting, all the formal details and the proofs.

Revisiting Yasuda et al.'s Biometric Authentication Protocol: Are you Private Enough?

Problem statement and related work. Abidin, Pagnin and Mitrokotsa [4] showed that Yasuda *et al.*'s privacy-preserving biometric authentication protocol [87] is vulnerable to an ad-hoc biometric template recovery attack, and thus can no longer be considered fully privacy-preserving. Among the enablers of Abidin's attack is the fact that the attacker is a malicious computational server. In this paper, we redeem Yasuda's protocol and propose a mitigation to the aforementioned attack.

Contributions and their implications. In this paper, we put forward a generic strategy to transform privacy-preserving BAPs that are secure in the honest-but-curious model into schemes that can tolerate internal malicious attackers. The stronger security guarantee is derived by employing verifiable computation techniques during the matching process. Specifically, we define $\mathsf{BFR} + \mathsf{SHE}$, a biometric authentication protocol that essentially augments Yasuda *et al.*'s proposal [87] with Backes *et al.*'s verifiable computation scheme [6] and is no longer vulnerable to Abidin's attack [4].

We remark that, $\mathsf{BFR} + \mathsf{SHE}$ is still affected by the unavoidable leakage of information inherent to BAPs that employ the Hamming distance in the matching process [68]. However, for the leakage to actually happen, the adversary needs to already hold a matching template, and [68] shows that from a theoretical point of view finding a matching biometric template is an NP-hard problem.

Statement of contributions. This paper is the outcome of Jing Liu's successful master thesis project under the supervision of Aikaterini Mitrokotsa and myself. I contributed with constant support for technical matters during the development of the master thesis and shaped up the results into a publishable paper.

Conclusions and Outlook

Our research isn't finished and much is left to do
For instance, proving theorems completely in haiku

Trotta Gnam [46]

This Ph.D. thesis contributes to the body of knowledge in data and user authentication. It provides high-level explanations of four authentication methods and six state-of-the-art papers that investigate homomorphic signatures, server-aided signature verification, distance-bounding authentication and biometric authentication. This thesis brings in new constructions and aims to inspire further research.

Among the directions for future investigation that stem from the contributions of this thesis we highlight the following. **Paper A** and **Paper B** show how to construct multi-key homomorphic authenticators, but do not aim to give succinct instantiations. Constructing multi-key schemes with authenticators of size independent of the number of users involved in the computation is an open challenge, if one does not want to rely on strong cryptographic tools that are likely to be based on non-falsifiable assumptions (*e.g.,* SNARKs as proposed in [60]). Other directions of research in this area include: combining authentication and confidentiality so that the entity who runs the homomorphic evaluation (*e.g.,* the cloud) does not learn the data over which it computes; and developing *context-hiding* schemes that achieve privacy by revealing no non-trivial information about the computations' inputs. **Paper C** raises awareness about the need for more efficient verifiable computation schemes for bilinear-pairing evaluation that would render a wide range of signature schemes accessible to resource-limited devices via server-aided verification techniques. **Paper D** opens up a new scene for distance-bounding authentication and therefore calls for creative application scenarios in two-hop and multi-hop settings. Finally, **Paper E** and **Paper F** address privacy concerns in biometric authentication and identify the need for new tools to achieve non-leaky biometric template matching.

In addition to the six papers collected in Part II, during my Ph.D. I had several successful collaborations that resulted in the publications reported in the **List of Publications** at the beginning of this thesis. Figure 8 provides subway map inspired representation of my research work so far. Papers are represented as stations, and the four lines follow the paths of data/user privacy privacy, multi-key features, constrained settings and new attacks. The two, black, right-most stations in Figure 8 are outlooks of two on-going works that I describe in what follows.

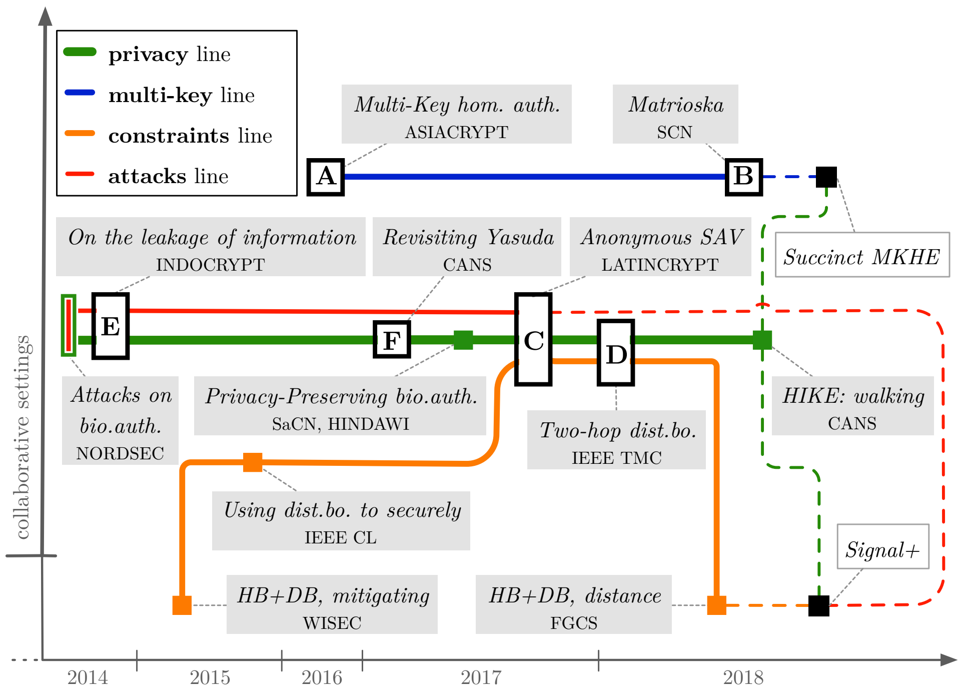


Figure 8: A subway-style map of the papers I contributed to during my Ph.D. The works are organized by the time of publication (or due date) on the x axis, and the size of the supported collaborative setting on the y axis (starting from two users and increasing progressively). Connections between papers are represented as 'subway lines' between 'stations'. The lines are named after the four main themes of my Ph.D. The Latin letters **A-F** refer to the corresponding papers appended to this thesis. Dashed lines lead to results currently under development and highlight directions for future work.

Paper *Succinct MKHE* in Figure 8 puts forward an original way to achieve fully succinct ciphertexts in multi-key additive homomorphic encryption. Exploiting the algebraic structure of some additive homomorphic encryption schemes, we define a new encryption scheme that is a hybrid of secret-key and public-key mechanisms. Our objective is to develop a scheme that supports linearly homomorphic computations on data encrypted by different users and has ciphertexts of constant-length. Paper *Signal+* investigates how to obtain secure asynchronous messaging under the presence of very powerful adversaries. The starting point is the widely deployed Signal protocol. We identify some weaknesses in the design of Signal and propose mitigations and improvements. Our two major goals are to change the trust assumptions of the Signal protocol and to develop a new approach to the ratchet mechanism that supports persistent entity authentication (partnering).

To conclude, I hope this thesis presents a pleasant tour in the land of data and user authentication. Authentication is only one side of the complex polyhedron of security goals in the cryptography world. I am confident that the authentication protocols and schemes we have now and will develop in the future will allow us to happily and safely collaborate in this digital Era even under the presence of malicious entities. Thus, I wish you all to *be more and be merry!*

Bibliography

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2010, pp. 209–236.

[2] Masayuki Abe and Tatsuaki Okamoto. "Provably Secure Partially Blind Signatures". In: *CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2000, pp. 271–286.

[3] Aysajan Abidin and Aikaterini Mitrokotsa. "Security Aspects of Privacy-Preserving Biometric Authentication Based on Ideal Lattices and Ring-LWE". In: *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*. IEEE. 2014, pp. 60–65.

[4] Aysajan Abidin, Elena Pagnin, and Aikaterini Mitrokotsa. "Attacks on Privacy-Preserving Biometric Authentication". In: *Proceedings of the 19th Nordic Conference on Secure IT Systems (NordSec 2014)*. Springer. 2014, pp. 293–294.

[5] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. "A Framework for Analyzing RFID Distance Bounding Protocols". In: vol. 19. 2. IOS Press, 2011, pp. 289–317.

[6] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. "ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data". In: *2015 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society Press, 2015, pp. 271–286.

[7] Manuel Barbosa, Thierry Brouard, Stéphane Cauchie, and Simão Melo de Sousa. "Secure Biometric Authentication with Improved Accuracy". In: *ACISP 08*. Ed. by Yi Mu, Willy Susilo, and Jennifer Seberry. Vol. 5107. LNCS. Wollongong, Australia: Springer, Heidelberg, Germany, 2008, pp. 21–36.

[8] Philippe Béguin and Jean-Jacques Quisquater. "Fast Server-Aided RSA Signatures Secure Against Active Attacks". In: *CRYPTO'95*. Ed. by Don Coppersmith. Vol. 963. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 1995, pp. 57–69.

[9] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract)". In: *30th ACM STOC*. Dallas, TX, USA: ACM Press, 1998, pp. 419–428.

[10] Adam Bender, Jonathan Katz, and Ruggero Morselli. "Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. New York, NY, USA: Springer, Heidelberg, Germany, 2006, pp. 60–79.

[11] Alexandra Boldyreva. “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme”. In: *PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Miami, FL, USA: Springer, Heidelberg, Germany, 2003, pp. 31–46.

[12] James Bolling. “A Window to Your Health”. In: *Special Issue: Retinal Diseases: Capacity and Examples Jacksonville Medicine* 51.9 (2000).

[13] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson. “Chip and Skim: Cloning EMV Cards with the Pre-Play Attack”. In: *Security and Privacy (SP)*. IEEE. 2014, pp. 49–64.

[14] Dan Boneh, Xavier Boyen, and Hovav Shacham. “Short Group Signatures”. In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2004, pp. 41–55.

[15] Dan Boneh and David Mandell Freeman. “Homomorphic Signatures for Polynomial Functions”. In: *EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. LNCS. Tallinn, Estonia: Springer, Heidelberg, Germany, 2011, pp. 149–168.

[16] Dan Boneh and David Mandell Freeman. “Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures”. In: *PKC 2011*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. LNCS. Taormina, Italy: Springer, Heidelberg, Germany, 2011, pp. 1–16.

[17] Dan Boneh, Ben Lynn, and Hovav Shacham. “Short Signatures from the Weil Pairing”. In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Gold Coast, Australia: Springer, Heidelberg, Germany, 2001, pp. 514–532.

[18] Dan Boneh, Emily Shen, and Brent Waters. “Strongly Unforgeable Signatures Based on Computational Diffie-Hellman”. In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. LNCS. New York, NY, USA: Springer, Heidelberg, Germany, 2006, pp. 229–240.

[19] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. “Practical and Provably Secure Distance-Bounding”. In: *ISC 2013*. Ed. by Yvo Desmedt. Vol. 7807. LNCS. Dallas, TX, USA: Springer, Heidelberg, Germany, 2015, pp. 248–258.

[20] Stefan Brands and David Chaum. “Distance-Bounding Protocols (Extended Abstract)”. In: *EUROCRYPT’93*. Ed. by Tor Helleseth. Vol. 765. LNCS. Lofthus, Norway: Springer, Heidelberg, Germany, 1994, pp. 344–359.

[21] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. “Threshold Ring Signatures and Applications to Ad-hoc Groups”. In: *CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2002, pp. 465–480.

[22] Julien Bringer, Herve Chabanne, Melanie Favre, Alain Patey, Thomas Schneider, and Michael Zohner. “GSHADE: Faster Privacy-Preserving Distance Computation and Biometric Identification”. In: *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*. ACM. 2014, pp. 187–198.

[23] Julien Bringer, Hervé Chabanne, and Alain Patey. “Shade: Secure Hamming Distance Computation from Oblivious Transfer”. In: *FC 13*. Springer. 2013, pp. 164–176.

[24] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. "Redactable Signatures for Tree-Structured Data: Definitions and Constructions". In: *ACNS 10*. Ed. by Jianying Zhou and Moti Yung. Vol. 6123. LNCS. Beijing, China: Springer, Heidelberg, Germany, 2010, pp. 87–104.

[25] Jan Camenisch and Anna Lysyanskaya. "A Signature Scheme with Efficient Protocols". In: *SCN 02*. Ed. by Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano. Vol. 2576. LNCS. Amalfi, Italy: Springer, Heidelberg, Germany, 2003, pp. 268–289.

[26] Srdjan Capkun, Karim M. El Defrawy, and Gene Tsudik. "Group Distance Bounding Protocols - (Short Paper)". In: *TRUST 11*. Pittsburgh, PA, USA, 2011, pp. 302–312.

[27] Jae Choon Cha and Jung Hee Cheon. "An Identity-Based Signature from Gap Diffie-Hellman Groups". In: *PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Miami, FL, USA: Springer, Heidelberg, Germany, 2003, pp. 18–30.

[28] David Chaum. "Blind Signatures for Untraceable Payments". In: *CRYPTO 82*. Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman. Santa Barbara, CA, USA: Plenum Press, New York, USA, 1982, pp. 199–203.

[29] David Chaum and Eugène van Heyst. "Group Signatures". In: *EUROCRYPT 91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Brighton, UK: Springer, Heidelberg, Germany, 1991, pp. 257–265.

[30] Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel, and Matthew Thompson. "Relay Cost Bounding for Contactless EMV Payments". In: *FC 2015*. Ed. by Rainer Böhme and Tatsuaki Okamoto. Vol. 8975. LNCS. San Juan, Puerto Rico: Springer, Heidelberg, Germany, 2015, pp. 189–206.

[31] Sherman S. M. Chow, Man Ho Au, and Willy Susilo. "Server-Aided Signatures Verification Secure against Collusion Attack (Short Paper)". In: *ASIACCS 11*. Ed. by Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong. Hong Kong, China: ACM Press, 2011, pp. 401–405.

[32] Vasek Chvatal. "A Greedy Heuristic for the Set-Covering Problem". In: *Mathematics of operations research* 4.3 (1979), pp. 233–235.

[33] John Horton Conway. "On Numbers and Games". In: *London Mathematical Society Monographs*. 6. Academic Press London-New-San Francisco, 1976.

[34] Ashok Kumar Das and Adrijit Goswami. "A Secure and Efficient Uniqueness-and-Anonymity-Preserving Remote User Authentication Scheme for Connected Health Care". In: *Journal of Medical Systems* 37.3 (2013), p. 9948.

[35] John Daugman. "How Iris Recognition Works". In: *The essential guide to image processing*. Elsevier, 2009, pp. 715–739.

[36] Yvo Desmedt. "Computer security by redefining what a computer is". In: *Proceedings on the 1992-1993 workshop on New security paradigms*. ACM. 1993, pp. 160–166.

[37] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654.

[38] Saar Drimer and Steven J. Murdoch. "Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks". In: *USENIX 97*. Boston, MA, USA, 2007.

[39] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. "Multi-key Homomorphic Authenticators". In: *ASIACRYPT 2016, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. LNCS. Hanoi, Vietnam: Springer, Heidelberg, Germany, 2016, pp. 499–530.

[40] Dario Fiore and Elena Pagnin. "Matrioska: A Compiler for Multi-Key Homomorphic Signatures". In: *SCN 18*. LNCS. Amalfi, Italy: Springer, Heidelberg, Germany, 2018.

[41] Aurélien Francillon, Boris Danev, and Srdjan Capkun. "Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars". In: *NDSS 2011*. San Diego, CA, USA: The Internet Society, 2011.

[42] Sébastien Gambs, Cristina Onete, and Jean-Marc Robert. "Prover Anonymous and deniable Distance-Bounding Authentication". In: *ASIACCS 14*. Ed. by Shiho Moriai, Trent Jaeger, and Kouichi Sakurai. Kyoto, Japan: ACM Press, 2014, pp. 501–506.

[43] Rosario Gennaro and Daniel Wichs. "Fully Homomorphic Message Authenticators". In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Bengalore, India: Springer, Heidelberg, Germany, 2013, pp. 301–320.

[44] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. Bethesda, MD, USA: ACM Press, 2009, pp. 169–178.

[45] Marc Girault and David Lefranc. "Server-Aided Verification: Theory and Practice". In: *ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. LNCS. Chennai, India: Springer, Heidelberg, Germany, 2005, pp. 605–623.

[46] Trotta Gnam. "Zero-Knowledge Made Easy so It Won't Make You Dizzy - (A Tale of Transaction Put in Verse About an Illicit Kind of Commerce)". In: *SCN 16*. Ed. by Vassilis Zikas and Roberto De Prisco. Vol. 9841. LNCS. Amalfi, Italy: Springer, Heidelberg, Germany, 2016, pp. 191–197.

[47] Shafi Goldwasser and Mihir Bellare. "Lecture Notes on Cryptography". In: *http://www. cs. ucsd. edu/users/mihir/papers/gb.html* (2015).

[48] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks". In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 281–308.

[49] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. "Leveled Fully Homomorphic Signatures from Standard Lattices". In: *47th ACM STOC*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. Portland, OR, USA: ACM Press, 2015, pp. 469–477.

[50] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". In: *ACM CCS 06*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. Alexandria, Virginia, USA: ACM Press, 2006, pp. 89–98.

[51] Gerhard P. Hancke and Markus G. Kuhn. "An RFID Distance Bounding Protocol". In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, (SecureComm)*. 2005, pp. 67–73.

[52] Jens Hermans, Roel Peeters, and Cristina Onete. "Efficient, Secure, Private Distance Bounding Without Key Updates". In: *Security and privacy in wireless and mobile networks (WiSec)*. ACM. 2013, pp. 207–218.

[53] Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Ràfols. "Short Attribute-Based Signatures for Threshold Predicates". In: *CT-RSA 2012*. Ed. by Orr Dunkelman. Vol. 7178. LNCS. San Francisco, CA, USA: Springer, Heidelberg, Germany, 2012, pp. 51–67.

[54] Anil K Jain, Karthik Nandakumar, and Abhishek Nagar. "Biometric Template Security". In: Hindawi Publishing Corp., 2008, p. 113.

[55] Markus Jakobsson and Ari Juels. "Proofs of Work and Bread Pudding Protocols". In: *Secure Information Networks: Communications and Multimedia Security (CMS '99)*. Springer, 1999, pp. 258–272.

[56] Markus Jakobsson and Susanne Wetzel. "Secure Server-Aided Signature Generation". In: *PKC 2001*. Ed. by Kwangjo Kim. Vol. 1992. LNCS. Cheju Island, South Korea: Springer, Heidelberg, Germany, 2001, pp. 383–401.

[57] Ayman Jarrous and Benny Pinkas. "Secure Hamming Distance Based Computation and Its Applications". In: *ACNS 09*. Ed. by Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud. Vol. 5536. LNCS. Paris-Rocquencourt, France: Springer, Heidelberg, Germany, 2009, pp. 107–124.

[58] Ari Juels and Stephen A. Weis. "Authenticating Pervasive Devices with Human Protocols". In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2005, pp. 293–308.

[59] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. "The Swiss-Knife RFID Distance Bounding Protocol". In: *ICISC 08*. Ed. by Pil Joong Lee and Jung Hee Cheon. Vol. 5461. LNCS. Seoul, Korea: Springer, Heidelberg, Germany, 2009, pp. 98–115.

[60] Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, and Sherman S. M. Chow. "A Zoo of Homomorphic Signatures: Multi-Key and Key-Homomorphism". Cryptology ePrint Archive, Report 2016/834, `http://eprint.iacr.org/2016/834`. 2016.

[61] Byoungcheon Lee, Heesun Kim, and Kwangjo Kim. "Strong Proxy Signature and its Applications". In: *Proceedings of SCIS*. Vol. 2001. 2001, pp. 603–608.

[62] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. "Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures Without Trapdoors". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Vienna, Austria: Springer, Heidelberg, Germany, 2016, pp. 1–31.

[63] Benoît Libert, Thomas Peters, and Moti Yung. "Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 2015, pp. 296–316.

[64] Chae Hoon Lim and Pil Joong Lee. "Server (Prover/Signer)-Aided Verification of Identity Proofs and Signatures". In: *EUROCRYPT'95*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Vol. 921. LNCS. Saint-Malo, France: Springer, Heidelberg, Germany, 1995, pp. 64–78.

[65] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. "Attribute-Based Signatures". In: *CT-RSA 2011*. Ed. by Aggelos Kiayias. Vol. 6558. LNCS. San Francisco, CA, USA: Springer, Heidelberg, Germany, 2011, pp. 376–392.

[66] Konstantinos Markantonakis, Lishoy Francis, Gerhard Hancke, and Keith Mayes. “Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones”. In: *Radio Frequency Identification System Security: RFIDsec* 12 (2012), p. 21.

[67] Elena Pagnin, Carlo Brunetta, and Pablo Picazo-Sanchez. “HIKE: Walking the Privacy Trail”. In: *Cryptology and Network Security (CANS)*. LNCS. Naples, Italy, 2018.

[68] Elena Pagnin, Christos Dimitrakakis, Aysajan Abidin, and Aikaterini Mitrokotsa. “On the Leakage of Information in Biometric Authentication”. In: *INDOCRYPT ’14*. Ed. by Willi Meier and Debdeep Mukhopadhyay. Vol. 8885. LNCS. New Delhi, India: Springer, Heidelberg, Germany, 2014, pp. 265–280.

[69] Elena Pagnin, Gerhard P. Hancke, and Aikaterini Mitrokotsa. “Using Distance-Bounding Protocols to Securely Verify the Proximity of Two-hop Neighbours”. In: *IEEE Communications Letters* 19.7 (2015), pp. 1173–1176.

[70] Elena Pagnin, Jing Liu, and Aikaterini Mitrokotsa. “Revisiting Yasuda et al.’s Biometric Authentication Protocol: Are you Private Enough?” In: *Cryptology and Network Security (CANS)*. LNCS. Hong Kong, 2017.

[71] Elena Pagnin and Aikaterini Mitrokotsa. “Privacy-preserving biometric authentication: challenges and directions”. In: *Security and Communication Networks* (2017). Article ID 7129505.

[72] Elena Pagnin, Aikaterini Mitrokotsa, and Keisuke Tanaka. “Anonymous Single-Round Server-Aided Verification”. In: *5th International Conference on Cryptology and Information Security in Latin America* (2017).

[73] Elena Pagnin, Anjia Yang, Gerhard P. Hancke, and Aikaterini Mitrokotsa. “HB+DB, Mitigating Man-in-the-Middle Attacks against HB+ with Distance Bounding”. In: *Security & Privacy in Wireless and Mobile Networks (WiSec)*. ACM. 2015, 3:1–3:6.

[74] Elena Pagnin, Anjia Yang, Qiao Hu, Gerhard Hancke, and Aikaterini Mitrokotsa. “HB+ DB: Distance Bounding Meets Human Based Authentication”. In: *Future Generation Computer Systems* 80 (2018), pp. 627–639.

[75] LS Penrose. “Dermatoglyphic topology”. In: *Nature* 205.4971 (1965), pp. 544–546.

[76] Charles Rackoff and Daniel R. Simon. “Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack”. In: *CRYPTO’91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 1992, pp. 433–444.

[77] Jason Reid, Juan Manuel González Nieto, Tee Tang, and Bouchra Senadji. “Detecting Relay Attacks with Timing-Based Protocols”. In: *ASIACCS 07*. Ed. by Feng Bao and Steven Miller. Singapore: ACM Press, 2007, pp. 204–213.

[78] M. Savvides, B. V. K. Vijaya Kumar, and P. K. Khosla. “Cancelable biometric filters for face recognition”. In: *International Conference on Pattern Recognition, ICPR*. Vol. 3. 3. 2004, pp. 922–925.

[79] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. “Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems”. In: *AFRICACRYPT 09*. Ed. by Bart Preneel. Vol. 5580. LNCS. Gammarth, Tunisia: Springer, Heidelberg, Germany, 2009, pp. 198–216.

[80] Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys. "A framework for analyzing template security and privacy in biometric authentication systems". In: *IEEE Transactions on Information Forensics and Security* 7.2 (2012), pp. 833–841.

[81] William Stallings. *Cryptography and network security: principles and practice.* Pearson Education, 2003.

[82] Alex Stoianov. "Cryptographically secure biometrics". In: *Biometric Technology for Human Identification VII.* Vol. 7667. International Society for Optics and Photonics. 2010, p. 76670.

[83] Zhiwei Wang. "A new construction of the server-aided verification signature scheme". In: *Mathematical and Computer Modelling* 55.1 (2012), pp. 97–101.

[84] Zhiwei Wang, Licheng Wang, Yixian Yang, and Zhengming Hu. "Comment on Wu et al.'s Server-aided Verification Signature Schemes." In: *International Journal of Network Security* 10.2 (2010), pp. 158–160.

[85] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. "Provably secure server-aided verification signatures". In: *Computers & Mathematics with Applications* 61.7 (2011), pp. 1705 –1723.

[86] A. Yang, E. Pagnin, A. Mitrokotsa, G. P. Hancke, and D. S. Wong. "Two-hop Distance-Bounding Protocols: Keep your Friends Close". In: *IEEE Transactions on Mobile Computing* 17.7 (2018), pp. 1723–1736.

[87] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokomaya, and T. Kashiba. "Practical packing method in somewhat homomorphic encryption". In: *DPM/SETOP.* Vol. 8147. LNCS. Springer Berlin Heidelberg, 2013, pp. 34–50.

[88] Naser Zaeri. "Minutiae-Based fingerprint extraction and recognition". In: *Biometrics.* InTech, 2011.

Part II

Collection of Papers

Paper A

Multi-Key Homomorphic Authenticators

DARIO FIORE, AIKATERINI MITROKOTSA, LUCA NIZZARDO, AND ELENA PAGNIN

Abstract. Homomorphic authenticators (HAs) enable a client to authenticate a large collection of data elements $m_1, \ldots, m_t$ and outsource them, along with the corresponding authenticators, to an untrusted server. At any later point, the server can generate a *short* authenticator vouching for the correctness of the output y of a function f computed on the outsourced data, i.e., $y = f(m_1, \ldots, m_t)$. Recently researchers have focused on HAs as a solution, with minimal communication and interaction, to the problem of delegating computation on outsourced data. The notion of HAs studied so far, however, only supports executions (and proofs of correctness) of computations over data authenticated by a single user. Motivated by realistic scenarios (ubiquitous computing, sensor networks, etc.) in which large datasets include data provided by multiple users, we study the concept of *multi-key homomorphic authenticators*. In a nutshell, multi-key HAs are like HAs with the extra feature of allowing the holder of public evaluation keys to compute on data authenticated under different secret keys. In this paper, we introduce and formally define multi-key HAs. Secondly, we propose a construction of a multi-key homomorphic signature based on standard lattices and supporting the evaluation of circuits of bounded polynomial depth. Thirdly, we provide a construction of multi-key homomorphic MACs based only on pseudorandom functions and supporting the evaluation of low-degree arithmetic circuits. Albeit being less expressive and only secretly verifiable, the latter construction presents interesting efficiency properties.

Keywords. Homomorphic Signatures, Homomorphic Message Authentication.

Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), 2016.

Multi-Key Homomorphic Authenticators

1 Introduction

The technological innovations offered by modern IT systems are changing the way digital data is collected, stored, processed and consumed. As an example, think of an application where data is collected by some organizations (e.g., hospitals), stored and processed on remote servers (e.g., the Cloud) and finally consumed by other users (e.g., medical researchers) on other devices. On one hand, this computing paradigm is very attractive, particularly as data can be shared and exchanged by multiple users. On the other hand, it is evident that in such scenarios one may be concerned about security: while the users that collect and consume the data may trust each other (up to some extent), trusting the Cloud can be problematic for various reasons. More specifically, two main security concerns to be addressed are those about the *privacy* and *authenticity* of the data stored and processed in untrusted environments.

While it is widely known that privacy can be solved in such a setting using, e.g., homomorphic encryption [27], in this work we focus on the orthogonal problem of providing authenticity of data during computation. Towards this goal, our contribution is on advancing the study of *homomorphic authenticators* (HAs), a cryptographic primitive that has been the subject of recent work [9, 26, 30, 33].

Homomorphic Authenticators. Using an homomorphic authenticator (HA) scheme a user Alice can authenticate a collection of data items $m_1, \dots, m_t$ using her secret key, and send the authenticated data to an untrusted server. The server can execute a program $\mathcal{P}$ on the authenticated data and use a public evaluation key to generate a value $\sigma_{\mathcal{P},y}$ vouching for the correctness of $y = \mathcal{P}(m_1, \dots, m_t)$. Finally, a user Bob who is given the tuple $(\mathcal{P}, y, \sigma_{\mathcal{P},y})$ and Alice's verification key can use the authenticator to verify the authenticity of y as output of the program $\mathcal{P}$ executed on data authenticated by Alice. In other words, Bob can check that the server did not tamper with the computation's result. Alice's verification key can be either secret or public. In the former case, we refer to the primitive as *homomorphic MACs* [11, 26], while in the latter we refer to it as *homomorphic signatures* [9]. One of the attractive features of HAs is that the authenticator $\sigma_{\mathcal{P},y}$ is *succinct*, i.e., much shorter than $\mathcal{P}$'s input size. This means that the server can execute a program on a huge amount of data and convince Bob of its correctness by sending him only a short piece of information. As discussed in previous work (e.g., [5, 26, 30]), HAs provide a nice solution, with minimal communication and interaction, to the problem of delegating computations on outsourced data, and thus can be preferable to verifiable computation (more details on this comparison appear in Section 1.1).

Our Contribution: Multi-Key Homomorphic Authenticators. Up to now, the notion of HAs has inherently been single-key, i.e., homomorphic computations are allowed only on data authenticated using the same secret key. This characteristic is obviously a limitation and prevents HA schemes from suiting scenarios where the data is provided (and authenticated) by multiple users. Consider the previously mentioned example of healthcare institutions which need to compute on data collected by several hospitals or even some remote-monitored patients. Similarly, it is often required to compute statistics for time-series data collected from multiple users e.g., to monitor usage data in smart metering, clinical research or to monitor the safety of buildings. Another application scenario is in distributed networks of sensors. Imagine for instance a network of sensors where each sensor is in charge of collecting data about air pollution in a certain area of a city, it sends its data to a Cloud server, and then a central control unit asks the Cloud to compute on the data collected by the sensors (e.g., to obtain the average value of air pollution in a large area).

A trivial solution to address the problem of computing on data authenticated by multiple users is to use homomorphic authenticators in such a way that all data providers share the *same* secret authentication key. The desired functionality is obviously achieved since data would be authenticated using a single secret key. This approach however has several drawbacks. The first one is that users need to coordinate in order to agree on such a key. The second one is that in such a setting there would be no technical/legal way to differentiate between users (e.g., to make each user accountable for his/her duties) as any user can impersonate all the other ones. The third and more relevant reason is that sharing the same key exposes the overall system to way higher risks of attacks and makes disaster recovery more difficult: if a single user is compromised the whole system is compromised too, and everything has to be reinitialized from scratch.

In contrast, this paper provides an innovative solution through the notion of *multi-key homomorphic authenticators* (multi-key HAs). This primitive guarantees that the corruption of one user affects the data of that user only, but does not endanger the authenticity of computations among the other (un-corrupted) users of the system. Moreover, the proposed system is dynamic, in the sense that compromised users can be assigned new keys and be easily reintegrated.

1.1 An Overview of Our Results

Our contribution is mainly threefold. First of all, we elaborate a suitable definition of multi-key HAs. Second, we propose the first construction of a multi-key homomorphic signature (i.e., with public verifiability) which is based on standard lattices and supports the evaluation of circuits of bounded polynomial depth. Third, we present a multi-key homomorphic MAC that is based only on pseudorandom functions and supports the evaluation of low-degree arithmetic circuits. In spite of being less expressive and only secretly verifiable, this last construction is way more efficient than the signature scheme. In what follows, we elaborate more on our results.

Multi-Key Homomorphic Authenticators: What are they? At a high level, multi-key HAs are like HAs with the additional property that one can execute a program $\mathcal{P}$ on data authenticated using different secret keys. In multi-key HAs, Bob verifies using the verification keys of all users that provided inputs to $\mathcal{P}$. These features make multi-key HAs a perfect candidate for applications where multiple users gather and outsource data. Referring to our previous examples, using multi-key HAs each sensor can authenticate and outsource to the Cloud the data it collects; the Cloud can compute statistics on the authenticated data and provide the central control unit with the result along with a certificate vouching for its correctness.

An important aspect of our definition is a mechanism that allows the verifier to keep track of the users that authenticated the inputs of $\mathcal{P}$, i.e., to know which user contributed to each input wire of $\mathcal{P}$. To formalize this mechanism, we build on the model of *labeled data and programs* of Gennaro and Wichs [26] (we refer the reader to Section 3 for details). In terms of security, multi-key HAs allow the adversary to corrupt users (i.e., to learn their secret keys); yet this knowledge should not help the adversary in tampering with the results of programs which involve inputs of honest (i.e., uncorrupted) users only. Our model allows to handle compromised users in a similar way to what occurs with classical digital signatures: a compromised user could be banned by means of a certificate revocation, and could easily be re-integrated via a new key pair.[2] Thinking of the sensor network application, if a sensor in the field gets compromised, the data provided by other sensors remains secure, and a new sensor can be easily introduced in the system with new credentials.

Finally, we require multi-key homomorphic authenticators to be *succinct* in the sense that the size of authenticators is bounded by some fixed polynomial in $(\lambda, n, \log t)$, where λ is the security parameter, n is the number of users contributing to the computation and t is the total number of inputs of $\mathcal{P}$. Although such dependence on n may look undesirable, we stress that it is still meaningful in many application scenarios where n is much smaller than t. For instance, in the application scenario of healthcare institutions a few hospitals can provide a large amount of data from patients.

A Multi-Key Homomorphic Signature for All Circuits. After setting the definition of multi-key homomorphic authenticators, we proceed to construct multi-key HA schemes. Our first contribution is a multi-key homomorphic signature that supports the evaluation of boolean circuits of depth bounded by a fixed polynomial in the security parameter. The scheme is proven secure based on the small integer solution (SIS) problem over standard lattices [37], and tolerates adversaries that corrupt users non-adaptively.[3] Our technique is inspired by the ones developed by Gorbunov, Vaikuntanathan and Wichs [30] to construct a (single-key) homomorphic signature. Our key contribution is on providing a new representation of the signatures that enables to homomorphically compute over them even if they were generated using different keys. Furthermore, our scheme enjoys an additional property, not fully satisfied by [30]: every user can authenticate separately every data item m_i of a collection $m_1, \dots m_t$, and the correctness of computations is guaranteed even when computing on not-yet-full datasets. Although it is possible to modify the scheme in [30] for signing data items separately, the security would only work against adversaries that query the whole dataset. In contrast, we prove our scheme to be secure under a stronger security definition where the adversary can *adaptively* query the various data items, and it can try to cheat by pretending to possess signatures on data items that it never queried (so-called Type 3 forgeries). We highlight that the scheme in [30] is *not* secure under the stronger definition (with Type 3 forgeries) used in this paper, and we had to introduce new techniques to deal with this scenario. This new property is particularly interesting as it enables users to authenticate and outsource data items in a streaming fashion, without ever having to store the whole dataset. This is useful in applications where the dataset size can

[2]Here we mean that this process does not add more complications than the ones already existing for classical digital signatures (e.g., relying on PKI mechanisms).

[3]Precisely, our "core" scheme is secure against adversaries that make non-adaptive signing queries; this is upgraded to adaptive security via general transformations.

be very large or not fixed a priori.

A MULTI-KEY HOMOMORPHIC MAC FOR LOW-DEGREE CIRCUITS. Our second construction is a multi-key homomorphic MAC that supports the evaluation of arithmetic circuits whose degree d is at most polynomial in the security parameter, and whose inputs come from a small number n of users. For results of such computations the corresponding authenticators have at most size $s = \binom{n+d}{d}$.[4] Notably, the authenticator's size is completely independent of the total number of inputs of the arithmetic circuit. Compared to our multi-key homomorphic signature, this construction is only secretly verifiable (i.e., Bob has to know the secret verification keys of all users involved in the computation) and supports a class of computations that is less expressive; also its succinctness is asymptotically worse. In spite of these drawbacks, our multi-key homomorphic MAC achieves interesting features. From the theoretical point of view, it is based on very simple cryptographic tools: a family of pseudorandom functions. Thus, the security relies only on one-way functions. On the practical side, it is particularly efficient: generating a MAC requires only one pseudo-random function evaluation and a couple of field operations; homomorphic computations boil down to additions and multiplications over a multi-variate polynomial ring $\mathbb{F}_p[X_1, \ldots, X_n]$.

1.2 Related Work

Homomorphic MACs and Signatures. Homomorphic authenticators have received a lot of attention in previous work focusing either on homomorphic signatures (publicly verifiable) or on homomorphic MACs (private verification with a secret key). The notion of homomorphic signatures was originally proposed by Johnson *et al.* [33]. The first schemes that appeared in the literature were homomorphic only for linear functions [8, 13–15, 23] and found important applications in network coding and proofs of retrievability. Boneh and Freeman [9] were the first to construct homomorphic signatures that can evaluate more than linear functions over signed data. Their scheme could evaluate bounded-degree polynomials and its security was based on the hardness of the SIS problem in ideal lattices in the random oracle model. A few years later, Catalano *et al.* [16] proposed an alternative homomorphic signature scheme for bounded-degree polynomials. Their solution is based on multi-linear maps and bypasses the need for random oracles. More interestingly, the work by Catalano *et al.* [16] contains the first mechanism to verify signatures faster than the running time of the verified function. Recently, Gorbunov *et al.* [30] have proposed the first (leveled) fully homomorphic signature scheme that can evaluate arbitrary circuits of bounded polynomial depth over signed data. Some important advances have been also achieved in the area of homomorphic MACs. Gennaro and Wichs [26] have proposed a fully homomorphic MAC based on fully homomorphic encryption. However, their scheme is not secure in the presence of verification queries. More efficient schemes have been proposed later [5, 11, 12] that are secure in the presence of verification queries and are more efficient at the price of supporting only restricted homomorphisms. Finally, we note that Agrawal *et al.* [1] considered a notion of *multi-key* signatures for network coding, and proposed a solution which works for linear functions only. Compared to this work, our contribution shows a full-fledged framework for multi-key homomorphic authenticators, and provides solutions that address a more expressive class of computations.

[4] Note that s can be bounded by $poly(n)$ for constant d, or by $poly(d)$ for constant n.

Verifiable Computation. Achieving correctness of outsourced computations is also the aim of *verifiable delegation of computation* (VC) [6, 18, 20, 25, 29, 38]. In this setting, a client wants to delegate the computation of a function f on input x to an untrusted cloud-server. If the server replies with y, the client's goal is to verify the correctness of $y = f(x)$ spending less resources than those needed to execute f. As mentioned in previous work (e.g., [26, 30]) a crucial difference between verifiable computation and homomorphic authenticators is that in VC the verifier has to know the input of the computation – which can be huge – whereas in HAs one can verify by only knowing the function f and the result y. Moreover, although some results of verifiable computation could be re-interpreted to solve scenarios similar to the ones addressed by HAs, results based on VC would still present several limitations. For instance, using homomorphic authenticators the server can prove correctness of $y = f(x)$ with a single message, without needing any special encoding of f from the delegator. Second, HAs come naturally with a composition property which means that the outputs of some computations on authenticated data (which is already authenticated) can be fed as input for follow-up computations. This feature is of particular interest to parallelize and or distribute computations (e.g., MapReduce). Emulating this composition within VC systems is possible by means of certain non-interactive proof systems [7] but leads to complex statements and less natural realizations. A last advantage is that by using HAs, clients can authenticate various (small) pieces of data independently and without storing previously outsourced data. In contrast, most VC systems require clients to encode the whole input in 'one shot', and often such encoding can be used in a single computation only.

Multi-Client Verifiable Computation. Another line of work, closely related to ours is that on *multi-client verifiable computation* [17, 31]. This primitive, introduced by Choi *et al.* [17], aims to extend VC to the setting where inputs are provided by multiple users, and one of these users wants to verify the result's correctness. Choi *et al.* [17] proposed a definition and a multi-client VC scheme which generalizes that of Gennaro *et al.* [25]. The solution in [17], however, does not consider malicious or colluding clients. This setting was addressed by Gordon *et al.* in [31], where they provide a scheme with stronger security guarantees against a malicious server or an arbitrary set of malicious colluding clients.

It is interesting to notice that in the definition of multi-client VC all the clients but the one who verifies can encode inputs independently of the function to be later executed on them. One may thus think that the special case in which the verifier provides no input yields a solution similar to the one achieved by multi-key HAs. However, a closer look at the definitions and the existing constructions of multi-client VC reveals three main differences. (1) In multi-client VC, in order to prove the correctness of an execution of a function f, the server has to wait a message from the verifier which includes some encoding of f. This is not necessary in multi-key HAs where the server can directly prove the correctness of f on previously authenticated data with a single message and without any function's encoding. (2) The communication between the server and the verifier is at least linear in the total number of inputs of f: this can be prohibitive in the case of computations on very large inputs (think of TBytes of data). In contrast, with multi-key HAs the communication between the server and the verifier is proportional only to the number of users, and depends only logarithmically on the total number of inputs. (3) In multi-client VC an encoding of one input can be used in a single computation. Thus, if a user wants to first upload data on the server to later execute many functions on it, then the user has to provide as many encodings as the number of functions to be executed. In contrast, multi-key HAs allow one to encode (i.e., authenticate) every input only once and to use it for proving correctness of computations an *unbounded* number of times.

2 Preliminaries

We collect here the notation and basic definitions used throughout the paper.

Notation. The Greek letter λ is reserved for the security parameter of the schemes. A function $\epsilon(\lambda)$ is said to be *negligible* in λ (denoted as $\epsilon(\lambda) = \mathsf{negl}(\lambda)$) if $\epsilon(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$. When a function can be expressed as a polynomial we often write it as $poly(\cdot)$. For any $n \in \mathbb{N}$, we refer to $[n]$ as $[n] := \{1, \ldots, n\}$. Moreover, given a set $\mathcal{S}$, the notation $s \xleftarrow{\$} \mathcal{S}$ stays for the process of sampling s uniformly at random from $\mathcal{S}$.

Definition 1.1 (Statistical Distance). *Let X, Y denote two random variables with support $\mathcal{X}, \mathcal{Y}$ respectively; the statistical distance between X and Y is defined as $\mathbf{SD}(X,Y) := \frac{1}{2}(\sum_{u \in \mathcal{X} \cup \mathcal{Y}} \mid \Pr[X = u] - \Pr[Y = u] \mid)$. If $\mathbf{SD}(X,Y) = \mathsf{negl}(\lambda)$, we say that X and Y are statistically close and we write $X \overset{\mathsf{stat}}{\approx} Y$.*

Definition 1.2 (Entropy [19]). *The min-entropy of a random variable X is defined as*

$$\mathbf{H}_\infty(X) := -\mathsf{log}\big(\max_x \Pr[X = x]\big).$$

The (average-) conditional min-entropy of a random variable X conditioned on a correlated variable Y is defined as

$$\mathbf{H}_\infty(X \mid Y) := -\mathsf{log}\Big(\mathop{\mathbf{E}}_{y \leftarrow Y}\Big[\max_x \Pr[X = x \mid Y = y]\Big]\Big).$$

The optimal probability of an unbounded adversary guessing X when given a correlated value Y is $2^{-\mathbf{H}_\infty(X|Y)}$.

Lemma 1.1 ([19]). *Let X, Y be arbitrarily random variables where the support of Y lies in $\mathcal{Y}$. Then $\mathbf{H}_\infty(X \mid Y) \geq \mathbf{H}_\infty(X) - \mathsf{log}(\mid \mathcal{Y} \mid)$.*

3 Multi-Key Homomorphic Authenticators

In this section, we present our new notion of *Multi-Key Homomorphic Authenticators* (multi-key HAs). Intuitively, multi-key HAs extend the existing notions of homomorphic signatures [9] and homomorphic MACs [26] in such a way that one can homomorphically compute a program $\mathcal{P}$ over data authenticated using different secret keys. For the sake of verification, in multi-key HAs the verifier needs to know the verification keys of all users that provided inputs to $\mathcal{P}$. Our definitions are meant to be general enough to be easily adapted to both the case in which verification keys are public and the one where verification keys are secret. In the former case, we call the primitive *multi-key homomorphic signatures* whereas in the latter case we call it *multi-key homomorphic MACs*.

As already observed in previous work about HAs, it is important that an authenticator $\sigma_{\mathcal{P},y}$ does not authenticate a value y out of context, but only as the output of a program $\mathcal{P}$ executed on previously authenticated data. To formalize this notion, we build on the model of *labeled data and programs* of Gennaro and Wichs [26]. The idea of this model is that every data item is authenticated under a unique label ℓ. For example, in scenarios where the data is outsourced, such labels can be thought of as a way to index/identify the remotely stored data. A labeled program $\mathcal{P}$, on the other hand, consists of a circuit f where every input wire i has a label ℓ_i. Going back to the outsourcing example, a labeled program is a way to specify on what portion of the outsourced data one should execute a circuit f. More formally, the notion of labeled programs of [26] is recalled below.

Labeled Programs [26]. A labeled program $\mathcal{P}$ is a tuple $(f, \ell_1, \dots, \ell_n)$, such that $f : \mathcal{M}^n \to \mathcal{M}$ is a function of n variables (e.g., a circuit) and $\ell_i \in \{0,1\}^*$ is a label for the i-th input of f. Labeled programs can be composed as follows: given $\mathcal{P}_1, \dots, \mathcal{P}_t$ and a function $g : \mathcal{M}^t \to \mathcal{M}$, the composed program $\mathcal{P}^*$ is the one obtained by evaluating g on the outputs of $\mathcal{P}_1, \dots, \mathcal{P}_t$, and it is denoted as $\mathcal{P}^* = g(\mathcal{P}_1, \dots, \mathcal{P}_t)$. The labeled inputs of $\mathcal{P}^*$ are all the distinct labeled inputs of $\mathcal{P}_1, \dots \mathcal{P}_t$ (all the inputs with the same label are grouped together and considered as a unique input of $\mathcal{P}^*$). Let $f_{id} : \mathcal{M} \to \mathcal{M}$ be the identity function and $\ell \in \{0,1\}^*$ be any label. We refer to $\mathcal{I}_\ell = (f_{id}, \ell)$ as the identity program with label ℓ. Note that a program $\mathcal{P} = (f, \ell_1, \dots, \ell_n)$ can be expressed as the composition of n identity programs $\mathcal{P} = f(\mathcal{I}_{\ell_1}, \dots, \mathcal{I}_{\ell_n})$.

Using labeled programs to identify users. In our notion of multi-key homomorphic authenticators, one wishes to verify the outputs of computations executed on data authenticated under *different* keys. A meaningful definition of multi-key HAs thus requires that the authenticators are not out of context also with respect to the set of keys that contributed to the computation. To address this issue, we assume that every user has an identity id in some identity space ID, and that the user's keys are associated to id by means of any suitable mechanism (e.g., PKI). Next, in order to distinguish among data of different users and to identify to which input wires a certain user contributed, we assume that the label space contains the set ID. Namely, in our model a data item is assigned a label $\ell := (\mathsf{id}, \tau)$, where id is a user's identity, and τ is a tag; this essentially identifies uniquely a data item of user id with index τ. For compatibility with previous notions of homomorphic authenticators, we assume that data items can be grouped in datasets, and one can compute only on data within the same dataset. In our definitions, a dataset is identified by an arbitrary string Δ.[5]

Definition 1.3 (Multi-Key Homomorphic Authenticator)**.** *A multi-key homomorphic authenticator scheme* MKHAut *consists of a tuple of PPT algorithms* (Setup, KeyGen, Auth, Eval, Verify) *satisfying the following properties:* authentication correctness, evaluation correctness, succinctness *and* security. *The five algorithms work as follows:*

$\mathsf{Setup}(1^\lambda)$**.** *The setup algorithm takes as input the security parameter λ and outputs some public parameters* pp*. These parameters include (at least) descriptions of a tag space $\mathcal{T}$, an identity space ID, the message space $\mathcal{M}$ and a set of admissible functions $\mathbb{F}$. Given $\mathcal{T}$ and ID, the label space of the scheme is defined as their cartesian product $L := \mathsf{ID} \times \mathcal{T}$. For a labeled program $\mathcal{P} = (f, \ell_1, \dots, \ell_t)$ with labels $\ell_i := (\mathsf{id}_i, \tau_i) \in L$, we use $\mathsf{id} \in \mathcal{P}$ as compact notation*

[5]Although considering the dataset notion complicates the definition, it also provides some benefits, as we illustrate later in the constructions. For instance, when verifying for the same program $\mathcal{P}$ over different datasets, one can perform some pre-computation that makes further verifications cheap.

for $\mathsf{id} \in \{\mathsf{id}_1, \ldots, \mathsf{id}_t\}$*. The* pp *are input to all the following algorithms, even when not specified.*

$\mathsf{KeyGen}(\mathsf{pp})$**.** *The key generation algorithm takes as input the public parameters and outputs a triple of keys* $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$*, where* sk *is a secret authentication key,* ek *is a public evaluation key and* vk *is a verification key which could be either public or private.*[6]

$\mathsf{Auth}(\mathsf{sk}, \Delta, \ell, \mathsf{m})$**.** *The authentication algorithm takes as input an authentication key* sk*, a dataset identifier* Δ*, a label* $\ell = (\mathsf{id}, \tau)$ *for the message* m*, and it outputs an authenticator* σ*.*

$\mathsf{Eval}(f, \{(\sigma_i, \mathsf{EKS}_i)\}_{i\in[t]})$**.** *The evaluation algorithm takes as input a* t*-input function* $f : \mathcal{M}^t \longrightarrow \mathcal{M}$*, and a set* $\{(\sigma_i, \mathsf{EKS}_i)\}_{i\in[t]}$ *where each* σ_i *is an authenticator and each* EKS_i *is a set of evaluation keys.*[7]

$\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \mathsf{m}, \sigma)$**.** *The verification algorithm takes as input a labeled program* $\mathcal{P} = (f, \ell_1, \ldots, \ell_t)$*, a dataset identifier* Δ*, the set of verification keys* $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$ *corresponding to those identities* id *involved in the program* $\mathcal{P}$*, a message* m *and an authenticator* σ*. It outputs* 0 *(reject) or* 1 *(accept).*

Authentication Correctness. Intuitively, a Multi-Key Homomorphic Autenticator has authentication correctness if the output of $\mathsf{Auth}(\mathsf{sk}, \Delta, \ell, \mathsf{m})$ verifies correctly for m as the output of the identity program $\mathcal{I}_\ell$ over the dataset Δ. More formally, a scheme MKHAut satisfies authentication correctness if for all public parameters $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda)$, any key triple $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, any label $\ell = (\mathsf{id}, \tau) \in \mathbb{L}$ and any authenticator $\sigma \leftarrow \mathsf{Auth}(\mathsf{sk}, \Delta, \ell, \mathsf{m})$, we have $\mathsf{Verify}(\mathcal{I}_\ell, \Delta, \mathsf{vk}, \mathsf{m}, \sigma)$ outputs 1 with all but negligible probability.

Evaluation Correctness. Intuitively, this property says that running the evaluation algorithm on signatures $(\sigma_1, \ldots, \sigma_t)$ such that each σ_i verifies for m_i as the output of a labeled program $\mathcal{P}_i$ over the dataset Δ, it produces a signature σ which verifies for $f(\mathsf{m}_1, \ldots, \mathsf{m}_t)$ as the output of the composed program $f(\mathcal{P}_1, \ldots, \mathcal{P}_t)$ over the dataset Δ. More formally, let us fix the public parameters $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda)$, a set of key triples $\{(\mathsf{sk}_{\mathsf{id}}, \mathsf{ek}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})\}_{\mathsf{id}\in\tilde{\mathsf{ID}}}$ for some $\tilde{\mathsf{ID}} \subseteq \mathsf{ID}$, a dataset Δ, a function $g : \mathcal{M}^t \to \mathcal{M}$, and any set of program/message/authentica-tor triples $\{(\mathcal{P}_i, m_i, \sigma_i)\}_{i\in[t]}$ such that $\mathsf{Verify}(\mathcal{P}_i, \Delta, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, \mathsf{m}_i, \sigma_i) = 1$ for all $i \in [t]$. Let $\mathsf{m}^* = g(\mathsf{m}_1, \ldots, \mathsf{m}_t)$, $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$, and $\sigma^* = \mathsf{Eval}(g, \{(\sigma_i, \mathsf{EKS}_i)\}_{i\in[t]})$ where $\mathsf{EKS}_i = \{\mathsf{ek}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}$. Then, $\mathsf{Verify}(\mathcal{P}^*, \Delta, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{m}^*, \sigma^*) = 1$ holds with all but negligible probability.

Succinctness. A multi-key HA is said to be *succinct* if the size of every authenticator depends only logarithmically on the size of a dataset. However, we allow authenticators to depend on the number of keys involved in the computation. More formally, let $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda)$, $\mathcal{P} = (f, \ell_1, \ldots, \ell_t)$ with $\ell_i = (\mathsf{id}_i, \tau_i)$, $\{(\mathsf{sk}_{\mathsf{id}}, \mathsf{ek}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})\}_{\mathsf{id}\in\mathcal{P}}$, and $\sigma_i \leftarrow \mathsf{Auth}(\mathsf{sk}_{\mathsf{id}_i}, \Delta, \ell_i, \mathsf{m}_i)$ for all $i \in [t]$. A multi-key HA is said to be *succinct* if there exists a fixed polynomial p such that $|\sigma| = p(\lambda, n, \log t)$ where $\sigma = \mathsf{Eval}(g, \{(\sigma_i, \mathsf{ek}_{\mathsf{id}_i})\}_{i\in[t]})$ and $n = |\{\mathsf{id} \in \mathcal{P}\}|$.

[6]As mentioned earlier, the generated triple $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$ will be associated to an identity $\mathsf{id} \in \mathsf{ID}$. When this connection becomes explicit, we will refer to $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$ as $(\mathsf{sk}_{\mathsf{id}}, \mathsf{ek}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$.

[7]The motivation behind the evaluation-keys set EKS_i is that, if σ_i authenticates the output of a labeled program $\mathcal{P}_i$, then $\mathsf{EKS}_i = \{\mathsf{ek}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}$ should be the set of evaluation keys corresponding to identities involved in the computation of $\mathcal{P}_i$.

Remark 1.1. Succinctness is one of the crucial properties that make multi-key HAs an interesting primitive. Without succinctness, a trivial multi-key HA construction is the one where Eval outputs the concatenation of all the signatures (and messages) given in input, and the verifier simply checks each message-signature pair and re-computes the function by himself. Our definition of succinctness, where signatures can grow linearly with the number of keys but logarithmically in the total number of inputs, is also non-trivial, especially when considering settings where there are many more inputs than keys (in which case, the above trivial construction would not work). Another property that can make homomorphic signatures an interesting primitive is privacy—context-hiding—as considered in prior work. Intuitively, context-hiding guarantees that signatures do not reveal information on the original inputs. While we leave the study of context-hiding for multi-key HAs for future work, we note that a trivial construction that is context-hiding but not succinct can be easily obtained with the additional help of non-interactive zero-knowledge proofs of knowledge: the idea is to extend the trivial construction above by requiring the evaluator to generate a NIZK proof of knowledge of valid input messages and signatures that yield the public output. In this sense, we believe that succinctness is the most non-trivial property to achieve in homomorphic signatures, and this is what we focus on in this work.

Security. Intuitively, our security model for multi-key HAs guarantees that an adversary, without knowledge of the secret keys, can only produce authenticators that were either received from a legitimate user, or verify correctly on the results of computations executed on the data authenticated by legitimate users. Moreover, we also give to the adversary the possibility of corrupting users. In this case, it must not be able to cheat on the outputs of programs that get inputs from uncorrupted users only. In other words, our security definition guarantees that the corruption of one user affects the data of that user only, but does not endanger the integrity of computations among the other (un-corrupted) users of the system. We point out that preventing cheating on programs that involve inputs of corrupted users is inherently impossible in multi-key HAs, at least if general functions are considered. For instance, consider an adversary who picks the function $(x_1 + x_2 \mod p)$ where x_1 is supposed to be provided by user Alice. If the adversary corrupts Alice, it can use her secret key to inject any input authenticated on her behalf and thus bias the output of the function at its will.

The formalization of the intuitions illustrated above is more involved. For a scheme MKHAut we define security via the following experiment between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ ($\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}(\lambda)$):

Setup. $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ to obtain the public parameters pp that are sent to $\mathcal{A}$.

Authentication Queries $\mathcal{A}$ can adaptively submit queries of the form $(\Delta, \ell, \mathsf{m})$, where Δ is a dataset identifier, $\ell = (\mathsf{id}, \tau)$ is a label in $\mathsf{ID} \times \mathcal{T}$ and $\mathsf{m} \in \mathcal{M}$ are messages of his choice. $\mathcal{C}$ answers as follows:

- If $(\Delta, \ell, \mathsf{m})$ is the first query for the dataset Δ, $\mathcal{C}$ initializes an empty list $L_\Delta = \emptyset$ and proceeds as follows.
- If $(\Delta, \ell, \mathsf{m})$ is the first query with identity id, $\mathcal{C}$ generates keys $(\mathsf{sk_{id}}, \mathsf{ek_{id}}, \mathsf{vk_{id}}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{pp})$ (that are implicitly assigned to identity id), gives $\mathsf{ek_{id}}$ to $\mathcal{A}$ and proceeds as follows.
- If $(\Delta, \ell, \mathsf{m})$ is such that $(\ell, \mathsf{m}) \notin L_\Delta$, $\mathcal{C}$ computes $\sigma_\ell \xleftarrow{\$} \mathsf{Auth}(\mathsf{sk_{id}}, \Delta, \ell, \mathsf{m})$ (note that $\mathcal{C}$ has already generated keys for the identity id), returns σ_ℓ to $\mathcal{A}$, and updates the list $L_\Delta \leftarrow L_\Delta \cup (\ell, \mathsf{m})$.

- If $(\Delta, \ell, \mathsf{m})$ is such that $(\ell, \cdot) \in L_\Delta$ (which means that the adversary had already made a query $(\Delta, \ell, \mathsf{m}')$ for some message m'), then $\mathcal{C}$ ignores the query.

Verification Queries $\mathcal{A}$ is also given access to a verification oracle. Namely, the adversary can submit a query $(\mathcal{P}, \Delta, \mathsf{m}, \sigma)$, and $\mathcal{C}$ replies with the output of $\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \mathsf{m}, \sigma)$.

Corruption The adversary has access to a corruption oracle. At the beginning of the game, the challenger initialises an empty list $L_{\mathsf{corr}} = \emptyset$ of corrupted identities; during the game, $\mathcal{A}$ can adaptively query identities $\mathsf{id} \in \mathsf{ID}$. If $\mathsf{id} \notin L_{\mathsf{corr}}$, then $\mathcal{C}$ replies with the triple $(\mathsf{sk}_{\mathsf{id}}, \mathsf{ek}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$ (that is generated using KeyGen if not done before) and updates the list $L_{\mathsf{corr}} \leftarrow L_{\mathsf{corr}} \cup \mathsf{id}$. If $\mathsf{id} \in L_{\mathsf{corr}}$, then $\mathcal{C}$ replies with the triple $(\mathsf{sk}_{\mathsf{id}}, \mathsf{ek}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$ assigned to id before.

Forgery In the end, $\mathcal{A}$ outputs a tuple $(\mathcal{P}^*, \Delta^*, \mathsf{m}^*, \sigma^*)$. The experiment outputs 1 if the tuple returned by $\mathcal{A}$ is a forgery (defined below), and 0 otherwise.

Definition 1.4 (Forgery). *Consider an execution of* $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}(\lambda)$ *where* $(\mathcal{P}^*, \Delta^*, \mathsf{m}^*, \sigma^*)$ *is the tuple returned by the adversary in the end of the experiment, with* $\mathcal{P}^* = (f^*, \ell_1^*, \dots, \ell_t^*)$. *This is a forgery if* $\mathsf{Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{m}^*, \sigma^*) = 1$, *for all* $\mathsf{id} \in \mathcal{P}^*$ *we have that* $\mathsf{id} \notin L_{\mathsf{corr}}$ *(i.e., no identity involved in* $\mathcal{P}^*$ *is corrupted), and either one of the following properties is satisfied:*

Type 1: L_{Δ^*} *has not been initialized during the game (i.e., the dataset* Δ^* *was never queried).*

Type 2: *For all* $i \in [t]$, $\exists(\ell_i^*, \mathsf{m}_i) \in L_{\Delta^*}$, *but* $\mathsf{m}^* \neq f^*(\mathsf{m}_1, \dots, \mathsf{m}_t)$ *(i.e.,* m^* *is not the correct output of* $\mathcal{P}^*$ *when executed over previously authenticated messages).*

Type 3: *There exists a label* ℓ^* *such that* $(\ell^*, \cdot) \notin L_{\Delta^*}$ *(i.e.,* $\mathcal{A}$ *never made a query with label* ℓ^**).*

We say that a HA scheme MKHAut is *secure* if for every PPT adversary $\mathcal{A}$, its advantage $\mathbf{Adv}^{\mathsf{HomUF\text{-}CMA}}_{\mathsf{MKHAut},\mathcal{A}}(\lambda) = \Pr[\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}(\lambda) = 1]$ is negligible.

Remark 1.2 (Comparison with previous security definitions). Our security notion can be seen as the multi-key version of the one proposed by Gennaro and Wichs in [26] (in their model our Type 3 forgeries are called 'Type 1' as they do not consider multiple datasets). We point out that even in the special case of a single key, our security definition is stronger than the ones used in previous work [9, 16, 23, 30] (with the only exception of [26]). The main difference lies in our definition of Type 3 forgeries. The intuitive idea of this kind of forgeries is that an adversary who did not receive an authenticated input labeled by a certain ℓ^* cannot produce a valid authenticator for the output of a computation which has ℓ^* among its inputs. In [9, 30] these forgeries were not considered at all, as the adversary is assumed to query the dataset always in full. Other works [11, 16, 23] consider a weaker Type 3 notion, which deals with the concept of "well defined programs", where the input wire labeled by the missing label ℓ^* is required to "contribute" to the computation (i.e., it must change its outcome). The issue with such a Type 3 definition is that it may not be efficient to test if an input contributes to a function, especially if the admissible functions are general circuits. In contrast our definition above is simpler and efficiently testable since it simply considers a Type 3 forgery one where the labeled program $\mathcal{P}^*$ involves an un-queried input.

Multi-Key Homomorphic Signatures. As previously mentioned, our definitions are general enough to be easily adapted to either case in which verification is secret or public. The only difference is whether the adversary is allowed to see the verification keys in the security experiment. When the verification is public, we call the primitive *multi-key homomorphic signatures.* More formally:

Definition 1.5 (Multi-Key Homomorphic Signatures). *A multi-key homomorphic signature is a multi-key homomorphic authenticator in which verification keys are also given to the adversary in the security experiment.*

Note that making verification keys public also allows to slightly simplify the security experiment by removing the verification oracle (the adversary can run the verification by itself). In the sequel, when referring to multi-key homomorphic signatures we adapt our notation to the typical one of digital signatures, namely we denote $\mathsf{Auth}(\mathsf{sk}, \Delta, \ell, \mathsf{m})$ as $\mathsf{Sign}(\mathsf{sk}, \Delta, \ell, \mathsf{m})$, and call its outputs *signatures.*

Non-Adaptive Corruption Queries. In our work, we consider a relaxation of the security definition in which the adversaries ask for corruptions in a non-adaptive way. More precisely, we say that an adversary $\mathcal{A}$ makes *non-adaptive corruption* queries if for every identity id asked to the corruption oracle, id was not queried earlier in the game to the authentication oracle or the verification oracle. For this class of adversaries, it is easy to see that corruption queries are essentially of no help as the adversary can generate keys on its own. More precisely, the following proposition holds (see the full version [22] for the proof).

Proposition 1.1. MKHAut *is secure against adversaries that do* not *make corruption queries if and only if* MKHAut *is secure against adversaries that make* non-adaptive *corruption queries.*

Weakly-Adaptive Secure multi-key HAs. In our work, we also consider a weaker notion of security for multi-key HAs in which the adversary has to declare all the queried messages at the beginning of the experiment. More precisely, we consider a notion in which the adversary declares only the messages and the respective tags that will be queried, for every dataset and identity, without, however, needing to specify the names of the datasets or of the identities. In a sense, the adversary $\mathcal{A}$ is adaptive on identities and dataset names, but not on tags and messages. The definition is inspired by the one, for the single-key setting, of Catalano *et al.* [16].

To define the notion of weakly-adaptive security for multi-key HAs, we introduce here a new experiment $\mathsf{Weak\text{-}HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}$, which is a variation of experiment $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}$ (Definition 1.3) as described below.

Definition 1.6 (Weakly-Secure Multi-Key Homomorphic Authenticators). *In the security experiment* $\mathsf{Weak\text{-}HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}$*, before the setup phase, the adversary $\mathcal{A}$ sends to the challenger $\mathcal{C}$ a collection of sets of tags $\mathcal{T}_{i,k} \subseteq \mathcal{T}$ for $i \in [Q_{\mathsf{id}}]$ and $k \in [Q_\Delta]$, where Q_{id} and Q_Δ are, respectively, the total numbers of distinct identities and datasets that will be queried during the game. Associated to every set $\mathcal{T}_{i,k}$, $\mathcal{A}$ also sends a set of messages $\{m_\tau\}_{\tau \in \mathcal{T}_{i,k}}$. Basically the adversary declares, prior to key generation, all the messages and tags that it will query later on; however $\mathcal{A}$ is not required to specify identity and dataset names. Next, the adversary receives the public parameters from $\mathcal{C}$ and can start the query-phase. Verification queries are handled as in* $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}$*. For authentication queries, $\mathcal{A}$ can adaptively submit pairs* (id, Δ) *to $\mathcal{C}$. The challenger then replies with a set of authenticators*

$\{\sigma_\tau\}_{\tau \in \mathcal{T}_{i,k}}$, *where indices i, k are such that* id *is the i-th queried identity, and Δ is the k-th queried dataset.*

An analogous security definition of weakly-secure multi-key homomorphic signatures is trivially obtained by removing a verification oracle.

In the full version of this paper, we present two generic transformations that turn weakly secure multi-key homomorphic authenticator schemes into adaptive secure ones. Our first transformation holds in the standard model and works for schemes in which the tag space $\mathcal{T}$ has polynomial size, while the second one avoids this limitation on the size of $\mathcal{T}$ but holds in the random oracle model.

4 Our Multi-Key Fully Homomorphic Signature

In this section, we present our construction of a multi-key homomorphic signature scheme that supports the evaluation of arbitrary circuits of bounded polynomial depth. The scheme is based on the *SIS* problem on standard lattices, a background of which is provided in the next section. Precisely, in Section 4.2 we present a scheme that is weakly-secure and supports a single dataset. Later, in Section 4.3 we discuss how to extend the scheme to handle multiple datasets, whereas the support of adaptive security can be obtained via the applications of our transformations as shown in [22].

4.1 Lattices and Small Integer Solution Problem

We recall here notation and some basic results about lattices that are useful to describe our homomorphic signature construction.

For any positive integer q we denote by $\mathbb{Z}_q$ the ring of integers modulo q. Elements in $\mathbb{Z}_q$ are represented as integers in the range $(-\frac{q}{2}, \frac{q}{2}]$. The absolute value of any $x \in \mathbb{Z}_q$ (denoted with $|x|$) is defined by taking the modulus q representative of x in $(-\frac{q}{2}, \frac{q}{2}]$, i.e., take $y = x \mod q$ and then set $|x| = |y| \in [0, \frac{q}{2}]$. Vectors and matrices are denoted in bold. For any vector $\mathbf{u} := (u_i, \ldots, u_n) \in \mathbb{Z}_q^n$, its infinity norm is $\|\mathbf{u}\|_\infty := \max_{i \in [n]} |u_i|$, and similarly for a matrix $\mathbf{A} := [a_{i,j}] \in \mathbb{Z}_q^{n \times m}$ we write $\|\mathbf{A}\|_\infty := \max_{i \in [n], j \in [m]} |a_{i,j}|$.

The Small Integer Solution Problem (SIS). For integer parameters n, m, q and β, the $\mathsf{SIS}(n, m, q, \beta)$ problem provides to an adversary $\mathcal{A}$ a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and requires $\mathcal{A}$ to find a vector $\mathbf{u} \in \mathbb{Z}_q^n$ such that $\mathbf{u} \neq \mathbf{0}$, $\|\mathbf{u}\|_\infty \leq \beta$, and $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$. More formally,

Definition 1.7 (SIS [37]). *Let $\lambda \in \mathbb{N}$ be the security parameter. For values $n = n(\lambda), m = m(\lambda), q = q(\lambda), \beta = \beta(\lambda)$, defined as functions of λ, the $\mathsf{SIS}(n, m, q, \beta)$ hardness assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$\Pr\left[\mathbf{A} \cdot \mathbf{u} = \mathbf{0} \ \wedge \ \mathbf{u} \neq \mathbf{0} \ \wedge \ \|\mathbf{u}\|_\infty \leq \beta : \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}, \mathbf{u} \leftarrow \mathcal{A}(1^\lambda, \mathbf{A})\right] \leq \mathsf{negl}(\lambda).$$

For standard lattices, the *SIS* problem is known to be as hard as solving certain worst-case instances of lattice problems [2, 34, 35, 37], and is also implied by the hardness of learning with error (we refer any interested reader to the cited papers for the technical details about the parameters).

In our paper, we assume that for any $\beta = 2^{\mathsf{poly}(\lambda)}$ there are some $n = \mathsf{poly}(\lambda)$, $q = 2^{\mathsf{poly}(\lambda)}$, with $q > \beta$, such that for all $m = \mathsf{poly}(\lambda)$ the $\mathsf{SIS}(n, m, q, \beta)$ hardness assumption holds. This parameters choice assures that hardness of worst-case lattice problems holds with sub-exponential approximation factors.

Trapdoors for Lattices. The *SIS* problem is hard to solve for a random matrix $\mathbf{A}$. However, there is a way to sample a random $\mathbf{A}$ together with a *trapdoor* such that *SIS* becomes easy to solve for that $\mathbf{A}$, given the trapdoor. Additionally, it has been shown that there exist "special" (non random) matrices $\mathbf{G}$ for which *SIS* is easy to solve as well. The following lemma summarizes the above known results (similar to a lemma in [10]):

Lemma 1.2 ([3, 4, 28, 36])**.** *There exist efficient algorithms* $\mathsf{TrapGen}$, SamPre, Sam *such that the following holds: given integers* $n \geq 1$, $q \geq 2$, *there exist some* $m^* = m^*(n, q) = O(n \log q)$, $\beta_{\mathsf{sam}} = \beta_{\mathsf{sam}}(n, q) = O(n\sqrt{\log q})$ *such that for all* $m \geq m^*$ *and all* k *(polynomial in* n*) we have:*

1. $\mathsf{Sam}(1^m, 1^k, q) \to \mathbf{U}$ *samples a matrix* $\mathbf{U} \in \mathbb{Z}_q^{m\times k}$ *such that* $\|\mathbf{U}\|_\infty \leq \beta_{\mathsf{sam}}$ *(with probability* 1*).*

2. *For* $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{U} \leftarrow \mathsf{Sam}(1^m, 1^k, q)$, $\mathbf{V} := \mathbf{AU}$, $\mathbf{V}' \xleftarrow{\$} \mathbb{Z}_q^{n\times k}$, $\mathbf{U}' \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}', \mathsf{td})$, *we have the following statistical indistinguishability (negligible in* n*)*

$$\mathbf{A} \overset{\mathsf{stat}}{\approx} \mathbf{A}' \quad \textit{and} \quad (\mathbf{A}, \mathsf{td}, \mathbf{U}, \mathbf{V}) \overset{\mathsf{stat}}{\approx} (\mathbf{A}, \mathsf{td}, \mathbf{U}', \mathbf{V}')$$

and $\mathbf{U}' \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}', \mathsf{td})$ *always satisfies* $\mathbf{AU}' = \mathbf{V}'$ *and* $\|\mathbf{U}'\|_\infty \leq \beta_{\mathsf{sam}}$.

3. *Given* n, m, q *as above, there is an efficiently and deterministically computable matrix* $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ *and a deterministic polynomial-time algorithm* $\mathbf{G}^{-1}$ *that on input* $\mathbf{V} \in \mathbb{Z}_q^{n\times k}$ *(for any integer* k*) outputs* $\mathbf{R} = \mathbf{G}^{-1}(\mathbf{V})$ *such that* $\mathbf{R} \in \{0, 1\}^{m\times k}$ *and* $\mathbf{GR} = \mathbf{V}$.

4.2 Our Multi-Key Homomorphic Signature for Single Data-set

In this section, we present our multi-key homomorphic signature that supports the evaluation of boolean circuits of bounded polynomial depth. Our construction is inspired by the (single-key) one of Gorbunov *et al.* [30], with the fundamental difference that in our case we enable computations over data signed using different secret keys. Moreover, our scheme is secure against Type 3 forgeries. We achieve this via a new technique which consists into adding to every signature a component that specifically protects against this type of forgeries. We prove the scheme to be weakly-secure under the *SIS* hardness assumption.

Parameters. Before describing the scheme, we discuss how to set the various parameters involved. Let λ be the security parameter, and let $d = d(\lambda) = \mathsf{poly}(\lambda)$ be the bound on the depth of the circuits supported by our scheme. We define the set of parameters used in our scheme $\mathsf{Par} = \{n, m, q, \beta_{\mathsf{SIS}}, \beta_{\mathsf{max}}, \beta_{\mathsf{init}}\}$ in terms of λ, d and of the parameters required by the trapdoor algorithm in Lemma 1.2: $m^*, \beta_{\mathsf{sam}}$, where $m^* = m^*(n, q) := O(n \log q)$ and $\beta_{\mathsf{sam}} := O(n\sqrt{\log q})$. More precisely, we set: $\beta_{\mathsf{max}} := 2^{\omega(\log\lambda)d}$; $\beta_{\mathsf{SIS}} := 2^{\omega(\log\lambda)}\beta_{\mathsf{max}}$; $n = \mathsf{poly}(\lambda)$; $q = \mathcal{O}(2^{\mathsf{poly}(\lambda)}) > \beta_{\mathsf{SIS}}$ is a prime (as small as possible) so that the $\mathsf{SIS}(n, m', q, \beta_{\mathsf{SIS}})$ assumption holds for all $m' = \mathsf{poly}(\lambda)$; $m = \max\{m^*, n \log q + \omega(\log(\lambda))\} = \mathsf{poly}(\lambda)$ and, finally, $\beta_{\mathsf{init}} := \beta_{\mathsf{sam}} = \mathsf{poly}(\lambda)$.

Construction. The PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ which define our construction of Multi-key Homomorphic Signatures work as follows:

$\mathsf{Setup}(1^\lambda)$**.** The setup algorithm takes as input the security parameter λ and generates the public parameters pp which include: the bound on the circuit depth d (which defines the class $\mathbb{F}$ of functions supported by the scheme, i.e., boolean circuits of depth d), the set $\mathsf{Par} = \{n, m, q, \beta_{\mathsf{SIS}}, \beta_{\mathsf{max}}, \beta_{\mathsf{init}}\}$, the set $\mathcal{U} = \{\mathbf{U} \in \mathbb{Z}_q^{m\times m} : \|\mathbf{U}\|_\infty \leq \beta_{\mathsf{max}}\}$, the set $\mathcal{V} = \{\mathbf{V} \in \mathbb{Z}_q^{n\times m}\}$, descriptions of the message space $\mathcal{M} = \{0, 1\}$, the tag space $\mathcal{T} = [\mathsf{T}]$, and the identity space $\mathsf{ID} = [\mathsf{C}]$, for integers $\mathsf{T}, \mathsf{C} \in \mathbb{N}$. In this construction, the tag space is of polynomial size, i.e., $\mathsf{T} = \mathsf{poly}(\lambda)$ while the identity space is essentially unbounded, i.e., we set $\mathsf{C} = 2^\lambda$. Also recall that $\mathcal{T}$ and ID immediately define the label space $\mathbb{L} = \mathsf{ID} \times \mathcal{T}$. The final output is $\mathsf{pp} = \{d, \mathsf{Par}, \mathcal{U}, \mathcal{V}, \mathcal{M}, \mathcal{T}, \mathsf{ID}\}$. We assume that these public parameters pp are input of all subsequent algorithms, and often omit them from the input explicitly.

$\mathsf{KeyGen}(\mathsf{pp})$**.** The key generation algorithm takes as input the public parameters pp and generates a key-triple $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$ defined as follows. First, it samples T random matrices $\mathbf{V}_1, ..., \mathbf{V}_\mathsf{T} \xleftarrow{\$} \mathcal{V}$. Second, it runs $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ to generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ along with its trapdoor td. Then, it outputs $\mathsf{sk} = (\mathsf{td}, \mathbf{A}, \mathbf{V}_1, \ldots, \mathbf{V}_\mathsf{T})$, $\mathsf{ek} = \mathbf{A}$, $\mathsf{vk} = (\mathbf{A}, \mathbf{V}_1, \ldots, \mathbf{V}_\mathsf{T})$. Note that it is possible to associate the key-triple to an identity $\mathsf{id} \in \mathsf{ID}$, when we need to stress this explicitly we write $(\mathsf{sk}_\mathsf{id}, \mathsf{ek}_\mathsf{id}, \mathsf{vk}_\mathsf{id})$. We also observe that the key generation process can be seen as the combination of two independent sub-algorithms [8] KeyGen_1 and KeyGen_2, where $\{\mathbf{V}_1, \ldots \mathbf{V}_\mathsf{T}\} \leftarrow \mathsf{KeyGen}_1(\mathsf{pp})$ and $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{KeyGen}_2(\mathsf{pp})$.

$\mathsf{Sign}(\mathsf{sk}, \ell, \mathsf{m})$**.** The signing algorithm takes as input a secret key sk, a label $\ell = (\mathsf{id}, \tau)$ for the message m and it outputs a signature $\sigma := (\mathsf{m}, \mathsf{z}, \mathsf{I}, \mathbf{U}_\mathsf{id}, \mathbf{Z}_\mathsf{id})$ where $\mathsf{I} = \{\mathsf{id}\}$, $\mathbf{U}_\mathsf{id}$ is generated as $\mathbf{U}_\mathsf{id} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}_\ell - \mathsf{m}\mathbf{G}, \mathsf{td})$ (using the algorithm SamPre from Lemma 1.2), $\mathsf{z} = \mathsf{m}$ and $\mathbf{Z}_\mathsf{id} = \mathbf{U}_\mathsf{id}$. The two latter terms are responsible for protection against Type 3 forgeries. Although they are redundant for fresh signatures, their value will become different from $(\mathsf{m}, \mathbf{U}_\mathsf{id})$ during homomorphic operations, as we clarify later on. More generally, in our construction signatures are of the form $\sigma := (\mathsf{m}, \mathsf{z}, \mathsf{I}, \{\mathbf{U}_\mathsf{id}\}_{\mathsf{id}\in\mathsf{I}}, \{\mathbf{Z}_\mathsf{id}\}_{\mathsf{id}\in\mathsf{I}})$ with $\mathsf{I} \subseteq \mathsf{ID}$ and $\mathbf{U}_\mathsf{id}, \mathbf{Z}_\mathsf{id} \in \mathcal{U}, \forall\, \mathsf{id} \in \mathsf{I}$.

$\mathsf{Eval}\big(f, \{(\sigma_i, \mathsf{EKS}_i)\}_{i\in[t]}\big)$**.** The evaluation algorithm takes as input a t-input function $f : \mathcal{M}^t \longrightarrow \mathcal{M}$, and a set of pairs $\{(\sigma_i, \mathsf{EKS}_i)\}_{i\in[t]}$ where each σ_i is a signature and each EKS_i is a set of evaluation keys. In our description below we treat f as an arithmetic circuit over $\mathbb{Z}_q$ consisting of addition and multiplication gates.[9] Therefore, we only describe how to evaluate homomorphically a fan-in-2 addition (resp. multiplication) gate as well as a unary multiplication-by-constant gate.

Let g be a fan-in-2 gate with left input $\sigma_\mathsf{L} := (\mathsf{m}_\mathsf{L}, \mathsf{z}_\mathsf{L}, \mathsf{I}_\mathsf{L}, \mathbf{U}_\mathsf{L}, \mathbf{Z}_\mathsf{L})$ and right input $\sigma_\mathsf{R} := (\mathsf{m}_\mathsf{R}, \mathsf{z}_\mathsf{R}, \mathsf{I}_\mathsf{R}, \mathbf{U}_\mathsf{R}, \mathbf{Z}_\mathsf{R})$. To generate the signature $\sigma := (\mathsf{m}, \mathsf{z}, \mathsf{I}, \mathbf{U}, \mathbf{Z})$ on the gate's output one proceeds as follows. First set $\mathsf{I} = \mathsf{I}_\mathsf{L} \cup \mathsf{I}_\mathsf{R}$. Second, "expand" $\mathbf{U}_\mathsf{L} := \{\mathbf{U}_\mathsf{L}^\mathsf{id}\}_{\mathsf{id}\in\mathsf{I}_\mathsf{L}}$ as:

[8] This splitting will be used to extend our multi-key homomorphic signature scheme from supporting a single dataset to support multiple datasets. This extension holds in the standard model and is described in Section 4.3.

[9] We point out that considering f as an arithmetic circuit over $\mathbb{Z}_q$ is enough to describe any boolean circuits consisting of NAND gates as $\mathsf{NAND}(\mathsf{m}_1, \mathsf{m}_2) = 1 - \mathsf{m}_1 \cdot \mathsf{m}_2$ holds for $\mathsf{m}_1, \mathsf{m}_2 \in \{0, 1\}$.

$$\hat{\mathbf{U}}^{\mathsf{id}}_{\mathsf{L}} = \begin{cases} \mathbf{0} & \text{if } \mathsf{id} \notin \mathsf{I}_{\mathsf{L}} \\ \mathbf{U}^{\mathsf{id}}_{\mathsf{L}} & \text{if } \mathsf{id} \in \mathsf{I}_{\mathsf{L}} \end{cases}, \quad \forall\, \mathsf{id} \in \mathsf{I}\,.$$

where $\mathbf{0}$ denotes an $(m \times m)$-matrix with all zero entries. Basically, we extend the set to be indexed over all identities in $\mathsf{I} = \mathsf{I}_{\mathsf{L}} \cup \mathsf{I}_{\mathsf{R}}$ by inserting zero matrices for identities in $\mathsf{I} \setminus \mathsf{I}_{\mathsf{L}}$. The analogous expansion process is applied to $\mathbf{U}_{\mathsf{R}} := \{\mathbf{U}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}_{\mathsf{L}}}$, $\mathbf{Z}_{\mathsf{L}} := \{\mathbf{Z}_{\mathsf{L}}{}^{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}_{\mathsf{R}}}$ and $\mathbf{Z}_{\mathsf{R}} := \{\mathbf{Z}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}_{\mathsf{R}}}$, denoting the expanded sets $\{\hat{\mathbf{U}}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}}$, $\{\hat{\mathbf{Z}}^{\mathsf{id}}_{\mathsf{L}}\}_{\mathsf{id}\in\mathsf{I}}$ and $\{\hat{\mathbf{Z}}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}}$ respectively.

Next, depending on whether g is an addition or multiplication gate one proceeds as follows.

Addition gate. If g is additive, compute $\mathsf{m} = \mathsf{m}_{\mathsf{L}} + \mathsf{m}_{\mathsf{R}}$, $\mathsf{z} = \mathsf{z}_{\mathsf{L}} + \mathsf{z}_{\mathsf{R}}$, $\mathbf{U} = \{\mathbf{U}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}} := \{\hat{\mathbf{U}}^{\mathsf{id}}_{\mathsf{L}} + \hat{\mathbf{U}}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}}$ and $\mathbf{Z} = \{\mathbf{Z}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}} := \{\hat{\mathbf{Z}}^{\mathsf{id}}_{\mathsf{L}} + \hat{\mathbf{Z}}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}}$.

If we refer to β_{L} and β_{R} as $\|\mathbf{U}_{\mathsf{L}}\|_\infty := \mathsf{max}\{\|\mathbf{U}^{\mathsf{id}}_{\mathsf{L}}\|_\infty : \mathsf{id} \in \mathsf{I}_{\mathsf{L}}\}$ and $\|\mathbf{U}_{\mathsf{R}}\|_\infty := \mathsf{max}\{\|\mathbf{U}^{\mathsf{id}}_{\mathsf{R}}\|_\infty : \mathsf{id} \in \mathsf{I}_{\mathsf{R}}\}$ respectively, then for any fan-in-2 addition gate it holds $\beta := \|\mathbf{U}\|_\infty = \beta_{\mathsf{L}} + \beta_{\mathsf{R}}$. The same noise growth applies to $\mathbf{Z}$.

Multiplication gate. If g is multiplicative, compute $\mathsf{m} = \mathsf{m}_{\mathsf{L}} \cdot \mathsf{m}_{\mathsf{R}}$, $\mathsf{z} = \mathsf{z}_{\mathsf{L}} + \mathsf{z}_{\mathsf{R}}$, define $\mathbf{V}_{\mathsf{L}} = \sum_{\mathsf{id}\in\,\mathsf{I}_{\mathsf{L}}} \mathbf{A}_{\mathsf{id}}\mathbf{U}_{\mathsf{id}} + \mathsf{m}_{\mathsf{L}}\mathbf{G}$, set

$$\mathbf{U} = \{\mathbf{U}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}} := \{\mathsf{m}_{\mathsf{R}}\hat{\mathbf{U}}^{\mathsf{id}}_{\mathsf{L}} + \hat{\mathbf{U}}^{\mathsf{id}}_{\mathsf{R}} \cdot \mathbf{G}^{-1}(\mathbf{V}_{\mathsf{L}})\}_{\mathsf{id}\in\mathsf{I}}$$

and $\mathbf{Z} = \{\mathbf{Z}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}} := \{\hat{\mathbf{Z}}^{\mathsf{id}}_{\mathsf{L}} + \hat{\mathbf{Z}}^{\mathsf{id}}_{\mathsf{R}}\}_{\mathsf{id}\in\mathsf{I}}$.

Letting β_{L} and β_{R} as defined before, then for any fan-in-2 multiplication gate it holds $\beta := \|\mathbf{U}\|_\infty = |\mathsf{m}_{\mathsf{R}}|\beta_{\mathsf{L}} + m\beta_{\mathsf{R}}$, while the noise growth of $\mathbf{Z}$ is the same as in the addition gate.

Multiplication by constant gate. Let g be a unary gate representing a multiplication by a constant $a \in \mathbb{Z}_q$, and let its single input signature be $\sigma_{\mathsf{R}} := (\mathsf{m}_{\mathsf{R}}, \mathsf{z}_{\mathsf{R}}, \mathsf{I}_{\mathsf{R}}, \mathbf{U}_{\mathsf{R}}, \mathbf{Z}_{\mathsf{R}})$. The output $\sigma := (\mathsf{m}, \mathsf{z}, \mathsf{I}, \mathbf{U}, \mathbf{Z})$ is obtained by setting $\mathsf{m} = a \cdot \mathsf{m}_{\mathsf{R}} \in \mathbb{Z}_q$, $\mathsf{z} = \mathsf{z}_{\mathsf{R}}$, $\mathsf{I} = \mathsf{I}_{\mathsf{R}}$, $\mathbf{Z} = \mathbf{Z}_{\mathsf{R}}$, and $\mathbf{U} = \{\mathbf{U}^{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}}$ where, for all $\mathsf{id} \in \mathsf{I}$, $\mathbf{U}^{\mathsf{id}} = a \cdot \mathbf{U}^{\mathsf{id}}_{\mathsf{R}}$ or, alternatively, $\mathbf{U}^{\mathsf{id}} = \mathbf{U}^{\mathsf{id}}_{\mathsf{R}} \cdot \mathbf{G}^{-1}(a \cdot \mathbf{G})$. In the first case, the noise parameter becomes $\beta := \|\mathbf{U}\|_\infty = |a|\beta_{\mathsf{L}}$ (thus a needs to be *small*), whereas in the second case it holds $\beta := \|\mathbf{U}\|_\infty \leq m\beta_{\mathsf{L}}$, which is independent of a's size.

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \mathsf{m}, \sigma)$**.** The verification algorithm takes as input a labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_n)$, the set of the verification keys $\{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$ of users involved in the program $\mathcal{P}$, a message m and a signature $\sigma = (\mathsf{m}, \mathsf{z}, \mathsf{I}, \mathbf{U}, \mathbf{Z})$. It then performs three main checks and outputs 0 if at least one check fails, otherwise it returns 1.

Firstly, it checks if the list of identities declared in σ corresponds to the ones in the labels of $\mathcal{P}$:

$$\mathsf{I} = \{\mathsf{id} : \mathsf{id} \in \mathcal{P}\} \tag{1}$$

Secondly, from the circuit f (again seen as an arithmetic circuit) and the values $\{\mathbf{V}_{\ell_1}, \ldots, \mathbf{V}_{\ell_t}\}$ contained in the verification keys, it computes two values $\mathbf{V}^*$ and $\mathbf{V}^+$ proceeding gate by gate as follows. Given as left and right input matrices $\mathbf{V}^*_{\mathsf{L}}, \mathbf{V}^*_{\mathsf{R}}$ (resp. $\mathbf{V}^+_{\mathsf{L}}, \mathbf{V}^+_{\mathsf{R}}$), at every *addition gate* one computes

$\mathbf{V}^* = \mathbf{V}^*_{\mathsf{L}} + \mathbf{V}^*_{\mathsf{R}}$ (resp. $\mathbf{V}^+ = \mathbf{V}^+_{\mathsf{L}} + \mathbf{V}^+_{\mathsf{R}}$); at every *multiplication gate* one computes $\mathbf{V}^* = \mathbf{V}^*_{\mathsf{R}}\mathbf{G}^{-1}\mathbf{V}^*_{\mathsf{L}}$ (resp. $\mathbf{V}^+ = \mathbf{V}^+_{\mathsf{L}} + \mathbf{V}^+_{\mathsf{R}}$). Every gate representing a multiplication by a constant $a \in \mathbb{Z}_q$, on input $\mathbf{V}^*_{\mathsf{R}}$ (resp. $\mathbf{V}^+_{\mathsf{R}}$) outputs $\mathbf{V}^* = a\cdot\mathbf{V}^*_{\mathsf{R}}$ (resp. $\mathbf{V}^+ = \mathbf{V}^+_{\mathsf{R}}$). Note that the computation of $\mathbf{V}^+$ is essentially the computation of a linear function $\mathbf{V}^+ = \sum_{i=1}^{t} \gamma_i \cdot \mathbf{V}_{\ell_i}$, for some coefficients γ_i that depend on the structure of the circuit f.

Thirdly, the verification algorithm parses $\mathbf{U} = \{\mathbf{U}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}}$ and $\mathbf{Z} = \{\mathbf{Z}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}}$ and checks:

$$\|\mathbf{U}\|_\infty \leq \beta_{\mathsf{max}} \quad \text{and} \quad \|\mathbf{Z}\|_\infty \leq \beta_{\mathsf{max}} \tag{2}$$

$$\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\mathbf{U}_{\mathsf{id}} + \mathsf{m}\cdot\mathbf{G} = \mathbf{V}^* \tag{3}$$

$$\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\mathbf{Z}_{\mathsf{id}} + \mathsf{z}\cdot\mathbf{G} = \mathbf{V}^+ \tag{4}$$

Finally, it is worth noting that the computation of the matrices $\mathbf{V}^*$ and $\mathbf{V}^+$ can be precomputed (or performed offline), prior to seeing the actual signature σ. In the multiple dataset extension of Section 4.3 this precomputation becomes particularly beneficial as the same $\mathbf{V}^*, \mathbf{V}^+$ can be re-used every time one wants to verify for the same labeled program $\mathcal{P}$ (i.e., one can verify faster, in an amortized sense, than that of running f).

In the following paragraphs, we analyse succinctness and security of the proposed construction; for what regards correctness we just give an intuition and refer the interested reader to the full version of this paper available in [22].

Noise Growth and Succinctness. First we analyse the noise growth of the components $\mathbf{U}, \mathbf{Z}$ in the signatures of our MKHSig construction. In particular we need to show that when starting from "fresh" signatures, in which the noise is bounded by β_{init}, and we apply an admissible circuit, then one ends up with signatures in which the noise is within the allowable amount β_{max}.

An analysis similar to the one of Gorbunov *et al.* [30] is applicable also to our construction whenever the admissible functions are boolean circuits of depth d composed only of NAND gates.

Let us first consider the case of the $\mathbf{U}$ component of the signatures. At every NAND gate, if $\|\mathbf{U}_{\mathsf{L}}\|_\infty, \|\mathbf{U}_{\mathsf{R}}\|_\infty \leq \beta$, the noise of the resulting $\mathbf{U}$ is at most $(m+1)\beta$. Therefore, if the circuit has depth d, the noise of the matrix $\mathbf{U}$ at the end of the evaluation is bounded by $\|\mathbf{U}\|_\infty \leq \beta_{\mathsf{init}} \cdot (m+1)^d \leq 2^{O(\log\lambda)d} \leq \beta_{\mathsf{max}}$. For what regards the computation performed over the matrices $\mathbf{Z}$, we observe that we perform only additions (or identity functions) over them. This means that at every gate of any f, the noise in the $\mathbf{Z}$ component at most doubles. Given that we consider depth-d circuits we have that $\|\mathbf{Z}\|_\infty \leq \beta_{\mathsf{init}} \cdot 2^d \leq 2^{O(\log\lambda)+d} \leq \beta_{\mathsf{max}}$. Finally, by inspection one can see that the size of every signature σ on a computation's output involving n users is at most $(1 + 2^d + n\lambda + 2n\beta_{\mathsf{max}})$ that is $O(n \cdot p(\lambda))$ for some fixed polynomial $p(\cdot)$.

Authentication Correctness. This is rather simple and follows from the noise growth property mentioned above and by observing that equation $\mathbf{A}_{\mathsf{id}}\mathbf{U}_{\mathsf{id}} + \mathsf{m}\mathbf{G} = \mathbf{V}_\ell = \mathbf{V}^*$ holds by construction.

Evaluation Correctness. Evaluation correctness follows from two main facts: the noise growth mentioned earlier, and the preservation of the invariant ($\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\mathbf{U}_{\mathsf{id}}$

$+ \mathsf{m}\mathbf{G}) = \mathbf{V}^*$. At every gate, it is easy to see that the expansion of $\mathbf{U}$ still preserves the invariant for both left and right inputs. For additive gates, assuming validity of the inputs, i.e., $\mathbf{V}_\mathsf{L} = \sum_{\mathsf{id}\in \mathsf{I}_\mathsf{L}} \mathbf{A}_\mathsf{id}\mathbf{U}^\mathsf{id}_\mathsf{L} + \mathsf{m}_\mathsf{L}\mathbf{G}$ (and similarly $\mathbf{V}_\mathsf{R}$) and by construction of $\mathbf{U}_\mathsf{id} = \mathbf{U}^\mathsf{id}_\mathsf{L} + \mathbf{U}^\mathsf{id}_\mathsf{R}$, one obtains $\sum_{\mathsf{id}\in \mathsf{I}_\mathsf{L}\cup \mathsf{I}_\mathsf{R}} \mathbf{A}_\mathsf{id}\mathbf{U}_\mathsf{id} + (\mathsf{m}_\mathsf{L} + \mathsf{m}_\mathsf{R})\mathbf{G} = \mathbf{V}_\mathsf{L} + \mathbf{V}_\mathsf{R} = \mathbf{V}^*$. For what regards multiplicative gates, by construction of every $\mathbf{U}_\mathsf{id}$ we obtain $\sum_{\mathsf{id}\in \mathsf{I}} \mathbf{A}_\mathsf{id}\mathbf{U}_\mathsf{id} + \mathsf{m}\mathbf{G} := \sum_{\mathsf{id}\in \mathsf{I}} \mathbf{A}_\mathsf{id}(\mathsf{m}_\mathsf{R}\hat{\mathbf{U}}^\mathsf{id}_\mathsf{L} + \hat{\mathbf{U}}^\mathsf{id}_\mathsf{R}\mathbf{G}^{-1}(\mathbf{V}_\mathsf{L})) + (\mathsf{m}_\mathsf{L}\mathsf{m}_\mathsf{R})\mathbf{G}$. Grouping by m_R and applying the definition of $\mathbf{V}_\mathsf{L}$, the equation can be rewritten as $\mathsf{m}_\mathsf{R}\mathbf{V}_\mathsf{L} + \left(\sum_{\mathsf{id}\in \mathsf{I}} \mathbf{A}_\mathsf{id}\ \hat{\mathbf{U}}^\mathsf{id}_\mathsf{R}\right) \mathbf{G}^{-1}(\mathbf{V}_\mathsf{L})$. If now we write $\mathsf{m}_\mathsf{R}\mathbf{V}_\mathsf{L}$ as $\mathsf{m}_\mathsf{R}\mathbf{G}\mathbf{G}^{-1}(\mathbf{V}_\mathsf{L})$ and we group by $\mathbf{G}^{-1}(\mathbf{V}_\mathsf{L})$, we get $\left[\left(\sum_{\mathsf{id}\in \mathsf{I}_\mathsf{R}} \mathbf{A}_\mathsf{id}\mathbf{U}^\mathsf{id}_\mathsf{R}\right) + \mathsf{m}_\mathsf{R}\mathbf{G}\right] \mathbf{G}^{-1}(\mathbf{V}_\mathsf{L}) = \mathbf{V}^*$, where the last equation follows from the definitions of $\mathbf{V}_\mathsf{R}$ and $\mathbf{V}^*$. Correctness of computations over the matrices $\mathbf{Z}$ is quite analogous.

Security. The following theorem states the security of the scheme MKHSig.

Theorem 1.1. *If the* $\mathsf{SIS}(n, m \cdot Q_\mathsf{id}, q, \beta_\mathsf{SIS})$ *hardness assumption holds,* $\mathsf{MKHSig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ *is a multi-key homomorphic signature weakly-adaptive secure against adversaries that make signing queries involving at most* Q_id *different identities and that make non-adaptive corruption queries.*

Proof. Note that we can deal with corruptions via our generic result of Proposition 1.1. Therefore it is sufficient to prove the security against adversaries that make no corruptions. Moreover, since this scheme works for a single dataset note that Type 1 forgeries cannot occur.

For the proof let us recall how the weakly-adaptive security experiment (Definition 1.6) works for our multi-key homomorphic signature scheme MKHSig. This is a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ that has four main phases:

(1) $\mathcal{A}$ declares an integer Q representing the number of different identities that it will ask in the signing queries. Moreover, for every $i \in [Q]$ $\mathcal{A}$ sends to $\mathcal{C}$ a set $\mathcal{T}_i \subseteq \mathcal{T} := \{\tau_1, \ldots, \tau_\mathsf{T}\}$ and a set of pairs $\{(\mathsf{m}_\tau, \tau)\}_{\tau\in\mathcal{T}_i}$.

(2) $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ to obtain the public parameters and sends them to $\mathcal{A}$.

(3) $\mathcal{A}$ adaptively queries identities $\mathsf{id}_1, \ldots, \mathsf{id}_Q$. When $\mathcal{C}$ receives the query id_i it generates a key-triple $(\mathsf{sk}_{\mathsf{id}_i}, \mathsf{ek}_{\mathsf{id}_i}, \mathsf{vk}_{\mathsf{id}_i})$ by running $\mathsf{KeyGen}(\mathsf{pp})$, and for all labels $\ell = (\mathsf{id}_i, \tau)$ such that $\tau \in \mathcal{T}_i$ it runs $\sigma^i_\tau \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}_i}, \ell, \mathsf{m}_\tau)$. Then $\mathcal{C}$ sends to $\mathcal{A}$: the public keys $\mathsf{vk}_{\mathsf{id}_i} := (\mathbf{A}_{\mathsf{id}_i}, \{\mathbf{V}_\ell\}_{\tau\in\mathcal{T}})$ and $\mathsf{ek}_{\mathsf{id}_i} := (\mathbf{A}_{\mathsf{id}_i})$, and the signatures $\{\sigma^i_\tau\}_{\tau\in\mathcal{T}_i}$.

(4) The adversary produces a forgery consisting of a labeled program $\mathcal{P}^* = (f^*, \ell^*_1, \ldots, \ell^*_t)$ where $f^* \in \mathbb{F}, f^* : \mathcal{M}^t \to \mathcal{M}$, a message m^* and a signature σ^*.

$\mathcal{A}$ wins the non-adaptive security game if $\mathsf{Verify}(\mathcal{P}^*, \{\mathsf{vk}_\mathsf{id}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{m}^*, \sigma^*) = 1$ and one of the following conditions holds:

Type 2 Forgery: there exist messages $\mathsf{m}_{\ell^*_1}, \ldots, \mathsf{m}_{\ell^*_t}$ s.t. $\mathsf{m}^* \neq f^*(\mathsf{m}_{\ell^*_1}, \ldots, \mathsf{m}_{\ell^*_t})$ (i.e., m^* is not the correct output of $\mathcal{P}^*$ when executed over previously signed messages).

Type 3 Forgery: there exists at least one label $\ell^* = (\mathsf{id}^*, \tau^*)$ that was not queried by $\mathcal{A}$.

Consider a variation of the above game obtained modifying phase (3) as follows: (3)$'$ $\mathcal{C}$ picks an instance $\mathbf{A} \in \mathbb{Z}_q^{n\times m'}$ of the $\mathsf{SIS}(n, m', q, \beta_{\mathsf{SIS}})$ problem for $m' = m \cdot Q = \mathsf{poly}(\lambda)$, and parse $\mathbf{A} := (\mathbf{A}_{\mathsf{id}_1}|\dots|\mathbf{A}_{\mathsf{id}_Q}) \in \mathbb{Z}_q^{n\times m'}$ as the concatenation of Q different blocks of $n \times m$ matrices.
Next, when $\mathcal{C}$ receives the i-th query id_i from $\mathcal{A}$, it does the following:

- it samples a matrix $\mathbf{U}_{\mathsf{id}_i,\tau} \xleftarrow{\$} \mathcal{U}$ such that $\|\mathbf{U}_{\mathsf{id}_i,\tau}\| \leq \beta_{\mathsf{init}}$;
- for all $\ell := (\mathsf{id}_i, \tau)$ with $\tau \in \mathcal{T}_i$, $\mathcal{C}$ computes $\mathbf{V}_\ell = \mathbf{A}_{\mathsf{id}_i}\mathbf{U}_{\mathsf{id}_i,\tau} + \mathsf{m}_\tau \cdot \mathbf{G}$;
- for all $\ell := (\mathsf{id}_i, \tau)$ with $\tau \notin \mathcal{T}_i$, $\mathcal{C}$ computes $\mathbf{V}_\ell = \mathbf{A}_{\mathsf{id}_i}\mathbf{U}_{\mathsf{id}_i,\tau} + b_{i,\tau} \cdot \mathbf{G}$, where $b_{i,\tau} \xleftarrow{\$} \{0, 1\}$.
- $\mathcal{C}$ sends to $\mathcal{A}$ the public keys $\mathsf{vk}_{\mathsf{id}_i} := (\mathbf{A}_{\mathsf{id}_i}, \{\mathbf{V}_\ell\}_{\tau\in\mathcal{T}})$ and $\mathsf{ek}_{\mathsf{id}_i} := (\mathbf{A}_{\mathsf{id}_i})$, along with signatures $\{\sigma^i_\tau\}_{\tau\in\mathcal{T}_i}$ where $\sigma^i_\tau := (\mathsf{m}_\tau, \mathsf{m}_\tau, \mathsf{I} := \{\mathsf{id}_i\}, \mathbf{U}_{\mathsf{id}_i,\tau}, \mathbf{U}_{\mathsf{id}_i,\tau})$.

Clearly, if $\mathbf{A}$ is a uniformly random matrix so is each block $\{\mathbf{A}_{\mathsf{id}_i}\}_{i\in[Q]}$.
Due to point (2) of Lemma 1.2, since $(\mathbf{A}_{\mathsf{id}_i}\mathbf{U}_{\mathsf{id}_i,\tau})$ is statistically indistinguishable from a random matrix, all the matrices $\mathbf{V}_\ell$ generated in (3)$'$ are statistically close to the ones generated in (3). Thus, the two games are statistically indistinguishable. At this point we show that for every PPT adversary $\mathcal{A}$ which produces a forgery in the modified game we can construct a PPT algorithm $\mathcal{B}$ that solves the $\mathsf{SIS}(n, m \cdot Q, q, \beta_{\mathsf{SIS}})$ problem. $\mathcal{B}$ receives an SIS instance $\mathbf{A} := (\mathbf{A}_{\mathsf{id}_1}|\dots|\mathbf{A}_{\mathsf{id}_Q}) \in \mathbb{Z}_q^{n\times mQ}$ and simulates the modified game to $\mathcal{A}$ by acting exactly as the challenger $\mathcal{C}$ described above. Then, once $\mathcal{A}$ outputs its forgery, according to the forgery's type, $\mathcal{B}$ proceeds as described below.

Type 2 Forgeries. Let $(\mathcal{P}^* := (f^*, \ell^*_1, \dots, \ell^*_t), \mathsf{m}^*, \sigma^* := (\mathsf{m}^*, \mathsf{z}^*, \mathsf{I}^*, \mathbf{U}^*, \mathbf{Z}^*))$ be a Type 2 forgery produced by $\mathcal{A}$ in the modified game. Moreover let $\sigma = (\mathsf{m}, \mathsf{z}, \mathsf{I}, \mathbf{U}, \mathbf{Z})$ be the signature obtained by honestly applying Eval to the signatures corresponding to labels $\ell^*_1, \dots, \ell^*_t$ that were given to $\mathcal{A}$. Parse $\mathbf{U} := \{\mathbf{U}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}}$ and notice that by the correctness of the scheme we have that $\mathsf{m} = f^*(\mathsf{m}_{\ell^*_1}, \dots, \mathsf{m}_{\ell^*_t})$, $\mathsf{I} = \{\mathsf{id} : \mathsf{id} \in \mathcal{P}^*\}$, and $\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\mathbf{U}_{\mathsf{id}} + \mathsf{m} \cdot \mathbf{G} = \mathbf{V}^*$. Moreover, by definition of Type 2 forgery recall that $\mathsf{m}^* \neq f^*(\mathsf{m}_{\ell^*_1}, \dots, \mathsf{m}_{\ell^*_t})$ and that the tuple satisfies verification. In particular, satisfaction of check (1) implies that $\mathsf{I} = \mathsf{I}^*$, while check (3) means $\sum_{\mathsf{id}\in\mathsf{I}^*} \mathbf{A}_{\mathsf{id}}\mathbf{U}^*_{\mathsf{id}} + \mathsf{m}^* \cdot \mathbf{G} = \mathbf{V}^*$. Combining the two equations above we obtain $\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\tilde{\mathbf{U}}_{\mathsf{id}} = \tilde{\mathsf{m}}\cdot\mathbf{G}$, where $\tilde{\mathsf{m}} = \mathsf{m}-\mathsf{m}^* \neq \mathbf{0}$ and, for all $\mathsf{id} \in \mathsf{I}$, $\tilde{\mathbf{U}}_{\mathsf{id}} = \mathbf{U}^*_{\mathsf{id}} - \mathbf{U}_{\mathsf{id}} \in \mathcal{U}$ such that $\|\tilde{\mathbf{U}}_{\mathsf{id}}\|_\infty \leq \beta_{\mathsf{max}}$. Notice that there must exist at least one $\bar{\mathsf{id}} \in \mathsf{I}$ for which $\tilde{\mathbf{U}}_{\bar{\mathsf{id}}} \neq \mathbf{0}$.

Moreover, for all $\mathsf{id} \in \{\mathsf{id}_1, \dots, \mathsf{id}_Q\} \setminus \mathsf{I}$, define $\tilde{\mathbf{U}}_{\mathsf{id}} = \mathbf{0}$ and set $\tilde{\mathbf{U}} = \begin{pmatrix} \tilde{\mathbf{U}}_{\mathsf{id}_1} \\ \vdots \\ \tilde{\mathbf{U}}_{\mathsf{id}_Q} \end{pmatrix} \in \mathbb{Z}_q^{mQ\times m}$. Then, we have $\mathbf{A}\tilde{\mathbf{U}} = \tilde{\mathsf{m}} \cdot \mathbf{G}$.

Next $\mathcal{B}$ samples $\mathbf{r} \xleftarrow{\$} \{0, 1\}^{mQ}$, sets $\mathbf{s} = \mathbf{Ar} \in \mathbb{Z}_q^n$, and computes $\mathbf{r}' = \mathbf{G}^{-1}(\tilde{\mathsf{m}}^{-1}\cdot\mathbf{s})$, so that $\mathbf{r}' \in \{0, 1\}^m$ and $\tilde{\mathsf{m}} \cdot \mathbf{Gr}' = \mathbf{s}$. Finally, $\mathcal{B}$ outputs $\mathbf{u} = \tilde{\mathbf{U}}\mathbf{r}' - \mathbf{r} \in \mathbb{Z}_q^{mQ}$. We conclude the proof by claiming that the vector $\mathbf{u}$ returned by $\mathcal{B}$ is a solution of the SIS problem for the matrix $\mathbf{A}$. To see this observe that

$$\mathbf{A}(\tilde{\mathbf{U}}\mathbf{r}' - \mathbf{r}) = (\mathbf{A}\tilde{\mathbf{U}})\mathbf{r}' - \mathbf{Ar} = \tilde{\mathsf{m}} \cdot \mathbf{G} \cdot \mathbf{G}^{-1}(\tilde{\mathsf{m}}^{-1} \cdot \mathsf{s}) - \mathbf{s} = \mathbf{0} \; .$$

and $\|\mathbf{u}\|_\infty \leq (2m + 1)\beta_{\mathsf{max}} \leq \beta_{\mathsf{SIS}}$.

It remains to show that $\mathbf{u} \neq \mathbf{0}$. We show that this is the case (i.e., $\tilde{\mathbf{U}}\mathbf{r}' \neq \mathbf{r}$) with overwhelming probability by using an entropy argument (the same argument used in [30]). In particular, this holds for any (worst case) choice of $\mathbf{A}, \tilde{\mathbf{U}}, \tilde{\mathsf{m}}$, and only based on the random choice of $\mathbf{r} \xleftarrow{\$} \{0,1\}^{mQ_{\mathsf{id}}}$. The intuition is that, even if $\mathbf{r}' = \mathbf{G}^{-1}(\mathsf{s}\hat{\mathsf{m}}^{-1})$ depends on $\mathbf{s} = \mathbf{A}\mathbf{r}$, $\mathbf{s}$ is too small to reveal much information about the random $\mathbf{r}$. More precisely, we have that $\mathbf{H}_\infty(\mathbf{r} \mid \mathbf{r}') \geq \mathbf{H}_\infty(\mathbf{r} \mid \mathbf{A}\mathbf{r})$ because $\mathbf{r}'$ is chosen deterministically based on $\mathbf{s} = \mathbf{A}\mathbf{r}$. Due to the Lemma 1.1, we have that $\mathbf{H}_\infty(\mathbf{r} \mid \mathbf{A}\mathbf{r}) \geq \mathbf{H}_\infty(\mathbf{r}) - \mathsf{log}(|\mathcal{S}|)$, where $\mathcal{S}$ is the space of all possible $\mathbf{s}$. Since $\mathbf{s} \in \mathbb{Z}_q^n$, $|\mathcal{S}| = q^n$, and then $\log(|\mathcal{S}|) = \log(q^n) = \log((2^{\mathsf{log} q})^n) = n\log((2^{\log q})) = n \log q$. Regarding $\mathbf{H}_\infty(\mathbf{r})$, since $\mathbf{H}_\infty(X) := -\mathsf{log}\big(\max_x \Pr[X = x]\big)$, we have $\mathbf{H}_\infty(\mathbf{r}) = -\mathsf{log}\big(2^{-mQ}\big) = mQ \geq m$. Then, $\mathbf{H}_\infty(\mathbf{r} \mid \mathbf{r}') \geq \mathbf{H}_\infty(\mathbf{r}) - \mathsf{log}(\mathcal{S}) \geq m - n\log q = \omega(\log\lambda)$. Since we know that for random variables X, Y the optimal probability of an unbounded adversary guessing X given the correlated value Y is $2^{-\mathbf{H}_\infty(X|Y)}$, then $\Pr[\mathbf{r} = \tilde{\mathbf{U}}\mathbf{r}'] \leq 2^{-\mathbf{H}_\infty(\mathbf{r}|\mathbf{r}')} \leq 2^{-\omega(\log\lambda)} = \mathsf{negl}(\lambda)$.

Type 3 Forgery. Let $(\mathcal{P}^* := (f^*, \ell_1^*, \ldots, \ell_t^*), \mathsf{m}^*, \sigma^* := (\mathsf{m}^*, \mathsf{z}^*, \mathsf{I}^*, \mathbf{U}^*, \mathbf{Z}^*))$ be a Type 3 forgery produced by $\mathcal{A}$ in the modified game such that there exists (at least) one label $\ell_j^* = (\mathsf{id}^*, \tau^*)$ such that $\mathsf{id}^* = \mathsf{id}_i$ but $\tau^* \notin \mathcal{T}_i$.[10] Actually, without loss of generality we can assume that there is exactly one of such labels; if this is not the case, one could indeed redefine another adversary that makes more queries until it misses only this one. Note that for such a tag $\tau^* \notin \mathcal{T}_i$, $\mathcal{B}$ simulated $\mathbf{V}_{\mathsf{id}_i,\tau^*} = \mathbf{A}\mathbf{U}_{\mathsf{id}_i,\tau^*} + b_{i,\tau^*}\mathbf{G}$ for a randomly chosen bit $b_{i,\tau^*} \xleftarrow{\$} \{0,1\}$, that is perfectly hidden from $\mathcal{A}$.

By definition of Type 3 forgery, the tuple passes verification, and in particular check (4) $\sum_{\mathsf{id}\in\mathsf{I}^*} \mathbf{A}_{\mathsf{id}}\mathbf{Z}^*_{\mathsf{id}} + \mathsf{z}^* \cdot \mathbf{G} = \mathbf{V}^+ = \sum_{i=1}^t \gamma_i \cdot \mathbf{V}_{\ell_i^*}$ where the right hand side of the equation holds by construction of the verification algorithm. Moreover, let $\sigma = (\mathsf{m}, \mathsf{z}, \mathsf{I}, \mathbf{U}, \mathbf{Z})$ be the signature obtained by honestly applying Eval to the signatures corresponding to labels $\ell_1^*, \ldots, \ell_t^*$; in particular for the specific, missing, label ℓ_j^* $\mathcal{B}$ uses the values $\mathbf{U}_{\mathsf{id}_i,\tau^*}, b_{i,\tau^*}$ used to simulate $\mathbf{V}_{\mathsf{id}_i,\tau^*}$. Parsing $\mathbf{Z} := \{\mathbf{Z}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}}$, notice that by correctness it holds $\mathsf{I} = \{\mathsf{id} : \mathsf{id} \in \mathcal{P}^*\}$ and $\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\mathbf{Z}_{\mathsf{id}} + \mathsf{z} \cdot \mathbf{G} = \mathbf{V}^+$ where $\mathsf{z} = \sum_{i=1, i\neq j}^t \gamma_i \mathsf{m}_i + \gamma_j b_{i,\tau^*}$. Now, the observation is that every $\gamma_i \leq 2^d < q$, i.e., $\gamma_i \neq 0 \mod q$. Since b_{i,τ^*} is random and perfectly hidden to $\mathcal{A}$ we have that with probability $1/2$ it holds $\mathsf{z} \neq \mathsf{z}^*$.

Thus, if $\mathsf{z} \neq \mathsf{z}^*$, $\mathcal{B}$ combines the equalities on $\mathbf{V}^+$ to come up with an equation $\sum_{\mathsf{id}\in\mathsf{I}} \mathbf{A}_{\mathsf{id}}\tilde{\mathbf{Z}}_{\mathsf{id}} = \tilde{\mathsf{z}} \cdot \mathbf{G}$ where $\tilde{\mathsf{z}} = \mathsf{z} - \mathsf{z}^* \neq 0 \mod q$ and, for all $\mathsf{id} \in \mathsf{I}$, $\tilde{\mathbf{Z}}_{\mathsf{id}} = \mathbf{Z}^*_{\mathsf{id}} - \mathbf{Z}_{\mathsf{id}} \in \mathcal{U}$ such that $\|\tilde{\mathbf{Z}}_{\mathsf{id}}\|_\infty \leq \beta_{\mathsf{max}}$.

Finally, using the same technique as in the case of Type 2 forgeries, $\mathcal{B}$ can compute a vector $\mathbf{u}$ that is a solution of SIS with overwhelming probability, i.e., $\mathbf{A}\mathbf{u} = 0$. Therefore, we have proven that if an adversary $\mathcal{A}$ can break the MKHSig scheme with non negligible probability, then $\mathcal{C}$ can use such an $\mathcal{A}$ to break the SIS assumption for $\mathbf{A}$ with non negligible probability as well. $\square$

A Variant with Unbounded Tag Space in the Random Oracle Model. In this section, we show that the construction of multi-key homomorphic signatures of Section 4.2 can be easily modified in order to have short public keys and to support an unbounded tag space $\mathcal{T} = \{0,1\}^*$. Note that once arbitrary tags are allowed, the scheme also allows to handle *multiple datasets* for free. In fact, one can always extend tags to include the dataset name, i.e., simply redefine each tag τ as consisting of two substrings $\tau = (\Delta, \tau')$ where Δ is the dataset name and τ' the actual tag. The idea of modifying the scheme to support an unbounded tag space is simple and was also suggested in [30] for their construction. Instead of sampling matrices

[10] It is easy to see that the case in which id^* is new would imply the generation of a new $\mathbf{A}_{\mathsf{id}^*}$, which would make the verification equations hold with negligible probability (over the random choice of $\mathbf{A}_{\mathsf{id}^*}$).

$\{\mathbf{V}_{\mathsf{id},1}, \dots \mathbf{V}_{\mathsf{id},\mathsf{T}}\}$ in KeyGen, one can just choose a random string $r_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$ and define every $\mathbf{V}_{\mathsf{id},\tau} := \hat{\mathsf{H}}(r_{\mathsf{id}}, \tau)$ where $\hat{\mathsf{H}} : \{0,1\}^* \to \mathcal{V}$ is a hash function chosen in Setup (modeled as a random oracle in the proof). In all the remaining algorithms, every time one needs $\mathbf{V}_{\mathsf{id},\tau}$, this is obtained using $\hat{\mathsf{H}}$.

For this modified scheme, we also provide an idea of how the security proof of Theorem 1.1 has to be modified to account for these changes. The main change is the simulation of hash queries, which is done as follows.

Before phase (1), where $\mathcal{A}$ declares its queries, $\mathcal{B}$ simply answers every query $\hat{\mathsf{H}}(r, \tau)$ with a randomly chosen $\mathbf{V} \xleftarrow{\$} \mathcal{V}$. Afterwards, once $\mathcal{A}$ has declared all its queries, $\mathcal{B}$ chooses $r_{\mathsf{id}_1}, \dots, r_{\mathsf{id}_Q} \xleftarrow{\$} \{0,1\}^\lambda$ and programs the random oracle so that, for all $\tau \in \mathcal{T}_i$, $\hat{\mathsf{H}}(r_{\mathsf{id}_i}, \tau) = \mathbf{V}_{\mathsf{id}_i,\tau}$ where $\mathbf{V}_{\mathsf{id}_i,\tau}$ is the same matrix generated in the phase (3) of the modified game. On the other hand, for all $\tau \notin \mathcal{T}_i$, $\hat{\mathsf{H}}(r_{\mathsf{id}_i}, \tau) = \mathbf{V}_{\mathsf{id}_i,\tau}$ where $\mathbf{V}_{\mathsf{id}_i,\tau} = \mathbf{A}_{\mathsf{id}}\mathbf{U}_{\mathsf{id}_i,\tau} + b_{i,\tau}\mathbf{G}$ for a randomly chosen $\mathbf{U}_{\mathsf{id}_i,\tau} \xleftarrow{\$} \mathcal{U}$. All other queries $\hat{\mathsf{H}}(r, \tau)$ where $r \neq r_{\mathsf{id}_i}, \forall i \in [Q]$ are answered with random $\mathbf{V} \xleftarrow{\$} \mathcal{V}$. With this simulation, it is not hard to see that, from $\mathcal{A}$'s forgery $\mathcal{B}$ can extract a solution for SIS (except for some negligible probability that $\mathcal{A}$ guesses one of r_{id_i} before seeing it).

4.3 From a Single Dataset to Multiple Datasets

In this section, we present a generic transformation to convert a single-dataset MKHSig scheme into a scheme that supports multiple datasets. The intuition behind this transformation is similar to the one employed in [30] and implicitly used in [13, 16], except that here we have to use additional techniques to deal with the multi-key setting. We combine a standard signature scheme $\mathsf{NH.Sig}$ (non-homomorphic) with a single dataset multi-key homomorphic signature scheme MKHSig'. The idea is that for every new dataset Δ, every user generates fresh keys of the multi-key homomorphic scheme MKHSig' and then uses the standard signature scheme $\mathsf{NH.Sig}$ to sign the dataset identifier Δ together with the generated public key. More precisely, in our transformation we assume to start with (single-dataset) multi-key homomorphic signature schemes in which the key generation algorithm can be split into two independent algorithms: KeyGen_1 that outputs some public parameters related to the identity id, and KeyGen_2 which outputs the actual keys. Differently than [30], in our scheme the signer does not need to sign the whole dataset at once, nor has to fix a bound N on the dataset size (unless such a bound is already contained in MKHSig').

In more details, let $\mathsf{NH.Sig} = (\mathsf{NH.KeyGen}, \mathsf{NH.Sign}, \mathsf{NH.Verify})$ be a standard (non-homomorphic) signature scheme, and let $\mathsf{MKHSig}' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Eval}', \mathsf{Verify}')$ be a single-dataset multi-key homomorphic signature scheme. We construct a multi-dataset multi-key homomorphic signature scheme $\mathsf{MKHSig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ as follows.

$\mathsf{Setup}(1^\lambda)$. The setup algorithm samples parameters of the single-dataset multi-key homomorphic signature scheme, $\mathsf{pp}' \leftarrow \mathsf{Setup}'(1^\lambda)$, together with a description of a PRF $F : K \times \{0,1\}^* \to \{0,1\}^\rho$, and outputs $\mathsf{pp} = (\mathsf{pp}', F)$.

$\mathsf{KeyGen}(\mathsf{pp})$. The key generation algorithm runs $\mathsf{NH.KeyGen}$ to get $(\mathsf{pk}_{\mathsf{id}}^{\mathsf{NH}}, \mathsf{sk}_{\mathsf{id}}^{\mathsf{NH}})$, a pair of keys for the standard signature scheme. In addition, it runs KeyGen_1 to generate user-specific public parameters $\mathsf{pp}_{\mathsf{id}}$, and chooses a seed K_{id} for the PRF F. The final output is the vector $(\mathsf{sk}_{\mathsf{id}}, \mathsf{ek}_{\mathsf{id}}, \mathsf{vk}_{\mathsf{id}})$: where $\mathsf{sk}_{\mathsf{id}} = (\mathsf{sk}_{\mathsf{id}}^{\mathsf{NH}}, \mathsf{K}_{\mathsf{id}})$, $\mathsf{ek}_{\mathsf{id}} = (\mathsf{pp}_{\mathsf{id}})$ and $\mathsf{vk}_{\mathsf{id}} = (\mathsf{pk}_{\mathsf{id}}^{\mathsf{NH}}, \mathsf{pp}_{\mathsf{id}})$.

$\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \Delta, \ell, \mathsf{m})$**.** The signing algorithm proceeds as follows. First it samples the keys of the single-dataset multi-key homomorphic signature scheme by feeding randomness $F_{\mathsf{K}_{\mathsf{id}}}(\Delta)$ to KeyGen_2, i.e., it runs $\mathsf{KeyGen}_2(\mathsf{pp}; F_{\mathsf{K}_{\mathsf{id}}}(\Delta))$ to obtain the keys $(\mathsf{sk}^{\Delta}_{\mathsf{id}}, \mathsf{ek}^{\Delta}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}})$.[11] The algorithm then runs $\sigma' {\leftarrow} \mathsf{Sign}'(\mathsf{sk}^{\Delta}_{\mathsf{id}}, \ell, \mathsf{m})$, and uses the non-homomorphic scheme to sign the concatenation of the public key $\mathsf{vk}^{\Delta}_{\mathsf{id}}$ and the dataset identifier Δ, i.e., $\sigma^{\Delta}_{\mathsf{id}} \leftarrow \mathsf{NH.Sign}(\mathsf{sk}^{\mathsf{NH}}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}}|\Delta)$, The output is the tuple $\sigma := (\mathsf{I} = \{\mathsf{id}\}, \sigma', \mathsf{par}_{\Delta})$ where $\mathsf{par}_{\Delta} = \{(\mathsf{ek}^{\Delta}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}}, \sigma^{\Delta}_{\mathsf{id}})\}$. Note that the use of the PRF allows every signer (having the same K_{id}) to generate the same keys of the scheme MKHSig' on the same dataset Δ.

$\mathsf{Eval}(f, \{(\sigma_i, \mathsf{EKS}_i)\}_{i\in[t]})$**.** For each $i \in [t]$, the algorithm parses every signature as $\sigma_i := (\mathsf{I}_i, \sigma'_i, \mathsf{par}_{\Delta,i})$ with $\mathsf{par}_{\Delta,i} = \{\mathsf{ek}^{\Delta}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}}, \sigma^{\Delta}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}_i}$, and sets $\mathsf{EKS}'_i = \{\mathsf{ek}^{\Delta}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{I}_i}$. It computes $\sigma' \leftarrow \mathsf{Eval}'(f, \{\sigma'_i, \mathsf{EKS}'_i\}_{i\in[t]})$, defines $\mathsf{I} = \cup^{t}_{i=1}\mathsf{I}_i$ and $\mathsf{par}_{\Delta} = \cup^{t}_{i=1}\mathsf{par}_{\Delta,i}$. The final output is $\sigma = (\mathsf{I}, \sigma', \mathsf{par}_{\Delta})$.

$\mathsf{Verify}(\mathcal{P}, \Delta, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \mathsf{m}, \sigma)$**.** The verification algorithm begins by parsing the verification keys as $\mathsf{vk}_{\mathsf{id}} := (\mathsf{pk}^{\mathsf{NH}}_{\mathsf{id}}, \mathsf{pp}_{\mathsf{id}})$ for each $\mathsf{id} \in \mathsf{I}$, and also the signature as $\sigma = (\mathsf{I}, \sigma', \mathsf{par}_{\Delta})$ with $\mathsf{par}_{\Delta} = \{(\mathsf{ek}^{\Delta}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}}, \sigma^{\Delta}_{\mathsf{id}})\}_{\mathsf{id}\in\mathsf{I}}$. Then, it proceeds with two main steps. First, for each $\mathsf{id} \in \mathsf{I}$, it verifies the standard signature $\sigma^{\Delta}_{\mathsf{id}}$ on the public key of the single-dataset multi-key homomorphic scheme and the given dataset, i.e., it checks whether $\mathsf{NH.Verify}(\mathsf{pk}^{\mathsf{NH}}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}}|\Delta, \sigma^{\Delta}_{\mathsf{id}}) = 1, \forall\, \mathsf{id} \in \mathsf{I}$. If at least one of the previous equations is not satisfied, the algorithm returns 0, otherwise it proceeds to the second check and returns the output of $\mathsf{Verify}'(\mathcal{P}, \{\mathsf{pp}_{\mathsf{id}}, \mathsf{vk}^{\Delta}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}, \mathsf{m}, \sigma')$.

Authentication Correctness. Correctness of the scheme substantially follows from the correctness of the regular signature scheme $\mathsf{NH.Sig}$, the single-dataset multi-hey homomorphic scheme MKHSig' and the PRF F.

Evaluation Correctness. Evaluation correctness follows directly from the correctness of the evaluation algorithm Eval' of the single-dataset MKHSig scheme, the correctness of $\mathsf{NH.Sig}$ and of the PRF.

Security. Intuitively, the security of the scheme follows from two main observations. First, no adversary is able to fake the keys of the single-dataset multi-key homomorphic signature scheme, due to the security of the standard signature scheme and the property of pseudo-random functions. Secondly, no adversary can tamper with the results of Eval for a specific dataset as this would correspond to breaking the security of the single-dataset MKHSig' scheme.

Theorem 1.2. *If F is a secure pseudo-random function, $\mathsf{NH.Sig}$ is an unforgeable signature scheme and MKHSig' is a secure single-dataset multi-key homomorphic signature scheme, then the MKHSig scheme for multiple datasets described in Section 4.3 is secure against adversaries that make static corruptions of keys and produce forgeries as in Definition 3.2.*

The full proof of Theorem 1.2 is given in [22].

5 Our Multi-Key Homomorphic MAC from OWFs

In this section, we describe our construction of a multi-key homomorphic authenticator with private verification keys and supporting the evaluation of low-degree

[11] Here we assume that a ρ-bits string is sufficient, otherwise it can always be stretched using a PRG.

arithmetic circuits. More precisely, for a computation represented by an arithmetic circuit of degree d and involving inputs from n distinct identities, the final authenticator has size $\binom{n+d}{d}$, that is bounded by $poly(n)$ (for constant d) or by $poly(d)$ (for constant n). Essentially, the authenticators of our scheme grow with the degree of the circuit and the number of distinct users involved in the computation, whereas their size remains *independent* of the total number of inputs / users. This property is particularly desirable in contexts that involve a small set of users each of which contributes with several inputs.

Although our multi-key homomorphic MAC supports less expressive computations than our homomorphic signatures of Section 4, the scheme comes with two main benefits. First, it is based on a simple, general assumption: it relies on pseudo-random functions and thus is secure only assuming existence of one-way functions (OWF). Second, the scheme is very intuitive and efficient: fresh MACs essentially consist only of two $\mathbb{F}_p$ field elements (where p is a prime of λ bits) and an identity identifier; after evaluation, the authenticators consist of $\binom{n+d}{d}$ elements in $\mathbb{F}_p$, and homomorphic operations are simply additions and multiplications in the multi-variate polynomial ring $\mathbb{F}_p[X_1, \ldots, X_n]$.

We describe the five algorithms of our scheme MKHMac below. We note that our solution is presented for single data set only. However, since it admits labels that are arbitrarily long strings it is straight-forward to extend the scheme for handling multiple data sets: simply redefine each tag τ as consisting of two substrings $\tau = (\Delta, \tau')$ where Δ is the dataset name and τ' the actual tag.

$\mathsf{Setup}(1^\lambda)$**.** The setup algorithm generates a λ-bit prime p and let the message space be $\mathcal{M} := \mathbb{F}_p$. The set of identities is $\mathsf{ID} = [\mathsf{C}]$ for some integer bound $\mathsf{C} \in \mathbb{N}$, while the tag space consists of arbitrary binary strings, i.e., $\mathcal{T} = \{0,1\}^*$. The set $\mathbb{F}$ of admissible functions is made up of all arithmetic circuits whose degree d is bounded by some polynomial in the security parameter. The setup algorithm outputs the public parameters pp which include the descriptions of $\mathcal{M}, \mathsf{ID}, \mathcal{T}, \mathbb{F}$ as in Section 3, as well as the description of a PRF family $F : \mathcal{K} \times \{0,1\}^* \to \mathbb{F}_p$ with seed space $\mathcal{K}$. The public parameters define also the authenticator space. Each authenticator σ consists of a pair (I, y) where $\mathsf{I} \subseteq \mathsf{ID}$ and y is in the C-variate polynomial ring $\mathbb{F}_p[X_1, \ldots, X_\mathsf{C}]$. More precisely, if C is set up as a very large number (e.g., $\mathsf{C} = 2^\lambda$) the polynomials y can still live in some smaller sub-rings of $\mathbb{F}_p[X_1, \ldots, X_\mathsf{C}]$.

$\mathsf{KeyGen}(\mathsf{pp})$**.** The key generation algorithm picks a random $x \xleftarrow{\$} \mathbb{F}_p^*$, a PRF seed $K \xleftarrow{\$} \mathcal{K}$, and outputs $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$ where $\mathsf{sk} = \mathsf{vk} = (x, K)$ and ek is void.

$\mathsf{Auth}(\mathsf{sk}, \ell, \mathsf{m})$**.** In order to authenticate the message m with label $\ell = (\mathsf{id}, \tau) \in \mathsf{ID} \times \mathcal{T}$, the authentication algorithm produces an authenticator $\sigma = (\mathsf{I}, \mathsf{y})$ where $\mathsf{I} \subseteq \mathsf{ID}$ and $\mathsf{y} \in \mathbb{F}_p[X_\mathsf{id}] \subset \mathbb{F}_p[X_1, \ldots, X_\mathsf{C}]$. The set I is simply $\{\mathsf{id}\}$. The polynomial y is a degree-1 polynomial in the variable X_id such that $\mathsf{y}(0) = \mathsf{m}$ and $\mathsf{y}(x_\mathsf{id}) = F(K_\mathsf{id}, \ell)$. Note that the coefficients of $\mathsf{y}(X_\mathsf{id}) = y_0 + y_\mathsf{id} X_\mathsf{id} \in \mathbb{F}_p[X_\mathsf{id}]$ can be efficiently computed with the knowledge of x_id by setting $y_0 = \mathsf{m}$ and $y_\mathsf{id} = \frac{F(K_\mathsf{id}, \ell) - \mathsf{m}}{x_\mathsf{id}}$. Moreover, y can be compactly represented by only giving the coefficients $y_0, y_\mathsf{id} \in \mathbb{F}_p$.

$\mathsf{Eval}(f, \{\sigma_k\}_{k \in [t]})$**.** Given a t-input arithmetic circuit $f : \mathbb{F}_p^t \to \mathbb{F}_p$, and the t authenticators $\{\sigma_k := (\mathsf{I}_k, \mathsf{y}_k)\}_k$, the evaluation algorithm outputs $\sigma = (\mathsf{I}, \mathsf{y})$ obtained in the following way. First, it determines all the identities involved in the computation by setting $\mathsf{I} = \cup_{k=1}^t \mathsf{I}_k$. Then every polynomial

y_k is "expanded" into a polynomial $\hat{\mathsf{y}}_k$, defined on the variables X_{id} corresponding to all the identities in I. This is done using the canonical embedding $\mathbb{F}_p[X_{\mathsf{id}} : \mathsf{id} \in \mathsf{I}_k] \hookrightarrow \mathbb{F}_p[X_{\mathsf{id}} : \mathsf{id} \in \mathsf{I}]$. It is worth noticing that the terms of $\hat{\mathsf{y}}_k$ that depend on variables in $\mathsf{I} \smallsetminus \mathsf{I}_k$ have coefficient 0. Next, let $\hat{f} : \mathbb{F}_p[X_{\mathsf{id}} : \mathsf{id} \in \mathsf{I}]^t \to \mathbb{F}_p[X_{\mathsf{id}} : \mathsf{id} \in \mathsf{I}]$ be the arithmetic circuit corresponding to the given f, i.e., $\hat{f}$ is the same as f except that additions (resp. multiplications) in $\mathbb{F}_p$ are replaced by additions (resp. multiplications) over the polynomial ring $\mathbb{F}_p[X_{\mathsf{id}} : \mathsf{id} \in \mathsf{I}]$. Finally, y is obtained as $\mathsf{y} = \hat{f}(\hat{\mathsf{y}}_1, \ldots, \hat{\mathsf{y}}_t)$.

$\mathsf{Verify}(\mathcal{P}, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}}, \mathsf{m}, \sigma)$. Let $\mathcal{P} = (f, \ell_1, \ldots, \ell_t)$ be a labeled program where f is a degree-d arithmetic circuit and every label is of the form $\ell_k = (\mathsf{id}_k, \tau_k)$. Let $\sigma = (\mathsf{I}, \mathsf{y})$ where $\mathsf{I} = \{\bar{\mathsf{id}}_1, \ldots, \bar{\mathsf{id}}_n\}$ with $\bar{\mathsf{id}}_i \neq \bar{\mathsf{id}}_j$ for $i \neq j$. The verification algorithm outputs 1 (accept) if and only if the authenticator satisfies the following three checks. Otherwise it outputs 0 (reject).

$$\begin{aligned}
\{\bar{\mathsf{id}}_1, \ldots, \bar{\mathsf{id}}_n\} &= \{\mathsf{id} : \mathsf{id} \in \mathcal{P}\}, && (5)\\
\mathsf{y}(0, \ldots, 0) &= \mathsf{m}\ , && (6)\\
\mathsf{y}(x_{\bar{\mathsf{id}}_1}, \ldots, x_{\bar{\mathsf{id}}_n}) &= f(F(K_{\mathsf{id}_1}, \ell_1), \ldots, F(K_{\mathsf{id}_t}, \ell_t))\ . && (7)
\end{aligned}$$

In the remainder of the section, we discuss the efficiency and succinctness of our MKHMac and prove the correctness of our scheme. We conclude with the security analysis of the proposed MKHMac scheme.

Succinctness. Let us consider the case of an authenticator σ which was obtained after running Eval on a circuit of degree d and taking inputs from n distinct identities. Note that every σ consists of two elements: a set $\mathsf{I} \subseteq [\mathsf{C}]$ and a polynomial $\mathsf{y} \in \mathbb{F}_p[X_{\bar{\mathsf{id}}} : \bar{\mathsf{id}} \in \mathsf{I}]$.

For the set I, it is easy to see that $|\mathsf{I}| = n$ and I can be represented with $n \log \mathsf{C}$ bits. The other part of the authenticator, y, is instead an n-variate polynomial in $\mathbb{F}_p[X_{\bar{\mathsf{id}}_1}, \ldots, X_{\bar{\mathsf{id}}_n}]$ of degree d. Since the circuit degree is d, the maximum number of coefficients of y is $\binom{n+d}{d}$. More precisely, the total size of y depends on the particular representation of the multi-variate polynomial y which is chosen for implementation. In [22] we discuss some possible representations (further details can also be found in [32]). For example, when employing the *sparse representation* of polynomials, the size of y is bounded by $O(nt \log d)$ where t is the number of non-zero coefficients in y (note that in the worst case, a polynomial $\mathsf{y} \in \mathbb{F}_p[X_{\bar{\mathsf{id}}_1}, \ldots, X_{\bar{\mathsf{id}}_n}]$ of degree d has at most $t = \binom{n+d}{d}$ non-zero coefficients). Thus, setting $\log \mathsf{C} \approx \log p \approx \lambda$, we have that the size in bits of the authenticator σ is $|\sigma| \leq \lambda n + \lambda \binom{n+d}{d}$. Ignoring the security parameter, we have that $|\sigma| = \mathsf{poly}(n)$ when d is constant, or $|\sigma| = \mathsf{poly}(d)$ when n is constant.

Efficiency of Eval. In what follows, we discuss the cost of computing additions and multiplications over authenticators in our MKHMac scheme. Let $\sigma^{(i)} = (\mathsf{I}^{(i)}, \mathsf{y}^{(i)})$, for $i = 1, 2$ be two authenticators and consider the operation $\sigma = \mathsf{Eval}(g, \sigma^{(1)}, \sigma^{(2)})$ where g is a fan-in-2 addition or multiplication gate. In both cases the set I of identities of $\sigma = (\mathsf{I}, \mathsf{y})$ is obtained as the union $\mathsf{I} = \mathsf{I}^{(1)} \cup \mathsf{I}^{(2)}$ that can be computed in time $O(n)$, where $n = |\mathsf{I}|$, assuming the sets $\mathsf{I}^{(1)}, \mathsf{I}^{(2)}$ are ordered. Regarding the computation of y from $\mathsf{y}^{(1)}$ and $\mathsf{y}^{(2)}$, one has to first embed each y_i into the ring $\mathbb{F}_p[X_{\bar{\mathsf{id}}} : \bar{\mathsf{id}} \in \mathsf{I}]$, and then evaluate addition (resp. multiplication) over $\mathbb{F}_p[X_{\bar{\mathsf{id}}} : \bar{\mathsf{id}} \in \mathsf{I}]$. Again, the costs of these operations depend on the adopted representation [24, 32].

Using the *sparse representation* of polynomials, expanding a y having t non-zero coefficients into an n-variate polynomial $\hat{\mathsf{y}}$ requires time at most $O(tn)$. To give an idea, such expansion indeed consists simply into inserting zeros in the correct positions of the exponent vectors of every non-zero monomial term of y. On the other hand, the complexity of operations (additions and multiplications) on polynomials using the *sparse representation* is usually estimated in terms of the number of monomial comparisons. The cost of such comparisons depends on the specific monomial ordering chosen, but is usually $O(n \log d)$, where n is the total number of variables and d is the maximum degree. Given two polynomials in sparse representation having t_1 and t_2 non-zero terms respectively, addition costs about $O(t_1 t_2)$ monomial comparisons (if the monomial terms are stored in sorted order the cost of addition drops to $O(t_1 + t_2)$), while multiplication requires to add (merge) t_2 intermediate products of t_1 terms each, and can be performed with $O(t_1 t_2 \log t_2)$ monomial comparisons [24].

Correctness. Authentication Correctness. By construction, each fresh authenticator $\sigma = (\mathsf{I}, \mathsf{y})$ of a message m labeled by $\ell := (\mathsf{id}, \tau)$ is of the form $\mathsf{I} = \{\mathsf{id}\}$ and $\mathsf{y}(X_{\mathsf{id}}) := y_0 + y_{\mathsf{id}} X_{\mathsf{id}} = \mathsf{m} + \frac{F(K_{\mathsf{id}}, \ell) - \mathsf{m}}{x_{\mathsf{id}}} X_{\mathsf{id}}$. Thus the set I satisfies equation (5) since $\{\mathsf{id} : \mathsf{id} \in \mathcal{I}_\ell\} = \{\mathsf{id}\}$. The two last verification checks (6) and (7) are automatically granted for the identity program $\mathcal{I}_\ell$ because $\mathsf{y}(0) = y_0 = \mathsf{m}$ and $\mathsf{y}(x_{\mathsf{id}}) = \mathsf{m} + \frac{F(K_{\mathsf{id}}, \ell) - \mathsf{m}}{x_{\mathsf{id}}} x_{\mathsf{id}} = F(K_{\mathsf{id}}, \ell)$.

Evaluation Correctness. The correctness of the Eval algorithm essentially comes from the structure of the multi-variate polynomial ring. We provide the detailed proof in the full version of the paper [22].

Security. In what follows we prove the security of our scheme against adversaries that make static corruptions, and produce forgeries according to the following restrictions.

Definition 1.8 (Weak Forgery). *Consider an execution of the experiment described in Section 3,* $\mathsf{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHAut}}(\lambda)$ *where* $(\mathcal{P}^*, \Delta^*, \mathsf{m}^*, \sigma^*)$ *is the tuple returned by the adversary at the end of the experiment, with* $\mathcal{P}^* = (f^*, \ell_1^*, \dots, \ell_t^*)$, Δ^* *a dataset identifier,* $\mathsf{m}^* \in \mathcal{M}$ *and* σ^* *an authenticator. First, we say that the labeled program* $\mathcal{P}^*$ *is* well-defined on a list L *if either one of the following two cases occurs:*

1. *There exists* $i \in [t]$ *such that* $(\ell_i^*, \cdot) \notin L$ *(i.e.,* $\mathcal{A}$ *never made a query with label* ℓ_i^**), and* $f^*(\{\mathsf{m}_j\}_{(\ell_j, \mathsf{m}_j) \in L} \cup \{\tilde{\mathsf{m}}_j\}_{(\ell_j, \cdot) \notin L})$ *outputs the same value for all possible choices of* $\tilde{\mathsf{m}}_j \in \mathcal{M}$*;*

2. L *contains the tuples* $(\ell_1^*, \mathsf{m}_1), \dots, (\ell_t^*, \mathsf{m}_t)$*, for some messages* $\mathsf{m}_1, \dots, \mathsf{m}_t$*.*

Then we say that $(\mathcal{P}^*, \Delta^*, \mathsf{m}^*, \sigma^*)$ *is a* weak forgery *if* $\mathsf{Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{vk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{m}^*, \sigma^*) = 1$ *and either one of the following conditions is satisfied:*

Type 1: L_{Δ^*} *was not initialized during the game (i.e.,* Δ^* *was never queried).*

Type 2: $\mathcal{P}^*$ *is well-defined on* L_{Δ^*} *but* $\mathsf{m}^* \neq f^*(\{\mathsf{m}_j\}_{(\ell_j, \mathsf{m}_j) \in L_{\Delta^*}} \cup \{0\}_{\ell_j \notin L_{\Delta^*}})$ *(i.e.,* m^* *is not the correct output of* $\mathcal{P}^*$ *when executed over previously authenticated messages).*

Type 3: $\mathcal{P}^*$ *is not well-defined on* L_{Δ^*}*.*

Although Definition 1.8 is weaker than our Definition 3.2, we stress that the above definition still protects the verifier from adversaries that try to cheat on the output of a computation. In more details, the difference between Definition 1.8 and Definition 3.2 is the following: if f^* has an input wire that has never been authenticated during the game (a Type 3 forgery in Definition 3.2), but f^* is *constant* with respect to such input wire, then the above definition does not consider it a forgery. The intuitive reason why such a relaxed definition still makes sense is that "irrelevant" inputs would not help in any case the adversary to cheat on the output of f^*. Definition 1.8 is essentially the multi-key version of the forgery definition used in previous (single-key) homomorphic MAC works, e.g., [11]. As discussed in [23] testing whether a program is well-defined may not be doable in polynomial time in the most general case (i.e., every class of functions). However, in [12] it is shown how this can be done efficiently via a probabilistic test in the case of arithmetic circuits of degree d over a finite field of order p such that $d/p < 1/2$. Finally, we notice that for our MKHMac Type 1 forgeries cannot occur as the scheme described here supports only one dataset.[12]

Theorem 1.3. *If F is a pseudo-random function then the multi-key homomorphic MAC described in Section 5 is secure against adversaries that make static corruptions of keys and produce forgeries as in Definition 1.8.*

Note that we can deal with corruptions via our generic result of Proposition 1.1. Therefore, it is sufficient to prove the security against adversaries that make no corruptions. The proof is done via a chain of games following this (intuitive) path. First, we rule out adversaries that make Type 3 forgeries. Intuitively, this can be done as the adversary has never seen one of the inputs of the computation, and in particular an input which can change the result. Second, we replace every PRF instance with a truly random function. Note that at this point the security of the scheme is information theoretic. Third, we change the way to answer verification queries that are candidates to be Type 2 forgeries. Finally, we observe that in this last game the adversary gains no information on the secret keys x_i and thus has negligible probability of making a Type 2 forgery. Due to space restrictions, the detailed and formal proofs appear in only in the full version [22].

6 Conclusions

In this paper, we introduced the concept of multi-key homomorphic authenticators, a cryptographic primitive that enables an untrusted third party to execute a function f on data authenticated using different secret keys in order to obtain a value certifying the correctness of f's result, which can be checked with knowledge of corresponding verification keys. In addition to providing suitable definitions, we also propose two constructions: one which is publicly verifiable and supports general boolean circuits, and a second one that is secretly verifiable and supports low-degree arithmetic circuits. Although our work does not address directly the problem of privacy, extensions of our results along this direction are possible, and we leave the details to future investigation. A first extension is defining a notion of context-hiding for multi-key HAs. Similarly to the single key setting, this property should guarantee that authenticators do not reveal non-trivial information about the

[12]As noted at the beginning of the section the extension to multiple datasets is straightforward given that tags are arbitrary strings. When such extension is applied it is easy to see that Type 1 forgeries are Type 3 ones in the underlying scheme.

computation's inputs. The second extension has to do with preventing the Cloud from learning the data over which it computes. In this case, we note that multi-key HAs can be executed on top of homomorphic encryption following an approach similar to that suggested in [21]. Finally, an interesting problem left open by our work is to find multi-key HA schemes where authenticators have size independent of the number of users involved in the computation.

Acknowledgements. This work was partially supported by the COST Action IC1306 through a STSM grant to Elena Pagnin, the European Union's H2020 Research and Innovation Programme under grant agreement 688722 (NEXTLEAP), the Spanish Ministry of Economy under project reference TIN2015-70713-R (DEDETIS) and a Juan de la Cierva fellowship to Dario Fiore, and by the Madrid Regional Government under project N-Greens (ref. S2013/ICE-2731). This work was also partially supported by the the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no 608743, the VR grant PRECIS no 621-2014-4845 and the STINT grant Secure, Private & Efficient Healthcare with wearable computing no IB2015-6001.

Bibliography

[1] Shweta Agrawal, Dan Boneh, Xavier Boyen, and David Mandell Freeman. "Preventing Pollution Attacks in Multi-source Network Coding". In: *Public Key Cryptography.* Springer. 2010, pp. 161–176.

[2] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *28th ACM STOC.* ACM Press, 1996, pp. 99–108.

[3] Miklós Ajtai. "Generating Hard Instances of the Short Basis Problem". In: *Automata, Languages and Programming, 26th International Colloquium.* 1999, pp. 1–9.

[4] Joël Alwen and Chris Peikert. "Generating shorter bases for hard random lattices". In: *Theory of Computing Systems* 48.3 (2011), pp. 535–553.

[5] Michael Backes, Dario Fiore, and Raphael M. Reischuk. "Verifiable delegation of computation on outsourced data". In: *ACM CCS.* ACM Press, 2013, pp. 863–874.

[6] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. "Verifiable Delegation of Computation over Large Datasets". In: vol. 6841. Springer, Heidelberg, 2011, pp. 111–131.

[7] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. "Recursive composition and bootstrapping for SNARKS and proof-carrying data". In: *45th ACM STOC.* ACM Press, 2013, pp. 111–120.

[8] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. "Signing a Linear Subspace: Signature Schemes for Network Coding". In: *PKC.* Vol. 5443. Springer, Heidelberg, 2009, pp. 68–87.

[9] Dan Boneh and David Mandell Freeman. "Homomorphic Signatures for Polynomial Functions". In: vol. 6632. Springer, Heidelberg, 2011, pp. 149–168.

[10] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. "Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits". In: *EUROCRYPT.* Vol. 8441. Springer, Heidelberg, 2014, pp. 533–556.

[11] Dario Catalano and Dario Fiore. "Practical Homomorphic MACs for Arithmetic Circuits". In: vol. 7881. Springer, Heidelberg, 2013, pp. 336–352.

[12] Dario Catalano, Dario Fiore, Rosario Gennaro, and Luca Nizzardo. "Generalizing Homomorphic MACs for Arithmetic Circuits". In: vol. 8383. Springer, Heidelberg, 2014, pp. 538–555.

[13] Dario Catalano, Dario Fiore, and Luca Nizzardo. “Programmable Hash Functions Go Private: Constructions and Applications to (Homomorphic) Signatures with Shorter Public Keys”. In: *CRYPTO*. Vol. 9216. Springer, 2015, pp. 254–274.

[14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. “Adaptive Pseudo-free Groups and Applications”. In: *EUROCRYPT*. Vol. 6632. Springer, 2011, pp. 207–223.

[15] Dario Catalano, Dario Fiore, and Bogdan Warinschi. “Efficient Network Coding Signatures in the Standard Model”. In: *PKC*. Vol. 7293. Springer, 2012, pp. 680–696.

[16] Dario Catalano, Dario Fiore, and Bogdan Warinschi. “Homomorphic Signatures with Efficient Verification for Polynomial Functions”. In: *CRYPTO*. Vol. 8616. Sringer, 2014, pp. 371–389.

[17] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. “Multi-Client Non-interactive Verifiable Computation”. In: *TCC*. Vol. 7785. Springer, 2013, pp. 499–518.

[18] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. “Improved Delegation of Computation Using Fully Homomorphic Encryption”. In: vol. 6223. Springer, Heidelberg, 2010, pp. 483–501.

[19] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data”. In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2004, pp. 523–540.

[20] Dario Fiore and Rosario Gennaro. “Publicly verifiable delegation of large polynomials and matrix computations, with applications”. In: *ACM CCS*. ACM Press, 2012, pp. 501–512.

[21] Dario Fiore, Rosario Gennaro, and Valerio Pastro. “Efficiently Verifiable Computation on Encrypted Data”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 844–855.

[22] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. “Multi-Key Homomorphic Authenticators”. In: *IACR Cryptology ePrint Archive* (2016).

[23] David Mandell Freeman. “Improved Security for Linearly Homomorphic Signatures: A Generic Framework”. In: *PKC*. Vol. 7293. Springer, 2012, pp. 697–714.

[24] Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithms for Computer Algebra*. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[25] Rosario Gennaro, Craig Gentry, and Bryan Parno. “Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers”. In: *CRYPTO*. Vol. 6223. Springer, 2010, pp. 465–482.

[26] Rosario Gennaro and Daniel Wichs. “Fully Homomorphic Message Authenticators”. In: *ASIACRYPT*. Vol. 8270. Springer, 2013, pp. 301–320.

[27] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *41st ACM STOC*. ACM Press, 2009, pp. 372–381.

[28] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. “Trapdoors for hard lattices and new cryptographic constructions”. In: *ACM STOC*. ACM Press, 2008, pp. 197–206.

[29] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating computation: interactive proofs for muggles". In: *ACM STOC*. ACM Press, 2008, pp. 113–122.

[30] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. "Leveled Fully Homomorphic Signatures from Standard Lattices". In: *47th ACM STOC*. ACM Press, 2015, pp. 469–477.

[31] S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. "Multi-Client Verifiable Computation with Stronger Security Guarantees". In: *TCC*. Vol. 9015. Springer, 2015, pp. 144–168.

[32] Joris van der Hoeven and Grgoire Lecerf. "On the bit-complexity of sparse polynomial and series multiplication". In: *Journal of Symbolic Computation* 50 (2013), pp. 227 –254.

[33] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. "Homomorphic Signature Schemes". In: *CT-RSA*. Vol. 2271. Springer, Heidelberg, 2002, pp. 244–262.

[34] Daniele Micciancio. "Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor". In: *SIAM Journal on Computing* 34.1 (2004), pp. 118–169.

[35] Daniele Micciancio and Chris Peikert. "Hardness of SIS and LWE with small parameters". In: *CRYPTO*. Vol. 8042. Springer, 2013, pp. 301–320.

[36] Daniele Micciancio and Chris Peikert. "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller". In: *EUROCRYPT*. Vol. 7237. Springer, 2012, pp. 700–718.

[37] Daniele Micciancio and Oded Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In: *45th FOCS*. IEEE, 2004, pp. 372–381.

[38] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. "How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption". In: *TCC*. Vol. 7194. Springer, 2012, pp. 422–439.

Paper B

Matrioska: A Compiler for Multi-Key Homomorphic Signatures

Dario Fiore and Elena Pagnin

Abstract. Multi-Key Homomorphic Signatures (MKHS) enable clients in a system to sign and upload messages to an untrusted server. At any later point in time, the server can perform a computation C on data provided by t different clients, and return the output y and a short signature $\sigma_{C,\mathsf{y}}$ vouching for the correctness of y as the output of the function C on the signed data. Interestingly, MKHS enable verifiers to check the validity of the signature using solely the public keys of the signers whose messages were used in the computation. Moreover, the signatures $\sigma_{C,\mathsf{y}}$ are succinct, namely their size depends at most linearly in the number of clients, and only logarithmically in the total number of inputs of C.

Existing MKHS are constructed based either on standard assumptions over lattices (Fiore *et al.*, ASIACRYPT'16), or on non-falsifiable assumptions (SNARKs) (Lai *et al.*, ePrint'16). In this paper, we investigate connections between single-key and multi-key homomorphic signatures. We propose a generic compiler, called Matrioska, which turns any (sufficiently expressive) single-key homomorphic signature scheme into a multi-key scheme. Matrioska establishes a formal connection between these two primitives and is the first alternative to the only known construction under standard falsifiable assumptions. Our result relies on a novel technique that exploits the homomorphic property of a single-key HS scheme to compress an arbitrary number of signatures from t different users into only t signatures.

Keywords. Homomorphic Signatures, Multi-Key Homomorphic Signatures.

Proceedings of the 11th Conference on Security and Cryptography for Networks (SCN), 2018.

MATRIOSKA: A COMPILER FOR MULTI-KEY HOMOMORPHIC SIGNATURES

1 Introduction

Consider a scenario where a user Alice uploads a collection of data items $x_1, \ldots, x_\mathsf{n}$ to an untrusted server. Later on, the server executes a computation $\mathcal{P}$ on this data and sends the result $\mathsf{y} = \mathcal{P}(x_1, \ldots, x_\mathsf{n})$ to another user Bob. *How can Bob be sure that* y *is the correct result obtained by running* $\mathcal{P}$ *on Alice's data?*
A trivial solution to this problem could be obtained by employing digital signatures: Alice could sign each data item x_i and send to the server the signatures $\sigma_1, \ldots, \sigma_\mathsf{n}$. Next, to convince Bob, a server can send along with y the original inputs with their signatures, and Bob should check that $\mathsf{y} = \mathcal{P}(x_1, \ldots, x_\mathsf{n})$ and that each σ_i is a valid signature for x_i. While this solution solves the above security concern, it has a clear efficiency drawback: it requires communication between the server and the verifier Bob that is *linear* in the input size of $\mathcal{P}$. This cost is undesirable and can even be unacceptable if Bob is cannot store the $x_1, \ldots, x_\mathsf{n}$.

Homomorphic Signatures. A solution to the above problem that achieves both security and efficiency can be obtained by using *homomorphic signatures* (HS). With this primitive, Alice can use her secret key to sign $x_1, \ldots, x_\mathsf{n}$ and sends the signed data items to the server. The server can use a special procedure Eval that, on input a program $\mathcal{P}$ and a collection of signatures $\sigma_1, \ldots, \sigma_\mathsf{n}$, outputs a signature $\sigma_{\mathcal{P},\mathsf{y}}$. Given Alice's public key and a triple $(\mathcal{P}, \mathsf{y}, \sigma_{\mathcal{P},\mathsf{y}})$, Bob (or anyone else) can get convinced that y is the correct output of $\mathcal{P}$ on inputs $(x_1, \ldots, x_\mathsf{n})$ signed by Alice. Very informally, homomorphic signatures are secure in the sense that an untrusted server (without knowing Alice's secret key) must not be able to convince the verifier of a false result. An additional property that makes this cryptographic primitive interesting and non-trivial is that signatures must be *succinct.* This means that the size of $\sigma_{\mathcal{P},\mathsf{y}}$ must be significantly smaller than $\mathcal{P}$'s input size, *e.g., size*$(\sigma_{\mathcal{P},\mathsf{y}}) = O(\log \mathsf{n})$.

The notion of homomorphic signatures was proposed by Desdmedt [16] and first formalized by Johnson *et al.* [24]. Boneh *et al.* [4] proposed the first scheme for computing linear functions over signed vectors and showed an application to preventing pollution attacks in linear network coding. Following [4], a long series of works (e.g., [1, 2, 6, 8, 9, 11–13, 15, 19, 20, 26]) addressed the problem of constructing linearly-homomorphic signatures obtaining new schemes that improved on multiple fronts, such as efficiency, security, and privacy. A few more works addressed the problem of constructing schemes for more expressive functionalities [5, 7, 14, 23]. Boneh and Freeman [5] proposed the first scheme for polynomial functions based on lattices, which was later improved by Catalano, Fiore and Warinschi [14] based on multilinear maps. In 2015, Gorbunov, Vaikuntanathan and Wichs [23] constructed the first HS scheme for arbitrary circuits of bounded depth from standard lattices.

Multi-Key Homomorphic Signatures. In a recent work, Fiore *et al.* [17] initiated the study of *multi-key homomorphic signatures* (MK-HS). In a nutshell, MK-HS are homomorphic signatures that allow for computing on data signed using different secret keys. This capability extends that one of previously known homomorphic signatures, and is useful in all those applications where one wants to compute on data provided (and signed) by multiple users. In addition to formally defining the notion of multi-key homomorphic signatures, Fiore *et al.* proposed a construction of MK-HS based on lattices that supports bounded depth circuits. Their scheme is obtained by extending the techniques of the single-key scheme of Gorbunov *et al.* [23]. Another recent work by Lai *et al.* [25] shows how to build an MK-HS using SNARKs and digital signatures. However, since SNARKs are likely to be based on non-falsifiable assumptions [22], the resulting MK-HS also relies on non standard assumptions.

1.1 Our Contribution

In this work, we continue the study of multi-key homomorphic signatures. Our main interest is to identify connections between multi-key homomorphic signatures and their single-key counterpart. In particular, we provide the first generic method to construct multi-key homomorphic signatures from (sufficiently expressive) single-key HS schemes. Our main contribution is a compiler, called $\mathsf{Matrioska}$, that yields the following result:

Theorem 3.1 (Informal). *Let* HS *be a homomorphic signature scheme for circuits of polynomial size. Then, for a constant* t *representing the number of distinct keys involved in a computation, there exists a multi-key homomorphic signature scheme* $\mathsf{MKHS}(\mathsf{HS}, t)$ *for circuits of polynomial size. Furthermore, if* HS *has signatures bounded by a fixed polynomial* $p(\lambda)$*,* $\mathsf{MKHS}(\mathsf{HS}, t)$ *has signatures bounded by* $t \cdot p(\lambda)$*.*

Our result essentially shows that for a sufficiently expressive class of functions multi-key and single-key homomorphic signatures are equivalent. Our construction is the first to establish a formal connection between these two primitives without resorting to powerful primitives such as SNARKs which only yield constructions from non-falsifiable assumptions. Also, we propose a new methodology to construct MK-HS, which is the first alternative to the only known construction from standard assumptions [17]. In particular, while the techniques in [17] are specific to an algebraic lattice setting, our construction works in a generic fashion and as such it will allow to immediately obtain new MK-HS schemes from any future proposal of single-key HS.

Our MK-HS construction is quite involved and its efficiency is, admittedly, theoretical. In particular, in order to support circuits of (polynomial) size s, we need to start from a single-key HS scheme that supports circuits of size $s^{{c_s}^{t-1}}$, where t is the number of distinct keys involved in the computation and c_s is some constant that depends on the single-key HS scheme. Therefore our generic construction generates multi-key homomorphic signature schemes that can support computations among a constant number of keys (*i.e.,* users) only.

Nevertheless, our MK-HS scheme has succinct signatures that have size $t \cdot p(\lambda)$, which is non-trivial as it is independent of the total number of inputs involved in the computation. Indeed, even in the multi-key setting a trivial solution to build MK-HS from digital signatures (and even from HS) would require communication linear in the total number of inputs of a computation, i.e., $O(\mathsf{n} \cdot t)$, assuming each user provides n inputs.

An overview of our techniques. The main challenge in constructing an MK-HS scheme generically from a single-key one is to obtain a construction with succinct signatures. In particular, obtaining succinctness requires some mechanism to "compress" $\mathsf{n} \cdot t$ signatures into some information that can at most depend linearly on $\log \mathsf{n}$ and t. While single-key HS allow for compressing signatures pertaining to the same key, this property seems of no utility when one needs to compute on signatures pertaining to different keys, if nothing about their structure can be assumed.[13] To overcome this challenge, we devise a novel technique that allows us to compress $\mathsf{n} \cdot t$ signatures from t different users into t signatures; for this we show how to use the homomorphic property of the single-key HS scheme in order to inductively "prove" that the signatures of the first i users verify correctly on the corresponding inputs.

In what follows we illustrate the core idea of our technique considering, for simplicity, the two-client case $t = 2$, and assuming each users contributes to the computation with n inputs.

Let $C : \{0,1\}^{2 \cdot \mathsf{n}} \to \{0,1\}$ be the circuit we wish to evaluate. Given the messages $\mathsf{m}_1, \dots \mathsf{m}_\mathsf{n}$ by user id_1 and $\mathsf{m}_{\mathsf{n}+1}, \dots \mathsf{m}_{2 \cdot \mathsf{n}}$ by user id_2, we wish to authenticate the output of $\mathsf{y} = C(\mathsf{m}_1, \dots, \mathsf{m}_{2 \cdot \mathsf{n}})$. Let σ_i be the signature for the message m_i; in particular the first n signatures and the last n signatures are associated to different secret keys.

The initial step is to construct a $(2 \cdot \mathsf{n})$-input circuit E_0 such that $E_0(x_1, \dots, x_{2\mathsf{n}}) = 1$ iff $C(x_1, \dots, x_{2\mathsf{n}}) = \mathsf{y}$. Second, define a new circuit $E_1 : \{0,1\}^\mathsf{n} \to \{0,1\}$ that is E_0 with the last n inputs hardwired: $E_1(x_1, \dots, x_\mathsf{n}) = E_0(x_1, \dots, x_\mathsf{n}, \mathsf{m}_{\mathsf{n}+1}, \dots, \mathsf{m}_{2\mathsf{n}})$. Now E_1 is a circuit that has inputs by a single client only, thus we can run $\hat{\sigma}_1 \leftarrow \mathsf{HS.Eval}(E_1, \mathsf{pk}_1, \sigma_1, \dots, \sigma_\mathsf{n})$. By the correctness of the single-key homomorphic signature scheme it must hold $\mathsf{HS.Verify}(E_1, \mathsf{pk}_1, \hat{\sigma}_1, 1) = 1$. At this point, we already compressed the signatures $\sigma_1, \dots, \sigma_\mathsf{n}$ into a single signature $\hat{\sigma}_1$. This is however not yet sufficient for succinctness because verifying $\hat{\sigma}_1$ requires the circuit E_1, which in turn requires to transmit to the verifier n messages $(\mathsf{m}_{\mathsf{n}+1}, \dots, \mathsf{m}_{2\mathsf{n}})$ to let him reconstruct E_1.

This is where the inductive reasoning, and our new technique, begins. Very intuitively, we use the signatures of the second user to "prove" that $\mathsf{HS.Verify}(E_1, \mathsf{pk}_1, \hat{\sigma}_1, 1) = 1$, without letting the verifier run this verification explicitly. Let us see $\mathsf{H} = \mathsf{HS.Verify}((E_1, (\tau_1, \dots, \tau_\mathsf{n})), \mathsf{pk}_1, \hat{\sigma}_1, 1)$ as a binary string with the description of a (no input) circuit. Look for the bits of H where the values $\mathsf{m}_{\mathsf{n}+1}, \dots, \mathsf{m}_{2\mathsf{n}}$ are embedded. We can define a new circuit description E_2 that is the same as H except that the hardwired values $\mathsf{m}_{\mathsf{n}+1}, \dots, \mathsf{m}_{2\mathsf{n}}$ are replaced with input gates. Thus E_2 is an n-input circuit satisfying $E_2(\mathsf{m}_{\mathsf{n}+1}, \dots, \mathsf{m}_{2\mathsf{n}}) = \mathsf{HS.Verify}(E_1, \mathsf{pk}_1, \hat{\sigma}_1, 1)$, which returns 1 by correctness of HS.

Now, the crucial observation is that E_2 is a circuit on inputs by the second client only. Thus, we can run $\hat{\sigma}_2 \leftarrow \mathsf{HS.Eval}(E_2, \mathsf{pk}_2, \sigma_{\mathsf{n}+1}, \dots, \sigma_{2\mathsf{n}})$. By the correctness of the HS scheme, $\mathsf{HS.Verify}(E_2, \mathsf{pk}_2, \hat{\sigma}_2, 1) = 1$. Note that E_2 does not contain any of the messages $\mathsf{m}_1, \dots, \mathsf{m}_{2 \cdot \mathsf{n}}$ hardwired; in particular E_2 is completely determined by C, y, pk_1, $\hat{\sigma}_1$ and a description of $\mathsf{HS.Verify}$. Hence, given $(\hat{\sigma}_1, \hat{\sigma}_2)$ the verifier can reconstruct E_2 and check if $\mathsf{HS.Verify}(E_2, \mathsf{pk}_2, \hat{\sigma}_2, 1) = 1$. Intuitively, this proves that for some messages signed by the second user $E_2(\mathsf{m}_{\mathsf{n}+1}, \dots, \mathsf{m}_{2\mathsf{n}}) = 1$. By the correctness of HS, this in turn implies $E_1(\mathsf{m}_1, \dots, \mathsf{m}_\mathsf{n}) = 1$ for some messages signed by the first user; and by construction of E_1 the latter implies $C(\mathsf{m}_1, \dots, \mathsf{m}_{2\mathsf{n}}) = \mathsf{y}$.

Our compiler, extends the above idea to multiple users, showing that at each step i the problem consists in proving correctness of a computation E_{i-1} that de-

[13]This is the case if one aims for a generic single-key to multi-key construction. In contrast, knowing for example the algebraic structure of signatures can be of help, as exploited in [17].

pends only on the inputs of user i, while inputs of users $> i$ are hardwired into it. This means that a progressive application of this idea lets the hardwired inputs progressively disappear up to the point of obtaining a circuit E_t which has no input hardwired and thus can be reconstructed by the verifier. This is the only computation explicitly checked by the verifier. By construction, E_t encodes the nested execution of several single-key HS verifications (from which our compiler's name "Matrioska"), and validity of E_t implicitly implies that each E_i returns 1 (even if the verifier does not know E_i itself). In this description we favor intuition to precision. A detailed presentation can be found in Section 3.

2 Preliminaries

Notation. The security parameter of our schemes is denoted by λ. For any $n \in \mathbb{N}$, we use $[n]$ to denote the set $[n] := \{1, \dots, n\}$. The symbol lg denotes the logarithm in base 2; $||$ denotes the string concatenation, *e.g.,* $(00)||(10) = (0010)$; bold font letters, *e.g.,* $\vec{\sigma} = (\sigma_1, \dots, \sigma_\mathsf{n})$, denote vectors. A function $\epsilon(\lambda)$ is said negligible in λ (denoted as $\epsilon(\lambda) = \mathsf{negl}(\lambda)$) if $\epsilon(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$. Also, we often write $\mathsf{poly}(\cdot)$ to denote a function that can be expressed as a polynomial.

2.1 Circuits

We use a modeling of circuits similar to the one in [3]. We define circuits as 6-tuples $C = (\mathsf{n}, \mathsf{u}, \mathsf{q}, \mathsf{L}, \mathsf{R}, \mathsf{G})$. The value $\mathsf{n} \geq 1$ denotes the number of inputs to the circuit, $\mathsf{u} \geq 1$ is the number of outputs and $\mathsf{q} \geq 1$ is the number of gates. Let w denote the total number of wires in the circuit. For the circuits considered in this work $\mathsf{w} = \mathsf{n} + \mathsf{q}$. The functions L and R define respectively the left and right input wire to any given gate $g \in [\mathsf{q}]$, formally, $\mathsf{L}, \mathsf{R} : [\mathsf{q}] \to [\mathsf{w}] \cup \{0\}$. Finally, $\mathsf{G} : [\mathsf{q}] \to \{0, 1\}$ encodes the gates by mapping each gate $g \in [\mathsf{q}]$ into a single bit G_g. In our construction we treat circuit descriptions C as binary strings. Similarly to [3], the size of our circuit description is quasi-linear in the number of wires: $|C| \in O(\mathsf{w} \lg(\mathsf{w}))$. Differently from [3], we number gates from 1 to q (instead of from $\mathsf{n} + 1$ to $\mathsf{n} + \mathsf{q}$) and label the outgoing wire of a gate g as $g + \mathsf{n}$. Moreover, we introduce the 0 wire to denote *constant output* gates, *e.g.,* no-input gates or gates that have the same output independently of the input values, and allow for a gate to have the same left and right input, *i.e.,* $\mathsf{L}(g) \leq \mathsf{R}(g) < g + \mathsf{n}$. The largest component in the string C is the descriptions of the function L (and R), that is a sequence of q values in $[\mathsf{w}] \cup \{0\}$, therefore $|\mathsf{L}| = |\mathsf{R}| = \mathsf{q} \lg(\mathsf{w} + 1)$. Hence, for a fixed and reasonable encoding it holds $|C| \in O(\mathsf{w} \lg(\mathsf{w}))$.

As an example of a circuit consider the following EQ^y circuit (that will be used in our generic compiler) $EQ^\mathsf{y} = \big(1, 1, 5, (01134), (02325), (\mathsf{y}, 1, 1, 1, 1)\big)$.

We explain the procedure to evaluate a 1-output, n-input circuit and refer the reader to [18] for the general case. Given $(x_1, \dots, x_\mathsf{n})$ and the circuit description $C = (\mathsf{n}, 1, \mathsf{q}, \mathsf{L}, \mathsf{R}, \mathsf{G})$, compute $\mathsf{y} = C(x_1, \dots, x_\mathsf{n})$ as follows. Retrieve the label of the left and right input wires to gate $g = i$, for $i = 1, 2, \dots, \mathsf{q}$. Let $l \leftarrow \mathsf{L}(i)$ and $r \leftarrow \mathsf{L}(i)$. Create a new variable $x_{\mathsf{n}+i} \in \{0, 1\}$. If $l = 0 = r$, g is a constant gate, assign $x_{\mathsf{n}+i} \leftarrow \mathsf{G}(i)$. Otherwise, by definition $l \neq 0 \neq r$, retrieve the values x_l and x_r, and return $x_{\mathsf{n}+i} \leftarrow x_l$ if $\mathsf{G}(i) = 0$, or $x_{\mathsf{n}+i} \leftarrow \mathsf{NAND}(x_l, x_r)$ if $\mathsf{G}(i) = 1$. The output is $x_{\mathsf{n}+\mathsf{q}} = \mathsf{y} = C(x_1, \dots, x_\mathsf{n})$.

Another interesting operation on circuits is circuit composition. Given two circuits, C_1 and C_2, we say that C_1 is *composable* with C_2 if $\mathsf{u}_1 = \mathsf{n}_2$. Intuitively, composition connects each output wire of C_1 with one input wire of C_2. We denote the circuit composition as $C_3 = C_1 \triangleright C_2$. The resulting circuit $C_3 = (\mathsf{n}_3, \mathsf{u}_3, \mathsf{q}_3, \mathsf{L}_3, \mathsf{R}_3, \mathsf{G}_3)$ is defined as: $\mathsf{n}_3 = \mathsf{n}_1$, $\mathsf{u}_3 = \mathsf{u}_2$, $\mathsf{q}_3 = \mathsf{q}_1 + \mathsf{q}_2$. Let w_i be the number of wires in C_i, then

$$\mathsf{L}_3 = \begin{cases} \mathsf{L}_1(i) & \text{for } i \in [\mathsf{w}_1] \\ 0 & \text{for } i \in [\mathsf{w}_1 + \mathsf{w}_2] \setminus [\mathsf{w}_1] \text{ and } \mathsf{L}_2(i - \mathsf{w}_1) = 0 \\ \mathsf{L}_2(i - \mathsf{w}_1) + \mathsf{w}_1 - \mathsf{u}_1 & \text{for } i \in [\mathsf{w}_1 + \mathsf{w}_2] \setminus [\mathsf{w}_1] \text{ and } \mathsf{L}_2(i - \mathsf{w}_1) \neq 0 \end{cases}$$

Note that the entries of L_3 that are set to 0 preserve constant output gates. The right-input function R_3 is defined analogously. The right-input function R_3 is defined analogously. Finally, $\mathsf{G}_3 = \mathsf{G}_1 || \mathsf{G}_2$.

2.2 Multi-Key Homomorphic Signatures

We start by recalling the notion of labeled programs of Gennaro and Wichs [21].

Labeled Programs [21]. A labeled program $\mathcal{P}$ is a tuple $(C, \ell_1, \dots, \ell_t)$, such that $C : \mathcal{M}^t \to \mathcal{M}$ is a function of t variables (*e.g.,* a circuit) and $\ell_i \in \{0,1\}^*$ is a label for the i-th input of C. Labeled programs can be composed as follows: given $\mathcal{P}_1, \dots, \mathcal{P}_n$ and a function $G : \mathcal{M}^n \to \mathcal{M}$, the composed program $\mathcal{P}^*$ is the one obtained by evaluating G on the outputs of $\mathcal{P}_1, \dots, \mathcal{P}_n$, and it is denoted as $\mathcal{P}^* = G(\mathcal{P}_1, \dots, \mathcal{P}_n)$. The labeled inputs of $\mathcal{P}^*$ are all the distinct labeled inputs of $\mathcal{P}_1, \dots, \mathcal{P}_n$ (all the inputs with the same label are grouped together and considered as a unique input of $\mathcal{P}^*$).

We recall the definitions of Fiore *et al.* [17] for multi-key homomorphic authenticators, adapted to the case of signature schemes only. Following [17], we consider labels where $\ell = (\mathsf{id}, \tau)$, such that id is a given client identity and τ is a tag which refers to the client's input data. To ease the reading, we use the compact and improper notation $\mathsf{id} \in \mathcal{P}$ meaning that there exists at least one index label ℓ in the description of $\mathcal{P} = (C, (\ell_1, \dots, \ell_\mathsf{n}))$ such that $\ell = (\mathsf{id}, \tau)$ for some string τ.

Definition 3.1 (Multi-Key Homomorphic Signature [17]). *A multi-key homomorphic signature scheme* MKHS *is a tuple of five PPT algorithms* MKHS = (MKHS.Setup, MKHS.KeyGen, MKHS.Sign, MKHS.Eval, MKHS.Verify) *that satisfies the properties of* authentication correctness, evaluation correctness, succinctness *and* security. *The algorithms are defined as follows:*

MKHS.Setup(1^λ). *The setup algorithm takes as input the security parameter λ and outputs some public parameters* pp *including a description of an identity space* ID*, a tag space $\mathcal{T}$ (these implicitly define the label space $\mathcal{L} = \mathsf{ID} \times \mathcal{T}$), a message space $\mathcal{M}$ and a set of admissible functions $\mathcal{F}$. The* pp *are input to all the following algorithms, even when not specified.*

MKHS.KeyGen(pp). *The key generation algorithm takes as input the public parameters and outputs a pair of keys* (sk, pk)*, where* sk *is a secret signing key, while* pk *is the public evaluation and verification key.*

MKHS.Sign(sk, Δ, ℓ, m). *The sign algorithm takes as input a secret key* sk*, a dataset identifier Δ, a label $\ell = (\mathsf{id}, \tau)$ for the message* m*, and it outputs a signature σ.*

MKHS.Eval($\mathcal{P}, \Delta, \{(\sigma_i, \mathsf{pk}_{\mathsf{id}_i})\}_{i \in [\mathsf{n}]}$). *The evaluation algorithm takes as input a labeled program $\mathcal{P} = (C, (\ell_1, \dots, \ell_\mathsf{n}))$, where $C : \mathcal{M}^\mathsf{n} \longrightarrow \mathcal{M}$ is an* n*-input circuit, a dataset identifier Δ and a set of signature and public-key pairs $\{(\sigma_i, \mathsf{pk}_{\mathsf{id}_i})\}_{i \in [\mathsf{n}]}$. The output is an homomorphic signature σ.*

$\mathsf{MKHS.Verify}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \mathsf{m}, \sigma)$. *The verification algorithm takes as input a labeled program $\mathcal{P} = (C, (\ell_1, \dots, \ell_\mathsf{n}))$, a dataset identifier Δ, the set of public keys $\{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}$ corresponding to those identities* id *involved in $\mathcal{P}$, a message* m *and an homomorphic signature σ. It outputs* 0 *(reject) or* 1 *(accept).*

Remark 3.1 (Single/Multi-Hop Evaluation). Similarly to fully homomorphic encryption, we call a (multi-key) homomorphic signature i-Hop if the Eval algorithm can be executed on its own outputs up to i times. We call *single-hop* a scheme where Eval can be executed only on fresh signatures, i.e., generated by Sign, whereas a multi-hop scheme is a scheme that is i-Hop for all i.

Authentication Correctness. A multi-key homomorphic signature satisfies authentication correctness if for all public parameters $\mathsf{pp} \leftarrow \mathsf{MKHS.Setup}(1^\lambda)$, any key pair $(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{MKHS.KeyGen}(\mathsf{pp})$, any dataset identifier Δ, any label $\ell = (\mathsf{id}, \tau) \in \mathcal{L}$, any message $\mathsf{m} \in \mathcal{M}$ and any signature $\sigma \leftarrow \mathsf{MKHS.Sign}(\mathsf{sk}, \Delta, \ell, \mathsf{m})$, it holds that

$$\Pr\left[\mathsf{MKHS.Verify}(\mathcal{I}_\ell, \Delta, \mathsf{pk}, \mathsf{m}, \sigma) = 1\right] \geq 1 - \mathsf{negl}\ .$$

Evaluation Correctness. A multi-key homomorphic signature satisfies evaluation correctness if

$$\Pr\left[\mathsf{MKHS.Verify}(\mathcal{P}', \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}'}, \mathsf{m}', \sigma') = 1\right] \geq 1 - \mathsf{negl}$$

where the equality holds for a fixed description of the public parameters $\mathsf{pp} \leftarrow \mathsf{MKHS.Setup}(1^\lambda)$, an arbitrary set of honestly generated keys $\{(\mathsf{sk_{id}}, \mathsf{pk_{id}})\}_{\mathsf{id}\in\tilde{\mathsf{ID}}}$ for some $\tilde{\mathsf{ID}} \subseteq \mathsf{ID}$, with $|\tilde{ID}| = t$, a dataset identifier Δ, a function $C : \mathcal{M}^\mathsf{n} \to \mathcal{M}$, and any set of program/message/signature triples $\{(\mathcal{P}_i, \mathsf{m}_i, \sigma_i)\}_{i\in[\mathsf{n}]}$ such that $\mathsf{MKHS.Verify}(\mathcal{P}_i, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}_i}, \mathsf{m}_i, \sigma_i) = 1$ for all $i \in [\mathsf{n}]$, and $\mathsf{m}' = g(\mathsf{m}_1, \dots, \mathsf{m_n})$, $\mathcal{P}' = g(\mathcal{P}_1, \dots, \mathcal{P}_\mathsf{n})$, and $\sigma' = \mathsf{Eval}(C, \{(\sigma_i, PK_i)\}_{i\in[n]})$ where $PK_i = \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}_i}$.

Succinctness. Succinctness is one of the crucial properties that make multi-key homomorphic signatures an interesting primitive. Intuitively, a MKHS scheme is succinct if the size of every signature depends only logarithmically on the size of a dataset. More formally, let $\mathsf{pp} \leftarrow \mathsf{MKHS.Setup}(1^\lambda)$, $\mathcal{P} = (C, (\ell_1, \dots, \ell_\mathsf{n}))$ with $\ell_i = (\mathsf{id}_i, \tau_i)$, $(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{MKHS.KeyGen}(\mathsf{pp})$ for all $\mathsf{id} \in [n]$. and $\sigma_i \leftarrow \mathsf{MKHS.Sign}(\mathsf{sk}_{\mathsf{id}_i}, \Delta, \ell_i, \mathsf{m}_i)$, for all $i \in [\mathsf{n}]$, then MKHS has succinct signatures if there exists a fixed polynomial $\mathsf{poly}(\cdot)$ such that $size(\sigma) = \mathsf{poly}(\lambda, t, \log \mathsf{n})$ where $\sigma = \mathsf{MKHS.Eval}(\mathcal{P}, \{(\sigma_i, \mathsf{pk}_{\mathsf{id}_i})\}_{i\in[n]})$.

Security. We adopt Fiore *et al.*'s security model [17]. Very intuitively, a multi-key homomorphic signature scheme is secure if the adversary, who can request to multiple users signatures on messages of its choice, can produce only signatures that are either the ones it received, or ones that are obtained by correctly executing the Eval algorithm. In addition, in the multi-key setting the adversary is also allowed to corrupt users but this shall not affect the integrity of computations performed on data signed by other (un-corrupted) users of the system. Formally, we define the $\mathsf{MK\text{-}HomUF\text{-}CMA}$ security experiment below

Setup. The challenger $\mathcal{C}$ runs $\mathsf{MKHS.Setup}(1^\lambda)$ and sends the public parameters pp to the adversary $\mathcal{A}$.

Sign Queries. The adversary can adaptively submit queries of the form $(\Delta, \ell, \mathsf{m})$, where Δ is a dataset identifier, $\ell = (\mathsf{id}, \tau)$ is a label in $\mathsf{ID} \times \mathcal{T}$ and $\mathsf{m} \in \mathcal{M}$ is a message. The challenger answers performing all the 1-4 checks below:

1. If (ℓ, m) is the first query for the dataset Δ, the challenger initializes an empty list $L_\Delta = \emptyset$.
2. If $(\Delta, \ell, \mathsf{m})$ is the first query with identity id, the challenger generates the keys for that identity: $(\mathsf{sk_{id}}, \mathsf{pk_{id}}) \leftarrow \mathsf{KeyGen(pp)}$. and proceeds to step 3.
3. If $(\Delta, \ell, \mathsf{m})$ is such that $(\ell, \mathsf{m}) \notin L_\Delta$, the challenger computes $\sigma \leftarrow \mathsf{MKHS.Sign}(\mathsf{sk_{id}}, \Delta, \ell, \mathsf{m})$ (this is possible since $\mathcal{C}$ has already generated the keys for the identity id). Then the challenger updates the list $L_\Delta \leftarrow L_\Delta \cup (\ell, \mathsf{m})$ and returns $(\sigma, \mathsf{pk_{id}})$ to $\mathcal{A}$.
4. If $(\Delta, \ell, \mathsf{m})$ is such that $(\ell, \cdot) \notin L_\Delta$, that is, the adversary had already made a query $(\Delta, \ell, \mathsf{m}')$ for some message m', the challenger ignores the query. Note that for a given (Δ, ℓ) pair only one message can be obtained.

Corruption Queries. At the beginning of the game, the challenger initialises an empty list $\mathsf{L_{corr}} = \emptyset$ of corrupted identities. During the game, the adversary can adaptively perform corruption queries by sending $\mathsf{id} \in \mathsf{ID}$ to the challenger. If $\mathsf{id} \notin \mathsf{L_{corr}}$ the challenger updates the list $\mathsf{L_{corr}} \leftarrow \mathsf{L_{corr}} \cup \mathsf{id}$ and answers the query with the pair $(\mathsf{sk_{id}}, \mathsf{pk_{id}})$ generated using KeyGen (if not done before). If $\mathsf{id} \in \mathsf{L_{corr}}$ the challenger replies with keys $(\mathsf{sk_{id}}, \mathsf{pk_{id}})$ assigned to id before.

Forgery. At the end of the game, $\mathcal{A}$ outputs a tuple $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$. The experiment outputs 1 if the tuple returned by $\mathcal{A}$ is a forgery (defined below), and 0 otherwise.

A MK-HS scheme MKHS is *unforgeable* if for every PPT adversary $\mathcal{A}$, its advantage $\mathrm{Adv}^{\mathsf{MKHS}}_{\mathcal{A}}(\lambda) = \Pr[\mathsf{MK\text{-}HomUF\text{-}CMA}_{\mathcal{A},\mathsf{MKHS}}(\lambda) = 1]$ is $\mathsf{negl}(\lambda)$.

Definition 3.2 (Forgery). *We consider an execution of* $\mathsf{MK\text{-}HomUF\text{-}CMA}$ *where* $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ *is the tuple returned by* $\mathcal{A}$ *at the end of the experiment. Let* $\mathcal{P}^* = (C^*, \ell^*_1, \ldots, \ell^*_{\mathsf{n}})$. *The adversary's output is said to be a successful forgery against the multi-key homomorphic signature scheme if:* $\mathsf{MKHS.Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*) = 1$ *and at least one of the following conditions hold:*

***Type-1 forgery**: the dataset* Δ^* *was never initialised.*

***Type-2 forgery**: for all* $\mathsf{id} \in \mathcal{P}^*$, $\mathsf{id} \notin \mathsf{L_{corr}}$ *and* $(\ell^*_i, \mathsf{m}_i) \in L_{\Delta^*}$ *for all* $i \in [\mathsf{n}]$, *but* $\mathsf{y}^* \neq C^*(\mathsf{m}_1, \ldots, \mathsf{m_n})$.

***Type-3 forgery**: there exists (at least) one index* $i \in [\mathsf{n}]$ *such that* ℓ^*_i *was never queried, i.e.,* $(\ell^*_i, \cdot) \notin L_{\Delta^*}$ *and* $\mathsf{id}_i \notin \mathsf{L_{corr}}$ *is a non-corrupted identity.*

Non-adaptive corruption queries. We also recall a proposition given in [17], which shows that it is sufficient to prove security for *non-adaptive* corruption queries. This is a setting where the adversary $\mathcal{A}$ can perform corruption queries only on identities for which no signature query had already been performed. This proposition can be used to simplify security proofs.

Proposition 3.1 ([17]). MKHS *is secure against adversaries that do not make corruption queries if and only if* MKHS *is secure against adversaries that make non-adaptive corruption queries.*

2.3 Homomorphic Signatures

Despite some minor syntactic modifications, homomorphic signatures can be seen as a special case of multi-key homomorphic signatures for algorithms that run on inputs by a single user only. For the purpose of this work, single-key homomorphic signature schemes are defined by five PPT algorithms $\mathsf{HS} = (\mathsf{HS.Setup}, \mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.Eval}, \mathsf{HS.Verify})$ that have the same input-output behavior as the corresponding algorithms in MKHS except:

- There is no identity space ID and the labels are simply $\ell = \tau$.

- The evaluation algorithm $\mathsf{HS.Eval}$ takes as input a circuit C, a single public key pk and a set of signatures $\sigma_1, \ldots, \sigma_{\mathsf{n}}$. In particular $\mathsf{HS.Eval}$ runs without labels or dataset identifier.

- The verification algorithm $\mathsf{HS.Verify}$ accepts inputs from a single user only, *i.e.,* the labeled program $\mathcal{P}$ is of the form $\mathcal{P} = (C, (\tau_1, \ldots, \tau_{\mathsf{n}}))$ and only one public key pk is provided.

The properties of authentication and evaluation correctness are analogous to the ones for MKHS in the case of computations on inputs by a single client. Regarding succinctness, a homomorphic signature scheme HS has *succinct signatures* if the size of any signature σ output by $\mathsf{HS.Eval}$ depends only logarithmic in the number n inputs to the labelled program, *i.e., size*$(\sigma) = \mathsf{poly}(\lambda, \log(\mathsf{n}))$.

Finally, we observe that the specialization to the single-key setting of the above security definition corresponds to the strong-adaptive security definition of HS that is formalized in [10]. In particular, the definitions in [10] allow for a simple treatment of Type-3 forgeries. In [10] it is also shown that HS constructions for circuits that are secure in this stronger model can be generically built, e.g., from [23].

3 The Matrioska compiler

In this section, we present Matrioska: a generic compiler from a single-key homomorphic signature scheme $\mathsf{HS} = (\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.Eval}, \mathsf{HS.Verify})$ to a (single-hop) multi-key scheme $\mathsf{MKHS} = (\mathsf{MKHS.KeyGen}, \mathsf{MKHS.Sign}, \mathsf{MKHS.Eval}, \mathsf{MKHS.Verify})$. The result is summarized in the following theorem:

Theorem 3.2. *Let* HS *be a homomorphic signature scheme that is correct and unforgeable. Then, for any given integer number* $\mathsf{T} \geq 1$ *there exists a multi-key homomorphic signature scheme* $\mathsf{MKHS}(\mathsf{HS}, \mathsf{T})$ *that supports computations on signatures generated using at most* T *distinct keys, it is correct and unforgeable. Furthermore, if* HS *supports circuits of maximum size s and maximum depth d and it has succinctness l, then* $\mathsf{MKHS}(\mathsf{HS}, \mathsf{T})$ *on* T *distinct users has succinctness* $\mathsf{T} \cdot l$*, and can support circuits of size* s' *and depth* d' *provided that* $s > (s')^{{c_s}^{\mathsf{T}-1}}$ *and* $d > \max\{d', d_{\mathsf{HSV}}((s')^{{c_s}^{\mathsf{T}-1}}, \lambda)\}$*, where* d_{HSV} *and* c_s *are a function and a non-negative constant that depend from the single-key scheme* HS*.*

More precisely, d_{HSV} expresses the depth of the circuit for the verification algorithm $\mathsf{HS.Verify}$ as a function of its input length (which includes the description of the labeled program $\mathcal{P}$); c_s is a constant such that the size of $\mathsf{HS.Verify}$ on input a circuit C is $size(C)^{c_s}$. Notice that by efficiency of HS such c_s exists, and d_{HSV} can, in the worst case, be written as $size(C)^{c_d}$ for some other constant c_d.

Theorem 3.2 can be instantiated in two ways. If HS is a fully-homomorphic signature (whose existence is not yet known), then for any $s' = \mathsf{poly}(\lambda)$ and for any constant number T, we are guaranteed that HS is executed on poly-sized circuits. Otherwise, if HS is an HS for circuits of bounded polynomial depth (and of any, or bounded, polynomial size), as *e.g.,* [23], then for any $s' = \mathsf{poly}(\lambda)$ and for any fixed number of keys T, we can derive a polynomial bound d on the depth. The proof of Theorem 3.2 is constructive. First we show a method to define MKHS given a HS scheme and a value T. Next, in a sequence of lemmas, we prove all the properties stated in the theorem.

Our construction is rather involved. Therefore, in the next section we first illustrate our ideas for a simple case of a computation that takes inputs from three different users, and then, in Section 3.2, we describe the full compiler.

3.1 An intuition: the three-client case

We provide here a simplified example to explain the core idea of our $\mathsf{Matrioska}$ compiler. To ease the exposition we consider the case $t = 3$ (three clients with identities $\mathsf{id}_1, \mathsf{id}_2$ and id_3) and deliberately remove dataset identifiers. A detailed description for $t = \mathsf{n} = 3$ can be found in the full version of this paper [18].

Let $\mathcal{P} = (C, (\ell_1, \dots, \ell_\mathsf{n}))$ be a labelled program, where C a (n)-input circuit (with $\mathsf{n} = \mathsf{n}_1 + \mathsf{n}_2 + \mathsf{n}_3$) and the labels $\ell_i = (\mathsf{id}_i, \tau_i)$ are ordered, *i.e.,* first n_1 inputs belong to client id_1, the subsequent n_2 to id_2 and the last n_3 inputs to id_3. Let σ_i be the signature on message m_i for the label ℓ_i. For simplicity assume that $C(\mathsf{m}_1, \dots, \mathsf{m}_\mathsf{n}) = \mathsf{y} = 1$.

Step 1. We want extract from C a circuit that contains only inputs by clients id_2 and id_3. To this end, we define E_1 as the partial evaluation of C on the messages $\mathsf{m}_{\mathsf{n}_1+1}, \dots, \mathsf{m}_\mathsf{n}$. Thus, E_1 is an n_1-input circuit with hardwired in it the inputs by clients id_2 and id_3. In our framework E_1 is obtained with two basic operations on the bit string C: (1) setting any gate g with left or right input wire in $[\mathsf{n}] \setminus [\mathsf{n}_1]$ to be a constant gate (*i.e.,* setting the bits $\mathsf{L}(g)$ and $\mathsf{R}(g)$ to 0), and (2) initializing the now constant gate to the value m_i for $i \in [\mathsf{n}] \setminus [\mathsf{n}_1]$. At this point we obtained a circuit with inputs of a single client only, and we can run $\hat{\sigma}_1 \leftarrow \mathsf{HS.Eval}(E_1, \mathsf{pk}_{\mathsf{id}_1}, \sigma_1, \dots, \sigma_{\mathsf{n}_1})$. By construction $E_1(\mathsf{m}_1, \dots, \mathsf{m}_{\mathsf{n}_1}) = C(\mathsf{m}_1, \dots, \mathsf{m}_\mathsf{n}) = 1$, therefore $\mathsf{HS.Verify}((E_1, (\tau_1, \dots, \tau_{\mathsf{n}_1})), \mathsf{pk}_{\mathsf{id}_1}, \hat{\sigma}_1, 1) = 1$.

Step 2. The actual inductive procedure begins now. We wish to verify the correctness of $\hat{\sigma}_1$ using the messages input by client id_2 as variables. Consider the input to the (single-client) verification as the string $S_1 = ((E_1, (\tau_1, \dots, \tau_{\mathsf{n}_1})), \mathsf{pk}_{\mathsf{id}_1}, \hat{\sigma}_1, 1)$. Recall that to construct the circuit E_1 we used the messages $\mathsf{m}_{\mathsf{n}_1+1}, \dots \mathsf{m}_\mathsf{n}$ (hard-wired in its gate description). To free the inputs by client id_2 we modify S_1 in the following way: (1) identify the gates that contain the messages $\mathsf{m}_{\mathsf{n}_1+1}, \dots, \mathsf{m}_{\mathsf{n}_1+\mathsf{n}_2}$, (2) turn these gates into input gates by setting the left/right wires to the opportune values w (using $\mathcal{P}$). Let us consider $\mathsf{HS.Verify}$ on the modified string S_1, this is a proper circuit E_2 such that $E_2(\mathsf{m}_{\mathsf{n}_1+1}, \dots, \mathsf{m}_{\mathsf{n}_1+\mathsf{n}_2}) = \mathsf{HS.Verify}((E_1, (\tau_1, \dots, \tau_{\mathsf{n}_1})), \mathsf{pk}_{\mathsf{id}_1}, \hat{\sigma}_1, 1) = 1$. Being E_2 a single-client circuit we can run $\hat{\sigma}_2 \leftarrow \mathsf{HS.Eval}(E_2, \mathsf{pk}_{\mathsf{id}_2}, \sigma_{\mathsf{n}_1+1}, \dots, \sigma_{\mathsf{n}_1+\mathsf{n}_2})$.

Step 3. This is analogous to **Step 2**: we wish to verify the correctness of $\hat{\sigma}_2$ using the messages input by client id_3 as variables and define a circuit that is completely determined by public values, no hard-wired message value. Let $S_2 = ((E_2, (\tau_{\mathsf{n}_1+1}, \dots, \tau_{\mathsf{n}_1+\mathsf{n}_2})), \mathsf{pk}_{\mathsf{id}_2}, \hat{\sigma}_2, 1)$, we free the inputs by client id_3 as in **Step 2**. We define E_3 as the formal evaluation of $\mathsf{HS.Verify}$ on the modified string S_2. By construction it holds that $E_3(\mathsf{m}_{\mathsf{n}_1+\mathsf{n}_2+1}, \dots, \mathsf{m}_\mathsf{n}) = \mathsf{HS.Verify}((E_2, (\tau_{\mathsf{n}_1+1}, \dots, \tau_{\mathsf{n}_1+\mathsf{n}_2})), \mathsf{pk}_{\mathsf{id}_2}, \hat{\sigma}_2, 1) = 1$, and we can run $\hat{\sigma}_3 \leftarrow \mathsf{HS.Eval}(E_3, \mathsf{pk}_{\mathsf{id}_3}, \sigma_{\mathsf{n}_1+\mathsf{n}_2+1}, \dots, \sigma_\mathsf{n})$.

The multi-key homomorphic evaluation algorithm outputs $\hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3)$. The $\mathsf{Matrioska}$ verification procedure needs only reconstruct the final circuit E_3, as this is fully determined by the public values $(\mathcal{P}, \mathsf{pk}_{\mathsf{id}_1}, \mathsf{pk}_{\mathsf{id}_2}, \hat{\sigma}_1, \hat{\sigma}_2, \mathsf{HS.Verify}, 1)$. Let $\mathcal{E}_3 = (E_3, (\tau_{\mathsf{n}_1+\mathsf{n}_2+1}, \ldots, \tau_{\mathsf{n}}))$, the verification concludes by running the single-key verification algorithm: $\mathsf{HS.Verify}(\mathcal{E}_3, \mathsf{pk}_3, \hat{\sigma}_3, 1)$.

3.2 The Matrioska Compiler

In this section we describe our compiler in the general case of computing on signatures generated by t different keys.

Definition 3.3 (Matrioska)**.** *Let* $\mathsf{HS} = (\mathsf{HS.Setup}, \mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.Eval}, \mathsf{HS.Verify})$ *be a single-key homomorphic signature scheme, we define a multi-key homomorphic signature scheme* MKHS *as follows:*

$\mathsf{MKHS.Setup}(1^\lambda, \mathsf{T}, s', d') \to \mathsf{pp}$. *The set-up algorithm takes as input the security parameter λ, a positive integer T that represents a bound for the maximal number of distinct identities involved in the same homomorphic computation, and bounds $s', d' = \mathsf{poly}(\lambda)$ on the size and depth respectively of the circuits used in the* $\mathsf{MKHS.Eval}$ *and* $\mathsf{MKHS.Verify}$ *algorithms. Setup first uses T, s', d' to derive two integers s and d such that $s > (s')^{c_s^{\mathsf{T}-1}}$ and $d > \max\{d', d_{\mathsf{HSV}}((s')^{c_s^{\mathsf{T}-1}}, \lambda)\}$. Next, it runs $\mathsf{HS.Setup}(1^\lambda, s, d)$ to obtain a tag space $\mathcal{T}$ (which corresponds to the label space of* HS*), a message space $\mathcal{M}$ and a set of admissible circuits $\mathcal{F}$.*[14] *Labels of the multi-key scheme are defined as pairs $\ell = (\mathsf{id}, \tau) \in \mathsf{ID} \times \mathcal{T}$, where the first entry is a client-identity identifier. Labeled programs are of the form $\mathcal{P} = (C, (\ell_1, ..., \ell_t))$ with labels as above.*

$\mathsf{MKHS.KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$. *The key-generation algorithm runs* $\mathsf{HS.KeyGen}$ *to obtain a public-secret key pair. This key-pair will be associated to an identity $\mathsf{id} \in \mathsf{ID}$. When we need to distinguish among clients we make the dependency on the identity explicit, e.g., $(\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$.*

$\mathsf{MKHS.Sign}(\mathsf{sk}, \Delta, \ell, \mathsf{m}) \to \sigma$. *This algorithm takes as input a secret key sk, a data set identifier Δ (e.g., a string), a label $\ell = (\mathsf{id}, \tau)$ for the message m. It outputs*

$$\sigma \leftarrow \mathsf{HS.Sign}(\mathsf{sk}_{\mathsf{id}}, \Delta, \tau, \mathsf{m}). \tag{8}$$

Without loss of generality we assume that σ includes m.

$\mathsf{MKHS.Eval}(\mathcal{P}, \Delta, \{(\vec{\sigma}_i, \mathsf{pk}_{\mathsf{id}_i})\}_{i\in[t]}) \to \vec{\hat{\sigma}}$. *Let $\mathcal{P} = (C, (\ell_1, \ldots, \ell_{\mathsf{n}}))$, where $C = (\mathsf{n}, 1, \mathsf{q}, \mathsf{L}, \mathsf{R}, \mathsf{G})$ and the $\mathsf{n} \geq t$ labels are of the form $\ell_j = (\mathsf{id}_i, \tau_j)$ for some $i \in [t]$ and $\tau_j \in \mathcal{T}$, where $t \leq \mathsf{T}$.*

$\boxed{\text{The case } t = 1}$ *In this case all the n signatures belong to the same user, that is to say, there exists an identity $\mathsf{id} \in \mathsf{ID}$ such that for all $j \in [\mathsf{n}]$ the labels are of the form $\ell = (\mathsf{id}, \tau_j)$ for some $\tau_j \in \mathcal{T}$. Thus, it is possible to run the classical evaluation algorithm of* HS *and the output of the multi-key evaluation algorithm for $t = 1$ is:*

$$\vec{\hat{\sigma}} = \hat{\sigma}_{\mathsf{id}} \leftarrow \mathsf{HS.Eval}\big(E_0, \mathsf{pk}_{\mathsf{id}}, (\sigma_1^{\mathsf{id}}, \ldots, \sigma_{\mathsf{n}}^{\mathsf{id}})\big). \tag{9}$$

$\boxed{\text{The case } t \geq 2}$ *In this case the inputs to the labeled program belong to t distinct users. Without loss of generality, we assume that the labels are ordered per client*

[14]If HS works without these a-priori bounds, it is enough to run $\mathsf{HS.Setup}(1^\lambda)$.

identity, i.e., all the labels between ℓ_{t_j} and $\ell_{\mathsf{t}_{j+1}-1}$ are of the form $(\mathsf{id}_j, *)$. For each $i \in [t]$ the signature vector $\vec{\sigma}_i$ is $\vec{\sigma}_i = (\sigma^i_1, \ldots, \sigma^i_{\mathsf{n}_i})$ for opportune values $\mathsf{n}_i \in [\mathsf{n}-t+1]$ satisfying $\sum_{i=1}^{t} \mathsf{n}_i = \mathsf{n}$. Let $\mathsf{t}_i = (\sum_{j=0}^{i-1} \mathsf{n}_j) + 1$, where we set $\mathsf{n}_0 = 0$, then t_i corresponds to the index of first input of identity id_i. The multi-key homomorphic evaluation performs the following $t+1$ steps.

Step 0. *Given $\mathcal{P} = (C, (\ell_1, \ldots, \ell_{\mathsf{n}}))$ retrieve the messages corresponding to the labels $\ell_1, \ldots, \ell_{\mathsf{n}}$. For notation sake let m_j be the message corresponding to label ℓ_j. Compute the value $\mathsf{y} = C(\mathsf{m}_1, \ldots, \mathsf{m}_{\mathsf{n}})$. Define a single-input single-output circuit $EQ^{\mathsf{y}}(x)$ that outputs 1 if and only if $x = \mathsf{y}$.Construct $E_0 = C \triangleright EQ^{\mathsf{y}} = (\mathsf{n}, 1, \mathsf{q}_0, \mathsf{L}_0, \mathsf{R}_0, \mathsf{G}_0)$. The properties of EQ^{y} imply that:*

$$E_0(x_1, \ldots, x_{\mathsf{n}}) = 1 \text{ iff } C(x_1, \ldots, x_{\mathsf{n}}) = \mathsf{y}\ . \tag{10}$$

Note that E_0 can be constructed directly from C and y, moreover

$$E_0(\mathsf{m}_1, \ldots, \mathsf{m}_{\mathsf{n}}) = 1. \tag{11}$$

Step 1. *We build a n_1-input circuit E_1 that corresponds to a partial evaluation of E_0 on the inputs of identities id_j with $j > 1$. Given $\mathcal{E}_0 = (E_0, (\ell_1, \ldots, \ell_{\mathsf{n}}))$, the signatures $\vec{\sigma_1} = (\sigma^1_1, \ldots, \sigma^1_{\mathsf{n}_1})$ and the messages $\mathsf{m}_{\mathsf{n}_1+1}, \ldots, \mathsf{m}_{\mathsf{n}}$ do:*

- *Define the mask circuit $M_1 = (\mathsf{n}_1, \mathsf{n}, \mathsf{n}, \mathsf{L}'_1, \mathsf{R}'_1, \mathsf{G}'_1)$ where*

$$\mathsf{L}'_1(j) = \mathsf{R}'_1(j) = \begin{cases} 1 & \text{for } j \in [\mathsf{n}_1] \\ 0 & \text{for } j \in [\mathsf{n}] \setminus [\mathsf{n}_1] \end{cases} \quad \text{and } \mathsf{G}'_1 = \begin{cases} 0 & \text{for } j \in [\mathsf{n}_1] \\ \mathsf{m}_j & \text{for } j \in [\mathsf{n}] \setminus [\mathsf{n}_1] \end{cases} .$$

By construction $M_1(b_1, \ldots, b_{\mathsf{n}_1}) = (b_1, \ldots b_{\mathsf{n}_1}, \mathsf{m}_{\mathsf{n}_1+1}, \ldots, \mathsf{m}_{\mathsf{n}})$.

- *Compose M_1 with E_0 to obtain $E_1 = M_1 \triangleright E_0 = (\mathsf{n}_1, 1, \mathsf{q}_1, \mathsf{L}_1, \mathsf{R}_1, \mathsf{G}_1)$ where: $\mathsf{q}_1 = \mathsf{q}_0 + \mathsf{n}$; $\mathsf{G}_1 = (\mathsf{G}'_1 || \mathsf{G}_0)$; $\mathsf{L}_1(g) = \mathsf{L}'_1(g)$ for $g \in [\mathsf{n}]$, $\mathsf{L}_1(g) = (\mathsf{L}_0(g - \mathsf{n} + 1) + 1)$ for $g \in [\mathsf{n}+1, \mathsf{n}+\mathsf{q}_0]$ if $\mathsf{L}_0(g - \mathsf{n} + 1) \neq 0$ and 0 whenever $\mathsf{L}_0(g - \mathsf{n} + 1) = 0$. The function $\mathsf{R}_1(g)$ is defined analogously. Equation (11) implies*

$$E_1(\mathsf{m}_1, \ldots, \mathsf{m}_{\mathsf{n}_1}) = 1. \tag{12}$$

- *Compute $\hat{\sigma}_1 \leftarrow \mathsf{HS.Eval}(E_1, \mathsf{pk}_{\mathsf{id}_1}, \vec{\sigma}_1)$. This is possible since E_1 is a circuit involving only inputs of client id_1.*

Remark 3.2. Let $\mathcal{E}_1 = (E_1, (\tau_1, \ldots, \tau_{\mathsf{n}_1}))$. Equation (12) and the correctness of the HS scheme imply $\mathsf{HS.Verify}(\mathcal{E}_1, \Delta, \mathsf{pk}_{\mathsf{id}_1}, \hat{\sigma}_1, 1) = 1$.

Step i for *$i \in [2, t]$. The goal is to construct an n_i-input circuit E_i using $\mathcal{E}_{i-1} = (E_{i-1}, (\tau_{\mathsf{t}_i}, \ldots, \tau_{\mathsf{t}_{i+1}-1}))$, Δ, $\mathsf{pk}_{\mathsf{id}_i}$ and $\vec{\sigma_i} = (\sigma^i_1, \ldots \sigma^i_{\mathsf{n}_i})$. This will be possible using the circuits $\mathsf{HSV}_i = (\mathsf{n}_{\mathsf{HSV}_i}, 1, \mathsf{q}_{\mathsf{HSV}_i}, \mathsf{L}_{\mathsf{HSV}_i}, \mathsf{R}_{\mathsf{HSV}_i}, \mathsf{G}_{\mathsf{HSV}_i})$ for the (single-key) homomorphic signature verification against the value 1 .*[15]
Let $S_{i-1} = (\mathcal{E}_{i-1}, \Delta, \mathsf{pk}_{\mathsf{id}_{i-1}}, \vec{\sigma}_{i-1})$ be a string of $\mathsf{n}_{\mathsf{HSV}_i} = size(S_{i-1})$ bits. Set $g_1 = 1$. The gates of E_{i-1} that embed the n_i values input by identity id_i are located in the interval $I_i = [g_i, g_i + \mathsf{n}_i]$, where $g_i = 3\lg(N_{i-1}) + 2\mathsf{q}_{i-1}\lg(\mathsf{w}_{i-1}) + g_{i-1} + \mathsf{n}_{i-1}$ (see [18] for an explanation).

[15]The readers can consider the circuit HSV_i to be the representation of $\mathsf{HS.Verify}(\mathcal{E}_{i-1}, \cdot, \cdot, 1)$ where $\mathcal{E}_{i-1}$ is a labelled program for a circuit of size at most $O((\mathsf{n}_{\mathsf{HSV}_{i-1}} + \mathsf{q}_{\mathsf{HSV}_{i-1}})\lg(\mathsf{w}_{\mathsf{HSV}_{i-1}}))$.

- *Define the mask circuit* $M_i = (\mathsf{n}_i, \mathsf{n}_{\mathsf{HSV}i}, \mathsf{n}_{\mathsf{HSV}i}, \mathsf{L}'_i, \mathsf{R}'_i, \mathsf{G}'_i)$ *where*

$$\mathsf{L}'_i(g) = \mathsf{R}'_i(g) = \begin{cases} 0 & \text{if } g \in [\mathsf{n}_{\mathsf{HSV}i}] \setminus I_i \\ 1 & \text{if } g \in I_i \end{cases}$$

and

$$\mathsf{G}'_i(g) = \begin{cases} S_{i-1}(g) & \text{if } g \in [\mathsf{n}_{\mathsf{HSV}3}] \setminus I_i \\ 0 & \text{if } g \in I_i \end{cases}$$

Note that for gates g in the interval I_i, $\mathsf{L}'_i(g) = 1$ and $\mathsf{G}'_i(g) = 0$ which means that M_i outputs its n_i input bits exactly the interval I_i, while outside I_i the output of M_i is constant. In particular: $M_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) = S_{i-1}$.

- *Compose M_i with HSV_i to obtain $E_i = M_i \triangleright \mathsf{HSV}_i = (\mathsf{n}_i, 1, \mathsf{q}_i, \mathsf{L}_i, \mathsf{R}_i, \mathsf{G}_i)$ where: $\mathsf{q}_i = \mathsf{n}_{\mathsf{HSV}i} + \mathsf{q}_{\mathsf{HSV}_i}$; $\mathsf{G}_i = (\mathsf{G}'_i || \mathsf{G}_{\mathsf{HSV}_i})$; $\mathsf{L}_i(g) = \mathsf{L}'_i(g)$ for $g \in [\mathsf{n}_{\mathsf{HSV}i}]$, $\mathsf{L}_i(g) = \mathsf{L}_{\mathsf{HSV}_i}(g - \mathsf{n}_{\mathsf{HSV}i} + 1) + \mathsf{n}_i$ for $g \in [\mathsf{n}_{\mathsf{HSV}i} + 1, \mathsf{q}_i]$ if $\mathsf{L}_{\mathsf{HSV}_i}(g - \mathsf{n}_{\mathsf{HSV}i} + 1) \neq 0$, and 0 otherwise; and R_i is defined analogously. . Circuit composition ensures that*[16] *$E_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) = \mathsf{HS.Verify}(\mathcal{E}_{i-1}, \Delta, \mathsf{pk}_{\mathsf{id}_{i-1}}, \hat{\sigma}_{i-1}, 1)$. In particular, applying Remark 3.2 inductively we get:*

$$E_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) = 1 \tag{13}$$

Note that E_i can be constructed directly from $\mathcal{E}_0$ given the values $\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{n}}$ and the public data Δ, $\mathsf{pk}_{\mathsf{id}_j}, \hat{\sigma}_j$ for $j \in [i-1]$. In more details, for $i \in [2, t]$ consider the set of bit strings: $\mathsf{head}_i = (\mathsf{n}_i, 1, \mathsf{q}_i, \mathsf{L}_i, \mathsf{R}_i)$ and $\mathsf{tail}_i = (\tau_{\mathsf{t}_i}, \ldots, \tau_{\mathsf{t}_i+\mathsf{n}_i}, \Delta, \mathsf{pk}_{\mathsf{id}_{i-1}}, \hat{\sigma}_{i-1}, \mathsf{G}_{\mathsf{HSV}_i})$. *For every $i \in [2, t]$ head_i and tail_i are completely determined by the tags for identity id_{i-1}, the public key $\mathsf{pk}_{\mathsf{id}_{i-1}}$ and the evaluated signature $\hat{\sigma}_{i-1}$. It is immediate to see that head_i and tail_i are respectively the head and the tail of the circuit description of E_i. The heart of the string E_i is where "all the magic" happens:*

$$\mathsf{body}_i = (\mathsf{head}_{i-1}, \ldots, \mathsf{head}_2, \underbrace{0, \ldots, 0}_{(\mathsf{t}_{i+1}-1) = \sum_{j=1}^{i} \mathsf{n}_j} \mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{n}}, \mathsf{G}_0, \mathsf{tail}_2, \ldots, \mathsf{tail}_i) \tag{14}$$

In particular, for $i = t$ we have:

$$\begin{aligned} E_t &= \Big(\mathsf{head}_t \qquad \mathsf{body}_t \qquad \mathsf{tail}_t \Big) \\ &= \Big(\mathsf{head}_t, (\mathsf{head}_{t-1}, \ldots, \mathsf{head}_2, \underbrace{0, \ldots, 0}_{\mathsf{n}}, \mathsf{G}_0, \mathsf{tail}_2, \ldots, \mathsf{tail}_{t-1}), \mathsf{tail}_t \Big) \end{aligned} \tag{15}$$

Equation (15) shows that the circuit E_t is completely determined by the labeled program $\mathcal{E}_0$ (to get the tags and the gate description G_0), the dataset identifier Δ, the public keys $\mathsf{pk}_{\mathsf{id}_i}$ and the signatures $\hat{\sigma}_i$ for $i \in [t]$.

- *Compute $\hat{\sigma}_i \leftarrow \mathsf{HS.Eval}(E_i, \mathsf{pk}_{\mathsf{id}_i}, \vec{\sigma}_i)$.*

Remark 3.3. This is possible since E_i is a n_i-input circuit with inputs from the user id_i only. Equation (13) and the correctness of the HS scheme imply that

$$\mathsf{HS.Verify}(\mathcal{E}_i, \Delta, \mathsf{pk}_i, 1, \hat{\sigma}_i) = 1. \tag{16}$$

The output of the multi-key evaluation algorithm is the vector of t signatures: $\vec{\hat{\sigma}} = (\hat{\sigma}_1, \ldots, \hat{\sigma}_t)$.

[16]With abuse of notation one can think that $E_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) = M_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) \triangleright \mathsf{HSV}_i = \mathsf{HSV}_i(M_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}))$. Since $M_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) = S_{i-1}$ the claim follows by the definition of HSV_i.

$\mathsf{MKHS.Verify}(\mathcal{P}, \Delta, \{\mathsf{pk_{id}}\}_{\mathsf{id}\in\mathcal{P}}, \mathsf{y}, \vec{\hat{\sigma}}) \to \{0,1\}$. *The verification algorithm parses the labeled program as* $\mathcal{P} = (C, (\ell_1 \ldots, \ell_{\mathsf{n}}))$ *and checks the number* $1 \leq t \leq \mathsf{T}$ *of distinct identities present among the* n *labels.*

$\boxed{\textit{The case } t = 1}$ *In this case all the inputs to the labeled program* $\mathcal{P}$ *come from the same user and* $\vec{\hat{\sigma}} = \hat{\sigma}_{\mathsf{id}}$. *In other words, all the labels are of the form* $\ell_j = (\mathsf{id}, \tau_j)$ *for an* $\mathsf{id} \in \mathsf{ID}$ *and some* $\tau_j \in \mathcal{T}$. *Set* $\mathcal{E}_0 = (C, (\tau_1, \ldots, \tau_{\mathsf{n}}))$, *notice that we removed the identity from the labels. The multi-key verification returns the output of*

$$\mathsf{HS.Verify}(\mathcal{E}_0, \Delta, \mathsf{pk_{id}}, 1, \hat{\sigma}_{\mathsf{id}}). \tag{17}$$

$\boxed{\textit{The case } t \geq 2}$ *In this case the labeled program* $\mathcal{P}$ *contains labels with* $t \geq 2$ *distinct identities and* $\vec{\hat{\sigma}} = (\hat{\sigma}_1, \ldots, \hat{\sigma}_t)$. *Without loss of generality, we assume that the labels are ordered per client identity and* $\mathsf{n}_i \in [\mathsf{n} - t + 1]$ *is the number of labels with identity* id_i.
Define $E_0 = (\mathsf{n}, 1, \mathsf{q}_0, \mathsf{L}_0\mathsf{R}_0, \mathsf{G}_0)$ *as the circuit* $E_0 = C \triangleright EQ^{\mathsf{y}}$, *where* $EQ^{\mathsf{y}}(x)$ *is the a single-input single-output circuit that outputs 1 if and only if* $x = \mathsf{y}$. *Thus,* $E_0(x_1, \ldots, x_{\mathsf{n}}) = 1$ *whenever* $C(x_1, \ldots, x_{\mathsf{n}}) = \mathsf{y}$. *As noted in the Step 0 of the evaluation algorithm,* E_0 *is completely determined by* $\mathcal{P}$ *and* y.

To verify the signature $\vec{\hat{\sigma}}$, *the multi-key verification algorithm inductively creates the following strings for* $i \in [2, t]$:

$$\begin{aligned}
\mathsf{head}_i &= (\mathsf{n}_i, 1, \mathsf{q}_i = \mathsf{n}_{\mathsf{HSV}\,i} + \mathsf{q}_{\mathsf{HSV}_i}, \mathsf{L}_i = (\underbrace{0, \ldots, 0}_{(\sum_{j=1}^{i-1}\mathsf{n}_j)-bits}, \overbrace{1, \ldots, 1}^{\mathsf{n}_i-bits}, \underbrace{0, \ldots, 0}_{(\mathsf{n}-\sum_{j=1}^{i}\mathsf{n}_j)-bits}), \mathsf{R}_i = \mathsf{L}_i) \\
\mathsf{tail}_i &= (\tau_{\mathsf{t}_{i-1}}, \ldots, \tau_{\mathsf{t}_{i-1}+\mathsf{n}_{i-1}}, \Delta, \mathsf{pk}_{\mathsf{id}_{i-1}}, \hat{\sigma}_{i-1}, \mathsf{G}_{\mathsf{HSV}_i})
\end{aligned}$$

where, the circuit HSV_i *is the same as the one explained in* $\mathsf{MKHS.Eval}$, *i.e., the* HSV_i *is the (single-key) homomorphic signature verification against the value 1. At this point the verifier can combine all the pieces to (re)-construct the description of the circuit* E_t:

$$E_t \quad = \quad (\mathsf{head}_t, \ldots, \mathsf{head}_2, \underbrace{0, \ldots, 0}_{\mathsf{n}}, \mathsf{G}_0, \mathsf{tail}_2, \ldots, \mathsf{tail}_t). \tag{18}$$

Let $\mathcal{E}_t = (E_t, (\tau_{\mathsf{t}_t}, \ldots, \tau_{\mathsf{n}}))$, *where we removed* id_t *from the labels. The verification returns:*

$$\mathsf{HS.Verify}(\mathcal{E}_t, \Delta, \mathsf{pk}_{\mathsf{id}_t}, \hat{\sigma}_t, 1). \tag{19}$$

Remark 3.4. Note that the E_t constructed by the verifier via Equation (18) coincides with the one created by the evaluator via Equation (15).

3.3 Correctness and Succinctness of Matrioska

In what follows we show that the Matrioska scheme satisfies the properties stated in Theorem 3.2.

Succinctness. By construction, for a computation involving messages from t users, our signatures consist of t signatures of the single-input scheme. It is straightforward to see that if HS signatures have length bounded by some polynomial l, the size of Matrioska's signatures is $\leq t \cdot l$, which is, asymptotically, the same level of succinctness as the MK-HS construction by Fiore *et al.* [17].

Correctness. The following two lemmas reduce the authentication and evaluation correctness of Matrioska multi-key homomorphic signatures to the authentication and evaluation correctness, respectively, of the underlying single-key HS scheme.

Lemma 3.1. *Let* HS *be a single-key homomorphic signature scheme with authentication correctness, then the multi-key homomorphic signature scheme* $\mathsf{MKHS}(\mathsf{HS},\mathsf{T})$ *obtained from the* Matrioska *compiler of Definition 3.3 achieves authentication correctness.*

The proof is quite straightforward and uses the labeled identity program $\mathcal{I}_\ell = (C_{\mathsf{id}}, \ell)$. For details check [18].

Lemma 3.2. *Let* HS *be a single-key homomorphic signature scheme with evaluation correctness, then the multi-key homomorphic signature scheme* $\mathsf{MKHS}(\mathsf{HS},\mathsf{T})$ *obtained from the* Matrioska *compiler of Definition 3.3 achieves evaluation correctness.*

The evaluation correctness of Matrioska essentially follows from the evaluation correctness of HS and the way we (inductively) define the circuits E_i. Moreover, notice that our MK-HS scheme is single-hop, therefore we have to prove evaluation correctness with respect to computing on freshly generated signatures (given that authentication correctness is granted by the previous lemma). For a detailed proof check [18].

Circuit Growth. In what follows we analyze the size growth of the circuits E_i computed by the Matrioska compiler, and use this to prove the bounds in Theorem 3.2.

Lemma 3.3. *Let* HS *be a correct single-key homomorphic signature scheme that supports computations on circuits of (maximum) depth d and size s; then the multi-key homomorphic signature scheme* $\mathsf{MKHS}(\mathsf{HS},\mathsf{T})$ *obtained from the* Matrioska *compiler of Definition 3.3 supports homomorphic computations on circuits of size s' and depth d' provided that $s > (s')^{{c_s}^{\mathsf{T}-1}}$ and $d > \max\{d', d_{\mathsf{HSV}}((s')^{{c_s}^{\mathsf{T}-1}}, \lambda)\}$, where d_{HSV} and c_s are a function and a non-negative constant that depend on the single-key scheme* HS.

Intuitively, for $t = 1$, MKHS is running the plain algorithms of HS. and thus MKHS supports circuits of size $s' < s$ and depth $d' < \max\{d, d_{\mathsf{HSV}}(s)\}$. For $t > 1$ the Matrioska compiler runs HS.Eval and HS.Verify on every E_i including E_t. Since $\{E_i\}_{i\in[t]}$ is a sequence of circuits of increasing size and depth we need to make sure that the circuit given as input to MKHS will grow into an E_t that is supported by HS. The details can be found in [18].

3.4 Security of Matrioska

In this section we argue that Matrioska MKHS schemes are unforgeable provided that so is the underlying HS scheme. For the proof we rely on Proposition 3.1 from [17], which allows for a simpler treating of corruption queries. Due to space limit, the detailed proof appears in the full version of this paper [18] while below we give a proof sketch with the main intuition.

Lemma 3.4. *Let* HS *be a secure single-key homomorphic signature scheme. Then the multi-key homomorphic signature scheme* $\mathsf{MKHS}(\mathsf{HS},\mathsf{T})$ *obtained from the* Matrioska *compiler of Definition 3.3 is secure. In particular, for any PPT adversary $\mathcal{A}$ making signing queries on at most $Q_{\mathsf{id}} = \mathsf{poly}(\lambda)$ distinct identities, there is a PPT algorithm $\mathcal{B}$ such that:* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{MKHS}} \leq Q_{\mathsf{id}} \cdot \mathrm{Adv}_{\mathcal{B}}^{\mathsf{HS}}$.

Proof sketch. The idea is that a forger against our MKHS scheme must create a forgery for the HS scheme for at least one of the users, say $\mathsf{id}_{i^\star}$, involved in the computation. Thus the reduction $\mathcal{B}$, on input a public key pk, makes a guess for $j^* = i^\star$, programs $\mathsf{pk}_{\mathsf{id}_{j^*}} = \mathsf{pk}$ and generates all the other keys. This allows $\mathcal{B}$ to perfectly simulate all the signing queries (perfectly hiding j^* to $\mathcal{A}$).

When $\mathcal{A}$ returns $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$, with $\sigma^* = (\hat{\sigma}^*_1, \dots, \hat{\sigma}^*_t)$, the crucial part of the proof is showing the existence of an index $i^\star$ such that $\hat{\sigma}^*_{i^\star}$ is a forgery for HS. Specifically:

- $\underline{\sigma^* \text{ is of type-1}}$ (Δ^* is new). Then $i^\star = t$ and $\hat{\sigma}^*_t$ is a type-1 forgery against HS.
- $\underline{\sigma^* \text{ is of type-2}}$. This means: $E_0(\mathsf{m}_1, \dots, \mathsf{m}_\mathsf{n}) = 0$ while $\mathsf{HS.Verify}(\mathcal{E}_t, \mathsf{pk}_{\mathsf{id}_t}, 1, \hat{\sigma}^*_t) = 1$. Then we show that there must exist a "forking index" $i^\star \in [t]$ such that $E_{i-1}(\mathsf{m}_{\mathsf{t}_{i-1}}, \dots, \mathsf{m}_{\mathsf{t}_{i-1}+\mathsf{n}_{i-1}}) = 0$ but $\mathsf{HS.Verify}(\mathcal{E}_i, \mathsf{pk}_{\mathsf{id}_i}, \hat{\sigma}^*_i, 1) = 1$, that is, $\hat{\sigma}^*_{i^\star}$ is a type-2 forgery against HS for the labeled program $\mathcal{E}_i$.
- $\underline{\sigma^* \text{ is of type-3}}$. If $t = 1$, then $i^\star = 1$ and $\hat{\sigma}^*_1$ is a type-3 forgery against HS. If $t > 1$, let $i \in [t]$ be the first index such that $\exists\, j \in [\mathsf{n}] : \ell_j = (\mathsf{id}_i, \tau_j) \notin L_{\Delta^*}$, *i.e.*, the first identity for which a type-3 forgery condition holds. Then, either $\hat{\sigma}^*_i$ is a type-3 forgery for HS for identity id_i (and thus $i^\star = i$); or there is $i^\star > i$ such that $\hat{\sigma}^*_{i^\star}$ is a type-2 forgery against identity $\mathsf{id}_{i^\star}$. The latter can be argued by showing the existence of a "forking index" as in the previous case. In a nutshell, a type-3 forgery against MKHS comes either from a type-3 forgery at some index i, or, the i-th signature is incorrect and thus there must be a type-2 forgery at a later index to cheat on the fact that verification at index i is correct.

Therefore, if $j^* = i^\star$ (which happens with non-negligible probability $1/Q_{\mathsf{id}}$), $\mathcal{B}$ can convert $\mathcal{A}$'s forgery into one for its challenger.

4 Conclusions and Future work

In this paper, we presented Matrioska the first generic compiler based on falsifiable assumptions that establishes a formal connection between single-key HS and multi-key HS schemes. Matrioska introduces an original mechanism to gain multi-key features by levering the homomorphic property of a single-key HS scheme. The resulting signatures are succinct in the sense that their length depends solely on the number of signers involved in the homomorphic computation, and not on the total number of signatures input. Unfortunately, constructions obtained with Matrioska are of limited efficiency, as they require the single-key HS scheme to support circuits of size exponentially large in the maximum number of distinct signers involved in the computation. Achieving full signature succinctness remains an interesting goal for further developments, as well as investigating if Matrioska's approach could be used to enhance other cryptographic primitives with multi-key features.

Acknowledgements. This work was partially supported by the COST Action IC1306 through a STSM grant to Elena Pagnin. Dario Fiore was partially supported by the Spanish Ministry of Economy under project references TIN2015-70713-R (DEDETIS), RTC-2016-4930-7 (DataMantium), and by the Madrid Regional Government under project N-Greens (ref. S2013/ICE-2731).

Bibliography

[1] Nuttapong Attrapadung and Benoît Libert. "Homomorphic Network Coding Signatures in the Standard Model". In: *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. Lecture Notes in Computer Science. Springer, Heidelberg, Mar. 2011, pp. 17–34.

[2] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. "Computing on Authenticated Data: New Privacy Definitions and Constructions". In: *18th International Conference on the Theory and Application of Cryptology and Information Security*. 2012, pp. 367–385.

[3] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. "Foundations of garbled circuits". In: *ACM CCS 12: 19th Conference on Computer and Communications Security*. Ed. by Ting Yu, George Danezis, and Virgil D. Gligor. ACM Press, Oct. 2012, pp. 784–796.

[4] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. "Signing a Linear Subspace: Signature Schemes for Network Coding". In: *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*. Vol. 5443. Lecture Notes in Computer Science. Springer, Heidelberg, 2009, pp. 68–87.

[5] Dan Boneh and David Mandell Freeman. "Homomorphic Signatures for Polynomial Functions". In: *EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. LNCS. Tallinn, Estonia: Springer, Heidelberg, Germany, 2011, pp. 149–168.

[6] Dan Boneh and David Mandell Freeman. "Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures". In: *PKC 2011*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. LNCS. Taormina, Italy: Springer, Heidelberg, Germany, 2011, pp. 1–16.

[7] Dario Catalano and Dario Fiore. "Practical Homomorphic Message Authenticators for Arithmetic Circuits". In: *Journal of Cryptology* 31.1 (2018), pp. 23–59.

[8] Dario Catalano, Dario Fiore, Rosario Gennaro, and Konstantinos Vamvourellis. "Algebraic (Trapdoor) One-Way Functions and Their Applications". In: *TCC 2013: 10th Theory of Cryptography Conference*. Vol. 7785. Springer, Heidelberg, 2013, pp. 680–699.

[9] Dario Catalano, Dario Fiore, Rosario Gennaro, and Konstantinos Vamvourellis. "Algebraic (trapdoor) one-way functions: Constructions and applications". In: *Theoretical Computer Science* 592 (2015), pp. 143–165.

[10] Dario Catalano, Dario Fiore, and Luca Nizzardo. "On the Security Notions for Homomorphic Signatures". In: (2018).

[11] Dario Catalano, Dario Fiore, and Luca Nizzardo. "Programmable Hash Functions Go Private: Constructions and Applications to (Homomorphic) Signatures with Shorter Public Keys". In: *CRYPTO*. Vol. 9216. Springer, 2015, pp. 254–274.

[12] Dario Catalano, Dario Fiore, and Bogdan Warinschi. "Adaptive Pseudo-free Groups and Applications". In: *EUROCRYPT*. Vol. 6632. Springer, 2011, pp. 207–223.

[13] Dario Catalano, Dario Fiore, and Bogdan Warinschi. "Efficient Network Coding Signatures in the Standard Model". In: *PKC*. Vol. 7293. Springer, 2012, pp. 680–696.

[14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. "Homomorphic signatures with efficient verification for polynomial functions". In: *International Cryptology Conference*. Springer. 2014, pp. 371–389.

[15] Dario Catalano, Antonio Marcedone, and Orazio Puglisi. "Authenticating Computation on Groups: New Homomorphic Primitives and Applications". In: *20th International Conference on the Theory and Application of Cryptology and Information Security*. 2014, pp. 193–212.

[16] Y. Desmedt. "Computer security by redefining what a computer is". In: *NSPW*. 1993.

[17] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. "Multi-key homomorphic authenticators". In: *22nd International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2016, pp. 499–530.

[18] Dario Fiore and Elena Pagnin. "Matrioska: A Compiler for Multi-Key Homomorphic Signatures." Cryptology ePrint Archive, Report 2018/616, `http://eprint.iacr.org/2018/616`. 2018.

[19] David Mandell Freeman. "Improved Security for Linearly Homomorphic Signatures: A Generic Framework". In: *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*. Vol. 7293. Springer, Heidelberg, 2012, pp. 697–714.

[20] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. "Secure Network Coding over the Integers". In: *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*. Vol. 6056. Springer, Heidelberg, 2010, pp. 142–160.

[21] Rosario Gennaro and Daniel Wichs. "Fully Homomorphic Message Authenticators". In: *ASIACRYPT*. Vol. 8270. Springer, 2013, pp. 301–320.

[22] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *43rd Annual ACM Symposium on Theory of Computing*. Ed. by Lance Fortnow and Salil P. Vadhan. ACM Press, June 2011, pp. 99–108.

[23] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. "Leveled Fully Homomorphic Signatures from Standard Lattices". In: *47th ACM STOC*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. Portland, OR, USA: ACM Press, 2015, pp. 469–477.

[24] Robert Johnson, David Molnar, Dawn Song, and David Wagner. "Homomorphic signature schemes". In: *Cryptographers' Track at the RSA Conference*. Springer. 2002, pp. 244–262.

[25] Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, and Sherman S. M. Chow. "A Zoo of Homomorphic Signatures: Multi-Key and Key-Homomorphism". Cryptology ePrint Archive, Report 2016/834, `http://eprint.iacr.org/2016/834`. 2016.

[26] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. "Linearly Homomorphic Structure-Preserving Signatures and Their Applications". In: *Advances in Cryptology – CRYPTO 2013, Part II*. Vol. 8043. Springer, Heidelberg, 2013, pp. 289–307.

Appendix

B.i Circuit evaluation and composition of circuits

In this work we define circuits as 6-tuples $C = (\mathsf{n}, \mathsf{u}, \mathsf{q}, \mathsf{L}, \mathsf{R}, \mathsf{G})$. We consider circuits of fan-in 2 only as any constant fan-in K circuit can be made into this form by paying a constant factor in the circuit's depth and size. The largest component in the string C is the descriptions of the function L (and R), that is a sequence of q values in $[\mathsf{w}] \cup \{0\}$, therefore $|\mathsf{L}| = |\mathsf{R}| = \mathsf{q} \lg(\mathsf{w} + 1)$. Hence, for a fixed and reasonable encoding it holds $|C| \in O(\mathsf{w} \lg(\mathsf{w}))$.

To show that this formalism is meaningful we describe two fundamental operations on circuits: circuit evaluation and circuit composition.

Evaluating a Circuit. In order to evaluate a circuit on a given input, we need a gate-functionality function that translates each bit in (the description of) G into the output of a gate. Let $\mathsf{G}(g)$ denote the g-th bit in the description of G, we define the gate-functionality function $\gamma : [\mathsf{q}] \times \{0, 1\}^2 \to \{0, 1\}$ as:

$$\gamma(g, x_l, x_r) = \begin{cases} \mathsf{G}(g) & \text{if } \mathsf{L}(g) = 0, \\ x_l & \text{if } (\mathsf{L}(g) \neq 0 \text{ and } \mathsf{G}(g) = 0), \\ \mathsf{NAND}(x_l, x_r) & \text{if } (\mathsf{L}(g) \neq 0 \text{ and } \mathsf{G}(g) = 1) \end{cases}$$

Note that when $\mathsf{L}(g) = 0$ the gate-functionality of g is the constant-gate that always returns the value $\mathsf{G}(g) \in \{0, 1\}$. Otherwise, g is a *proper* gate: if $\mathsf{G}(g) = 0$ it returns the left input to g, while if $\mathsf{G}(g) = 1$ it returns the NAND between the two input values.

We define the evaluation function $\mathsf{ev_{circ}}$ on a circuit $C = (\mathsf{n}, \mathsf{u}, \mathsf{q}, \mathsf{L}, \mathsf{R}, \mathsf{G})$ and an n-bit string $(x_1, x_2, \ldots, x_\mathsf{n})$ as:

```
**process** $\mathsf{ev_{circ}}(C, (x_1, \ldots, x_\mathsf{n}))$
  **for** $g$ from 1 to $\mathsf{q}$ **do**:
      $l \leftarrow \mathsf{L}(g); r \leftarrow \mathsf{R}(g); x_g \leftarrow \gamma(g, x_l, x_r);$
  **return** $(x_{\mathsf{n}+\mathsf{q}-\mathsf{u}+1}, \ldots, x_{\mathsf{n}+\mathsf{q}})$.
```

We will often shorten $\mathsf{ev_{circ}}(C, (x_1, \ldots, x_\mathsf{n}))$ into $C(x_1, \ldots, x_\mathsf{n})$.

Sequential composition of circuits. Given two circuits, C_1 and C_2, we say that C_1 is *composable* with C_2 if $\mathsf{u}_1 = \mathsf{n}_2$. We denote the circuit composition as $C_3 = C_1 \triangleright C_2$. The resulting circuit $C_3 = (\mathsf{n}_3, \mathsf{u}_3, \mathsf{q}_3, \mathsf{L}_3, \mathsf{R}_3, \mathsf{G}_3)$ is defined as: $\mathsf{n}_3 = \mathsf{n}_1$, $\mathsf{u}_3 = \mathsf{u}_2$, $\mathsf{q}_3 = \mathsf{q}_1 + \mathsf{q}_2$. Let w_i be the number of wires in C_i, then

$$\mathsf{L}_3 = \begin{cases} \mathsf{L}_1(i) & \text{for } i \in [\mathsf{w}_1] \\ 0 & \text{for } i \in [\mathsf{w}_1 + \mathsf{w}_2] \setminus [\mathsf{w}_1] \text{ and } \mathsf{L}_2(i - \mathsf{w}_1) = 0 \\ \mathsf{L}_2(i - \mathsf{w}_1) + \mathsf{w}_1 - \mathsf{u}_1 & \text{for } i \in [\mathsf{w}_1 + \mathsf{w}_2] \setminus [\mathsf{w}_1] \text{ and } \mathsf{L}_2(i - \mathsf{w}_1) \neq 0 \end{cases}$$

The right-input function R_3 is defined analogously. Finally, $\mathsf{G}_3 = \mathsf{G}_1 || \mathsf{G}_2$.

B.ii Details of the three-client three-input case for Matrioska

Consider the case in which we want to authenticate the result of a *three*-input circuit $C = (3, 1, \mathsf{q}_C, \mathsf{L}_C, \mathsf{R}_C, \mathsf{G}_C)$ evaluated on *three* messages, each signed by a distinct client. For notation sake, we assume the clients have identities $\mathsf{id}_1 = 1, \mathsf{id}_2 = 2$ and $\mathsf{id}_3 = 3$. Let $\mathsf{m}_i \in \{0,1\}$ denote the input of party $i \in [3]$, and $\sigma_i \leftarrow \mathsf{HS.Sign}(\mathsf{sk}_i, \ell_i, \mathsf{m}_i)$ be the corresponding signature. Note that each σ_1, σ_2 and σ_3 is generated using a different secret key of the HS scheme. Moreover, in this example we are deliberately removing dataset identifiers for ease of exposition.

Step 0. Given the labeled program $\mathcal{P} = (C, (\ell_1, \ell_2, \ell_3))$ and the three messages $\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3$, compute the value $\mathsf{y} = C(\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3)$. Let $E_0 = (3, 1, \mathsf{q}_0, \mathsf{L}_0, \mathsf{R}_0, \mathsf{G}_0)$ be a circuit satisfying $E_0(x_1, x_2, x_3) = 1$ iff $C(x_1, x_2, x_3) = \mathsf{y}$, *e.g.,* $E_0 = C \triangleright EQ^{\mathsf{y}}$. Note that E_0 can be constructed using C and y solely without, knowing the values $\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3$ (see the definition given in the Section 2.1). By construction it holds that:

$$E_0(\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3) = 1. \tag{20}$$

Step 1. We build a single-input circuit E_1 that corresponds to E_0 where the last two inputs m_2 and m_3 are fixed and hardwired into it. In this way, we obtain a single-input single-client circuit on which we can run $\mathsf{HS.Eval}$ using the public key pk_1. In more details, given $\mathcal{E}_0 = (E_0, (\ell_1, \ell_2, \ell_3))$, the signature σ_1 and the messages $\mathsf{m}_2, \mathsf{m}_3$:

• Define a *mask* circuit $M_1 = (1, 3, 3, \mathsf{L}'_1, \mathsf{R}'_1, \mathsf{G}'_1)$ where $\mathsf{L}'_1 = \mathsf{R}'_1 = (1, 0, 0)$ and $\mathsf{G}'_1 = (0, \mathsf{m}_2, \mathsf{m}_3)$. The purpose of M_1 is to create ad-hoc inputs for E_0: given $b \in \{0,1\}$ as input, M_1 outputs $M_1(b) = (b, \mathsf{m}_2, \mathsf{m}_3)$. [17]

• *Compose* M_1 with E_0 to obtain $E_1 = M_1 \triangleright E_0 = (1, 1, \mathsf{q}_1, \mathsf{L}_1, \mathsf{R}_1, \mathsf{G}_1)$ where: $\mathsf{q}_1 = \mathsf{q}_0 + 3$ and $\mathsf{G}_1 = (\mathsf{G}'_1 || \mathsf{G}_0)$. Let $\mathsf{L}^\star_0$ and $\mathsf{R}^\star_0$ be the string representations of the functions:

$$\mathsf{L}^\star_0(i) = \begin{cases} \mathsf{L}_0(i) + 3 & \text{if } \mathsf{L}_0(i) \neq 0,\ (i \in [\mathsf{q}_0]) \\ 0 & \text{if } \mathsf{L}_0(i) = 0,\ (i \in [\mathsf{q}_0]) \end{cases},$$

$$\mathsf{R}^\star_0(i) = \begin{cases} \mathsf{R}_0(i) + 3 & \text{if } \mathsf{R}_0(i) \neq 0,\ (i \in [\mathsf{q}_0]) \\ 0 & \text{if } \mathsf{R}_0(i) = 0,\ (i \in [\mathsf{q}_0]) \end{cases}$$

The left/right input functions of E_1 are $\mathsf{L}_1 = (\mathsf{L}'_1 || \mathsf{L}^\star_0)$, $\mathsf{R}_1 = (\mathsf{R}'_1 || \mathsf{R}^\star_0)$.

Circuit composition ensures that $E_1(x_1) = E_0(x_1, \mathsf{m}_2, \mathsf{m}_3)$ and thus equation (20) implies

$$E_1(\mathsf{m}_1) = 1. \tag{21}$$

Note that E_1 can be constructed directly from E_0 given m_2 and m_3, in particular the value m_1 is not needed:

$$E_1 = \big(1, 1, \mathsf{n}_0 + 3, (100, \mathsf{L}^\star_0), (100, \mathsf{R}^\star_0), (0, \mathsf{m}_2, \mathsf{m}_3, \mathsf{G}_0)\big).$$

• *Compute* $\hat{\sigma}_1 \leftarrow \mathsf{HS.Eval}(E_1, \mathsf{pk}_1, \sigma_1)$. This is possible since E_1 is a one-input circuit. Moreover equation (21) and the correctness of the HS scheme imply that

$$\mathsf{HS.Verify}((E_1, \ell_1), \mathsf{pk}_1, \hat{\sigma}_1, 1) = 1. \tag{22}$$

Step 2. The actual inductive procedure begins now. The challenge is that $\hat{\sigma}_1$ cannot

[17] Recall that $\mathsf{L}'_1(1) = 1 \neq 0$ and $\mathsf{G}'_1(1) = 0$, thus the first gate outputs the input to the circuit.

be directly checked by the verifier as it does not know the messages $\mathsf{m}_2, \mathsf{m}_3$ needed to define the circuit E_1. Our idea is to write equation (22) as $\mathsf{HS.Verify}(S_1) = 1$ for a string $S_1 = ((E_1, \ell_1), \mathsf{pk}_1, \sigma_1)$ that contains two bits that are the messages $\mathsf{m}_2, \mathsf{m}_3$, and then we want to use $\mathsf{HS.Eval}$ to create a signature proving the correctness of the computation in (22). As we shall see, this is possible by repeating our previous technique, namely seeing $\mathsf{HS.Verify}(S_1)$ as a single-input function of m_2 in which m_3 is hardwired. Repeating this approach one more time, we will later be able to let m_3 also "disappear" and use $\mathsf{HS.Eval}$ on a circuit that can be reconstructed by the verifier without knowing $\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3$.

Coming back to this second step, in $\mathsf{MKHS.Eval}$ we proceed as follows. We define a mask circuit that outputs S_1 where the bit that embeds the value m_2 is substituted with the bit input to mask circuit. Next, we define E_2 as the composition of this mask and the circuit HSV_2 that is the verification circuit of the signature scheme when checking the authenticity of a signature against the value 1 on input of $size(E_1)$.[18]

Facts: The position of the gate that embeds the value m_2 in E_1 is by construction $g_2 = 3\lg(N_1) + 2\mathsf{q}_1 \lg(\mathsf{w}_1) + 2$, where N_1 is a given upper bound on the size of n_1 and q_1, indeed:

$$E_1 = \big(\underbrace{1, 1, \mathsf{q}_1,}_{3\lg(N_1)}\ \underbrace{\mathsf{L}_1, \mathsf{R}_1}_{2\mathsf{q}_1\lg(\mathsf{w}_1)}\ , \mathsf{G}_1 = (\underbrace{0, \mathsf{m}_2}_{2}, \mathsf{m}_3, \mathsf{G}_0)\big).$$

Since $S_1 = ((E_1, \ell_1), \mathsf{pk}_1, \sigma_1)$ is a valid input to HSV_2, we have that $|S_1| = \mathsf{n}_{\mathsf{HSV}2}$.

Given $S_1 = ((E_1, \ell_1), \mathsf{pk}_1, \hat{\sigma}_1)$ and the signature σ_2:

- Define a *mask* circuit $M_2 = (1, \mathsf{n}_{\mathsf{HSV}2}, \mathsf{n}_{\mathsf{HSV}2}, \mathsf{L}'_2, \mathsf{R}'_2, \mathsf{G}'_2)$ where

$$\mathsf{L}'_2(g) = \mathsf{R}'_2(g) = \begin{cases} 0 & \text{if } g \in [\mathsf{n}_{\mathsf{HSV}2}] \setminus \{g_2\} \\ 1 & \text{if } g = g_2 \end{cases}$$

and

$$\mathsf{G}'_2(g) = \begin{cases} S_1[g] & \text{if } g \in [\mathsf{n}_{\mathsf{HSV}2}] \setminus \{g_2\} \\ 0 & \text{if } g = g_2 \end{cases}$$

The purpose of M_2 is to create ad-hoc inputs for the circuit $\mathsf{HS.Verify}$. The output of M_2 is S_1 where we overwrite the second gate of E_1 to output the value input to the circuit M_2 – instead of the constant output m_2. In particular,

$$M_2(\mathsf{m}_2) = \big((\overbrace{1, 1, \mathsf{q}_1, \mathsf{L}_1, \mathsf{R}_1, \underbrace{(0, \mathsf{m}_2, \mathsf{m}_3, \mathsf{G}_0)}_{\mathsf{G}_1}}^{E_1}, \ell_1), \mathsf{pk}_1, \hat{\sigma}_1)$$

where all values should be seen as bit-strings.

- *Compose* M_2 with HSV_2 to obtain $E_2 = M_2 \triangleright \mathsf{HSV}_2 = (1, 1, \mathsf{q}_2, \mathsf{L}_2, \mathsf{R}_2, \mathsf{G}_2)$ where: $\mathsf{q}_2 = \mathsf{n}_{\mathsf{HSV}2} + \mathsf{q}_{\mathsf{HSV}_2}$; and $\mathsf{G}_2 = (\mathsf{G}'_2 || \mathsf{G}_{\mathsf{HSV}_2})$. Let $\mathsf{L}^\star_{\mathsf{HSV}_2}$ and $\mathsf{R}^\star_{\mathsf{HSV}_2}$ be the string representations of the functions:

$$\mathsf{L}^\star_{\mathsf{HSV}_2}(i) = \begin{cases} \mathsf{L}_{\mathsf{HSV}_2}(i) + \mathsf{n}_{\mathsf{HSV}2} & \text{if } \mathsf{L}_{\mathsf{HSV}_2}(i) \neq 0,\ (i \in [\mathsf{q}_{\mathsf{HSV}_2}]) \\ 0 & \text{if } \mathsf{L}_{\mathsf{HSV}_2}(i) = 0,\ (i \in [\mathsf{q}_{\mathsf{HSV}_2}]) \end{cases},$$
$$\mathsf{R}^\star_{\mathsf{HSV}_2}(i) = \begin{cases} \mathsf{R}_{\mathsf{HSV}_2}(i) + \mathsf{n}_{\mathsf{HSV}2} & \text{if } \mathsf{R}_{\mathsf{HSV}_2}(i) \neq 0,\ (i \in [\mathsf{q}_{\mathsf{HSV}_2}]) \\ 0 & \text{if } \mathsf{R}_{\mathsf{HSV}_2}(i) = 0,\ (i \in [\mathsf{q}_{\mathsf{HSV}_2}]) \end{cases}$$

[18] With abuse of notation $\mathsf{HSV}_2 = \mathsf{HS.Verify}((\cdot, \cdots), \cdot, \cdots, 1)$. The index $_2$ is used to keep track of the size of the input (and corresponding output) of the verification circuit.

The left/right input functions of E_2 are defined as $\mathsf{L}_2 = (\mathsf{L}'_2||\mathsf{L}^{\star}_{\mathsf{HSV}_2})$, $\mathsf{R}_2 = (\mathsf{R}'_2||\mathsf{R}^{\star}_{\mathsf{HSV}_2})$.

Circuit composition ensures that $E_2(\mathsf{m}_2) = \mathsf{HS.Verify}((E_1, \ell_1), \mathsf{pk}_1, \hat{\sigma}_1, 1)$, therefore by (22) it holds that:

$$E_2(\mathsf{m}_2) = 1 \tag{23}$$

Note that E_2 can be constructed directly from E_0 given solely m_3, and additional public data (*i.e.,* pk_1, $\hat{\sigma}_1$, $\mathsf{HS.Verify}$, $\mathcal{P}$, y), indeed:

$$\mathsf{G}_2 = (\overbrace{1, 1, \mathsf{q}_0 + 3, \underbrace{(1, 0, 0, \mathsf{L}^{\star}_0)}_{\mathsf{L}_1}, \underbrace{(1, 0, 0, \mathsf{R}^{\star}_0)}_{\mathsf{R}_1}, \underbrace{(0, \mathbf{0}, \mathsf{m}_3, \mathsf{G}_0}_{\mathsf{G}'_1}}^{g_2-th \text{ index}}, \ell_1, \mathsf{pk}_1, \hat{\sigma}_1, \mathsf{G}_{\mathsf{HSV}_2})$$

where all values should be seen as bit-strings.

• *Compute* $\hat{\sigma}_2 \leftarrow \mathsf{HS.Eval}(E_2, \mathsf{pk}_2, \sigma_2)$. This is possible since E_2 is a one-input circuit, and σ_2 is a signature on m_2. Indeed, equation (23) and the correctness of the HS scheme imply that

$$\mathsf{HS.Verify}((E_2, \ell_2), \mathsf{pk}_2, \hat{\sigma}_2, 1) = 1. \tag{24}$$

Step 3. We proceed inductively, along the line of Step 2.

Facts: The position of the gate that embeds the value m_3 in E_2 is by construction $g_3 = 3\lg(N_2) + 2\mathsf{q}_2 \lg(\mathsf{w}_2) + g_2 + 1$, where N_2 is a given upper bound on the size of n_2 and q_2, indeed:

$$E_2 = \big(\underbrace{1, 1, \mathsf{q}_2}_{3\lg(N_2)}, \underbrace{\mathsf{L}_2, \mathsf{R}_2}_{2\mathsf{q}_2\lg(\mathsf{w}_2)}, \mathsf{G}_2 = \underbrace{(1, 1, \mathsf{q}_1, \mathsf{L}_1, \mathsf{R}_1, (0, \mathbf{0}}_{g_2}, \mathsf{m}_3, \mathsf{G}_0, \ell_1 ... \mathsf{G}_{\mathsf{HSV}_2})\big).$$

Given $S_2 = ((E_2, \ell_2), \mathsf{pk}_2, \hat{\sigma}_2)$, and the signature σ_2:

• Define a *mask* circuit $M_3 = (1, \mathsf{n}_{\mathsf{HSV}3}, \mathsf{n}_{\mathsf{HSV}3}, \mathsf{L}'_3, \mathsf{R}'_3, \mathsf{G}'_3)$ where

$$\mathsf{L}'_3(g) = \mathsf{R}'_3(g) = \begin{cases} 0 & \text{if } g \in [\mathsf{n}_{\mathsf{HSV}3}] \setminus \{g_3\} \\ 1 & \text{if } g = g_3 \end{cases}$$

and

$$\mathsf{G}'_3(g) = \begin{cases} S_2[g] & \text{if } g \in [\mathsf{n}_{\mathsf{HSV}3}] \setminus \{g_3\} \\ 0 & \text{if } g = g_3 \end{cases}$$

The output of M_3 is S_2 where we overwrite the gate that embeds the constant value m_3 with a gate that outputs the value input to the circuit M_3. In particular, $M_3(\mathsf{m}_3) = S_2$.

• *Compose* M_3 with HSV_3 to obtain $E_3 = M_3 \triangleright \mathsf{HSV}_3 = (1, 1, \mathsf{q}_3, \mathsf{L}_3, \mathsf{R}_3, \mathsf{G}_3)$. Circuit composition ensures that $E_3(\mathsf{m}_3) = \mathsf{HS.Verify}((E_2, \ell_2), \mathsf{pk}_2, \hat{\sigma}_2, 1)$, therefore by (24) it holds that:

$$E_3(\mathsf{m}_3) = 1. \tag{25}$$

Note that E_3 can be constructed directly from $\mathcal{E}_0 = (E_0, \ell_1, \ell_2, \ell_3)$ given solely the public data $\mathsf{pk}_i, \hat{\sigma}_i$ for $i \in [2]$: indeed:

$$E_3 \quad = \quad \Big(1, 1, \mathsf{n}_{\mathsf{HSV}3} + \mathsf{q}_{\mathsf{HSV}_3}, \mathsf{L}_3 = (\overbrace{\underbrace{0..0..1}_{g_3} 0..0}^{\mathsf{n}_{\mathsf{HSV}3}}, \overbrace{\mathsf{L}^\star_{\mathsf{HSV}_3}}^{\mathsf{q}_{\mathsf{HSV}3}}), \mathsf{R}_3 = (\overbrace{\underbrace{0..0..1}_{g_3} 0..0}^{\mathsf{n}_{\mathsf{HSV}3}}, \overbrace{\mathsf{R}^\star_{\mathsf{HSV}_3}}^{\mathsf{q}_{\mathsf{HSV}3}}),$$

$$\underbrace{(\overbrace{0..01}^{g_2} 0..0, \mathsf{L}^\star_{\mathsf{HSV}_2})}$$

$$\mathsf{G}_3 = \big((1, 1, \mathsf{n}_{\mathsf{HSV}2} + \mathsf{q}_{\mathsf{HSV}_2},\ \mathsf{L}_2,\ \mathsf{R}_2, \mathsf{string}, \mathsf{G}_{\mathsf{HSV}_2}, \ell_2), \mathsf{pk}_2, \hat{\sigma}_2, \mathsf{G}_{\mathsf{HSV}_3})\big)\Big)$$

$$\overbrace{(1, 1, \mathsf{q}_0 + 3, \underbrace{(1, 0, 0, \mathsf{L}^\star_0)}_{\mathsf{L}_1}, \underbrace{(1, 0, 0, \mathsf{R}^\star_0)}_{\mathsf{R}_1}, (0, 0, 0, \mathsf{G}_0, \ell_1, \mathsf{pk}_1, \hat{\sigma}_1))}$$

where all values should be seen as bit-strings.

• *Compute* $\hat{\sigma}_3 \leftarrow \mathsf{HS.Eval}(E_3, \mathsf{pk}_3, \sigma_3)$. This is possible since E_3 is a one-input circuit. From Equation (25) and the correctness of the HS scheme we get:

$$\mathsf{HS.Verify}((E_3, \ell_3), \mathsf{pk}_3, \hat{\sigma}_3, 1) = 1. \tag{26}$$

The multi-key homomorphic evaluation algorithm outputs $\hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3)$.

In order to verify $\hat{\sigma}$ the verifier simply needs the labeled program $\mathcal{P} = (C, (\ell_1, \ell_2, \ell_3))$, the value y corresponding to the claimed output of $\mathcal{P}$, the three public keys pk_i for $i \in [3]$ and the multi-key homomorphic signature $\hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3)$. The verification process begins by constructing the circuit E_0. As noted before, this can be done given solely C and y. Next, the verifier computes directly the circuit E_3. This can be done using the available values $\ell_i, \mathsf{pk}_i, \hat{\sigma}_i$ for $i \in [2]$ and the (public) circuit descriptions of HSV_2 and HSV_3. Let $\mathcal{E}_3 = (E_3, \ell_3)$, the verification concludes by running:

$$\mathsf{HS.Verify}(\mathcal{E}_3, \mathsf{pk}_3, \hat{\sigma}_3, 1)$$

It is easy to see that by correctness, this returns 1, as stated in Equation (26).

B.ii.1 Computing the index where the messages of signer i are embedded

Here we explain the reasoning behind the definition of the index

$$g_i = 3\lg(N_{i-1}) + 2\mathsf{q}_{i-1}\lg(\mathsf{w}_{i-1}) + g_{i-1} + \mathsf{n}_{i-1}$$

in our compiler (step i in Definition 3.3). Recall that in this step we hold the circuit E_{i-1} and look for the positions in its gate description where the n_i values input by identity id_i are located. This will be an interval $I_i = [g_i, g_i + \mathsf{n}_i]$, for some index g_i. Essentially, g_i should jump over the description of the first part of the circuit $E_{i-1} = (\mathsf{n}_{i-1}, 1, \mathsf{q}_{i-1}, \mathsf{L}_{i-1}, \mathsf{R}_{i-1}, \mathsf{G}_{i-1})$ to select the bits in G_{i-1} that contain the values $\mathsf{m}_{(\mathsf{n}_{i-1}+1)}, \ldots, \mathsf{m}_{\mathsf{n}_i}$. The description of the values $\mathsf{n}_{i-1}, 1, \mathsf{q}_{i-1}$ covers the first $3\lg(N_{i-1})$ bits. Then, the left/right input functions $\mathsf{L}_{i-1}, \mathsf{R}_{i-1}$ are two strings of q_{i-1} wires covering additional $2\mathsf{q}_{i-1}\lg(\mathsf{w}_{i-1})$ bits. At this point we enter the gate description of E_{i-1} that brings with an accumulative addend of $g_{i-1} + \mathsf{n}_{i-1} + 1$ bits to reach the position where G_{i-1} contains the first message input by client id_i. By construction $\mathsf{G}_{i-1} = (\mathsf{G}'_{i-1}||\mathsf{G}_{\mathsf{HSV}_{i-1}})$, for consistency let $\mathsf{HSV}_1 = \mathsf{G}_0$. For $i \geq 2$ the gates in G'_{i-1} *embed* (a minor modification of) the string $S_{i-2} = ((E_{i-2}, \ell_{\mathsf{t}_{i-1}}, \dots, \ell_{\mathsf{t}_i - 1}), \mathsf{pk}_{\mathsf{id}_{i-2}}, \vec{\sigma}_{i-2})$, for consistency let $S_0 = (0..0\mathsf{m}_{\mathsf{n}_1+1}..\mathsf{m}_\mathsf{n})$ where the first n_1 entries are 0. When $i = 2$, $\mathsf{G}'_1 = S_0$, therefore the bit that contains the first input by client id_2 is the $\mathsf{t}_2 = (\mathsf{n}_1 + 1)$-th bit in G_1, which is consistent with the formula $g_1 + \mathsf{n}_1$ since we set $g_1 = 1$. Now for $i = 3$, G'_2 equals $S_1 = ((E_1, \ell_{\mathsf{n}_1+1}, \dots, \ell_{\mathsf{t}_2-1}), \mathsf{pk}_{\mathsf{id}_1}, \vec{\sigma}_1)$, except for the (n_3) bits where G'_1 has the gates initiated to the values input by id_3. This interval begins at the $3\lg(N_1) + 2\mathsf{q}_1 \lg(\mathsf{w}_1) + \mathsf{n}_1 + 1 = g_2$ bit of S_1. Therefore $g_3 = 3\lg(N_2) + 2\mathsf{q}_2 \lg(\mathsf{w}_2) + g_2 + \mathsf{n}_2$. This explains the recursive definition of g_i. in our representation of circuits the largest factor in the size of a circuit is determined by the description of the functions L and R. In particular, for a circuit C with $\mathsf{q} >> \mathsf{n}$ the asymptotic bound is $size(C) \sim \mathsf{q}\lg(\mathsf{q})$, where we approximate the number of wires w with the number of gates q.[19]

Let q_0 denote the number of gates in E_0, then: $size(E_0) \sim \mathsf{q}_0 \lg(\mathsf{q}_0)$. Let n be the total number of input to the computation (in case $\mathsf{n}=t$ each input belongs to a different user) then $size(E_1) \sim (\mathsf{q}_0 + \mathsf{n})\lg(\mathsf{q}_0 + \mathsf{n})$, since $\mathsf{n} < \mathsf{q}_0$ in this analysis we consider

$$size(E_1) \sim 2\mathsf{q}_0 \lg(2\mathsf{q}_0). \tag{27}$$

The actual growth begins with $i = 2$. From this point on, in fact, the verification circuit HSV_i is contained in E_i. Without loss of generality, we assume that on input a circuit of size Z the verification circuit $size(\mathsf{HSV})$ has size $size(\mathsf{HSV}) \sim size(Z)^{c_s}$, for a constant value $c_s \geq 1$ (dependent on the HS scheme). In our compiler this translates to $\mathsf{q}_{\mathsf{HSV}_i} \lg(\mathsf{q}_{\mathsf{HSV}_i}) \sim (\mathsf{q}_{i-1} \lg(\mathsf{q}_{i-1}))^{c_s}$, thus:

$$\begin{aligned}
size(E_2) &= size(M_2 \triangleright \mathsf{HSV}_2) = |(1, 1, \mathsf{q}_2, \mathsf{L}_2, \mathsf{R}_2, \mathsf{G}_2)| \\
&\sim \mathsf{q}_2 \lg(\mathsf{q}_2) \qquad (28)\\
&\sim \mathsf{q}_{\mathsf{HSV}_2} \lg(\mathsf{q}_{\mathsf{HSV}_2}) \qquad (29)\\
&\sim (2\mathsf{q}_0 \lg(2\mathsf{q}_0))^{c_s}. \qquad (30)
\end{aligned}$$

Where (28) is the usual approximation of the circuit's size with the number of gates, (29) comes from the fact that[20] $\mathsf{q}_2 = \mathsf{n}_{\mathsf{HSV}_2} + \mathsf{q}_{\mathsf{HSV}_2} \sim \mathsf{q}_{\mathsf{HSV}_2}$ and (30) is implied by our assumption $size(\mathsf{HSV}_2) \sim size(E_1)^{c_s}$ and (27). Following this reasoning inductively we get: $size(E_t) \sim (\mathsf{q}_0 \lg(\mathsf{q}_0))^{{c_s}^{t-1}}$. In other words, let s' denote the size of the circuit C given in input to $\mathsf{MKHS.Eval}$, then $size(E_t) = s'^{{c_s}^{t-1}} < s$. Regarding the depth growth we notice that if the function $d_{\mathsf{HSV}}(z, \lambda)$ is polynomial there exists a constant $c_d \geq 1$ such that $d_{\mathsf{HSV}}(z, \lambda) = z^{c_d}$. Then $depth(E_t) = size(E_{t-1})^{c_d} = s^{c_d \cdot {c_s}^{1-t}}$. However, if $d_{\mathsf{HSV}}(z, \lambda)$ is logarithmic then $depth(E_t) = \log(size(E_{t-1})) = ({c_s}^{1-t})s$.

Proof. We define a reduction $\mathcal{B}$ between an $\mathsf{MK\text{-}HomUF\text{-}CMA}$ forger $\mathcal{A}$ and an $\mathsf{HomUF\text{-}CMA}$ challenger $\mathcal{C}$. The reduction begins by initializing the identity counter q to 1 and by choosing a (random) index $j^* \leftarrow [Q_{\mathsf{id}}]$ as a guess for an identity on which $\mathcal{A}$ will make a forgery.

[19] Approximating $\mathsf{w} = \mathsf{q} + \mathsf{n}$ with q is a quite tight in our case, since all E_i are circuits with one single input.

[20]to upperbound we could put a factor 2 in this estimate: $\mathsf{q}_2 < 2\mathsf{q}_{\mathsf{HSV}_2}$ but since this is an asymptotic estimate and $\mathsf{q}_{\mathsf{HSV}_2} > 2$ it would not change much.

Setup. In the setup phase $\mathcal{B}$ starts the $\mathsf{HomUF\text{-}CMA}$ game for the scheme HS. The reduction uses the public parameters of the HS scheme given by its $\mathsf{HomUF\text{-}CMA}$ challenger $\mathcal{C}$ to generate pp for the MKHS scheme (*e.g.,* adding the ID set, redefining the labels). $\mathcal{B}$ sends pp to the adversary $\mathcal{A}$, and stores the public key pk provided by $\mathcal{C}$.

Sign queries. In the sign queries, the reduction $\mathcal{B}$ answers to queries $(\Delta, \ell = (\mathsf{id}, \tau), \mathsf{m})$ as follows:

1. If this is the first query for the dataset Δ, the reduction initializes an empty list $L_\Delta = \emptyset$ and proceeds to step 2.

2. If this is the first query with identity id *and* $q \neq j^*$, generate keys for the identity id running $(\mathsf{sk}_q, \mathsf{pk}_q) \leftarrow \mathsf{MKHS.KeyGen}(\mathsf{pp})$. If this is the first query with identity id *and* $q = j^*$, set $\mathsf{pk}_{j^*} = \mathsf{pk}$ ($\mathcal{A}$ is generating the user that $\mathcal{B}$ has guessed as to be the target for the forgery). Update the identity-index map $\omega : \mathsf{ID} \to [Q_{\mathsf{id}}]$ with $\omega(\mathsf{id}) = q$ and increase the identity counter $q \leftarrow q + 1$. Proceed to step 3 and 4.

3. If the query has not been asked before, (*i.e.,* $(\ell, \mathsf{m}) \notin L_\Delta$) *and* $\omega(\mathsf{id}) = i \neq j^*$, compute $\sigma \leftarrow \mathsf{MKHS.Sign}(\mathsf{sk}_i, \ell, \mathsf{m})$ (notice that in this case $\mathcal{B}$ knows the secret key). If $(\ell, \mathsf{m}) \notin L_\Delta$ *and* $\omega(\mathsf{id}) = j^*$, $\mathcal{B}$ queries its challenger $\mathcal{C}$ with $(\Delta, \tau, \mathsf{m})$ to obtain a signature σ. Finally, in both cases, the reduction updates the list of queried messages for the database Δ: $L_\Delta \leftarrow L_\Delta \cup \{(\ell, \mathsf{m})\}$ and returns (σ, pk_i) to $\mathcal{A}$ ($i \in [q]$).

4. If the query has the same label of a previous query on the same dataset, *i.e.,* there exists a message $\mathsf{m}' \in \mathcal{M}$ such that $(\ell, \mathsf{m}') \in L_\Delta$, ignore the query.

It is easy to see that up to this point $\mathcal{B}$ perfectly simulates the $\mathsf{MK\text{-}HomUF\text{-}CMA}$ game to $\mathcal{A}$.

Forgery. Let $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ denote the output of $\mathcal{A}$ at the end of the $\mathsf{MK\text{-}HomUF\text{-}CMA}$ security experiment (simulated by $\mathcal{B}$). Let id^* denote the identity corresponding to the index j^* chosen by $\mathcal{B}$, *i.e.,* $\omega(\mathsf{id}^*) = j^*$. If $\mathsf{id}^* \notin \mathcal{P}^*$ the reduction aborts. In this case indeed, $\mathcal{B}$ has for sure failed to guess one of the identities involved in the forgery made by $\mathcal{A}$. Otherwise, in what follows we show that the forgery $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ can be converted (except with some non-negligible error probability related to the wrong guess of j^*) into a single-key forgery for the $\mathsf{HomUF\text{-}CMA}$ experiment. In what follows we do an analysis case by case.

Single user. If $\mathcal{P}^*$ is a computation that involves a single user only, the reduction is perfect, *i.e.,* $\mathcal{B}$ can turn every forgery (of any type) output by $\mathcal{A}$ against MKHS into a forgery against HS by removing the identity id^* from the labels. Indeed, notice that if $\mathcal{B}$ reached this point, it did not abort, and thus $\mathsf{id}^* \in \mathcal{P}^*$.

Multi-user programs. If $\mathcal{P}^*$ involves $t > 1$ users, the reduction proceeds as follows.

Type-1 Forgery. Namely, it holds that both $\mathsf{MKHS.Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*) = 1$ and $L_{\Delta^*} = \emptyset$. We show that this corresponds to a type-1 forgery against HS for the t-th key in $\mathcal{P}^*$. By construction (see Equation (19)), $\mathsf{MKHS.Verify}$ outputs 1 if and only if $\mathsf{HS.Verify}((E_t, (\tau^*_{\mathsf{t}_t}, \ldots, \tau^*_{\mathsf{n}})), \Delta^*, \mathsf{pk}^*_{\mathsf{id}_t}, 1, \hat{\sigma}^*_t) = 1$. Moreover, since $L_{\Delta^*} = \emptyset$, the reduction never queried its challenger $\mathcal{C}$ on the dataset Δ^* either. The last two conditions ensure that $(E_t, (\tau^*_{\mathsf{t}_t}, \ldots, \tau^*_{\mathsf{n}})), \Delta^*, \mathsf{pk}^*_{\mathsf{id}_t}, 1, \hat{\sigma}^*_t)$ is a type-1 forgery against HS for the key pair $(\mathsf{pk}_{\mathsf{id}_t}, \mathsf{sk}_{\mathsf{id}_t})$.

Therefore, in this case the reduction $\mathcal{B}$ returns $(E_t, (\tau^*_{\mathsf{t}_t}, \ldots, \tau^*_{\mathsf{n}})), \Delta^*, \mathsf{pk}^*_{\mathsf{id}_t}, 1, \hat{\sigma}^*_t)$ to its challenger, if $j^* = \omega(\mathsf{id}_t)$ (*i.e.,* $\mathsf{pk}_{\mathsf{id}_t} = \mathsf{pk}_{j^*}$), and aborts otherwise.

Remark 3.5. The adversary $\mathcal{A}$ can possibly produce type-1 forgeries also for identities id_i with $i < t$. In this case, however, L^*_Δ is empty and therefore it is impossible to run the (single-key) verification algorithm on the circuits E_i for $i < t$, as this would require the knowledge of the messages $\mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i+1}, \ldots, \mathsf{m}_\mathsf{n}$ input by the last $t-i$ users.

Type-2 Forgery. Namely, the adversary returns $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ such that $\mathsf{MKHS.Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$=1 and $\mathsf{y}^* \neq C^*(\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{n})$, where $\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{n}$ are the messages queries by $\mathcal{A}$ for the respective labels in $\mathcal{P}^*$. In the following claim, we formally show that from any type-2 forgery against the MKHS scheme, it is possible to extract a type-2 forgery against the HS scheme (corresponding to at least one of the users involved in $\mathcal{P}^*$).

Claim 1. Let $t \geq 2$, and let $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ be such that $\mathsf{MKHS.Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$=1 and $\mathsf{y}^* \neq C^*(\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{n})$, with $\sigma^* = (\hat{\sigma}^*_1, \ldots, \hat{\sigma}^*_t)$. Then, there exists (at least) one index $i \in [t]$ such that $\hat{\sigma}^*_i$ is a type-2 forgery against the HS scheme (for an opportune function).

Proof. The claim follows from this inductive reasoning. Consider $\vec{\sigma}^* = (\hat{\sigma}^*_1, \ldots, \hat{\sigma}^*_t)$. By definition $E_0(\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{n}) = 1$ if and only if $C^*(\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{n}) = \mathsf{y}$ (see Equation (10)). Since $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ is a type-2 forgery $\mathsf{y}^* \neq \mathsf{y}$. Therefore $E_0(\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{n}) = 0$. The correctness of the $\mathsf{Matrioska}$ compiler therefore implies that $E_1(\mathsf{m}_1, \ldots, \mathsf{m}_{\mathsf{n}_1}) = 0$ as well, where E_1 is the circuit defined in $\mathsf{MKHS.Eval}$ with the messages of identities id_j, with $j > 1$, hardwired.

Given the signature $\hat{\sigma}^*_1$ there are two possible cases: either $\mathsf{HS.Verify}(\mathcal{E}_1, \mathsf{pk}_{\mathsf{id}_1}, \hat{\sigma}^*_1, 1) = 1$ or not. In the first case, $(\mathcal{E}_1, \mathsf{pk}_{\mathsf{id}_1}, \hat{\sigma}^*_1, 1)$ is a type-2 forgery against the HS scheme for the key-pair $(\mathsf{pk}_1, \mathsf{sk}_1)$, and thus we have found our forgery and the claim is proven with index $i = 1$. Otherwise, we proceed inductively to the next identity to show that the claim can be proven for $i > 1$.

By induction, let $i > 1$ and assume $E_{i-1}(\mathsf{m}_{\mathsf{t}_{i-1}}, \ldots, \mathsf{m}_{\mathsf{t}_{i-1}+\mathsf{n}_{i-1}}) = 0$. By construction of $\mathsf{Matrioska}$ it holds that $E_i(\mathsf{m}_{\mathsf{t}_i}, \ldots, \mathsf{m}_{\mathsf{t}_i+\mathsf{n}_i}) = 0$. Similarly to the case $i = 1$, note that for the signature $\hat{\sigma}^*_i$ there are two possible cases: either $\mathsf{HS.Verify}(\mathcal{E}_i, \mathsf{pk}_{\mathsf{id}_i}, \hat{\sigma}^*_i, 1) = 1$ or not. In the first case $(\mathcal{E}_i, \mathsf{pk}_{\mathsf{id}_i}, \hat{\sigma}^*_i, 1)$ is a type-2 forgery against the HS scheme for the key-pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, and thus we have found our forgery and the claim is proven with this index i. Otherwise, we proceed with index $i + 1$.

Finally, to show that such index i must exist, we notice that we cannot reach $i = t+1$ without finding a forgery. In fact, for $i = t+1$ we would have $E_t(\mathsf{m}_{\mathsf{t}_t}, \ldots, \mathsf{m}_\mathsf{n}) = 0$. However, by definition of type-2 forgery in $\mathsf{MK\text{-}HomUF\text{-}CMA}$ we have that $\mathsf{MKHS}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}^*_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}, \mathsf{y}^*, \sigma^*) = 1$, that is $\mathsf{HS.Verify}(\mathcal{E}_t, \mathsf{pk}_{\mathsf{id}_t}, 1, \hat{\sigma}^*_t) = 1$ (see Equation (19)). This immediately shows that $(\mathcal{E}_t, \mathsf{pk}_{\mathsf{id}_t}, 1, \hat{\sigma}^*_t)$ is a type-2 forgery against HS for the key $\mathsf{pk}_{\mathsf{id}_t}$. This completes the proof of the claim. □

Given the type-2 forgery produced by $\mathcal{A}$, $\mathcal{B}$ builds the circuit E_{j^*} using $\mathcal{P}^* = (C^*, (\ell_1, \dots, \ell_{\mathsf{n}}))$, the messages stored in L_{Δ^*} and the signatures $\hat{\sigma}_i^*$ for $i < j^*$. Let i be the index whose existence is granted by the previous claim. If $i = j^*$, $\mathcal{B}$ outputs to its challenger $\mathcal{C}$ the tuple $(\tilde{\mathcal{E}_{j^*}} = (E_{j^*}, (\tau_{\mathsf{t}_{j^*}}, \dots, \tau_{\mathsf{t}_{j^*}+\mathsf{n}_{j^*}})), 1, \hat{\sigma}_{j^*}^*)$ as its type-2 forgery against HS for the key pk_{j^*}. Otherwise, the reduction aborts as the guess of j^* was incorrect and $\hat{\sigma}_{j^*}^*$ is not guaranteed to be a forgery.

Type-3 Forgery. Namely, the adversary returns $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ such that $\mathsf{MKHS.Verify}(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$=1 and there exists one label, say $\ell_{i^\star} \in \mathcal{P}^*$, for which no sign query was performed, *i.e.,* $(\ell_{i^\star} = (\mathsf{id}_{i^\star}, \tau_{i^\star}), \cdot) \notin L_{\Delta^*}$. In the following claim we show that any such type-3 forgery against MKHS reduces to either a type-3 or a type-2 forgery against HS.

Claim 2. Let $t \geq 2$. If, at the end of the $\mathsf{MK\text{-}HomUF\text{-}CMA}$ security experiment, $\mathcal{A}$ outputs a type-3 forgery for the label $\ell_{i^\star} = (\mathsf{id}_{i^\star}, \tau_{i^\star}) \in \mathcal{P}^*$ then either (1) the forgery reduces to a type-3 forgery against HS for the identity $\mathsf{id}_{i^\star}$, or (2) there is (at least one) type-2 forgery against HS for an identity $\mathsf{id}_i \in \mathcal{P}^*$ with $i > i^\star$.

Proof. Let $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ be the type-3 forgery output by the adversary at the end of the $\mathsf{MK\text{-}HomUF\text{-}CMA}$ experiment. Since no sign query for the label $\ell_{i^\star}$ was performed during the security experiment, it is impossible to reconstruct any circuit E_i for $i < i^\star$. The motivation is that by construction the description of E_i includes all the messages m_j with $j \geq \mathsf{t}_i$, including $\mathsf{m}_{i^\star}$ (see Equation (14)). Therefore the first circuit that is publicly reconstructible is $E_{i^\star}$. Let $\mathcal{E}_{i^\star} = (E_{i^\star}, (\tau_{\mathsf{t}_{i^\star}}, \dots, \tau_{\mathsf{t}_{i^\star}+\mathsf{n}_{i^\star}}))$, there are two possible cases: either $\mathsf{HS.Verify}(\mathcal{E}_{i^\star}, \Delta^*, \mathsf{pk}_{i^\star}, 1, \hat{\sigma}_{i^\star}^*) = 1$ or it equals 0. It is immediate to see that in case the verification procedure outputs 1 $(\mathcal{E}_{i^\star}, \Delta^*, \mathsf{pk}_{i^\star}, 1, \hat{\sigma}_{i^\star}^*)$ satisfies all the requirements for a type-3 forgery on the key $\mathsf{pk}_{i^\star}$ against the scheme HS. Otherwise, the verification of the $i^\star$-th circuit fails, *i.e.,* $\mathsf{HS.Verify}(\mathcal{E}_{i^\star}, \Delta^*, \mathsf{pk}_{i^\star}, 1, \hat{\sigma}_{i^\star}^*) = 0$, and the correctness of the $\mathsf{Matrioska}$ compiler implies that $E_j(\mathsf{m}_{\mathsf{t}_j}, \dots, \mathsf{m}_{\mathsf{t}_j+\mathsf{n}_j}) = 0$ for all $j > i^\star$. However, by definition of type-3 forgery the final (multi-key) verification outputs 1. We are now in a situation similar to the one of type=2 forgeries. In particular for $\mathsf{MKHS.Verify}$ to output 1, it must hold that $\mathsf{HS.Verify}(\mathcal{E}_t, \Delta^*, \mathsf{pk}_{\mathsf{id}_t}, \hat{\sigma}_t^*, 1) = 1$ (see Equation (19)). Therefore there must be a type-2 forgery against the HS scheme for one identity id_j with $i^\star < j \leq t$. The latter follows by the same argument used in the previous Claim (details omitted). This concludes the proof of the claim. □

In light of the claim above, from a type-3 forgery $(\mathcal{P}^*, \Delta^*, \{\mathsf{pk}_{\mathsf{id}}^*\}_{\mathsf{id} \in \mathcal{P}^*}, \mathsf{y}^*, \sigma^*)$ the reduction $\mathcal{B}$ can derive either a type-3 forgery against its $\mathsf{HomUF\text{-}CMA}$ challenger (if $\mathsf{id}_{i^\star} = \mathsf{id}_{j^*}$ and $\mathsf{HS.Verify}(\mathcal{E}_{j^*}, \Delta^*, \mathsf{pk}_{j^*}, 1, \hat{\sigma}_{j^*}^*) = 1$) or a type-2 forgery (if $j^* > i^\star$ and $\mathsf{HS.Verify}(\mathcal{E}_{j^*}, \Delta^*, \mathsf{pk}_{j^*}, 1, \hat{\sigma}_{j^*}^*) = 1$ while $E_{j^*}(\mathsf{m}_{\mathsf{t}_{j^*}}, \dots, \mathsf{m}_{\mathsf{t}_{j^*}+\mathsf{n}_{j^*}}) = 0$, note that since $j^* > i^\star$ we know all the messages needed to define E_{j^*}).

Putting together all the cases analyzed above, one can see that, when $\mathcal{B}$ does not abort, it provides a perfect simulation to $\mathcal{A}$ and always finds a forgery against HS. Hence, $\mathrm{Adv}_{\mathcal{B}}^{\mathsf{HS}} = \mathrm{Adv}_{\mathcal{A}}^{\mathsf{MKHS}} \Pr[\mathcal{B}$ does not abort$]$. Since the simulation provided by $\mathcal{B}$ to $\mathcal{A}$ is perfect, the index j^* is completely hidden to $\mathcal{A}$. Also, $\mathcal{B}$ does not abort when j^* equals an appropriate index (in each forgery case), which happens with probability at least $1/Q_{\mathsf{id}}$. □

Paper C

Anonymous Single-Round Server-Aided Verification

Elena Pagnin, Aikaterini Mitrokotsa, and Keisuke Tanaka

Abstract. Server-Aided Verification (SAV) is a method that can be employed to speed up the process of verifying signatures by letting the verifier outsource part of its computation load to a third party. Achieving fast and reliable verification under the presence of an untrusted server is an attractive goal in cloud computing and internet of things scenarios.

In this paper, we describe a simple framework for SAV where the interaction between a verifier and an untrusted server happens via a single-round protocol. We propose a security model for SAV that refines existing ones and includes the new notions of SAV-*anonymity* and *extended unforgeability*. In addition, we apply our definitional framework to provide the first generic transformation from any signature scheme to a single-round SAV scheme that incorporates verifiable computation. Our compiler identifies two independent ways to achieve SAV-anonymity: *computationally*, through the privacy of the verifiable computation scheme, or *unconditionally*, through the adaptibility of the signature scheme.

Finally, we define three novel instantiations of SAV schemes obtained through our compiler. Compared to previous works, our proposals are the only ones which simultaneously achieve existential unforgeability and soundness against collusion.

Keywords. Server-Aided Verification, Digital Signatures, Anonymity, Verifiable Computation.

Proceedings of the 5th International Conference on Cryptology and Information Security (LATINCRYPT), 2017.

Anonymous Single-Round Server-Aided Verification

1 Introduction

The design of new efficient and secure signature schemes is often a challenging task, especially when the target devices on which the scheme should run have *limited resources*, as it happens in the Internet of Things (IoT). Nowadays many IoT devices can perform quite *expensive* computations. For instance, smartphones have gained significant computational power. Carrying out several expensive tasks, however, leads to undesirable consequences as, *e.g.,* draining the battery of the device [11]. We consider signed auctions as a motivating example in an IoT setting. In signed auctions, bidders sign their offers to guarantee that the amount is correct and that the offer belongs to them. The auctioneer considers a bid valid only if its signature is verified. Imagine that the auctioneer checks the validity of the bids using a resource-limited device. In this case, running the signature verification algorithm several times drastically affects the device's performance. In this setting one may wonder:

> Can an auctioneer *efficiently*, *securely* and *privately* check the authenticity of signed bids using a *resource-limited* device?

This paper addresses the above question in case the auctioneer has access to a computationally powerful, yet untrusted, server. This is indeed the setting of server-aided verification.

1.1 Previous Work

The concept of Server-Aided Verification (SAV) was introduced in the nineties in two independent works [1, 18], and refined for the case of signature and authentication schemes by Girault and Lefranc [15]. The aim of SAV is to guarantee security and reliability of the outcome of a verification procedure when part of the computation is offloaded from a trusted device, called the verifier, to an untrusted one, the server.

All existing security models for SAV consider existential forgery attacks, where the adversary, *i.e.,* the malicious server, tries to convince the verifier that an invalid signature is valid [8, 15, 21, 23, 25, 26]. Despite the fundamental theoretical contributions, [15] did not consider attack scenarios in which the malicious signer colludes with the server, *e.g.,* by getting control over the server, in order to tamper with the outcome of the server-aided verification of a signature. The so-called collusion attack was defined by Wu *et al.* in [25, 26] together with two SAV schemes claimed to be collusion-resistant. Subsequent works revisited the notion of signer-server collusion [8, 22, 23]. The most complete and rigorous definition of collusion attack is due to Chow *et al.* [8], who also showed that the protocols in [25] are no longer collusion resistant under the new definition [7, 8]. Recently, Cao *et al.* [7] rose new concerns about the artificiality and the expensive communication costs of the SAV in [26].

Chow *et al.* [8] showed that the enabler of many attacks against SAV is the absence of an integrity check on the results returned by the server. Integrity however, is not the only concern when outsourcing computations. In this paper, we address for the first time privacy concerns and we introduce the notion of anonymity in the context of SAV of signatures.

1.2 Contributions

The main motivation of this work is the need for formal and realistic definitions in the area of server-aided verification. To this purpose we:

- Introduce a formalism which allows for an intuitive description of *single-round* SAV signature schemes (Section 3);

- Define a security model that includes three new security notions: SAV-*anonymity* (Section 4.3), *extended* existential unforgeability and extended *strong* unforgeability (Section 4.1);

- Describe the *first compiler* to a SAV signature scheme from any signature and a verifiable computation scheme (Section 5). Besides its simplicity, our generic composition identifies *sufficient requirements* on the underlying primitives to achieve security. In particular, we prove that under mild assumptions our compiler provides: extended (existential/strong) unforgeability (Theorem 4.1); soundness against collusion (Theorem 4.2); and SAV-anonymity when either the employed verifiable computation is private (Theorem 4.3) or the signature scheme is adaptive (Theorem 4.4).

- Apply our generic composition to obtain *new* SAV schemes for the BLS signature [3] (Section 6.1), Waters' signature Wat [24] (Section 6.2) and the *first* SAV for the CL signature by Camenisch and Lysyanskaya [5] (Section 6.3). While preserving efficiency, our proposals achieve better security than previous works (Table C.1).

2 Preliminaries

Throughout the paper, $x \leftarrow A(y)$ denotes the output x of an algorithm A run with input y. If X is a finite set, by $x \xleftarrow{\$} X$ we mean x is sampled from the uniform distribution over the set X. The expression $cost(A)$ refers to the computational cost of running algorithm A. For any positive integer n, $[n] = \{1, \ldots, n\}$ and $\mathbb{G}_n$ is a group of order n. A function $f : \mathbb{N} \to \mathbb{R}$ is said to be *negligible* if $f(n) < 1/poly(n)$ for any polynomial $poly(\cdot)$ and any $n > n_0$, for suitable $n_0 \in \mathbb{N}$. Finally, ε denotes a negligible function.

2.1 Signature schemes

Signature schemes [4, 5, 13] enable one to sign a message in such a way that anyone can verify the signature and be convinced that the message was created by the signer. Formally,

Definition 4.1 (Signature scheme). *A signature scheme* $\Sigma = (\mathsf{SetUp}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *consists of four, possibly randomized, polynomial time algorithms where:*

$\mathsf{SetUp}(1^\lambda)$*: on input the security parameter* $\lambda \in \mathbb{N}$, *the setup algorithm returns the global parameters* gp *of the scheme, which include a description of the* *message and the signature spaces $\mathcal{M}$, $\mathcal{S}$. The* gp *are input to all the following algorithms, even when not specified.*

$\mathsf{KeyGen}()$*: the key generation outputs public-secret key pairs* $(\mathsf{pk}, \mathsf{sk})$*.*

$\mathsf{Sign}(\mathsf{sk}, m)$*: on input a secret key* sk *and a message $m \in \mathcal{M}$, the sign algorithm outputs a signature $\sigma \in \mathcal{S}$ for m.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$*: The verification algorithm is a deterministic algorithm that given a public key* pk*, a message* $\mathsf{m} \in \mathcal{M}$ *and a signature $\sigma \in \mathcal{S}$, outputs* $\mathsf{b} = 1$ *for acceptance, or* $\mathsf{b} = 0$ *for rejection.*

Definition 4.2 ((In)Valid signatures)**.** *Let Σ be a signature scheme. We say that a signature $\sigma \in \mathcal{S}$ is valid for a message $m \in \mathcal{M}$ under the key* pk *if* $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$*. Otherwise, we say that σ is invalid.*

In this paper, we refer to (in)valid *signatures* also as (in)valid message-signature *pairs.*

2.2 Verifiable computation

Verifiable computation schemes enable a client to delegate computations to one or more untrusted servers, in such a way that one can efficiently verify the correctness of the result returned by the server [2, 12]. Gennaro *et al.* [14] formalised private verification of outsourced computations as:

Definition 4.3 (Verifiable Computation scheme [14])**.** *A verifiable computation scheme* $\Gamma = (\mathsf{KeyGen}, \mathsf{ProbGen}, \mathsf{Comp}, \mathsf{Verify})$ *consists of four possibly randomized algorithms where:*

$\mathsf{KeyGen}(\lambda, f)$**:** *given the security parameter λ and a function f, the key generation algorithm produces a public key* pk*, that encodes the target function f, and a secret key* sk*.*

$\mathsf{ProbGen}(\mathsf{sk}, x)$**:** *given the secret key* sk *and the input data x, the problem generation algorithm outputs a public value ω_x and a private value τ_x.*

$\mathsf{Comp}(\mathsf{pk}, \omega_x)$**:** *given the public key* pk *and the encoded input ω_x, this algorithm computes ω_y, which is an encoding of $y = f(x)$.*

$\mathsf{Verify}(\mathsf{sk}, \tau_x, \omega_y)$**:** *given* sk*, τ_x and the encoded result ω_y, the verification algorithm returns y if ω_y is a valid encoding of $f(x)$, and $\perp$ otherwise.*

A verifiable scheme is efficient if verifying the outsourced computation requires less computational effort than computing the function f on the data x, *i.e.,*

$$cost(\mathsf{ProbGen}) + cost(\mathsf{Verify}) < cost(f(x)).$$

In the remainder of the paper, we often drop the indexes and write $\tau_x = \tau$, $\omega_x = \omega$, $\omega_y = \rho$ and denote by y the output of $\mathsf{Verify}(\mathsf{sk}, \tau, \rho)$.

3 Single-round server-aided verification

In the context of signatures, server-aided verification is a method to improve the efficiency of a resource-limited verifier by outsourcing part of the computation load required in the signature verification to a computationally powerful server. Intuitively SAV equips a signature scheme with:

- An additional SAV.VSetup algorithm that sets up the server-aided verification and outputs a public component pb (given to the server) and a private one pr (held by the verifier only). [21]
- An interactive protocol AidedVerify executed between the verifier and the server that outputs: 0 if the input signature is invalid; 1 if the input signature is valid; and $\perp$ otherwise, *e.g.,* when the server returns values that do not match the expected output of the outsourced computation.

In this work, we want to reduce the communication cost of AidedVerify and restrict this to a single-round (two-message) interactive protocol. This choice enables us to describe the AidedVerify protocol as a sequence of three algorithms: SAV.ProbGen (run by the verifier), SAV.Comp (run by the server) and SAV.Verify (run by the verifier). This limitation is less restrictive than it may appear: all the instantiations of SAV signature schemes in [15, 17, 21, 23, 25, 26, 28] are actually single-round SAV.

We define single-round server-aided verification signature schemes as:

Definition 4.4 (SAV). *A single-round server-aided verification signature scheme is defined by the following possible randomized algorithms:*

$\mathsf{SAV.Init}(1^\lambda)$**:** *on input the security parameter $\lambda \in \mathbb{N}$, the initialisation algorithm returns the global parameters* gp *of the scheme, which are input to all the following algorithms, even when not specified.*

$\mathsf{SAV.KeyGen}()$**:** *the key generation algorithm outputs a secret key* sk *(used to sign messages) and the corresponding public key* pk.

$\mathsf{SAV.VSetup}()$**:** *the server-aided verification setup algorithm outputs a public verification-key* pb *and a private one* pr.

$\mathsf{SAV.Sign}(\mathsf{sk}, m)$**:** *given a secret key* sk *and a message* m *the sign algorithm produces a signature σ.*

$\mathsf{SAV.ProbGen}(\mathsf{pr}, \mathsf{pk}, m, \sigma)$ **:** *on input the private verification key* pr*, the public key* pk*, a message m and a signature σ, this algorithm outputs a public-private data pair (ω, τ) for the server-aided verification.*

$\mathsf{SAV.Comp}(\mathsf{pb}, \omega)$**:** *on input the public verification key* pb *and ω the outsourced-computation algorithm returns ρ.*

$\mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, m, \sigma, \rho, \tau)$**:** *the verification algorithm takes as input the private verification-key* pr*, the public key* pk*, m, σ, ρ and τ. The output is $\Delta \in \{0, 1, \perp\}$.*

Intuitively, the output Δ of SAV.Verify has the following meanings:

[21] In [8, 25] the output of SAV.VSetup is called **Vstring**.

- $\Delta = 1$: the pair (m, σ) is considered valid and we say that (m, σ) *verifies in the server-aided sense*;
- $\Delta = 0$: the pair (m, σ) is considered invalid and we say that (m, σ) does *not verify in the server-aided sense*;
- $\Delta = \perp$: the server-aided verification has failed, ρ is rejected (not σ), and nothing is inferred about the validity of (m, σ).

Unless stated otherwise, from now on SAV refers to a single-round server-aided signature verification scheme as in Definition 4.4. Definition 4.4 implicitly allows to delegate the computation of several inputs, as long as all inputs can be sent in a single round, as a vector ω.

Completeness and efficiency of SAV are defined as follows.

Definition 4.5 (SAV completeness). *A* SAV *is said to be complete if for all* $\lambda \in \mathbb{N}$, $\mathsf{gp}{\leftarrow}\mathsf{SAV.Init}(1^\lambda)$*, for any* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SAV.KeyGen}()$*,* $(\mathsf{pb}, \mathsf{pr}) \leftarrow \mathsf{SAV.VSetup}()$ *and message* $m \xleftarrow{\$} \mathcal{M}$*; given* $\sigma \leftarrow \mathsf{SAV.Sign}(\mathsf{sk}, m)$*,* $(\omega, \tau) \leftarrow \mathsf{SAV.ProbGen}(\mathsf{pr}, \mathsf{pk}, m, \sigma)$ *and* $\rho \leftarrow \mathsf{SAV.Comp}(\mathsf{pb}, \omega)$*, it holds:*

$$\mathsf{Prob}[\mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, m, \sigma, \rho, \tau) = 1] > 1 - \varepsilon$$

where the probability is taken over the coin tosses of SAV.Sign*,* SAV.ProbGen*.*

Definition 4.6 (SAV efficiency). *A* SAV *for a signature scheme* $\Sigma = (\mathsf{SetUp}_\Sigma, \mathsf{KeyGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verify}_\Sigma)$ *is said to be efficient if the computational cost of the whole server-aided verification is less than the cost of running the standard signature verification, i.e.,*

$$\big(cost(\mathsf{SAV.ProbGen}) + cost(\mathsf{SAV.Verify})\big) < cost(\mathsf{Verify}_\Sigma) \,.$$

4 Security model

In server-aided verification there are two kinds of adversaries to be considered: the one that solely controls the server used for the aided-verification, and the one that additionally knows the secret key for signing (signer-server collusion). In the first case, we are mostly concerned about forgeries against the signature scheme, while in the second scenario we want to avoid some kind of repudiation [7]. Existing security models for SAV consider existential unforgeability (EUF) and soundness against collusion (SAC) [8, 25]. In this section, we extend the notion of EUF to capture new realistic attack scenarios and we consider for the first time signer anonymity in server-aided verification.

In what follows, the adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm. We denote by q_s (*resp.* q_v) the upper bound on the number of signature (*resp.* verification) queries in each query phase.

4.1 Unforgeability

Intuitively, a SAV signature scheme is unforgeable if a malicious server, taking part to the server-aided verification process, is not able to tamper with the output of the protocol. All the unforgeability notions presented in this section are based on the unforgeability under chosen message and verification attack (UF-ACMV) experiment:

Definition 4.7. *The unforgeability under chosen message and verification experiment* ($\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda]$) *goes as follows:*

Setup. *The challenger* $\mathcal{C}$ *runs the algorithms* SAV.Init, SAV.KeyGen *and* SAV.-VSetup *to obtain the system parameters* gp, *the key pair* (pk, sk), *and the public-private verification keys* (pb, pr). *The adversary* $\mathcal{A}$ *is given* pk, pb, *while* sk *and* pr *are withheld from* $\mathcal{A}$.

Query Phase I. *The adversary can make a series of queries which may be of the following two kinds:*

- *sign:* $\mathcal{A}$ *chooses a message* m *and sends it to* $\mathcal{C}$. *The challenger behaves as a signing oracle: it returns the value* $\sigma \leftarrow \mathsf{SAV.Sign}(\mathsf{sk}, \mathsf{m})$ *and stores the pair* (m, σ) *in an initially empty list* $L \subset \mathcal{M} \times \mathcal{S}$.

- *verify:* $\mathcal{A}$ *begins the interactive (single-round) protocol for server-aided verification by supplying a message-signature pair* (m, σ) *to its challenger.* $\mathcal{C}$ *simulates a verification oracle: it runs* $\mathsf{SAV.ProbGen}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}, \sigma) \to (\omega, \tau)$, *returns* ω *to* $\mathcal{A}$, *and waits for a second input. Upon receiving an answer* ρ *from the adversary, the challenger returns* $\Delta \leftarrow \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}, \sigma, \rho, \tau)$.

The adversary can choose its queries adaptively based on the responses to previous queries, and can interact with both oracles at the same time.

Challenge. $\mathcal{A}$ *chooses a message-signature pair* (m^*, σ^*) *and sends it to* $\mathcal{C}$. *The challenger computes* $(\hat{\omega}, \hat{\tau}) \leftarrow \mathsf{SAV.ProbGen}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*)$. *The value* $\hat{\tau}$ *is stored and withheld from* $\mathcal{A}$, *while* $\hat{\omega}$ *is sent to the adversary.*

Query Phase II. *In the second query phase the sign queries are as before, while the verify queries are answered using the same* $\hat{\tau}$ *generated for the challenge, i.e.,* $\mathcal{A}$ *submits only* ρ *and* $\mathcal{C}$ *replies with* $\Delta \leftarrow \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho, \hat{\tau})$.

Forgery. $\mathcal{A}$ *outputs the tuple* $(\mathsf{m}^*, \sigma^*, \rho^*)$.

The experiment outputs 1 if $(\mathsf{m}^*, \sigma^*, \rho^*)$ *is a forgery (see Definition 4.8), and 0 otherwise.*

Unlike unforgeability for digital signatures, in SAV the adversary can influence the outcome of the signature verification through the value ρ^*. Moreover, $\mathcal{A}$ can perform verification queries. This is a crucial requirement as the adversary cannot run SAV.Verify on its own, since pr and τ are withheld from $\mathcal{A}$. In practice, whenever the output of the server-aided verification is $\perp$ the verifier could abort and stop interacting with the malicious server. In this work, we ignore this case and follow the approach used in [8] and in verifiable computation [14] where the adversary 'keeps on querying' independently of the outcome of the verification queries.

Definition 4.8 (Forgery). *Consider an execution of the* UF-ACMV *experiment where* $(\mathsf{m}^*, \sigma^*, \rho^*)$ *is the tuple output by the adversary. We define three types of forgery:*

type-1a forgery: $(\mathsf{m}^*, \cdot) \notin L$ *and* $1 \leftarrow \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho^*, \hat{\tau})$.
type-1b forgery: $(\mathsf{m}^*, \sigma^*) \notin L$ *and* $1 \leftarrow \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho^*, \hat{\tau})$.
type-2 forgery: $(\mathsf{m}^*, \sigma^*) \in L$ *and* $0 \leftarrow \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho^*, \hat{\tau})$,

Existential unforgeability for SAV signature schemes is defined for a quite weak adversary: the second query phase is skipped and only type-1a forgeries are considered:

Definition 4.9 (Existential Unforgeability (EUF) [8]). *A* SAV *scheme is* (ε, q_s, q_v)-*existentially unforgeable under adaptive chosen message and verification attacks if* $\mathsf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda] = 1] < \varepsilon$ *and the experiment* $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda]$ *outputs 1 only on* ***type-1a forgeries****, and no query is performed in the* ***Query Phase II***.

This notion of unforgeability fails to capture some realistic attack scenarios. For instance, consider the case of signed auctions. The adversary is a bidder and wants to keep the price of the goods he is bidding on under a certain threshold. A simple way to achieve this goal is to get control over the server used for the SAV and prevent signatures of higher bids from verifying correctly. This motivates us to *extend* the notion of EUF in [8, 25] to also account for malicious servers tampering with the verification outcome of honestly generated message-signature pairs:

Definition 4.10 (Extended Existential Unforgeability (ExEUF)). *A* SAV *scheme is* (ε, q_s, q_v)*-extended existentially unforgeable under adaptive chosen message and verification attacks if* $\mathsf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda] = 1] < \varepsilon$ *and the experiment* $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda]$ *outputs 1 on* ***type-1a*** *and* ***type-2 forgeries****.*

Extended existential unforgeablility deals with a stronger adversary than the one considered in EUF: in ExEUF the adversary can perform two different types of forgeries and has access to an additional query phase (after setting the challenge). Resembling the notion of the strongly unforgeable signatures [4], we introduce *extended strong* unforgeability for SAV:

Definition 4.11 (Extended Strong Unforgeability (ExSUF)). *A* SAV *scheme is* (ε, q_s, q_v)*-extended strong unforgeable under adaptive chosen message and verification attacks if* $\mathsf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda] = 1] < \varepsilon$ *and* $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda]$ *outputs 1 on* ***type-1a, type-1b*** *and* ***type-2 forgeries****.*

In ExSUF there is no restriction on the pair (m^*, σ^*) chosen by the adversary: it can be a new message (type-1a), a new signature on a previously-queried message (type-1b) or an honestly generated pair obtained in the first Query Phase (type-2).

4.2 Soundness against collusion

In collusion attacks, the adversary controls the server used for the aided verification and holds the signer's secret key. This may happen when a malicious signer hacks the server and wants to tamper with the outcome of a signature verification. As a motivating example consider signed auctions. The owner of a good could take part to the auction (as the malicious signer) and influence its price. For instance, in order to increase the cost of the good, the malicious signer can produce an invalid signature for a high bid (message) and make other bidders overpay for it. To tamper with the verification of the invalid signature, the malicious signer can use the server and make his (invalid) signature verify when the bid is stated. However, in case no one outbids him, the malicious signer can repudiate the signature as it is actually invalid.

We define collusion as in [8], with two minor adaptations: (i) we use our single-round framework, that allows us to clearly state the information flow between $\mathcal{A}$ and $\mathcal{C}$; and (ii) we introduce a second query phase, after the challenge phase (to strengthen the adversary).

Definition 4.12 (Soundness Against Collusion (SAC)). *Define* $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ACVAuC}}[\lambda]$ *to be* $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{UF\text{-}ACMV}}[\lambda]$ *where:*
- in the ***Setup*** *phase,* $\mathcal{C}$ *gives to* $\mathcal{A}$ *all keys except* pr*, and*
- no sign query is performed, and
- the tuple $(\mathsf{m}^*, \sigma^*, \rho^*)$ *output by* $\mathcal{A}$ *at the end of the experiment is considered* ***forgery*** *if* $\Delta \leftarrow \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho^*, \hat{\tau})$ *is such that* $\Delta \neq \perp$ *and*

$\Delta \neq \mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho, \hat{\tau})$, *where* $\rho \leftarrow \mathsf{SAV.Comp}(\mathsf{pb}, \hat{\omega})$ *is generated honestly. A* SAV *signature scheme is* (ε, q_v)*-sound against adaptive chosen verification attacks under collusion if* $\mathsf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ACVAuC}}[\lambda] = 1] < \varepsilon$.

Definition 4.12 highlights connections between the notions of extended existential unforgeability and soundness against collusion. In particular, it is possible to think of collusion attacks as unforgeability attacks where $\mathcal{A}$ possesses the signing secret key sk (and thus no *sign* query is needed), and a forgery is a tuple for which the output of the server-aided verification does not coincide with the correct one, *e.g.,* if $\sigma^* \leftarrow \mathsf{SAV.Sign}(\mathsf{sk}, \mathsf{m}^*)$ then $\mathsf{SAV.Verify}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}^*, \sigma^*, \rho^*, \hat{\tau})$ returns 0.

4.3 Anonymity

We initiate the study of anonymity in the context of server-aided verification of signatures and provide the first definition of SAV-anonymity.

Consider the running setting of signed auctions. If a malicious server can distinguish whose signature it is performing the aided-verification of, it can easily 'keep out' target bidders from the auction by preventing their signatures from verifying (in the server aided sense). To prevent such an attack, bidders may want to hide their identity from the untrusted server. SAV-anonymity guarantees precisely this: the auctioneer (trusted verifier) learns the identities of the bidders (signers), while the untrusted server is not able to determine whose signature was involved in the SAV.

Definition 4.13 (SAV-anonymity). *A* SAV *scheme is* (ε, q_v) *-* SAV*-anonymous if* $\mathsf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SAV\text{-}anon}}[\lambda] = 1] < \frac{1}{2} + \varepsilon$ *and* $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{SAV\text{-}anon}}[\lambda]$ *is:*

Setup. *The challenger runs the algorithms* SAV.Init*,* SAV.VSetup *to obtain the system parameters and the verification keys* (pb, pr)*. Then it runs* SAV.KeyGen *twice to generate* $(\mathsf{sk}_0, \mathsf{pk}_0), (\mathsf{sk}_1, \mathsf{pk}_1)$ *and draws* $b \xleftarrow{\$} \{0, 1\}$. $\mathcal{C}$ *gives* pb, $\mathsf{pk}_0, \mathsf{pk}_1$ *to* $\mathcal{A}$ *and retains the secret values* pr, $\mathsf{sk}_0, \mathsf{sk}_1$.

Query I. $\mathcal{A}$ *can adaptively perform up to* q_v *partial-verification queries as follows. The adversary sends a pair* (m, i)*,* $i \in \{0, 1\}$ *to* $\mathcal{C}$*. The challenger computes* $\sigma \leftarrow \mathsf{SAV.Sign}(\mathsf{sk}_i, m)$*, runs* $\mathsf{SAV.ProbGen}(\mathsf{pr}, \mathsf{pk}_i, \mathsf{m}, \sigma) \to (\omega, \tau)$ *and returns* ω *to* $\mathcal{A}$.

Challenge. *The adversary chooses a message* m^* *to be challenged on, and sends it to* $\mathcal{C}$*. The challenger computes* $\sigma \leftarrow \mathsf{SAV.Sign}(\mathsf{sk}_b, \mathsf{m}^*)$ *and* $(\omega, \tau) \leftarrow \mathsf{SAV.ProbGen}(\mathsf{pr}, \mathsf{pk}_b, \mathsf{m}^*, \sigma)$*; and sends* ω *to the adversary.*

Query II. $\mathcal{A}$ *can perform another query phase, as in Query I.*

Output. *The adversary outputs a guess* $b^* \in \{0, 1\}$ *for the identity* b *chosen by* $\mathcal{C}$*. The experiment outputs 1 if* $b^* = b$ *and 0 otherwise.*

The fundamental difference between anonymity for signatures schemes [13, 27] and SAV-anonymity lies in the choice of the challenge message m^*. In the former case, it is chosen by the challenger at random, while in SAV we let the adversary select it. This change increases the adversary's power and reflects several application scenarios where $\mathcal{A}$ learns the messages (*e.g.,* bids in signed auctions). We remark that in SAV-anonymity the adversary does not have access to the verification outcome Δ, as this would correspond to having a verification oracle, which is not allowed in the anonymity game for signature schemes [13, 27].

5 A compiler for SAV

We present here the first generic compiler for server-aided verification of signatures. Our generic composition method allows to combine any signature scheme Σ with an efficient verifiable computation scheme Γ for a function f involved in the signature verification algorithm, and outputs $\mathsf{SAV}_{\Sigma}^{\Gamma}$, a single-round server-aided verification scheme for Σ.

The idea to employ verifiable computation in SAV comes from the following observation. All the attacks presented in [8] succeed because in the target SAV schemes the verifier never checks the validity of the values returned by the server. We leverage the efficiency and security properties of verifiable computation to mitigate such attacks.

5.1 Description of our compiler

Let $\Sigma = (\mathsf{SetUp}_{\Sigma}, \mathsf{KeyGen}_{\Sigma}, \mathsf{Sign}_{\Sigma}, \mathsf{Verify}_{\Sigma})$ be a signature scheme and $\Gamma = (\mathsf{KeyGen}^{\Gamma}, \mathsf{ProbGen}^{\Gamma}, \mathsf{Comp}^{\Gamma}, \mathsf{Verify}^{\Gamma})$ be a verifiable computation scheme.[22] In our generic composition, we identify a computationally-expensive sub-routine of Verify_{Σ} that we refer to as $\mathsf{Ver_H}$ (the *heavy* part of the signature verification); and we outsource $f = \mathsf{Ver_H}$ using the verifiable computation scheme Γ. To ease the presentation, we introduce:

$\mathsf{ProbGen}^{\mathsf{PRE}}$: This algorithm prepares the input to $\mathsf{ProbGen}^{\Gamma}$.

$\mathsf{Ver_L}$: This algorithm is the computationally *light* part of the signature verification. More precisely, $\mathsf{Ver_L}$ is Verify_{Σ} where $\mathsf{Ver_H}$ is replaced by the output y of Verify^{Γ}. It satisfies: $cost(\mathsf{Ver_L}) < cost(\mathsf{Verify}_{\Sigma})$ and $\mathsf{Ver_L}(\mathsf{pk}_{\Sigma}, m, \sigma, y) = \mathsf{Verify}_{\Sigma}(\mathsf{pk}, m, \sigma)$ whenever $y \neq \perp$.

Definition 4.14 ($\mathsf{SAV}_{\Sigma}^{\Gamma}$)**.** *Let Σ, Γ and f be as above. Our generic composition method for single-round server-aided verification signature scheme $\mathsf{SAV}_{\Sigma}^{\Gamma}$ is defined by the following possibly randomized algorithms:*

$\mathsf{SAV.Init}(1^{\lambda})$*: the initialisation algorithm outputs the global parameters* $\mathsf{gp} \leftarrow \mathsf{SetUp}_{\Sigma}(1^{\lambda})$*, which are implicitly input to all the algorithms.*

$\mathsf{SAV.KeyGen}()$*: this algorithm outputs* $(\mathsf{pk}_{\Sigma}, \mathsf{sk}_{\Sigma}) \leftarrow \mathsf{KeyGen}_{\Sigma}()$.

$\mathsf{SAV.Sign}(\mathsf{sk}_{\Sigma}, m)$*: the sign algorithm outputs* $\sigma \leftarrow \mathsf{Sign}_{\Sigma}(\mathsf{sk}_{\Sigma}, \mathsf{m})$.

$\mathsf{SAV.VSetup}()$*: the verification setup algorithm outputs a pair of verification keys* $(\mathsf{pk}^{\Gamma}, \mathsf{sk}^{\Gamma}) \leftarrow \mathsf{KeyGen}^{\Gamma}(\lambda, f)$*, where the function f is described in* gp.

$\mathsf{SAV.ProbGen}(\mathsf{sk}^{\Gamma}, \mathsf{pk}_{\Sigma}, m, \sigma)$*: this algorithm first runs* $\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}_{\Sigma}, m, \sigma) \rightarrow x$ *to produce an encoding of* pk_{Σ}*, m, σ. Then x is used to compute the output* $(\omega, \tau) \leftarrow \mathsf{ProbGen}^{\Gamma}(\mathsf{sk}^{\Gamma}, x)$.

$\mathsf{SAV.Comp}(\mathsf{pk}^{\Gamma}, \omega)$*: this algorithm returns* $\rho \leftarrow \mathsf{Comp}^{\Gamma}(\mathsf{pk}^{\Gamma}, \omega)$.

[22]To improve readability, we put the subscript Σ (*resp.* superscript Γ) to each algorithm related to the signature (*resp.* verifiable computation) scheme.

$\mathsf{SAV.Verify}(\mathsf{sk}^\Gamma, \mathsf{pk}_\Sigma, m, \sigma, \rho, \tau)$: *the verification algorithm executes* $\mathsf{Verify}^\Gamma(\mathsf{sk}^\Gamma, \rho, \tau) \to y$*; if* $y = \perp$*, it sets* $\Delta = \perp$ *and returns. Otherwise, it returns the output of* $\mathsf{Ver_L}(\mathsf{pk}_\Sigma, m, \sigma, y) \to \Delta \in \{0,1\}$.

Intuitively, the SAV.ProbGen algorithm prepares the inputs for the delegated computations (ω) and the private values for the verification of computations (τ). The SAV.Comp algorithm performs the verifiable delegation of the bilinear pairing computation, and returns ρ, which includes the encoding of the bilinear pairing and some additional values to prove the correctness of the performed operations. Finally, SAV.Verify checks the correctness of the values received by the server, and proceed with the (light-weight) verification of the signature, only if the server has behaved according to the protocol.

Completeness of $\mathsf{SAV}_\Sigma^\Gamma$. The correctness of $\mathsf{SAV}_\Sigma^\Gamma$ is a straight-forward computation assuming that Σ is complete and Γ is correct (see the extended version of this paper [19] for a detailed proof).

Efficiency of $\mathsf{SAV}_\Sigma^\Gamma$. It is immediate to check that $cost(\mathsf{Verify}_\Sigma) = cost(\mathsf{Ver_L}) + cost(\mathsf{Ver_H})$. The $\mathsf{ProbGen}^{\mathsf{PRE}}$ algorithm is just performing encodings of its inputs (usually projections), and does not involve computationally expensive operations.[23] By the efficiency of verifiable computation schemes we have: $cost(\mathsf{Ver_H}) > cost(\mathsf{ProbGen}^\Gamma) + cost(\mathsf{Verify}^\Gamma)$ and thus

$$cost(\mathsf{Verify}_\Sigma) > cost(\mathsf{ProbGen}^{\mathsf{PRE}}) + cost(\mathsf{ProbGen}^\Gamma) + cost(\mathsf{Verify}^\Gamma) + cost(\mathsf{Ver_L}),$$

which proves the last claim.

Our generic composition enjoys two additional features: it applies to *any* signature scheme and it allows to reduce the security of $\mathsf{SAV}_\Sigma^\Gamma$ to the security of its building blocks, Σ and Γ. To demonstrate the first claim, let us set $f = \mathsf{Ver_H} = \mathsf{Verify}_\Sigma$ and $\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}_\Sigma, \mathsf{m}, \sigma) \to x = (\mathsf{pk}_\Sigma, \mathsf{m}, \sigma)$. The correctness of Γ implies that $y = \mathsf{Ver_H}(x) = \mathsf{Verify}_\Sigma(\mathsf{pk}_\Sigma, \mathsf{m}, \sigma)$. In this case, $\mathsf{Ver_L}(\mathsf{pk}_\Sigma, \mathsf{m}, \sigma, y)$ is the function that returns 1 if $y = 1$ and 0 otherwise. We defer the proof of the second claim to the following section.

5.2 Security of our generic composition

The following theorems state the security of the compiler presented in Definition 4.14. Our approach is to identify sufficient requirements on Σ and Γ to guarantee specific security properties in the resulting $\mathsf{SAV}_\Sigma^\Gamma$ scheme. All the proofs can be found in the extended version of this paper [19]. We highlight that the results below apply to all our instantiations of the SAV signature schemes presented in Section 6, since these are obtained via our generic composition method.

Theorem 4.1 (Extended Unforgeability of $\mathsf{SAV}_\Sigma^\Gamma$)**.** *Let* Σ *be an* $(\varepsilon_\Sigma, q_s)$*-existentially (resp. strongly) unforgeable signature scheme, and* Γ *an* $(\varepsilon^\Gamma, q_v)$*-secure verifiable computation scheme. Then* $\mathsf{SAV}_\Sigma^\Gamma$ *is* $(\frac{\varepsilon_\Sigma + \varepsilon^\Gamma}{2}, q_s, q_v)$*-extended existentially (resp. strongly) unforgeable.*

The proof proceeds by reduction transforming type-1a (*resp.* type-1b) forgeries into existential (*resp.* strong) forgeries against Σ; and type-2 forgeries, into forgeries against the security of Γ.

[23]This claim will become clear after seeing examples of SAV signature schemes.

Theorem 4.2 (Soundness Against Collusion of $\mathsf{SAV}_\Sigma^\Gamma$)**.** *Let Σ be a correct signature scheme and Γ an $(\varepsilon^\Gamma, q_v)$-secure verifiable computation scheme. Then $\mathsf{SAV}_\Sigma^\Gamma$ is $(\varepsilon^\Gamma, q_v)$-sound against collusion.*

The intuition behind the proof of Theorem 4.2 is the same as in Theorem 4.1 for the case of type-2 forgeries.

We present now two independent ways to achieve SAV-anonymity for schemes obtained with our compiler: leveraging either the privacy of the verifiable computation scheme or the adaptibility of the signature scheme.

Theorem 4.3 (Anonymity of $\mathsf{SAV}_\Sigma^\Gamma$ from Private Verification)**.** *Let Σ be a correct signature scheme and Γ an $(\varepsilon^\Gamma, q_v)$-private verifiable computation scheme. Then $\mathsf{SAV}_\Sigma^\Gamma$ is $(\varepsilon^\Gamma, q_v)$-SAV-anonymous.*

Theorem 4.3 does not require Σ to be anonymous and SAV-anonymity comes directly from the privacy of the verifiable computation scheme.

Key-homomorphic signatures have been recently introduced by Derler and Slamanig [10]. In a nutshell, a signature scheme provides *adaptibility* of signatures [10] if given a signature σ for a message m under a public key pk, it is possible to publicly create a valid σ' for the same message m under a new public key pk'. In particular, there exists an algorithm Adapt that, given pk, m, σ and a shift amount h, returns a pair (pk', σ') for which $\mathsf{Verify}(\mathsf{pk}', \mathsf{m}, \sigma') = 1$ (cf. Definition 16 in [10] for a formal statement). [24]

Theorem 4.4 (Anonymity of $\mathsf{SAV}_\Sigma^\Gamma$ from Perfect Adaption)**.** *Let Σ be a signature scheme with perfect adaption and Γ a correct verifiable computation scheme. If the output of $\mathsf{ProbGen}^{\mathsf{PRE}}$ depends only on the adapted values, i.e., for all $\mathsf{pr}, \mathsf{pk}, \mathsf{m}, \sigma$ there is a function G such that:*

$$\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pr}, \mathsf{pk}, \mathsf{m}, \sigma) = \mathsf{G}(\mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma, \mathsf{h}), \mathsf{m})$$

for a randomly chosen shift amount h, then $\mathsf{SAV}_\Sigma^\Gamma$ is unconditionally SAV-anonymous.

Theorem 4.4 provides a new application of key-homomorphic signatures to anonymity. The proof is inspired to the tricks used in [10], intuitively SAV-anonymity follows from the indistinguishability of the output of Adapt from $(\mathsf{pk}'', \sigma'' \leftarrow \mathsf{Sign}(\mathsf{sk}'', \mathsf{m}))$ for a freshly generated key pair $(\mathsf{pk}'', \mathsf{sk}'')$. Many signatures based on the discrete logarithm problem enjoy this property, *e.g.,* BLS [3] and Wat [24].

6 New instantiations of SAV schemes

Our generic composition requires the existence of a verifiable computation scheme for a function $f = \mathsf{Ver}_\mathsf{H}$ used in the signature verification algorithm. To the best of the authors' knowledge, there are verifiable computation schemes for arithmetic circuits [9, 20] and bilinear pairings [6], but no result is yet known for simpler computations such as hash functions and group exponentiations. Following previous works' approach, we consider only SAV for pairing-based signatures [8, 21, 25, 28], since bilinear pairings are bottle-neck computations for resource-limited devices.[25]

[24] To provide an example, consider the BLS signature scheme [3]. Given $\mathsf{pk} = g^{\mathsf{sk}}$, $\mathsf{m} \in \{0,1\}^*$, $\sigma \in \mathbb{G}_p$ and $\mathsf{h} \in \mathbb{Z}p$, the output of Adapt can be defined as: $\mathsf{pk}' = \mathsf{pk} \cdot g^{\mathsf{h}}$ and $\sigma' = \sigma \cdot \mathsf{H}(\mathsf{m})^{\mathsf{h}}$. It is immediate to check that (σ', m) is a valid pair under pk'.

[25] To give benchmarks, let M_p denote the computational cost of a base field multiplication in $\mathbb{F}_p$ with $\log p = 256$, then computing z^a for any $z \in \mathbb{F}_p$ and $a \in [p]$ costs about $256 M_p$, while computing the Optimal Ate pairing on the BN curve requires about $16000 M_p$ (results extrapolated from Table 1 in [16]).

$\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}_\Sigma, m, \sigma)$: on input $\mathsf{pk}_\Sigma \in \mathbb{G}_1$, $m \in \{0,1\}^*$ and $\sigma \in \mathbb{G}_1$, the algorithm returns $x = \big((\mathsf{pk}_\Sigma, \mathsf{H}(m)), (\sigma, g)\big)$.

$\mathsf{Ver}_\mathsf{L}(\mathsf{pk}_\Sigma, m, \sigma, y)$: this algorithm is $\mathsf{Verify}_{\mathsf{BLS}}$ where the computation of the two pairings is replaced with the output $y = (y_1, y_2)$ of $\mathsf{Verify}^{\mathsf{CDS}_2}$. Formally, Ver_L checks whether $y_1 = y_2$, in which case it outputs $\Delta = 1$, otherwise it returns $\Delta = 0$.

Figure C.9: The core algorithms of $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$.

All our instantiations of SAV schemes are obtained using the compiler in Definition 4.14. Their security therefore follows from the results of Section 5.2, once shown that that the chosen schemes satisfy the hypothesis of the theorems. For conciseness, we only define the two algorithms $\mathsf{ProbGen}^{\mathsf{PRE}}$ and Ver_L. Appendix C.i contains thorough descriptions.

6.1 A secure SAV for BLS ($\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$)

The BLS signature by Boneh *et al.* [3] has been widely used for constructing server-aided verification schemes, *e.g.,* Protocols I and II in [25]. Cao *et al.* [7] and Chow *et al.* [8] have shown that all the existing SAV for BLS are neither existentially unforgeable, nor sound against collusion. This motivates us to propose $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$ (described in Figure C.9). As a verifiable scheme for the pairing computation, we employ *'a protocol for public variable A and B'* by Canard *et al.* [6], which we refer to as CDS_1. By the correctness of the CDS_1 scheme $y_2 = e(\mathsf{pk}_\Sigma, \mathsf{H}(\mathsf{m}))$ and $y_2 = e(\sigma, g)$, thus Ver_L has the same output as $\mathsf{Verify}_{\mathsf{BLS}}$. Given that BLS is strongly unforgeable in the random oracle model [3] and that CDS_1 is secure in the generic group model [6], $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$ is extended strongly unforgeable and sound against collusion. Our SAV scheme for the BLS is not SAV-anonymous: the signer's public key is given to the server for the aided verification. However, SAV-anonymity can be simply gained via the adaptability of BLS [10].

In $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$ the verifier does not need to perform *any* pairing computation. This is a very essential feature, especially if the verifying device has very limited computational power, *e.g.,* an RFID tag.

6.2 A secure SAV for Wat ($\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$)

Wu *et al.* [25] proposed a SAV for Waters' signature Wat [24], which is neither existentially unforgeable nor sound against collusion. Here we propose $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$ (described in Figure C.10), which is similar to Protocol III in [25], but has strong security guarantees thanks to the verifiable computation scheme for *'public A and B'* CDS_1 [6].

By the correctness of the CDS_1 scheme $y_1 = e(\sigma_1, g)$, and $y_2 = e(\mathsf{H}(\mathsf{m}), \sigma_2)$. Thus, Ver_L has the same output as $\mathsf{Verify}_{\mathsf{Wat}}$. Given that CDS_1 is secure in the generic group model [6], and that Wat is existentially unforgeable in the standard model [24] our $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$ is extended existential unforgeable and sound against collusion. Similarly to Protocol III in [25], $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$ achieves SAV-anonymity thanks to the perfect adaption of Wat [10].

$\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}_\Sigma, m, \sigma)$: given $\mathsf{pk}_\Sigma \in \mathbb{G}_1$, $m \in \{0,1\}^*$ and $\sigma \in \mathbb{G}_1$, select $\mathsf{h} \xleftarrow{\$} \mathbb{Z}p$, compute $(\mathsf{pk'}_\Sigma, \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}_\Sigma, m, \sigma, \mathsf{h})$, return $x = (\mathsf{pk'}_\Sigma, m, \sigma')$.

$\mathsf{Ver}_\mathsf{L}(\mathsf{pk'}_\Sigma, m, \sigma, y)$: this is $\mathsf{Verify}_{\mathsf{Wat}}$ where the computation of the two pairings is replaced with the outputs y_1, y_2 of $\mathsf{Verify}^{\mathsf{CDS}_1}$. Formally, Ver_L checks if $y_1 = \mathsf{pk'}_\Sigma \cdot y_2$, in which case it outputs $\Delta = 1$, otherwise it returns $\Delta = 0$.

Figure C.10: The core algorithms of $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$.

$\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}_\Sigma, m, \sigma)$: this algorithm simply returns the first two entries of the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, *i.e.,* $x = (\sigma_1, \sigma_2)$.

$\mathsf{Ver}_\mathsf{L}(\mathsf{pk}_\Sigma, m, \sigma, y)$: this algorithm is $\mathsf{Verify}_{\mathsf{CL}}$, except for two pairing computations which are replaced with the outcome $y = (\beta_1, \beta_2)$ of $\mathsf{Verify}^{\mathsf{CDS}_2}$. More precisely, the Ver_L algorithm computes $\alpha_1 = e(\sigma_1, Y)$, $\alpha_2 = e(X, \sigma_1)$ and $\alpha_3 = e(X, \sigma_2)^m$. It then checks whether $\alpha_1 = \beta_1$ and $\alpha_2 \cdot \alpha_3 = \beta_2$. If both of the conditions hold, the algorithm returns $\Delta = 1$, otherwise $\Delta = 0$.

Figure C.11: The core algorithms of $\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$.

6.3 The first SAV for CL ($\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$)

The verification of the BLS and the Wat signatures only requires the computation of two bilinear pairings. We want to move the focus to more complex signature schemes that would benefit more of server-aided verification. To this end, we consider scheme A by Camenish and Lysyanskaya [5], which we refer to as CL, where $\mathsf{Verify}_{\mathsf{CL}}$ involves the computation of five bilinear pairings. For verifiability we employ CDS_2, *'a protocol with public constant B and variable secret A'* by Canard *et al.* [6]. Our $\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$ scheme is reported in Figure C.11.
By the correctness of CDS_2 we have: $y_1 = \beta_1 = e(\sigma, g_1)$, and $y_2 = \beta_2 = e(H(m), \mathsf{pk}_\Sigma)$. Therefore Ver_L performs the same checks as $\mathsf{Verify}_{\mathsf{CL}}$ and the two algorithms have the same output. Given that CL is existential unforgeable in the standard model [5] and CDS_2 is secure and private in the generic group model [6], $\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$ is extended-existential unforgeable, sound against collusion and SAV-anonymous. Therefore $\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$ is an example of a scheme which is SAV-anonymous although the base signature scheme is not anonymous.

6.4 Comparison with previous work

Table C.1 gives a compact overview of how our SAV schemes compare to previous proposals in terms of unforgeability, soundness under collusion and SAV-anonymity. We report only the highest level of unforgeability that the scheme provides. A yes (*resp.* no) in the table states that the scheme does (*resp.* does not) achieve the property written at the beginning of the row, *e.g.,* Protocol III does not employ a verifiable computation scheme and provides SAV-anonymity. Every scheme or property is followed by a reference paper or the section where the claim is proven. Regarding efficiency, the computational cost of pairing-based algorithms is influ-

	Protocol I [25]	Protocol II [25]	$\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$	Protocol III [25]	$\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$	SAV-ZSS [15]	$\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$
signature	BLS [3]	BLS [3]	BLS [3]	Wat [24]	Wat [24]	ZSS [28]	CL [5]
verifiability	no	no	CDS_1 [6]	no	CDS_1 [6]	no	CDS_2 [6]
unforgeability	EUF [25]	no [8]	ExSUF (6.1)	no [19]	ExEUF (6.2)	EUF [15]	ExEUF (6.3)
collusion resistance	no [8]	no [19]	yes (6.1)	no [19]	yes (6.2)	no [19]	yes (6.3)
anonymity	no [19]	no [19]	no (6.1)	yes [19]	yes (6.2)	no [19]	yes (6.3)

Table C.1: Comparison among our SAV schemes and previous works: Protocol I (Figure 3 in [25]), Protocol II (Figure 5 in [25]), Protocol III (Figure 4 in [25]), SAV-ZSS [15] (depicted in Figure 1 in [25]).

enced by three main parameters: (*i*) the elliptic curve, (*ii*) the field size, and (*iii*) the bilinear pairing. As a result, it is impossible to state that a given algorithm is efficient for all pairings and for all curves, since even the computational cost of the most basic operations (*e.g.,* point addition) variates significantly with the above parameters. For example, CDS_2 provides a 70% efficiency gain[26] for the delegator (verifier) when the employed pairing is the Optimal Ate pairing on the KSS-18 curve [6], but is nearly inefficient when computed on the BN curve [16].

7 Conclusions

In this paper, we provided a framework for single-round server-aided verification signature schemes and introduced a security model which extends previous proposals towards more realistic attack scenarios and stronger adversaries. In addition, we defined the first generic composition method to obtain a SAV for any signature scheme using an appropriate verifiable computation scheme. Our compiler identifies for the first time sufficient requirements on the underlying primitives to ensure the security and anonymity of the resulting SAV scheme. In particular, we showed sufficient conditions to achieve both computational and unconditional SAV-anonymity. Finally, we introduced three new SAV signature schemes obtained via our generic composition method, that simultaneously achieve existential unforgeability and soundness against collusion.

Currently, Canard *et al.*'s is the only verifiable computation scheme for pairings available in the literature. Considering the wide applicability of bilinear pairings in cryptography, a more efficient verifiable computation scheme for these functions would render pairings a server-aided accessible computation to a large variety of resource-limited devices, such as the ones involved in IoT and cloud computing settings.

Acknowledgements We thank Dario Fiore (Assistant Research Professor) for providing useful comments. This work was partially supported by the Japanese Society for the Promotion of Science (JSPS), summer program, the SNSF project SwissSenseSynergy and the STINT project IB 2015-6001.

[26] Efficiency gain is the ratio $\big(cost(\mathsf{SAV.ProbGen}) + cost(\mathsf{SAV.Verify})\big)/cost(\mathsf{Verify}_{\Sigma})$.

Bibliography

[1] Philippe Béguin and Jean-Jacques Quisquater. “Fast Server-Aided RSA Signatures Secure Against Active Attacks”. In: *Advances in Cryptology - CRYPTO ’95*. 1995, pp. 57–69.

[2] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. “Verifiable delegation of computation over large datasets”. In: *Annual Cryptology Conference*. Springer. 2011, pp. 111–131.

[3] Dan Boneh, Ben Lynn, and Hovav Shacham. “Short Signatures from the Weil Pairing”. In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Gold Coast, Australia: Springer, Heidelberg, Germany, 2001, pp. 514–532.

[4] Dan Boneh, Emily Shen, and Brent Waters. “Strongly Unforgeable Signatures Based on Computational Diffie-Hellman”. In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. LNCS. New York, NY, USA: Springer, Heidelberg, Germany, 2006, pp. 229–240.

[5] Jan Camenisch and Anna Lysyanskaya. “Signature schemes and anonymous credentials from bilinear maps”. In: *Annual International Cryptology Conference*. Springer. 2004, pp. 56–72.

[6] Sébastien Canard, Julien Devigne, and Olivier Sanders. “Delegating a pairing can be both secure and efficient”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2014, pp. 549–565.

[7] Zhengjun Cao, Lihua Liu, and Olivier Markowitch. “On Two Kinds of Flaws in Some Server-aided Verification Schemes”. In: *International Journal of Network Security* 18.6 (2016), pp. 1054–1059.

[8] Sherman S. M. Chow, Man Ho Au, and Willy Susilo. “Server-Aided Signatures Verification Secure against Collusion Attack (Short Paper)”. In: *ASIACCS 11*. Ed. by Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong. Hong Kong, China: ACM Press, 2011, pp. 401–405.

[9] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. “Geppetto: Versatile verifiable computation”. In: *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE. 2015, pp. 253–270.

[10] David Derler and Daniel Slamanig. *Key-homomorphic signatures and applications to multiparty signatures*. Tech. rep. IACR Cryptology ePrint Archive 2016, 792, 2016.

[11] Xuhua Ding, Daniele Mazzocchi, and Gene Tsudik. “Experimenting with Server-Aided Signatures”. In: *NDSS*. 2002.

[12] Dario Fiore, Rosario Gennaro, and Valerio Pastro. “Efficiently verifiable computation on encrypted data”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014, pp. 844–855.

[13] Marc Fischlin. “Anonymous signatures made easy”. In: *International Workshop on Public Key Cryptography*. Springer. 2007, pp. 31–42.

[14] Rosario Gennaro, Craig Gentry, and Bryan Parno. “Non-interactive verifiable computing: Outsourcing computation to untrusted workers”. In: *Annual Cryptology Conference*. Springer. 2010, pp. 465–482.

[15] Marc Girault and David Lefranc. “Server-Aided Verification: Theory and Practice”. In: *ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. LNCS. Chennai, India: Springer, Heidelberg, Germany, 2005, pp. 605–623.

[16] Aurore Guillevic and Damien Vergnaud. “Algorithms for outsourcing pairing computation”. In: *CARDIS*. Springer. 2014, pp. 193–211.

[17] Fuchun Guo, Yi Mu, Willy Susilo, and Vijay Varadharajan. “Server-aided signature verification for lightweight devices”. In: *The Computer Journal* (2013), bxt003.

[18] Chae Hoon Lim and Pil Joong Lee. “Server (Prover/Signer)-Aided Verification of Identity Proofs and Signatures”. In: *EUROCRYPT’95*. Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Vol. 921. LNCS. Saint-Malo, France: Springer, Heidelberg, Germany, 1995, pp. 64–78.

[19] Elena Pagnin, Aikaterini Mitrokotsa, and Keisuke Tanaka. “Anonymous Single-Round Server-Aided Verification”. In: (2017). `http://eprint.iacr.org/2017/794`.

[20] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. “Pinocchio: Nearly practical verifiable computation”. In: *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE. 2013, pp. 238–252.

[21] Bin Wang. “A Server-Aided Verification Signature Scheme without Random Oracles”. In: *International Review on Computers & Software* 7 (7 2012), p. 3446.

[22] Zhiwei Wang. “A new construction of the server-aided verification signature scheme”. In: *Mathematical and Computer Modelling* 55.1 (2012), pp. 97–101.

[23] Zhiwei Wang, Licheng Wang, Yixian Yang, and Zhengming Hu. “Comment on Wu et al.’s Server-aided Verification Signature Schemes.” In: *International Journal of Network Security* 10.2 (2010), pp. 158–160.

[24] B. Waters. “Efficient identity-based encryption without random oracles”. In: *In EUROCRYPT 2005, in: Lecture Notes in Computer Science* 3494.114–127 (2005).

[25] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. “Provably secure server-aided verification signatures”. In: *Computers & Mathematics with Applications* 61.7 (2011), pp. 1705 –1723.

[26] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. “Server-aided verification signatures: Definitions and new constructions”. In: *International Conference on Provable Security*. Springer. 2008, pp. 141–155.

[27] Guomin Yang, Duncan S Wong, Xiaotie Deng, and Huaxiong Wang. “Anonymous signature schemes”. In: *International Workshop on Public Key Cryptography*. Springer. 2006, pp. 347–363.

[28] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. "An efficient signature scheme from bilinear pairings and its applications". In: *International Workshop on Public Key Cryptography.* Springer. 2004, pp. 277–290.

Appendix

C.i Detailed descriptions of our SAV schemes

In this Appendix we present thorough descriptions of the new SAV scheme proposed in this paper (Section 6). The complete explanations of the algorithms in $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$, $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{Wat}}$ and $\mathsf{SAV}^{\mathsf{CDS}_2}_{\mathsf{CL}}$ are presented in Figures C.12, C.13 and C.14 respectively.

For consistency, we adopt the multiplicative notation for describing the operation elliptic curve groups.

$\mathsf{SAV.Init}(1^\lambda) = \mathsf{SetUp}_{\mathsf{BLS}}(1^\lambda)$. This algorithm generates the global parameters of the scheme, that include: a Gap Diffie-Hellman bilinear group $(p, g_1, \mathbb{G}, \mathbb{G}_T, e)$ according to the security parameter λ; and a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ that maps messages $m \in \mathcal{M} = \{0,1\}^*$ to group elements in $\mathbb{G}$. The output is $\mathsf{gp} = (p, g_1, \mathsf{H}, \mathbb{G}, \mathbb{G}_T, e)$.
$\mathsf{SAV.KeyGen}() = \mathsf{KeyGen}_{\mathsf{BLS}}()$. The key generation algorithm draws a random $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs $(\mathsf{pk}, \mathsf{sk}) = ({g_1}^{\mathbf{x}}, \mathbf{x})$.
$\mathsf{SAV.VSetup}() = \mathsf{KeyGen}^{\mathsf{CDS}_1}()$. This algorithm outputs $\mathsf{pr} = \mathsf{void}$ and $\mathsf{pb} = (p, \mathbb{G}, \mathbb{G}_T, e, g, \hat{\beta})$, where $\hat{\beta} = e(g, g)$.
$\mathsf{SAV.Sign}(\mathsf{sk}, m) = \mathsf{Sign}_{\mathsf{BLS}}(\mathsf{sk}, \mathsf{m})$. The signing algorithm outputs $\sigma = \mathsf{H}(m)^{\mathbf{x}} \in \mathbb{G}$.
$\mathsf{SAV.ProbGen}(\mathsf{void}, \mathsf{pk}, m, \sigma)$. This algorithm runs $\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}, m, \sigma) \to \big((\mathsf{pk}, \mathsf{H}(m)), (\sigma, g)\big)$ and returns the outputs of $\mathsf{ProbGen}^{\mathsf{CDS}_1}$ on the two pairs $(\mathsf{pk}, \mathsf{H}(\mathsf{m}))$ and (σ, g). In details, for $(\mathsf{pk}, \mathsf{H}(\mathsf{m}))$ the problem generator algorithm selects two random values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}p$, computes the points $R_1 = \mathsf{pk}^{r_2^{-1}} g^{r_1}$, $R_2 = \mathsf{H}(\mathsf{m})^{r_1^{-1}} g^{r_2}$ and $\hat{U} = \hat{\beta}^{r_1 r_2}$. This process (with fresh randomness) is applied to the pair (σ, g) as well. The final outputs are $\omega = \big((\mathsf{pk}, \mathsf{H}(\mathsf{m}), R_1^{(1)}, R_2^{(1)}), (\sigma, g, R_1^{(2)}, R_2^{(2)})\big)$ and $\tau = \big((\hat{U}^{(1)}, r_1^{(1)}, r_2^{(1)}), (\hat{U}^{(2)} r_1^{(2)}, r_2^{(2)})\big)$.
$\mathsf{SAV.Comp}(\mathsf{pb}, \omega)$. The algorithm computes the following bilinear pairings: $\alpha_1^{(1)} = e(\mathsf{pk}, \mathsf{H}(\mathsf{m}))$, $\alpha_2^{(1)} = e(R_1^{(1)}, R_2^{(1)})\big(e(\mathsf{pk}, g)e(g, \mathsf{H}(\mathsf{m}))\big)^{-1}$, $\alpha_1^{(2)} = e(\sigma, g)$, $\alpha_2^{(2)} = e(R_1^{(2)}, R_2^{(2)})\big(e(\sigma, g)e(g, g)\big)^{-1}$. It returns $\rho = (\rho_1, \rho_2) = \big((\alpha_1^{(1)}, \alpha_2^{(1)}), (\alpha_1^{(2)}, \alpha_2^{(2)})\big)$.
$\mathsf{SAV.Verify}(\mathsf{void}, \mathsf{pk}, m, \sigma, \rho, \tau)$. The verification algorithm first runs $\mathsf{Verify}^{\mathsf{CDS}_1}(\rho_i, \tau_i)$ for $i \in [2]$, *i.e.,* checks whether $\alpha_2^{(i)} = \hat{U}^{(i)}(\alpha_1^{(i)})^{(r_1^{(i)} r_2^{(i)})^{-1}}$ and $\alpha_1 \in \mathbb{G}_T$. If any of the previous checks fails, the verification algorithm returns $\Delta = \perp$ and halts. Otherwise, it sets $y_i = \alpha_1^{(i)}$, for $i \in [2]$ and runs $\mathsf{Ver}_{\mathsf{L}}(\mathsf{pk}, m, \sigma, y)$, which returns $\Delta = 1$ if $y_1 = y_2$, and $\Delta = 0$ otherwise.

Figure C.12: $\mathsf{SAV}^{\mathsf{CDS}_1}_{\mathsf{BLS}}$: Our SAV for the BLS Signature in [3].

$\mathsf{SAV.Init}(1^\lambda) = \mathsf{SetUp}_{\mathsf{Wat}}(1^\lambda)$. This algorithm generates a bilinear group $(p, g_1, \mathbb{G}, \mathbb{G}_T, e)$ according to the security parameter λ; selects $n+1$ group elements $V_0, V_1, \dots V_n \xleftarrow{\$} \mathbb{G}$ and defines a function $\mathsf{H} : \{0,1\}^n \to \mathbb{G}$ as $\mathsf{H}(\mathsf{m}) = V_0(\prod_{i=1}^n V_i^{\mathsf{m}_i})$. The output is $\mathsf{gp} = (p, g_1, \mathsf{H}, \mathbb{G}, \mathbb{G}_T, e)$.
$\mathsf{SAV.KeyGen}() = \mathsf{KeyGen}_{\mathsf{Wat}}()$. The key generation algorithm draws a random $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs $(\mathsf{pk}, \mathsf{sk}) = (e(g_1, g_1)^{\mathbf{x}}, \mathbf{x})$.
$\mathsf{SAV.VSetup}() = \mathsf{KeyGen}^{\mathsf{CDS}_1}()$. This algorithm outputs $\mathsf{pr} = \mathsf{void}$ and $\mathsf{pb} = (p, \mathbb{G}, \mathbb{G}_T, e, g, \hat{\beta})$, where $\hat{\beta} = e(g, g)$.
$\mathsf{SAV.Sign}(\mathsf{sk}, m) = \mathsf{Sign}_{\mathsf{Wat}}(\mathsf{sk}, \mathsf{m})$. The signing algorithm picks a random $a \xleftarrow{\$} \mathbb{Z}p$ and outputs $\sigma = (\sigma_1, \sigma_2) = ({g_1}^{\mathbf{x}}(\mathsf{H}(m))^a, {g_1}^a) \in \mathbb{G}^2$.
$\mathsf{SAV.ProbGen}(\mathsf{void}, \mathsf{pk}, m, \sigma)$. This algorithm runs $\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}, m, \sigma) \to (\mathsf{pk}', \sigma')$ to create a signature for a new public key, *i.e.,* it picks two random values $\mathsf{h}, b \xleftarrow{\$} \mathbb{Z}p$ and sets $\mathsf{pk}' = \mathsf{pk}\hat{\beta}^{\mathsf{h}}$, $\sigma' = (g^{\mathsf{h}}\sigma_1\mathsf{H}(\mathsf{m})^b, \sigma_2 g^b)$. *(By the adaptivity of* Wat *if σ is a valid signature for* m *under* sk *with randomness a, then σ' is a valid signature for* m *under* $\mathsf{sk}' + \mathsf{h}$ *with randomness $a+b$).*
Secondly, the problem generation algorithm runs $\mathsf{ProbGen}^{\mathsf{CDS}_1}$ on (σ_1', g) and $(\mathsf{H}(\mathsf{m}), \sigma_2')$. In details, for each pair (A, B), the algorithm selects two random values $r_1, r_2 \xleftarrow{\$} \mathbb{Z}p$, computes the points $R_1 = A^{r_2^{-1}} g^{r_1}$, $R_2 = B^{r_1^{-1}} g^{r_2}$ and $\hat{U} = \hat{\beta}^{r_1 r_2}$. The final outputs are $\omega = (R_1^{(1)}, R_2^{(1)}, R_1^{(2)}, R_2^{(2)})$ and $\tau = (\mathsf{pk}'\hat{U}^{(1)}, r_1^{(1)}, r_2^{(1)}, \hat{U}^{(2)} r_1^{(2)}, r_2^{(2)})$.
$\mathsf{SAV.Comp}(\mathsf{pb}, \omega)$. The algorithm parses ω as $\big((R_1^{(1)}, R_2^{(1)}), (R_1^{(2)}, R_2^{(2)})\big)$; for each pair (A, B) it computes $\alpha_1 = e(A, B)$ and $\alpha_2 = e(R_1, R_2)\big(e(g, B), e(A, g)\big)^{-1}$. It returns $\rho = (\alpha_1^{(1)}, \alpha_2^{(1)}, \alpha_1^{(2)}, \alpha_2^{(2)})$.
$\mathsf{SAV.Verify}(\mathsf{void}, \mathsf{pk}, m, \sigma, \rho, \tau)$. The verification algorithm parses $\rho = (\rho^{(1)}, \rho^{(2)}) = \big((\alpha_1^{(1)}, \alpha_2^{(1)}), (\alpha_1^{(2)}, \alpha_2^{(2)})\big)$ and $\tau = (\mathsf{pk}', \tau^{(1)}, \tau^{(2)}) = \big((\hat{U}^{(1)}, r_1^{(1)}, r_2^{(1)}), (\hat{U}^{(2)} r_1^{(2)}, r_2^{(2)})\big)$. For $i \in [2]$ it runs $\mathsf{Verify}^{\mathsf{CDS}_2}(\rho^{(i)}, \tau^{(i)})$, *i.e.,* it checks if $\alpha_2^{(i)} = \hat{U}^{(i)}(\alpha_1^{(i)})^{(r_1^{(i)} r_2^{(i)})^{-1}}$ and $\alpha_1 \in \mathbb{G}_T$. If any of the previous checks fails, the verification algorithm returns $\Delta = \perp$ and halts. Otherwise, it returns $y^{(i)} = \alpha_1^{(i)}$ and runs $\mathsf{Ver}_{\mathsf{L}}(\mathsf{pk}, m, \sigma, y)$, which returns $\Delta = 1$ if $y^{(1)} = \mathsf{pk}'\, y^{(2)}$, and $\Delta = 0$ otherwise.

Figure C.13: $\mathsf{SAV}_{\mathsf{Wat}}^{\mathsf{CDS}_1}$: Our SAV for the Wat Signature in [24].

$\mathsf{SAV.Init}(1^\lambda) = \mathsf{SetUp_{CL}}(1^\lambda)$. The setup algorithm generates the global parameters of the scheme, that include a bilinear group $(q, \mathbb{G}, g, \mathbb{G}_T, \hat{g}, e)$.
$\mathsf{SAV.KeyGen}() = \mathsf{KeyGen_{CL}}()$. The key generation algorithm draws two random values $x, y \xleftarrow{\$} \mathbb{Z}q$, computes $g^x = X$, $g^y = Y$ and returns $\mathsf{pk} = (X, Y)$ and $\mathsf{sk} = (x, y)$.
$\mathsf{SAV.VSetup}() = \mathsf{KeyGen}^{\mathsf{CDS_2}}()$. This algorithm outputs $\mathsf{pr} = \mathsf{void}$ and $\mathsf{pb} = (p, \mathbb{G}, \mathbb{G}_T, e, G, B, \hat{\beta})$, where $G \xleftarrow{\$} \mathbb{G}$, $B = g$ and $\hat{\beta} = e(G, B)$.
$\mathsf{SAV.Sign}(\mathsf{sk}, m) = \mathsf{Sign_{CL}}(\mathsf{sk}, \mathsf{m})$. The sign algorithm picks a random $a \xleftarrow{\$} \mathbb{G}$ and outputs the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3) = (a, a^y, a^{x+mxy}) \in \mathbb{G}^3$.
$\mathsf{SAV.ProbGen}(\mathsf{void}, \mathsf{pk}, m, \sigma)$. This algorithm first runs $\mathsf{ProbGen}^{\mathsf{PRE}}(\mathsf{pk}, m, \sigma) \rightarrow (\sigma_2, \sigma_3)$. Then it runs $\mathsf{ProbGen}^{\mathsf{CDS_2}}$ on σ_2 and σ_3. In more details, for $i \in \{2, 3\}$ it selects three random values $r_1^{(i)}, r_2^{(i)}, u^{(i)} \xleftarrow{\$} \mathbb{Z}q$, computes the points $R_1^{(i)} = \sigma_i \cdot G^{r_1^{(i)}}$ and $R_2^{(i)} = \sigma_i^{u^{(i)}} \cdot G^{r_2^{(i)}}$, and calculates $\hat{X}_1^{(i)} = (\hat{\beta})^{r_1^{(i)}}$, $\hat{X}_2^{(i)} = (\hat{\beta})^{r_2^{(i)}}$. The final outputs are $\omega = (R_1^{(2)}, R_2^{(2)}, R_1^{(3)}, R_2^{(3)})$ and $\tau = (u^{(2)}, \hat{X}_1^{(2)}, \hat{X}_2^{(2)}, u^{(3)}, \hat{X}_1^{(3)}, \hat{X}_2^{(3)})$.
$\mathsf{SAV.Comp}(\mathsf{pb}, \omega)$. The algorithm parses $\omega = (R_1^{(2)}, R_2^{(2)}, R_1^{(3)}, R_2^{(3)})$ and returns $\rho = (e(R_1^{(2)}, g), e(R_2^{(2)}, g), e(R_1^{(3)}, g), e(R_2^{(2)}, g))$.
$\mathsf{SAV.Verify}(\mathsf{void}, \mathsf{pk}, m, \sigma, \rho, \tau)$. The verification algorithm first runs $\mathsf{Verify}^{\mathsf{CDS_2}}(\rho, \tau)$, *i.e.,* for $i \in \{2, 3\}$ it checks if $\alpha_2^{(i)} = \hat{X}_2^{(i)}(\alpha_1(\hat{X}_1^{(i)})^{-1})^u$ and $\alpha_1^{(i)} \in \mathbb{G}_T$. If any of the previous checks fails, the verification algorithm returns $\Delta = \perp$ and halts. Otherwise, the values $y^{(i)} = \beta_1^{(i)}(\hat{X}_1^{(i)})^{-1}$, for $i \in \{2, 3\}$ are used as input for $\mathsf{Ver_L}$. In details, $\mathsf{Ver_L}(\mathsf{pk}, m, \sigma, y = (y^{(2)}, y^{(3)}))$, computes: $\beta_1 = e(\sigma_1, Y)$, $\beta_2 = e(X, \sigma_1 \sigma_2^m)$. If both $\beta_1 = y(1)$ and $\beta_2 = y^{(2)}$, the algorithm returns $\Delta = 1$; otherwise it returns $\Delta = 0$.

Figure C.14: $\mathsf{SAV}_{\mathsf{CL}}^{\mathsf{CDS_2}}$: Our SAV for the CL Signature in [5].

Paper D

Two-hop Distance-Bounding Protocols: Keep your Friends Close

Anjia Yang, Elena Pagnin, Aikaterini Mitrokotsa, Gerhard P. Hancke, and Duncan S. Wong

Abstract. Authentication in wireless communications often depends on the physical proximity to a location. Distance-bounding (DB) protocols are cross-layer authentication protocols that are based on the round-trip-time of challenge-response exchanges and can be employed to guarantee physical proximity and combat relay attacks. However, traditional DB protocols rely on the assumption that the prover (e.g., user) is in the communication range of the verifier (e.g., access point); something that might not be the case in multiple access control scenarios in ubiquitous computing environments as well as when we need to verify the proximity of our two-hop neighbour in an ad-hoc network. In this paper, we extend traditional DB protocols to a two-hop setting i.e. when the prover is out of the communication range of the verifier and thus, they both need to rely on an untrusted in-between entity in order to verify proximity. We present a formal framework that captures the most representative classes of existing DB protocols and provide a general method to extend traditional DB protocols to the two-hop case (three participants). We analyse the security of two-hop DB protocols and identify connections with the security issues of the corresponding one-hop case. Finally, we demonstrate the correctness of our security analysis and the efficiency of our model by transforming five existing DB protocols to the two-hop setting and we evaluate their performance with simulated experiments.

Keywords. Distance-Bounding, Relay Attacks, Authentication Protocols.

IEEE Transactions on Mobile Computing (2018).

Two-hop Distance-Bounding Protocols: Keep your Friends Close

1 Introduction

Wireless communications have strong connections with proximity-based authentication. For instance, in multiple wireless access control scenarios, we gain access to a service and/or a place depending on our physical proximity to an access control point. Furthermore, wireless communications often rely on the cooperation of one-hop and two-hop neighbours (e.g., routing in wireless ad-hoc networks). Verifying the location of our neighbours and our proximity to an access point is usually performed by employing a secure neighbour discovery (SND) method [20]. Distance-bounding (DB) protocol is an important method for reliable SND, which is based on the round-trip-time of carefully designed challenge-response messages to provide an upper bound on the physical distance between two nodes. Although DB protocols provide a cryptographic proof of proximity for one-hop neighbours they cannot be employed when the prover is outside the communication range of the verifier.

In this paper, we investigate the extension of conventional (one-hop) DB protocols to a two-hop setting. More precisely, we examine how DB protocols designed for two participants –a (trusted) verifier $\mathcal{V}$ and a (usually untrusted) prover $\mathcal{P}$– can be extended to a three participants setting – a (trusted) verifier $\mathcal{V}$, a (potentially untrusted) prover that lies *two hops away* from $\mathcal{V}$, and an (untrusted) in-between entity, henceforth called the *linker* $\mathcal{L}$, which is in the communication range of both $\mathcal{P}$ and $\mathcal{V}$.

We should stress here that one-hop DB protocols can be employed when $\mathcal{P}$ is in the communication range of $\mathcal{V}$, in order to verify that $\mathcal{P}$ is in the vicinity of $\mathcal{V}$. However, when $\mathcal{P}$ is outside $\mathcal{V}$'s communication range, traditional one-hop DB protocols are not enough. In the latter case, there is a need for two-hop DB protocols that are able to verify that $\mathcal{L}$ is close to $\mathcal{V}$ and that $\mathcal{P}$ is close to $\mathcal{L}$ by measuring the time-of-flight of the exchanged messages.

Given the tremendous development of wireless communications and the new era of ubiquitous computing, two-hop distance-bounding can be useful in multiple important scenarios, such as the detection of wormhole attacks and wireless access control scenarios where the prover is located outside the communication range of the verifier.

A *wormhole* is an attack strategy, first described by Perrig *et al.* [14], for disrupting the normal operation of routing protocols. In a wormhole attack, an adversary advertises as the most attractive routing path to another node in the network, a path that passes through her. In this way, she is able to get under her control the communication between two nodes (i.e. the messages are sent to her in order to be forwarded). This implies that she can modify or even discard the messages that reach her and consequently disrupt the whole communication. In a wormhole attack, an adversary may have compromised a node $\mathcal{L}$ that is a one-hop neighbour of two nodes $\mathcal{P}$ and $\mathcal{V}$, while $\mathcal{P}$ is a two-hop neighbour (i.e. outside the communication range) of $\mathcal{V}$. For instance, consider the scenario where $\mathcal{V}$ wants to transmit a message to $\mathcal{P}$ and verify that $\mathcal{P}$ is $\mathcal{V}$'s two-hop neighbour. Here we can distinguish between two cases: *(i)* if $\mathcal{V}$ trusts $\mathcal{P}$, and *(ii)* if $\mathcal{P}$ is untrusted. If $\mathcal{V}$ trusts $\mathcal{P}$ but both $\mathcal{P}$ and $\mathcal{V}$ do not trust $\mathcal{L}$ then by running a conventional DB protocol twice (once run between $\mathcal{V}$ and $\mathcal{L}$ and once between $\mathcal{L}$ and $\mathcal{P}$), $\mathcal{V}$ could verify that $\mathcal{P}$ is indeed its two-hop neighbour. In the second case (untrusted $\mathcal{P}$), conventional (one-hop) DB protocols cannot directly verify the two-hop proximity of $\mathcal{P}$. The same problem exists when the adversary controls two nodes $\mathcal{L}_1$ and $\mathcal{L}_2$ instead of a single node $\mathcal{L}$ (Figure D.15). It is easy to see that in such a scenario, there is an eminent need for a mechanism to verify the two-hop proximity of an untrusted node $\mathcal{P}$ by relying on an untrusted one-hop neighbour ($\mathcal{L}$).

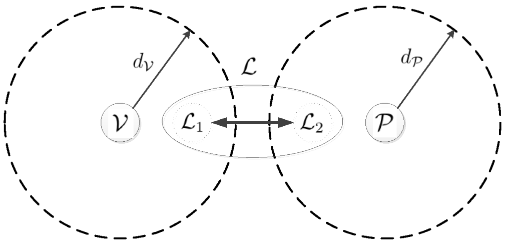


Figure D.15: A wormhole attack run by an external adversary that employs two malicious nodes $\mathcal{L}_1$ and $\mathcal{L}_2$. The communication ranges of $\mathcal{V}$ and $\mathcal{P}$ are indicated by $d_{\mathcal{V}}$ and $d_{\mathcal{P}}$ respectively.

As an additional motivation for the need of two-hop DB protocols we could consider access control problems when the prover (i.e. employed device) does not have direct access to the verifier (i.e. access point) but instead has to depend on an untrusted (in-between) node (device). This could be the case in multiple scenarios where smart devices are employed in ubiquitous computing environments, e.g., in a university campus, gaining access to a printer if the prover can prove that he is a two-hop neighbour of the verifier (i.e. printer).

Contributions. In this paper, we investigate how conventional (one-hop) DB protocols can be extended to a two-hop setting i.e. when the prover does not have direct access (i.e. not in the communication range) with the verifier, but instead has to rely an in-between untrusted party. We provide a general framework that can be employed to transform certain families of one-hop DB protocols to the two-hop (three participants) setting. We analyse the security of two-hop DB protocols and identify connections with the security issues of the corresponding one-hop case. Finally, we demonstrate the correctness of our security analysis and the efficiency of our model by implementing five different two-hop DB protocols and performing experimental attacks on them.

Organisation. The paper is organised as follows. Section 2 describes related work, while Section 3 describes conventional (one-hop) DB protocols. In Section 4 we describe the general structure of a two-hop DB protocol. Section 5 presents a formal model for analysing the security of two-hop DB protocols, while Section 6 presents the experimental evaluation of our proposed model based on five existing DB protocols. Finally, Section 6 concludes the paper.

2 Related Work

DB protocols are real-time challenge-response authentication protocols that attempt to simultaneously verify the credentials and the proximity of an untrusted prover. These protocols are mainly employed in settings where the adversary wants to fool a verifier (e.g., access point) into accepting a distant prover. DB protocols were initially proposed by Brands and Chaum [6] as an efficient countermeasure against relay attacks in ATM systems. They rely on the round-trip-time of multiple challenge-response pairs to determine an upper bound on the physical distance between a trusted verifier $\mathcal{V}$ and an untrusted prover $\mathcal{P}$. The final output of the protocol depends on (*i*) the estimated distance between the prover and the verifier, and (*ii*) the correctness of the received responses.

In 2005, Hancke and Kuhn [12] proposed a DB protocol resistant to the main two threats against DB protocols and introduced the concept of *registers* for DB protocols. Subsequently, many protocols were proposed that rely on the Hancke and Kuhn's protocol (e.g. [4, 16, 22, 24, 25]). In the recent decade, the interest in formulating and evaluating DB protocols has grown considerably, and different approaches have been presented [1, 2, 5, 10, 22, 23]. Although the majority of DB protocols present in the literature are based on secret key cryptography, there are also some recent proposals that rely on public key cryptography [11, 13, 15]. In our analysis, to facilitate understanding, we shall focus on DB protocols that rely on secret key cryptography. However, our results could also be easily applied to protocols that rely on public key cryptography.

We classify DB protocols into two main categories, depending on how the prover generates the responses:

Register-based DB protocols: In this case, the set of possible responses is composed of $\rho \geq 1$ vector(s), called register(s) [12]. The verifier's challenges indicate which register should be used for the calculation of the responses. Multiple existing DB protocols belong in this category, e.g., Brands-Chaum [6], where $\rho = 1$; Hancke-Kuhn [12], Swiss-knife [16], Bussard-Bagga [7] and Reid *et al.* [22], where $\rho = 2$; finally SKI [4] treats the case where $\rho \geq 3$.

Non-register-based DB protocols: There are very few protocols that fall in this category. In particular, Avoine *et al.* [2] proposed a DB protocol that employs a tree-based response function, where the prover's responses depend on all the challenges sent in the same protocol run. Furthermore, a DB protocol where the responses are based on challenge reflection with channel selection (CRCS) were introduced by Rasmussen and Capkun [21].

In this paper, we shall focus on DB protocols that are register-based, as these constitute the overwhelming majority of the proposals presented in the literature. We need to note though that non-register based DB protocols could easily be extended to a two-hop setting. However, the employed generalisation and notation does not capture the description of non-register based protocols, since they have more complicated structures (e.g., a tree-based response function). Henceforth, when referring to DB protocols we shall consider register-based DB protocols. Similar to part of our work, Mauw *et al.* [18] also constructed an abstract model for the register-based DB protocols based on finite state automata, provided security analysis of these protocols, and developed a new family of DB protocols which is resistant to mafia fraud attacks. As a comparison, we provide more comprehensive security analysis which captures all the three common attacks.

Table D.2: Notations

$\mathcal{V}/\ \mathcal{P}/\mathcal{L}$	:	honest verifier/prover/linker
$\mathcal{P}^*/\mathcal{L}^*$	:	dishonest prover/linker
$\mathcal{K}/\mathcal{M}/\mathcal{I}/\mathcal{C}/\mathcal{R}$	:	key/message/index/challenge/response space
X, W	:	the range of two strings generated in the non-time critical phases
n	:	the number of rounds in the DB phase
i	:	the index of the rounds in the DB phase
$g_0/g_1/g_2$	:	functions used in the non-time critical phases
f	:	the response function used in the DB phase
$x_{\mathcal{VL}}/x_{\mathcal{VP}}$	:	the secret key shared between the verifier and the linker/prover
$m_{\mathcal{V}}/m_{\mathcal{L}}/m_{\mathcal{P}}$	:	messages sent by the verifier/linker/prover in the initialisation phase
c_i/r_i	:	the verifier's challenge value / the prover's response value
ℓ_i	:	the linker's response value
w	:	the final signature sent by the prover
$\xi_{\mathcal{L}}$, $\xi_{\mathcal{P}}$	:	vectors used to produce the responses
Δt_i	:	the time difference between the time c_i was sent and r_i was received
$\mathsf{t}_{\mathsf{max}}$	:	the maximum round-trip-time allowed (as a bound on the distance)

Current DB protocols mostly consider a single prover bounding the distance of a single verifier. None of these proposals provide non-repudiation of the distance-bound between two parties to any third (untrusted) party. Our proposal allows the verifier to determine a distance bound on the linker (next-hop node) and verify the validity of the distance bound between the linker and the prover, even though the linker is not trusted. One interesting divergence from the two-party distance-bounding approach is performing distance-bounding with multiple parties [8]. This group distance-bounding verifies that all the parties are in close proximity. However, this still requires all the parties in the group to be able to communicate directly with each other to be able to complete the protocol. Our proposal allows a verifier to verify that two nodes are in close proximity (next-hop and two-hop) without directly communicating with the two-hop node. Centralized SND approaches can verify more than just next-hop neighbours but are based on the assumption that there are many nodes that can collaborate and aggregate data to a central system controller [17]. This approach often involves location-based methods that require the physical location of each node to be known [14]. Determining the location of a node requires additional network infrastructure and resources, especially indoors where Global Positioning Systems (GPS) are not so effective, while a system wide localization scheme still relies on accurate node-level neighbour detection to build secure connectivity maps [3]. There are several secure localization schemes that use DB protocols for the underlying distance estimation between nodes [9]. Our approach does not compete with these centralised approaches and can potentially assist them by allowing individual nodes to securely verify the proximity of next-hop and two-hop nodes.

We have recently introduced [19] the concept of two-hop distance-bounding that extends traditional DB protocols to a two-hop setting and proposed an approach on how some of the existing DB protocols could be modified to verify the proximity of both next-hop and two-hop neighbours. In this paper, we go beyond this and introduce a general model that covers most of the existing DB protocols and analyse its security for internal and external adversaries. Furthermore, we provide instantiations for the transformation of five existing DB protocols to a two-hop setting. To verify our theoretical analysis, we evaluate the five chosen protocols and provide an analysis of the success probability of the different types of attacks for a varying number of rounds. Finally, we provide simulated experiments that validate our theoretical results.

3 A General Model for Distance-Bounding Protocols

In this section, we describe a general structure of most existing DB protocols that belong in the register-based category. Table D.2 refers to the main notations used throughout the paper. Let $\mathcal{K}$ denote the set of keys, $\mathcal{M}$ the set of messages and $\mathcal{X}$ the set of session vectors. The challenges will be taken from the set $\mathcal{C}$, the responses from the set $\mathcal{R}$ while $\mathcal{I}$ denotes the set of indices[27]. Then we can make the following claim:

Claim: *All (existing) register-based* DB *protocols can be described using the seven (finite) sets* $\mathcal{K}, \mathcal{M}, \mathcal{X}, \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{W}$ *and the four maps,* g_0, g_1, g_2, f *defined as follows:* $g_0 : \mathcal{K} \to \mathcal{M}$, $g_1 : \mathcal{K} \times \mathcal{M} \times \mathcal{M} \to \mathcal{X}$, $g_2 : \mathcal{K} \times \mathcal{X} \times \mathcal{C}^n \times \mathcal{R}^n \to \mathcal{W}$, $f : \mathcal{K} \times \mathcal{X} \times \mathcal{C} \times \mathcal{I} \to \mathcal{R}$.

We shall demonstrate this claim via a proof-of-concept. More precisely, we shall construct a general (register-based) DB protocol that employs the functions g_0, g_1, g_2, f. By explicitly defining the functions g_0, g_1, g_2, f and the sets $\mathcal{K}, \mathcal{M}, \mathcal{X}, \mathcal{C}$, $\mathcal{R}, \mathcal{I}, \mathcal{W}$, one can obtain all register-based DB protocols. Specific instantiations for five existing DB protocols are given in Section 6.

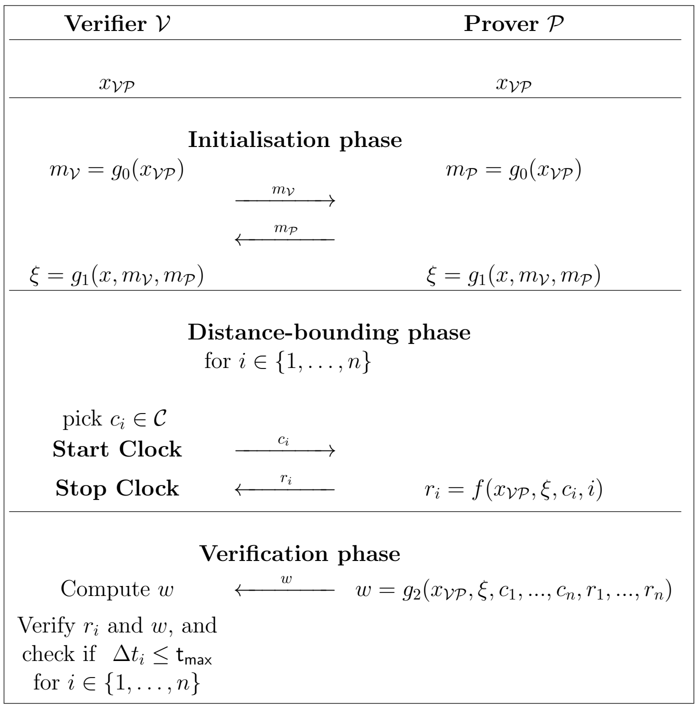

Verifier $\mathcal{V}$
Prover $\mathcal{P}$
$x_{\mathcal{VP}}$
$x_{\mathcal{VP}}$
Initialisation phase
$m_{\mathcal{V}} = g_0(x_{\mathcal{VP}})$
$m_{\mathcal{P}} = g_0(x_{\mathcal{VP}})$
$\xrightarrow{m_{\mathcal{V}}}$
$\xleftarrow{m_{\mathcal{P}}}$
$\xi = g_1(x, m_{\mathcal{V}}, m_{\mathcal{P}})$
$\xi = g_1(x, m_{\mathcal{V}}, m_{\mathcal{P}})$
Distance-bounding phase
for $i \in \{1, \ldots, n\}$
pick $c_i \in \mathcal{C}$
Start Clock $\xrightarrow{c_i}$
Stop Clock $\xleftarrow{r_i}$
$r_i = f(x_{\mathcal{VP}}, \xi, c_i, i)$
Verification phase
Compute w $\xleftarrow{w}$ $w = g_2(x_{\mathcal{VP}}, \xi, c_1, ..., c_n, r_1, ..., r_n)$
Verify r_i and w, and check if $\Delta t_i \leq \mathsf{t}_{\mathsf{max}}$ for $i \in \{1, \ldots, n\}$

Figure D.16: The general description of a register-based DB protocol.

We consider a trusted verifier $\mathcal{V}$ who is equipped with a clock, and an untrusted prover $\mathcal{P}$ that shares a secret key $x_{\mathcal{VP}} \in \mathcal{K}$. The general structure model for register-based DB protocols is depicted in Figure D.16 and is composed of the following phases:

[27]$\mathcal{I} = \{1, \ldots, n\}$ where n is the total number of rounds in the distance-bounding phase.

Initialisation Phase This first phase enables the verifier $\mathcal{V}$ and the prover $\mathcal{P}$ to initialise some values that will be used in the subsequent phases of the protocol. At this stage both parties can (*i*) generate some values (e.g., nonces, using the function g_0) and (*ii*) perform some preliminary computations (using the function g_1). More precisely, the functions g_0 and g_1 are defined as follows:
• $g_0 : \mathcal{K} \to \mathcal{M}$ is a possibly randomised function used to generate the initialisation messages: $m_\mathcal{V}$ (for the verifier) and $m_\mathcal{P}$ (for the prover). The function g_0 could be for example a random selection (e.g., generation of a nonce) [12].
• $g_1 : \mathcal{K} \times \mathcal{M} \times \mathcal{M} \to \mathcal{X}$ is a deterministic function used in the initialisation phase. It takes as input the shared secret key and two previously generated messages, and outputs $\xi = g_1(x_{\mathcal{VP}}, m_\mathcal{V}, m_\mathcal{P})$. The function g_1 could be for instance a hash function, a pseudorandom function (PRF), a commitment scheme [6] or it could just return void. The session vector ξ is usually representing a session key that will be used to generate the responses in the following phase.

Distance-Bounding Phase This is the only time-critical part of the protocol and consists of $n = |\mathcal{I}|$ rounds with the same structure. $\mathcal{V}$ picks a random element in $\mathcal{C}$ (a challenge), sends it to $\mathcal{P}$ and starts the clock. Upon receiving the prover's response, $\mathcal{V}$ stops the clock, stores the received data and records the round-trip-time Δt_i for $i \in \{1, \dots, n\}$. The main function in the whole DB protocol is the response function $f : \mathcal{K} \times \mathcal{X} \times \mathcal{C} \times \mathcal{I} \to \mathcal{R}$ that is called in this phase. In general, f outputs r_i according to the values of the secret key x, the session vector ξ, the challenge c_i and also the round i. In DB, the independency of the responses r_i follows from the independency of the challenges c_i and the domain definition of f.

Verification Phase This is the final stage of the protocol. In most existing DB protocols in this phase, the verifier checks that the received responses are correct, and all the recorded round-trip times (Δt_i) are smaller or equal to a pre-defined threshold $\mathsf{t_{max}}$ that denotes the maximum-round-trip-time between $\mathcal{P}$ and $\mathcal{V}$. Optionally, $\mathcal{P}$ can provide an additional message that $\mathcal{V}$ shall check. We could actually discriminate two main classes of DB protocols that provide an additional message to be checked (e.g., a MAC or a signature) following the protocol proposed by Brands and Chaum [6] or those where there is no additional message, following the paradigm of Hancke and Kuhn's protocol [12]. Eventually, if all the conditions are satisfied, the verifier states that the prover is close enough and authenticated. We model the final message produced by $\mathcal{P}$ through the function $g_2 : \mathcal{K} \times \mathcal{X} \times \mathcal{C}^n \times \mathcal{R}^n \to \mathcal{W}$. The exponent n is the number of rounds in the distance-bounding phase. The function g_2 computes the value (usually a vector) $w = g_2(x_{\mathcal{VP}}, \xi, c_1, \dots, c_n, r_1, \dots, r_n)$. The output w depends on the secret key $x_{\mathcal{VP}}$, the session vector ξ and possibly the transcript of the DB phase (n challenges and the n corresponding responses). Depending on the protocol, g_2 can be an open-commitment [6], a signature on the transcript [6, 16] or it might return a void value [4, 22].

It should be obvious to see that any register-based DB protocol fits in the presented framework. For instantiations of the functions g_0, g_1, g_2, f and the sets $\mathcal{K}, \mathcal{M}, \mathcal{X}, \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{W}$ we refer the reader to Section 6.

4 From One-Hop to Two-Hop Distance-Bounding

Although traditional (one-hop) DB protocols have important advantages on combating relay attacks and verifying the proximity of an untrusted prover $\mathcal{P}$ to a trusted verifier $\mathcal{V}$, they cannot be employed when $\mathcal{P}$ is beyond the communication range of $\mathcal{V}$. In this section, we describe how traditional DB protocols could be extended to the two-hop setting. In this setting, we consider three parties: an untrusted prover $\mathcal{P}$, a trusted verifier $\mathcal{V}$ and an untrusted in-between node $\mathcal{L}$. $\mathcal{P}$ and $\mathcal{V}$ are not in the communication range of each other while $\mathcal{L}$ is a one-hop neighbour of both $\mathcal{P}$ and $\mathcal{V}$. The goal of a two-hop DB protocol is to enable $\mathcal{V}$ in determining an upper bound of the distance of the next-hop node $\mathcal{L}$ as well as the two-hop neighbouring node $\mathcal{P}$ even when both of these nodes are untrusted. Figure D.17 depicts the configuration of a two-hop DB protocol under consideration. Following the same formalisation used in section 3, in this section we provide the general structure of a two-hop DB protocol.

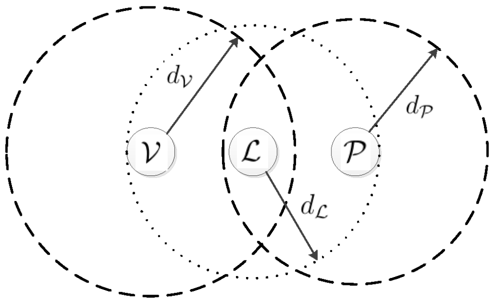


Figure D.17: The basic configuration of two-hop DB protocols: $\mathcal{P}$ lies at a two-hop distance from $\mathcal{V}$. Being outside $\mathcal{V}$'s communication rage, $\mathcal{P}$ relies on the in-between untrusted node $\mathcal{L}$, who is in the communication range of both $\mathcal{P}$ and $\mathcal{V}$. $d_{\mathcal{V}}$, $d_{\mathcal{L}}$ and $d_{\mathcal{P}}$ denote the communication ranges of $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{P}$ correspondingly.

4.1 Challenge-Response in two-hop Distance-Bounding

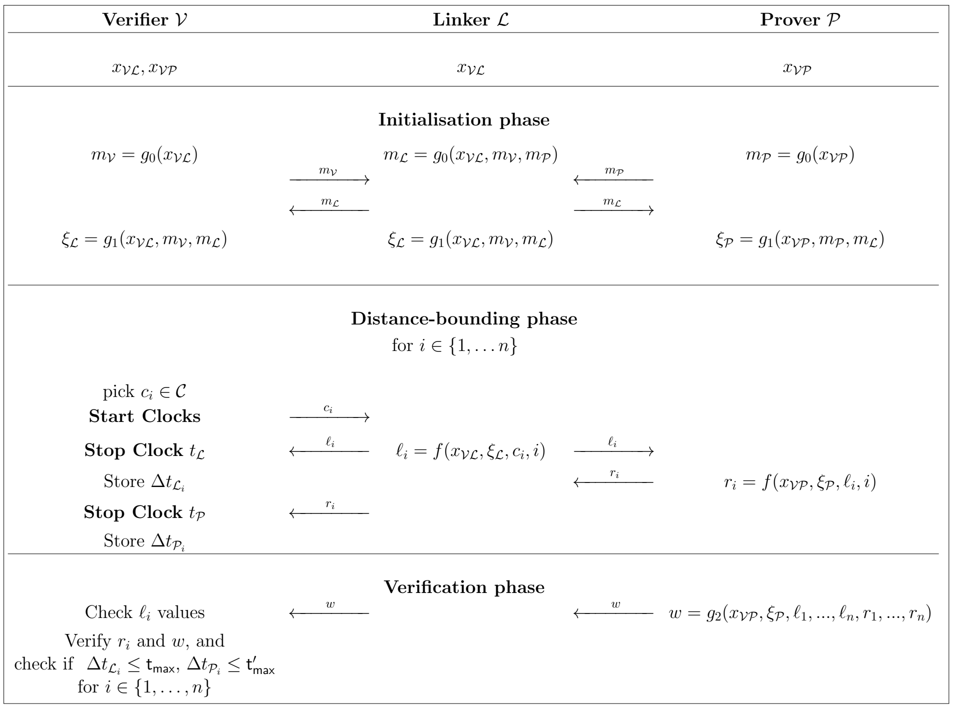
We assume that each entity in the setting under consideration broadcasts its messages. This condition implies that whenever the linker $\mathcal{L}$ sends a message to the verifier $\mathcal{V}$, this message will also be received by the prover $\mathcal{P}$ (since $\mathcal{P}$ lies in $\mathcal{L}$'s communication range). In particular, in the time-critical phase (i.e., the distance-bounding phase) of the two-hop DB protocol, any response ℓ_i produced by $\mathcal{L}$, will be interpreted as a challenge by $\mathcal{P}$. Therefore, we will refer to ℓ_i, the output of the linker in the time-critical phase, as the *challenge-response*, because it is a response to the challenge c_i (generated by $\mathcal{V}$) and also a challenge to $\mathcal{P}$ (who will return the final response r_i).

In our generalisation of two-hop DB protocols (depicted in Figure D.18), we let $\mathcal{L}$ use the same response function f as $\mathcal{P}$ (obviously f has different inputs for each entity). Because of this choice, the range $\mathcal{R}$ of the response function f shall be contained in the set of possible challenges, i.e., $\mathcal{R} \subseteq \mathcal{C}$. For security reasons explained in Section 5.3, we require that $\mathcal{C} = \mathcal{R}$. This assumption is actually satisfied by the large majority of existing DB protocols [6, 7, 12, 16, 22].

Our two-hop extension also applies to the register-based DB protocols for which $|\mathcal{C}| > |\mathcal{R}|$, e.g., [4]. Since the values ℓ_i are used both as challenges and as responses (for the entities $\mathcal{P}$ and $\mathcal{V}$ respectively), $\ell_i \in \mathcal{R} \cap \mathcal{C} = \mathcal{R}$, the prover knows *a priori* that some challenge-values are not possible. Thus, the security level of the corresponding two-hop DB protocols drops considerably. For this reason, we generalise Boureanu *et al.*'s DB protocol [4] to the two-hop case, only in the case where $\mathcal{C} = \mathcal{R}$.

4.2 General Construction of Two-Hop DB Protocols

We proceed now with the formal description of a two-hop DB protocol. Similarly to the one-hop case, we assume that the verifier $\mathcal{V}$ shares a secret key $x_{\mathcal{VP}}$ with the prover $\mathcal{P}$ and a secret key $x_{\mathcal{VL}}$ with the linker $\mathcal{L}$. Note that a linker itself could be a prover in another DB protocol execution. We recall that for the two-hop DB protocols under consideration the set of challenges and the set of responses coincide, i.e., $\mathcal{C} = \mathcal{R}$. The general structure of a two-hop DB protocol is depicted in Figure D.18 and is composed of the following three phases:



Verifier $\mathcal{V}$
Linker $\mathcal{L}$
Prover $\mathcal{P}$
$x_{\mathcal{VL}}, x_{\mathcal{VP}}$
$x_{\mathcal{VL}}$
$x_{\mathcal{VP}}$
Initialisation phase
$m_{\mathcal{V}} = g_0(x_{\mathcal{VL}})$
$m_{\mathcal{L}} = g_0(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{P}})$
$m_{\mathcal{P}} = g_0(x_{\mathcal{VP}})$
$\xrightarrow{m_{\mathcal{V}}}$
$\xleftarrow{m_{\mathcal{P}}}$
$\xleftarrow{m_{\mathcal{L}}}$
$\xrightarrow{m_{\mathcal{L}}}$
$\xi_{\mathcal{L}} = g_1(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{L}})$
$\xi_{\mathcal{L}} = g_1(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{L}})$
$\xi_{\mathcal{P}} = g_1(x_{\mathcal{VP}}, m_{\mathcal{P}}, m_{\mathcal{L}})$
Distance-bounding phase
for $i \in \{1, \dots n\}$
pick $c_i \in \mathcal{C}$
Start Clocks
$\xrightarrow{c_i}$
Stop Clock $t_{\mathcal{L}}$
$\xleftarrow{\ell_i}$
$\ell_i = f(x_{\mathcal{VL}}, \xi_{\mathcal{L}}, c_i, i)$
$\xrightarrow{\ell_i}$
Store $\Delta t_{\mathcal{L}_i}$
$\xleftarrow{r_i}$
$r_i = f(x_{\mathcal{VP}}, \xi_{\mathcal{P}}, \ell_i, i)$
Stop Clock $t_{\mathcal{P}}$
$\xleftarrow{r_i}$
Store $\Delta t_{\mathcal{P}_i}$
Verification phase
Check ℓ_i values
$\xleftarrow{w}$
$\xleftarrow{w}$
$w = g_2(x_{\mathcal{VP}}, \xi_{\mathcal{P}}, \ell_1, ..., \ell_n, r_1, ..., r_n)$
Verify r_i and w, and
check if $\Delta t_{\mathcal{L}_i} \leq \mathsf{t}_{\mathsf{max}}$, $\Delta t_{\mathcal{P}_i} \leq \mathsf{t}'_{\mathsf{max}}$
for $i \in \{1, \dots, n\}$

Figure D.18: The general structure of a two-hop DB protocol.

Initialisation Phase In this phase $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{P}$ calculate some values that will be used in the rest of the protocol. Initially, $\mathcal{V}$ and $\mathcal{P}$ send to $\mathcal{L}$ the messages $m_{\mathcal{V}} = g_0(x_{\mathcal{VL}}, \mathsf{void}, \mathsf{void})$ and $m_{\mathcal{P}} = g_0(x_{\mathcal{VP}}, \mathsf{void}, \mathsf{void})$ respectively. The linker $\mathcal{L}$ broadcasts its message $m_{\mathcal{L}} = g_0(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{P}})$, which can be related to $m_{\mathcal{V}}$ and $m_{\mathcal{P}}$, e.g., a concatenation of them, or be independent, e.g., a randomly selected nonce. Finally, all parties produce a (possibly different) value, which will be used in the next phase. In the two-hop DB we augment the input of the function $g_0 : \mathcal{K} \times \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ to include the case in which $\mathcal{L}$ transmits a manipulation of the messages $m_{\mathcal{V}}$ and $m_{\mathcal{P}}$ generated by $\mathcal{V}$ and $\mathcal{P}$ correspondingly.

Distance-Bounding Phase This phase consists of n time-critical rounds and it uses the response function $f : \mathcal{K} \times \mathcal{X} \times \mathcal{C} \times \mathcal{I} \to \mathcal{R}$. In each round $i \in \{1, \dots, n\}$, $\mathcal{V}$ generates a challenge c_i, transmits it and starts two clocks $t_{\mathcal{L}}$ and $t_{\mathcal{P}}$. The linker

$\mathcal{L}$ receives c_i and evaluates the function f on $x_{\mathcal{VL}}, \xi_{\mathcal{L}}, c_i$ and the round counter i to obtain value $\ell_i \in \mathcal{C}$. Then, $\mathcal{L}$ broadcasts ℓ_i, which will be read by $\mathcal{V}$ as the response to the challenge c_i and by the $\mathcal{P}$ as the i-th challenge. As soon as $\mathcal{V}$ receives ℓ_i it stops the clock $t_{\mathcal{L}}$ and stores the round-trip-time $\Delta t_{\mathcal{L}_i}$. The prover $\mathcal{P}$ replies to ℓ_i with the value $r_i = f(x_{\mathcal{VP}}, \xi_{\mathcal{P}}, \ell_i, i)$. Eventually, $\mathcal{L}$ replies r_i to the verifier, who stops the clock $t_{\mathcal{P}}$ and records the round-trip-time $\Delta t_{\mathcal{P}_i}$.

Verification Phase In this phase, $\mathcal{V}$ checks whether the responses ℓ_i, r_i, $\forall i \in \{1, \ldots, n\}$ are correct and whether the recorded round-trip-times satisfy the conditions $\Delta t_{\mathcal{L}_i} \leq \mathsf{t_{max}}$, and $\Delta t_{\mathcal{P}_i} \leq \mathsf{t'_{max}}$ where $\mathsf{t_{max}} = \mathsf{c} \cdot \mathsf{d}_{\mathcal{V}}$ and $\mathsf{t'_{max}} = \mathsf{c} \cdot (\mathsf{d}_{\mathcal{V}} + \mathsf{d}_{\mathcal{L}})$; c denotes the speed of light, and $d_{\mathcal{V}}$, $d_{\mathcal{P}}$ the communication ranges of $\mathcal{V}$ and $\mathcal{P}$ respectively. If we consider that all entities have the same communication range then $\mathsf{t'_{max}} = 2\mathsf{t_{max}}$. Moreover, in case g_2 outputs a non-void value w, the verifier will also check the correctness of it. Finally, if the verification succeeds, $\mathcal{V}$ states that $\mathcal{P}$ is within its two-hop communication range and authenticated.

In the verification phase, both the linker and the prover are authenticated in terms of the identity authentication and the distance checking.

5 Security Analysis in one-hop and two-hop Distance-Bounding

In this section, we provide the security analysis of the generalisations of one-hop and two-hop DB protocols presented in sections 3 and 4 respectively. We consider a list of threats in the two settings and show general formulas to compute the success probability of the best attacks for each of the threats under consideration against one-hop and two-hop DB protocols. The exact security level of a DB protocol can be computed using the provided formulas, and obviously depends on the specific properties of the employed functions (g_0, g_1, g_2, f). We explicitly calculate these values in Section 6 for several one-hop and two-hop instantiations of DB protocols. In this work we rely on the classical security assumption of DB, namely:

- The verifier $\mathcal{V}$ is honest, i.e., it behaves according to the protocol.
- All entities (honest, malicious and/or external attackers) are aware of how the DB protocol works, e.g., the functions g_0, g_1, g_2, f are public.
- All rounds $i \in \{1, \ldots, n\}$ are independent, which implies that all challenges are equi-probable at each round of the protocol (this is the usual case for most existing DB protocols).

5.1 Threat Model for one-hop DB protocols

The main objective of DB protocols is to protect against the following main threats:

- Distance Fraud (DF): In this case, a dishonest prover $\mathcal{P}^*$ attempts to prove that it is close to the verifier $\mathcal{V}$ while in reality it is far away.
- Mafia Fraud (MF): This threat involves three entities: an honest verifier $\mathcal{V}$, an untrusted prover $\mathcal{P}$ and an adversary $\mathcal{A}$ who acts as *man-in-the-middle* and is located close to $\mathcal{V}$. More precisely, $\mathcal{P}$ and $\mathcal{V}$ are not in close proximity and $\mathcal{A}$ attempts to shorten the distance between $\mathcal{P}$ and $\mathcal{V}$, by convincing $\mathcal{V}$ that it communicates with $\mathcal{P}$, while in reality both $\mathcal{P}$ and $\mathcal{V}$ are communicating

with $\mathcal{A}$. For instance, in order to achieve this, $\mathcal{A}$ could control two nodes[28] ($\mathcal{L}_2$ and $\mathcal{L}_1$ respectively) one near $\mathcal{P}$ and the other near $\mathcal{V}$ (as shown in Figure D.15). Note that, since DB protocols take into account the round-trip-time of the challenge-response pairs, in order to succeed $\mathcal{A}$ cannot simply relay the communication.

- Terrorist Fraud (TF): In this case, similarly to the *mafia fraud*, three entities are involved: a prover $\mathcal{P}^*$, an honest verifier $\mathcal{V}$ and an adversary $\mathcal{A}$ located close to $\mathcal{V}$. Also in this case, the adversary's goal is to shorten the distance between $\mathcal{P}$ and $\mathcal{V}$. However, in this threat the prover is dishonest and helps $\mathcal{A}$ to get authenticated, and more precisely to make it appear that $\mathcal{A}$ is the prover close to $\mathcal{V}$. The attack is successful if $\mathcal{P}^*$ does not reveal any (useful) information to the attacker.

5.2 General Security Analysis for one-hop DB protocols

Since the main threats against DB protocols are *distance fraud* (DF), *mafia fraud* (MF) and *terrorist fraud* (TF) [1, 5], in this section we describe the three attacks in terms of the properties of the functions g_0, g_1, g_2, f introduced in Section 3. We also provide general formulas to compute the attacker's best success probability for any register-based DB protocol captured by our general framework.

- One-hop DF: In this fraud, the attacker is a dishonest prover $\mathcal{P}^*$. In addition to the previously mentioned assumptions (introduction of Section 5), the adversary knows the secret key $x_{\mathcal{VP}}$ and can correctly compute ξ and $m_{\mathcal{P}}$. The attack is considered successful if and only if (a) $\Delta t_i < \mathsf{t}_{\max}$ and (b) $\mathcal{P}^*$'s responses r_i^* are correct, i.e., $r_i^* = r_i$ at any round i, where r_i is the honest response to challenge c_i sent by the verifier. Due to the actual distance between $\mathcal{P}^*$ and $\mathcal{V}$, in order to fool the verifier in point (a) the malicious prover has to send a response *before* receiving the corresponding challenge. In this way, the time-difference Δt_i measured by $\mathcal{V}$ will be smaller than the actual one. In order to achieve distance shortening, the responses are computed right after the initialization phase, before the time-critical DB phase. The best strategy to achieve (b) is for $\mathcal{P}^*$ to choose the response r_i^* that is most likely to happen, independently of the challenge c_i. This can be easily achieved by evaluating the function f on $x_{\mathcal{VP}}, \xi, i$ and try all the possible values for the challenge, obtaining a list $r_i^{(j)} = f(x_{\mathcal{VP}}, \xi, c^{(j)}, i)$, $j \in \{1, \ldots, |\mathcal{C}|\}$. In order to maximize the success probability, the malicious prover will take the value r_i^* that has the largest number pre-images in round i, i.e., the r_i^* that appears most frequently in the list $\{r_i^{(1)}, \ldots, r_i^{(|\mathcal{C}|)}\}$. Formally, $r_i^* = \max_{r \in \mathcal{R}} |\{c \in \mathcal{C}, r = f(x_{\mathcal{VP}}, \xi, c, i)\}|$, where the pre-images are taken according to the known values of $x_{\mathcal{VP}}, \xi$ and i. Let $p_{r_i} \in (0,1)$ be the probability that the attacker $\mathcal{P}^*$ guesses the correct r_i, and $\mathbb{P}_{\mathtt{DF1}}$ denote the success probability of the distance fraud in the one hop setting, then, assuming that all rounds are independent[29], the following holds:

$$\mathbb{P}_{\mathtt{DF1}} = \prod_{i=1}^{n} p_{r_i} \tag{D.31}$$

[28]Often, in order to ease the understanding the two adversarial-controlled nodes are *collapsed* into one single entity, the adversary $\mathcal{A}$.

[29]This assumption can be re-formulated as, all challenges are equi-probable at each round of the protocol, which is the case for all register-based protocol considered in this paper.

Let c_i denote the actual challenge for round i, $p_{r_i} = \Pr\left(r_i^* = f(x_{\mathcal{VP}}, \xi, c_i, i)\right) \geq \max\left\{\frac{1}{|\mathcal{R}|}, \frac{1}{|\mathcal{C}|}\right\}$. We refer the reader to Section 6.2 for concrete examples of values for p_{r_i}.

- One-hop MF: Differently from DF, in MF the attacker $\mathcal{A}$ is an entity external to the protocol, and therefore does not know $x_{\mathcal{VP}}, \xi$. The attack is considered successful if $\mathcal{A}$ manages to pass the protocol as if she were the prover $\mathcal{P}$ but is located in $\mathcal{A}$'s position (i.e., close to $\mathcal{V}$). Since, by assumption, $\mathcal{A}$ lies in the verifier's DB range, the only condition to have a valid forgery is that $\mathcal{A}$ produces answers r_i^* and a final transcript check w^* (if needed) that correctly pass the verification performed by $\mathcal{V}$.

  In order to output correct replies r_i^*, the adversary can adopt the classical (optimal) strategy against DB protocols: in the initialisation phase, relay the transmissions between $\mathcal{V}$ and $\mathcal{P}$; start a *pre-ask* session with the prover, i.e., query $\mathcal{P}$ for the responses r_i to challenges c_i^* of the attacker's choice; collect the final message $w = g_2(x_{\mathcal{VP}}, \xi, c_1^*, \dots c_n^*, r_1, \dots r_n)$. When the actual DB phase starts, $\mathcal{A}$ runs the protocol with $\mathcal{V}$ pretending to be $\mathcal{P}$. Every time the verifier's challenge equals the malicious pre-asked challenge ($c_i = c_i^*$) the attacker can correctly reply using the response $r_i^* = r_i$ collected during the *pre-ask* session. If $c_i = c_i^*$ for all $i \in \{1, \dots, n\}$, $\mathcal{A}$ can succeed by relaying $w^* = w$. On the other hand, if $c_i \neq c_i^*$ in at least one round $i \in \{1, \dots, n\}$, $\mathcal{A}$ returns a random element r_i^* from the set of responses $\mathcal{R}$, and has to tamper with the final message of the DB protocol. To formally define the probability of one-hop mafia fraud ($\mathbb{P}_{\mathtt{MF1}}$) we introduce three events: G_k : $\mathcal{A}$ guesses correctly exactly k challenges (among the n rounds); E_f : $\mathcal{A}$ successfully guesses the correct answer to all of the verifier's challenges; E_w : $\mathcal{A}$ forges the final message w. Then, by the law of total probability we have:

$$\mathbb{P}_{\mathtt{MF1}} = \sum_{k=0}^{n} \Pr(G_k) \Pr(E_f | G_k) \Pr(E_w | E_f \cap G_k) \tag{D.32}$$

  Let $\varepsilon_w(k) = Pr(E_w | E_f \cap G_k)$, for $k \in \{1, \dots, n\}$. It is immediate to see that, independently of the function g_2, $\varepsilon_w(n) = 1$. Indeed, when $k = n$ the adversary has successfully guessed all the verifier's challenges, and $\mathcal{A}$ can set $w^* = w$ obtained from $\mathcal{P}$ in the *pre-ask* phase. For $1 \leq k \leq n-1$, $\varepsilon_w(k)$ is either negligible, as it corresponds to the unforgeability of the employed signature/commitment scheme, or $\varepsilon_w(k) = 1$, e.g., when the output of g_2 is void, or it can be computed using solely public data. In any case, for $k < n$, $\varepsilon_w(k) = \varepsilon$ is a constant value. We can thus split the summation in Equation (D.32) into the $k = n$ term, and the $k < n$ term:

$$\begin{aligned} \mathbb{P}_{\mathtt{MF1}} &= \Pr(G_n) \Pr(E_f | G_n) \cdot 1 + \sum_{k=0}^{n-1} \Pr(G_k) \Pr(E_f | G_k) \cdot \epsilon \\ &= p_{c_i}^n + \epsilon \sum_{k=0}^{n-1} \binom{n}{k} p_{c_i}^k \left(1 - p_{c_i}\right)^{n-k} \left(\frac{1}{|\mathcal{R}|}\right)^{n-k} \end{aligned} \tag{D.33}$$

  The value p_{c_i} in the above expression corresponds to the probability of correctly guessing the challenge c_i. In our security model $p_{c_i} = \frac{1}{|\mathcal{C}|}$. Equation (D.33) translates the intuition highlighted before: a mafia fraud attack is successful if either $\mathcal{A}$ guesses all the challenges correctly, or $\mathcal{A}$ guesses correctly all the replies r_i^* for which $c_i^* \neq c_i$ and produces a valid w^*. In particular, when $|\mathcal{R}| = |\mathcal{C}|$, which is the case for most DB protocols, Equation (D.33) becomes: $\mathbb{P}_{\mathtt{MF1}} = \frac{1}{|\mathcal{C}|}^n +$ negligible terms, for an unforgeable g_2; and $\mathbb{P}_{\mathtt{MF1}} = \frac{1}{|\mathcal{C}|}^n \left(2 - \frac{1}{|\mathcal{C}|}\right)^n$, when g_2 is forgeable (i.e., $\epsilon = 1$).

- One-hop TF: Terrorist fraud is a challenging threat to defend against and different definitions are given regarding its success [1, 4, 10]. Although it is not possible to design a general formula to capture all the different definitions, we identify the common notion behind all of them: the malicious prover $\mathcal{P}^*$ is willing to help $\mathcal{A}$ as long as no compromising information about the long-term secret key $x_{\mathcal{VP}}$ is revealed. Existing DB protocols that are TF resistant (according to at least one of the cited definitions) prevent the forgery by forcing the prover to partially or fully disclose the secret key to the attacker. Table D.3 provides the values for the best success probability of TF of the protocols investigated in this paper.

5.3 General Security Analysis of two-hop DB protocols

We begin the security analysis of two-hop DB protocols by identifying two types of adversaries: *internal adversaries* and *external adversaries*. As the name suggests, internal adversaries are entities that take part in the protocol and pretend to be honest but actually behave in a dishonest way (we mark these malicious entities with a * symbol). Internal adversaries can be a dishonest prover $\mathcal{P}^*$ and/or a dishonest linker $\mathcal{L}^*$. An external adversary $\mathcal{A}$ is an entity (possibly controlling multiple entities) that is not supposed to take part in the protocol, however she interferes with it. In order to maximise the success of the attacks an external adversary would place herself between the $\mathcal{V}$ and the $\mathcal{L}$ or between the $\mathcal{L}$ and the $\mathcal{P}$. In general, malicious entities (internal or external) have two simultaneous goals: (a) to successfully pass the protocol (impersonating $\mathcal{P}$), and (b) shorten the distance between a legitimate entity (i.e., $\mathcal{P}$ or $\mathcal{L}$) and $\mathcal{V}$. On top of the assumptions in Section 5 for one-hop DB protocols, in the two-hop case we enable adversaries ($\mathcal{A}$, $\mathcal{P}^*$, $\mathcal{L}^*$) to send unilateral messages. For example, $\mathcal{L}^*$ is able to query $\mathcal{P}$ without $\mathcal{V}$ *receiving* the message(s) as well.

Internal Adversaries In two-hop DB protocols there are two possible attacks that involve solely internal adversaries, and the attack scenarios resemble the ones considered against one-hop DB protocols. However, the fact that the adversaries are *internal*, the protocol makes the resulting security analysis quite different from the one-hop cases. Since the linker is untrusted in the two-hop DB scenarios, the two corresponding attack cases are: 1) dishonest linker, honest prover ($\mathcal{L}^*$,$\mathcal{P}$); and 2) dishonest linker, dishonest prover ($\mathcal{L}^*$,$\mathcal{P}^*$), where we denote them as $\mathcal{L}^*\mathcal{P}$, and $\mathcal{L}^*\mathcal{P}^*$respectively.

- Case $\mathcal{L}^*\mathcal{P}$- ($\mathcal{L}^*$,$\mathcal{P}$): in this attack, the prover $\mathcal{P}$ is honest while the linker $\mathcal{L}^*$ is malicious. Although the setting resembles one-hop MF, in the two-hop case the fact that the attacker is an internal entity for the protocol implies that $\mathcal{L}^*$ has a greater advantage with respect to the classical MF attacker $\mathcal{A}$. More precisely, $\mathcal{L}^*$ additionally knows $x_{\mathcal{VL}}, \xi_{\mathcal{L}}$ and $m_{\mathcal{P}}$. By running a strategy similar to the one-hop MF, $\mathcal{L}^*$ can *pre-ask* $\mathcal{P}$ with challenge-responses ℓ_i^* of its choice. The values ℓ_i^* are chosen without knowing the corresponding challenge c_i coming from $\mathcal{V}$. In some cases, however, $\mathcal{L}^*$ is able to predict the exact value of some ℓ_i. Consider for instance the two-hop Hancke and Kuhn [12] DB protocol depicted in Figure D.20: the g_1 function in the initialisation phase generates two equal-length registers, say $\xi_{\mathcal{L}} = (R_0, R_1) \in \{0,1\}^{2n}$. In the time-critical phase, the response function f outputs $\ell_i = (R_{c_i})_i$ on an input challenge c_i, i.e., the challenge-response ℓ_i corresponding to the challenge c_i is the i-th entry of the c_i-th register. Whenever $(R_0)_i = (R_1)_i$, $\mathcal{L}^*$ can determine the correct $\ell_i = (R_0)_i$ without waiting for the challenge c_i. For the remaining rounds, in which $(R_0)_i \neq (R_1)_i$, $\mathcal{L}^*$ cannot pre-determine the exact challenge-response value and will simply choose the most likely one, i.e., ℓ_i^* s.t. $|\{c \in \mathcal{C}, \ell_i^* = f(x_{\mathcal{VL}}, \xi_{\mathcal{L}}, m_{\mathcal{P}}, c)\}| = \max_{\ell \in \mathcal{R}} |\{c \in \mathcal{C}, \ell = f(x_{\mathcal{VL}}, \xi_{\mathcal{L}}, m_{\mathcal{P}}, c)\}|$. During the actual DB phase, upon receiving c_i from $\mathcal{V}$, the malicious linker will find out whether its guess on ℓ_i^* was correct or not. Every time it holds that $\ell_i^* = (R_{c_i})_i$, $\mathcal{L}^*$ will use the value r_i that $\mathcal{P}$ honestly provided in the *pre-ask* session (for the prover it was the DB phase). Otherwise, $\mathcal{L}^*$ returns a random guess r_i^* on the value of $\mathcal{P}$'s response.

To formally definite the success probability of two-hop $\mathcal{L}^*\mathcal{P}$ attack, consider the following three events: G_k : $\mathcal{A}$ guesses correctly exactly k values $\ell_{i_1}, \ldots \ell_{i_k}$ (among the n rounds); E_f : $\mathcal{A}$ successfully guesses the correct answer r_i to the verifier's challenge c_i for all the n rounds; E_w : $\mathcal{A}$ forges the final message w. By the law of total probability we have:

$$\begin{aligned}
\Pr(\mathcal{L}^*, \mathcal{P}) &= \sum_{k=0}^{n} \Pr(G_k) \Pr(E_f | G_i) \Pr(E_w | E_f \cap G_k) \\
&= \left(\prod_{i=1}^{n} p_{\ell_i} \right) + \epsilon \left[\sum_{\substack{k\,=\,0,\\ I \subsetneq \{1,\ldots,n\},\\ |I|\,=\,k}}^{n-1} \binom{n}{k} \left(\prod_{i \in I} p_{\ell_i} \right) \cdot \right. \\
&\quad \left. \cdot \left(\prod_{i \in \{1,\ldots,n\} \setminus I} (1 - p_{\ell_i}) \right) \left(\frac{1}{|\mathcal{R}|} \right)^{n-k} \right] \qquad \text{(D.34)}
\end{aligned}$$

where, similarly to the case of one-hop mafia fraud, we split the summation into two terms: the $k = n$ case, corresponding to the case $\mathcal{A}$ guesses all the values ℓ_i correctly; and the case where $\mathcal{A}$ guesses correctly $k < n$ values of ℓ_i and outputs the correct responses $r_i^* = r_i$ for all $1 \leq i \leq n$ and forges w^*. Similarly to the one-hop mafia fraud case, ε_w denotes the probability that $\mathcal{L}^*$ forges the value $w = g_2(x_{\mathcal{VP}}, \xi_{\mathcal{P}}, \ell_1, \ldots, \ell_n, r_1^*, \ldots, r_n^*)$ with $r_j^* \neq r_j$ for some $j \in \{1, \ldots, n\}$.

We observe that $\Pr_{\mathcal{L}^*\mathcal{P}} = \mathbb{P}_{\texttt{MF1}}$ whenever $p_{\ell_i} = p_{c_i} = \frac{1}{|\mathcal{R}|}$ at each round. This corresponds to $\mathcal{L}^*$ actually having no *advantage* in pre-determining the values ℓ_i. Intuitively, the longer the output of g_1 (i.e., the larger the number of registers) the lower the value of p_{ℓ_i}, and the closer $\Pr_{\mathcal{L}^*\mathcal{P}}$ is to $\mathbb{P}_{\texttt{MF1}}$.

- Case $\mathcal{L}^*\mathcal{P}^*$- ($\mathcal{L}^*$, $\mathcal{P}^*$): In this attack, both the prover and the linker are malicious and collaborate with each other. Although this scenario resembles TF in the one-hop case, there is a substantial difference: $\mathcal{L}^*$ is an insider in the protocol and potentially can exploit information about the values of ℓ_i (as in the two-hop ($\mathcal{L}^*$, $\mathcal{P}$) case). However, differently from one-hop TF, $\mathcal{L}^*$ must be careful not to leak secret information (e.g., $x_{\mathcal{L}^*}$) to $\mathcal{P}^*$. We distinguish two scenarios for this attack:

(i) $\mathcal{P}^*$ helps $\mathcal{L}^*$ to pass one protocol run with probability 1. However, if $\mathcal{L}^*$ later on runs a two-hop $\mathcal{L}^*\mathcal{P}$ attack, she should have no advantage (i.e., the knowledge leaked by $\mathcal{P}^*$ does not increase the chances of $\mathcal{L}^*$ to cheat alone).

(ii) $\mathcal{P}^*$ helps $\mathcal{L}^*$ to pass one protocol run with a certain probability $\Pr(\mathcal{L}^*, \mathcal{P}^*) \leq 1$, without $\mathcal{P}^*$ or $\mathcal{L}^*$ leaking any secret information to each other.

Even though the two definitions appear distinct, they essentially capture the aim of the $\mathcal{L}^*\mathcal{P}^*$attack: the higher the chance to pass the DB protocol the larger amount of secret information needs to be leaked. The two expressions we provide are consistent with each other and respectively answer the questions (i) *Can $\mathcal{L}^*$ pass the two-hop* DB *protocol with the help of $\mathcal{P}^*$, without any of them leaking information that can be useful to cheat in a subsequent protocol run?* and (ii) *If no information is leaked by $\mathcal{L}^*$ or $\mathcal{P}^*$, this could be useful for future frauds, what is the probability that $\mathcal{L}^*$ and $\mathcal{P}^*$ together successfully succeed in cheating on the prover's distance?* In the sequel, when mentioning the adversary's success probability with respect to the $\mathcal{L}^*\mathcal{P}^*$ attack, we will refer only to the case (ii). Similar to the one-hop TF, it is not straight-forward to give an equation for the success probability of $\mathcal{L}^*\mathcal{P}^*$for the general two-hop DB protocol in Figure D.18. The main problem consists of defining the *leakage* of information of a DB protocol in a general way, since this quantity depends on the properties of the specific response function f, and thus differs for each DB protocol. We will compute the leakage of information explicitly for the five DB protocols considered in this paper in Section 6.2.

External Adversaries an external adversary $\mathcal{A}$ (man-in-the-middle) is a malicious entity that takes part in a two-hop DB protocol, as a ghost, i.e., $\mathcal{A} \notin \{\mathcal{P}^*, \mathcal{L}^*\}$. It can be located between $\mathcal{V}$ and $\mathcal{L}$, or between $\mathcal{L}$ and $\mathcal{P}$. In both cases, the adversary has no direct access to the secret key of the parties $\mathcal{L}$ and $\mathcal{P}$. Therefore, $\mathcal{A}$ is as powerless as a one-hop MF adversary. The success probability of $\mathcal{A}$ equals $\mathbb{P}_{\texttt{MF1}}$ in equation (D.33). We underline that there is no interest for an $\mathcal{A}$ settled between $\mathcal{V}$ and $\mathcal{L}$ to impersonate both $\mathcal{L}$ and $\mathcal{P}$, as this will only lower the success probability. Let us discuss two-hop collusions. If the prover $\mathcal{P}$ helps $\mathcal{A}$ to pass the protocol, we are exactly in the same situation as one-hop TF. Similarly, for the collusion between $\mathcal{L}$ and $\mathcal{A}$; however, in this case the probability should be higher, as $\mathcal{L}$ (and so $\mathcal{A}$) does not need to additionally evaluate function g_2. In general, the success probability of external attackers is always lower than obtained by internal ones, simply because internal adversaries have access to the same information as external ones and, in addition, possess at least one secret key needed in the protocol run.

6 Protocols Evaluation

We have provided the security analysis of the novel model for general two-hop DB protocols. In this section, we evaluate our proposed model on five existing DB protocols: Hancke and Kuhn [12], Brands and Chaum [6], Reid *et al.* [22], Swiss-Knife [16] and $\mathsf{SKI}_{\mathsf{extend}}$ from SKI [4]. We choose these five protocols because their employed functions (g_0, g_1, g_2, f) are representative of the majority of the existing DB protocols. According to our model in Section 5, these four functions are the key factors that determine any DB protocol, one-hop or two-hop. Therefore, all aforementioned functions influence the adversary's success probability in the three main frauds. In the following, we first give a representative example of transforming a one-hop DB protocol to two-hop and employ the Hancke and Kuhn's protocol [12] to a two-hop case. Then, we briefly revisit the other four selected protocols. Next, we calculate the best success probabilities of the main attacks against one-hop and two-hop DB protocols described in Sections 5.2 and 5.3 respectively. Finally, we verify our theoretical analysis through some simulation experiments for the three main attacks (distance fraud, mafia fraud and terrorist fraud).

6.1 The Selected DB Protocols

Hancke and Kuhn (HK) [12] protocol

In the Hancke-Kuhn protocol [12] (depicted in Figure D.19), the verifier $\mathcal{V}$ and the prover $\mathcal{P}$ share a secret key $x_{\mathcal{VP}}$. During the *initialisation phase* the two parties exchange some randomly selected nonces $m_{\mathcal{V}} = N_{\mathcal{V}}$ and $m_{\mathcal{P}} = N_{\mathcal{P}}$ through an appropriate function g_0 (e.g., a PRNG function). $\mathcal{V}$ and $\mathcal{P}$ evaluate a pseudorandom function (PRF) f (corresponding to g_1 in the general description of one-hop DB protocols) on the exchanged nonces and the shared key, obtaining two n-bits sequences, a_0 and a_1 ($\xi = a_0||a_1$). The *distance-bounding phase* consists of n rounds: for $i \in \{1, \ldots, n\}$, the verifier $\mathcal{V}$ sends a random bit c_i as a challenge to $\mathcal{P}$. Upon receiving c_i, the prover $\mathcal{P}$ sends $(a_{c_i})_i$ as a response back to the verifier. After the last round, the verifier checks whether the n responses $r_1, \ldots, r_n$ are correct and if each round-trip-time (denoted by Δt_i) is less than or equal to a pre-defined maximum delay-threshold $\mathsf{t_{max}}$. If all previous constraints hold, the verifier states that the prover is *close* enough and authenticated. There is no final message, i.e., the output w of g_2 is void.

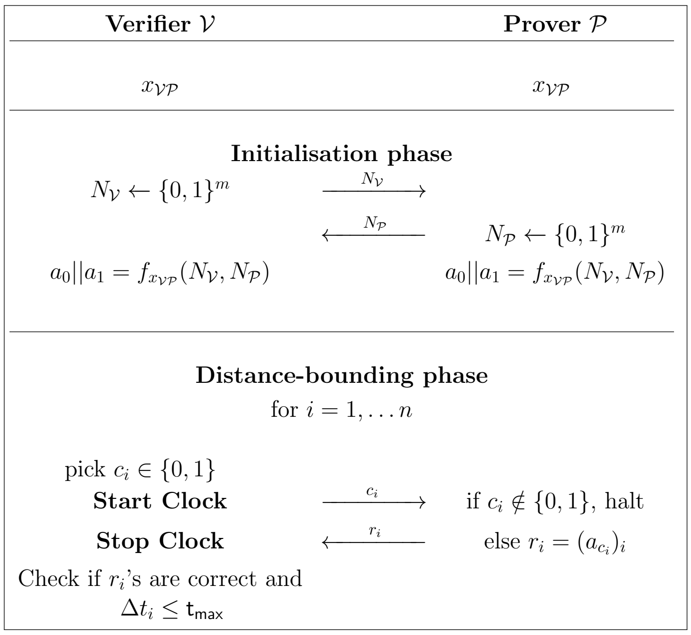

Verifier $\mathcal{V}$
Prover $\mathcal{P}$
$x_{\mathcal{VP}}$
$x_{\mathcal{VP}}$
Initialisation phase
$N_{\mathcal{V}} \leftarrow \{0,1\}^m$
$\xrightarrow{N_{\mathcal{V}}}$
$\xleftarrow{N_{\mathcal{P}}}$
$N_{\mathcal{P}} \leftarrow \{0,1\}^m$
$a_0||a_1 = f_{x_{\mathcal{VP}}}(N_{\mathcal{V}}, N_{\mathcal{P}})$
$a_0||a_1 = f_{x_{\mathcal{VP}}}(N_{\mathcal{V}}, N_{\mathcal{P}})$
Distance-bounding phase
for $i = 1, \ldots n$
pick $c_i \in \{0,1\}$
Start Clock
$\xrightarrow{c_i}$
if $c_i \notin \{0,1\}$, halt
Stop Clock
$\xleftarrow{r_i}$
else $r_i = (a_{c_i})_i$
Check if r_i's are correct and
$\Delta t_i \leq \mathsf{t_{max}}$

Figure D.19: The Hancke-Kuhn protocol.

Figure D.20 depicts the extension of the HK protocol. We assume that $\mathcal{P}$ and $\mathcal{L}$ respectively share secret keys $x_{\mathcal{VP}}$ and $x_{\mathcal{VL}}$ with the verifier $\mathcal{V}$ only. In the first

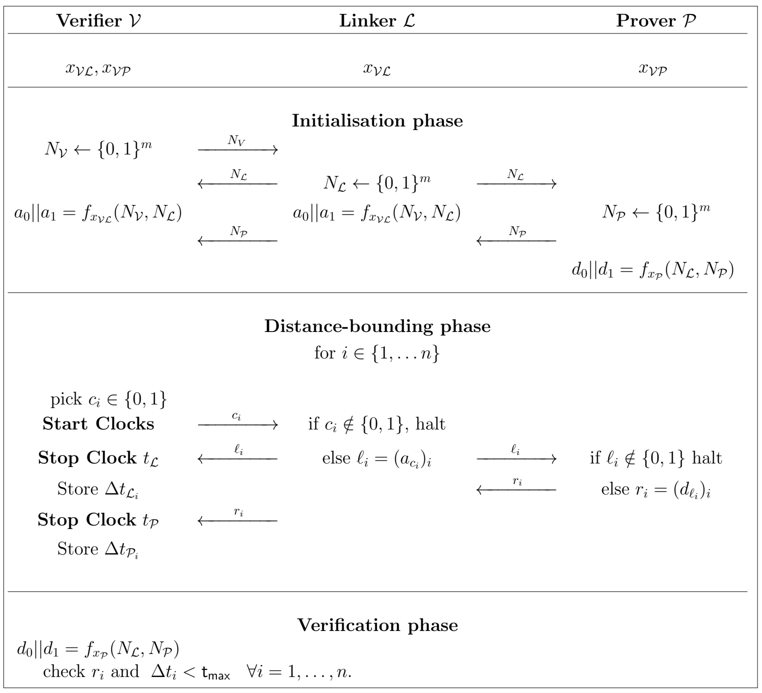

Verifier $\mathcal{V}$
Linker $\mathcal{L}$
Prover $\mathcal{P}$
$x_{\mathcal{VL}}, x_{\mathcal{VP}}$
$x_{\mathcal{VL}}$
$x_{\mathcal{VP}}$
Initialisation phase
$N_\mathcal{V} \leftarrow \{0,1\}^m$
$\xrightarrow{N_V}$
$\xleftarrow{N_\mathcal{L}}$
$N_\mathcal{L} \leftarrow \{0,1\}^m$
$\xrightarrow{N_\mathcal{L}}$
$a_0||a_1 = f_{x_{\mathcal{VL}}}(N_\mathcal{V}, N_\mathcal{L})$
$a_0||a_1 = f_{x_{\mathcal{VL}}}(N_\mathcal{V}, N_\mathcal{L})$
$N_\mathcal{P} \leftarrow \{0,1\}^m$
$\xleftarrow{N_\mathcal{P}}$
$\xleftarrow{N_\mathcal{P}}$
$d_0||d_1 = f_{x_\mathcal{P}}(N_\mathcal{L}, N_\mathcal{P})$
Distance-bounding phase
for $i \in \{1, \dots n\}$
pick $c_i \in \{0,1\}$
Start Clocks
$\xrightarrow{c_i}$
if $c_i \notin \{0,1\}$, halt
Stop Clock $t_\mathcal{L}$
$\xleftarrow{\ell_i}$
else $\ell_i = (a_{c_i})_i$
$\xrightarrow{\ell_i}$
if $\ell_i \notin \{0,1\}$ halt
Store $\Delta t_{\mathcal{L}_i}$
$\xleftarrow{r_i}$
else $r_i = (d_{\ell_i})_i$
Stop Clock $t_\mathcal{P}$
$\xleftarrow{r_i}$
Store $\Delta t_{\mathcal{P}_i}$
Verification phase
$d_0||d_1 = f_{x_\mathcal{P}}(N_\mathcal{L}, N_\mathcal{P})$
check r_i and $\Delta t_i < \mathsf{t_{max}}$ $\forall i = 1, \dots, n.$

Figure D.20: The two-hop Hancke-Kuhn DB protocol (extension to three participants)

phase, each participant ($\mathcal{V}$, $\mathcal{L}$ and $\mathcal{P}$) generates a random string of bits (nonce) ($m_\mathcal{V} = N_V$, $m_\mathcal{L} = N_L$ and $m_\mathcal{P} = N_P$ respectively) using a function g_0 (e.g., a PRNG). Each participant uses the two nonces as input to the PRF g_1 (with its corresponding key) to produce the variables $\xi_\mathcal{L} = a_0||a_1$ (for the verifier and linker) and $\xi_\mathcal{P} = d_0||d_1$ (for the prover).

In this way, $\mathcal{V}$ is able to check the correctness of $\mathcal{L}$'s responses during the DB phase. The DB phase consists of n rounds that run as follows. The verifier generates a (random) challenge-bit c_i and transmits it. The linker checks whether the received input is acceptable, if so it reads the i-th entry of the c_i-th register, namely $(a_{c_i})_i$, and transmits this bit, say ℓ_i. Both $\mathcal{V}$ and $\mathcal{P}$ get ℓ_i, the prover sends its reply $r_i = (d_{\ell_i})_i$ to the linker, who forwards it to the verifier.

The protocol ends with the *verification phase*, where $\mathcal{V}$ estimates the distance of $\mathcal{P}$ by computing an average of the times Δt_i that elapses between the instant when the challenge c_i is sent (by $\mathcal{V}$) and the instant when $\mathcal{V}$ receives the corresponding response r_i. $\mathcal{V}$ can check the authenticity of all the responses received from $\mathcal{P}$ and thus authenticate the prover. The function g_2 always returns $w = \mathsf{void}$, and we omit to write it.

Brands and Chaum (BC) [6]

In the Brands and Chaum's protocol, g_0 is a commitment, g_1 is a pseudorandom number generator (PRNG) and they are executed only in the prover side. Thus, $m_\mathcal{V}$ is a null string (i.e., there is no $m_\mathcal{V}$ in this protocol). g_2 is a combination of an open commitment and a signature scheme. The response function is $f = c_i \oplus (N_\mathcal{P})_i$, where $c_i \in \{0,1\}$.

Reid *et al.* [22]

In Reid *et al.*'s protocol, g_0, f, g_2 are the same as that of Hancke and Kuhn's protocol. The difference is in g_1, i.e., g_1 is a combination of a PRF and an encryption scheme Enc. In particular, $\xi = a_0||a_1$, where a_0 is an output of PRF and $a_1 = \mathsf{Enc}_{a_0}(x)$.

Swiss-Knife [16]

The Swiss-Knife protocol is an extension of Reid *et al.*'s protocol, and thus shares the same g_0 and the response function f. g_1 is a combination of a PRF and a one-time pad encryption function. In particular, $\xi = a_0||a_1$, where a_0 is an output of PRF and $a_1 = a_0 \oplus x$. The response function is $f = (a_{c_i})_i$, where $c_i \in \{0,1\}$. g_1, g_2 are PRF. Note that in the Swiss-Knife protocol, there is a fourth non-time critical phase (i.e. verification phase) where the verifier sends the prover the final message to authenticate himself. Thus, this protocol achieves mutual authentication. However, this does not influence the security analysis of this protocol (does not change any success probability). Therefore, hereafter when we talk about the Swiss-Knife protocol under our general DB model, we mean the Swiss-Knife protocol without the fourth slow (non-time critical) message.

SKI and $\mathsf{SKI}_{\mathsf{extend}}$ [4]

Boureanu *et al.* proposed the first family of provably secure DB protocols, named SKI. This family of DB protocols introduced the use of *circular-keying* PRF functions and PRF *masking* to provide resistance against a generalised version of mafia and terrorist as well as distance fraud attacks. In one of the instantiations proposed by Boureanu *et al.*, g_0 is a function that generates uniformly at random $m_{\mathcal{V}} = N_{\mathcal{V}}$ and $m_{\mathcal{P}} = N_{\mathcal{P}}$. g_1 is derived by masking the output of a PRF (that employs as input the random values $N_{\mathcal{V}}$ and $N_{\mathcal{P}}$) with a random mask M. The result is denoted by $\xi = a_1||a_2 = M \oplus f(x, N_{\mathcal{V}}, N_{\mathcal{P}})$. The response function is $f = (a_{c_i})_i$ for $c_i \in \{1,2\}$, $a_{c_i} \in \{0,1\}$ and $f = x'_i + (a_1)_i + (a_2)_i \mod 2$ for $c_i = 3$, where $c_i \in \{1,2,3\}$, $x' = L(x)$, and L is a linear transformation. There is no final message and thus w and g_2 are void. Obviously, a response is either 0 or 1 and a challenge belongs to $\{1,2,3\}$. This implies that the response space and the challenge space are different. To ease understand and employ the described approach of extending one-hop DB protocols to DB protocols, we adopt a modified version of the SKI family of protocols and derive a new protocol that we call $\mathsf{SKI}_{\mathsf{extend}}$. More precisely, in $\mathsf{SKI}_{\mathsf{extend}}$ we set both of the challenge and response space as $\{0,1,2\}$. Now the response function becomes $f = (a_{c_i})_i$ for $c_i \in \{0,1\}$ and $f = x'_i + (a_0)_i + (a_1)_i \mod 3$ for $c_i = 2$, where $a_0, a_1 \in \mathbb{F}_3$ are two vectors consisting of n random numbers, and thus $(a_0)_i, (a_1)_i, c_i \in \{0,1,2\}$.

6.2 Concrete Security Analysis for Five Selected DB Protocols

In the following, we shall show how to apply our security analysis model to compute the concrete success probabilities of the chosen five protocols in the cases of both one-hop and two-hop scenarios. In all these protocols, each round in the DB phase is independent of another, i.e., the response of the current round r_i is not influenced by previous r_{i-1} or c_{i-1}. Thus, we only need to compute the success probability for each round and then we can obtain the success probability for n rounds immediately.

Table D.3: Comparison of the best case success probabilities of attacks against five DB protocols. The values shown for the one-hop case are the ones provided by the authors in the corresponding papers.

	One-hop DF	One-hop MF	One-hop TF	Two-hop $\mathcal{L}^*\mathcal{P}$	Two-hop $\mathcal{L}^*\mathcal{P}^*$
BC	$(\frac{1}{2})^n$	$(\frac{1}{2})^n$	1	$(\frac{1}{2})^n$	1
HK	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$	1	$(\frac{7}{8})^n$	1
Reid *et al.*	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$	$(\frac{7}{8})^n$	1
Swiss-Knife	$(\frac{3}{4})^n$	$(\frac{1}{2})^n$	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$
$\mathsf{SKI}_{\mathsf{Extend}}$	$(\frac{17}{27})^n$	$(\frac{2}{3})^n$	$(\frac{7}{9})^n$	$(\frac{3}{4})^n$	$(\frac{25}{27})^n$

One-hop Distance Fraud

Using equation (D.31), we only need to calculate p_{r_i}. In the Brands and Chaum's protocol, the *i-th* response r_i equals to $c_i \oplus (N_{\mathcal{P}})_i$, where the attacker $\mathcal{P}^*$ knows $N_{\mathcal{P}}$ but not c_i, where $c_i, r_i \in \{0,1\}$. Thus, $p_{r_i} = \frac{1}{2}$ and we get $\mathbb{P}_{\mathtt{DF1}} = Pr(\mathcal{L}, \mathcal{P}^*) = (\frac{1}{2})^n$.

In the HK, Reid *et al.*, Swiss-Knife, and $\mathsf{SKI}_{\mathsf{extend}}$ protocols, the responses come from more than one register and thus the attacker gets a higher advantage. Let's take the HK protocol as an example, in which the response function f is $r_i = (a_{c_i})_i$, where $c_i, r_i \in \{0,1\}$, and a_0, a_1 are two n-bit secret registers (outputs of a PRF). For each round, the attacker already knows $(a_0)_i$ and $(a_1)_i$, but she has no idea of c_i. She can list all the possible $c_i \in \{0,1\}$ and thus can find the most likely response. With a probability of $\frac{1}{2}$ no matter if c_i equals 0 or 1, the response is the same (i.e., $(a_0)_i = (a_1)_i$), then the attacker definitely knows the correct r_i in advance. For the other half, the attacker randomly guesses r_i^* and thus $p_{r_i} = \frac{1}{2} * 1 + (1 - \frac{1}{2}) * \frac{1}{2} = \frac{3}{4}$. Using the same strategy for the other protocols we can get the specific values as shown in Table D.3.

One-hop Mafia Fraud

For the one-hop MF, we can apply equation D.33, which requires the value of $|C|$, $|R|$ and ε_w. Protocols like HK, Reid *et al.*, and $\mathsf{SKI}_{\mathsf{extend}}$ have no final signature, that is, w is a null string in these protocols. Therefore, for these protocols the value of ε_w is 1, which means the attacker does not need to forge w. In the BC and Swiss-Knife protocols, the attacker has to forge a valid w without knowing the secret key used in g_2. If g_2 is secure, then ε_w is negligible. Based on the above analysis, we take the HK protocol as an example, where $|\mathcal{C}| = |\mathcal{R}| = 2$, $\varepsilon_w = 1$ and thus, we get $\mathbb{P}_{\mathtt{MF1}} = (\frac{1}{2} + (1 - \frac{1}{2}) * \frac{1}{2} * \varepsilon_w)^n = (\frac{3}{4})^n$. The analysis for mafia fraud in BC, Reid *et al.*, Swiss-Knife, and $\mathsf{SKI}_{\mathsf{extend}}$ protocols is similar.

One-hop Terrorist Fraud

Protocols like BC, HK are not secure against terrorist fraud, since the malicious prover can always help the attacker to pass the current protocol run, without leaking any secret information, which means that the information that $\mathcal{P}^*$ gives to the attacker in the current protocol run does not help to increase the attacker's success probability in a future protocol run. Now we will focus on the Reid *et al.*, Swiss-Knife, and $\mathsf{SKI}_{\mathsf{extend}}$ protocols.

In the Reid *et al.* protocol, the prover has to give all the response registers to the adversary, which results in revealing the secret key x. Therefore, it prevents the one-hop TF. Both the Swiss-Knife and the $\mathsf{SKI}_{\mathsf{extend}}$ protocols use secret sharing schemes to prevent terrorist fraud. Suppose an (m, m) secret-sharing scheme is used. The prover $\mathcal{P}^*$ can at most give $(m-1)$ shares of the secrets (response registers) to the adversary, otherwise the secret will be revealed. In the case when a challenge requires a response that comes from the last share of the secrets that is not sent to the adversary, the adversary can send a random response as its answer. Thus, the success probability of one-hop TF is:

$$\mathbb{P}_{\texttt{TF1}} = \big(\frac{m-1}{m} \cdot 1 + \frac{1}{m} \cdot \frac{1}{m}\big)^n = \big(1 - \frac{1}{m} + \frac{1}{m^2}\big)^n \tag{D.35}$$

According to equation D.35, we can obtain the success probability of one-hop TF for Swiss-Knife and $\mathsf{SKI}_{\mathsf{extend}}$ are $(\frac{3}{4})^n$ and $(\frac{7}{9})^n$, respectively.

Two-hop $\mathcal{L}^*\mathcal{P}$

For the two-hop $\mathcal{L}^*\mathcal{P}$, according to equation (D.34), besides p_{c_i} and ε_w, we also need to compute p_{ℓ_i}, i.e., the probability that the attacker $\mathcal{L}^*$ knows ℓ_i definitely. In the two-hop BC protocol, there is only one response register and thus $\mathcal{L}^*$ has no way to assert ℓ_i, which means $p_{\ell_i} = 0$. Thus, $\Pr(\mathcal{L}^*, \mathcal{P}) = (\frac{1}{2})^n$. In the rest of the selected protocols, there are more than one register and thus $\mathcal{L}^*$ has the advantage in winning the two-hop $\mathcal{L}^*\mathcal{P}$ over in the one-hop MF. For instance, in the two-hop HK protocol there are two registers, while the probability of having both registers with the same value (either 0 or 1) is equal to $\frac{1}{2}$. Thus, we have $p_{\ell_i} = \frac{1}{2}$ and $\Pr(\mathcal{L}^*, \mathcal{P}) = (\frac{1}{2} + (1 - \frac{1}{2}) * (\frac{1}{2} + (1 - \frac{1}{2}) * \frac{1}{2}))^n = (\frac{7}{8})^n$. The analysis of the other protocols is similar and the results are shown in Table D.3.

Two-hop $\mathcal{L}^*\mathcal{P}^*$

Similar with one-hop terrorist fraud, protocols like two-hop BC and two-hop HK are not secure against two-hop $\mathcal{L}^*\mathcal{P}^*$, since the malicious prover can always help the linker to pass the current protocol run, without leaking any secret information. We only focus on the other three protocols.

In the two-hop Reid *et al.* protocol, the malicious linker $\mathcal{L}^*$ can predict half of ℓ_i's $(1 \leq i \leq n)$ correctly and can ask for the corresponding responses r_i's from $\mathcal{P}^*$ without leaking any secret information (this is because half of the responses are either from the first register or from the second register, and thus one of the registers will not leak secret information). As to the rest half of the responses, $\mathcal{P}^*$ can give both of the two register values to $\mathcal{L}^*$. With this method, $\mathcal{L}^*$ has all the correct responses r_i and thus she can pass the current protocol with probability of 1, but she does not have all the values of the two registers, which means she cannot recover the secret key. This means in the future protocols, without $\mathcal{P}^*$'s help, $\mathcal{L}^*$ has no advantage to win. Thus, the two-hop Reid *et al.* cannot prevent the two-hop $\mathcal{L}^*\mathcal{P}^*$ attack.

In both of the two-hop Swiss-Knife and the two-hop $\mathsf{SKI}_{\mathsf{extend}}$ protocols, the malicious linker $\mathcal{L}^*$ can predict some ℓ_i and thus can query for the corresponding correct response r_i which can be given by $\mathcal{P}$ without leaking any secret information. Below we provide the detailed analysis for the two-hop Swiss-Knife and $\mathsf{SKI}_{\mathsf{extend}}$.

For the two-hop Swiss-Knife protocol both $\mathcal{L}^*$ and $\mathcal{P}^*$ have two registers that store the candidate response strings, which are computed by the same response function f but with different secret keys. In particular, $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$ where a_0, b_0 are n-bits outputs of a PRF, $a_1 = a_0 \oplus x_{\mathcal{L}}$ and $b_1 = b_0 \oplus x_{\mathcal{P}}$. Table D.4 shows all the possible values of $(a_0)_i$ (resp. $(b_0)_i$) and $(a_1)_i$ (resp. $(b_1)_i$) in the *i-th* round, as well as the probability of those cases. Now we discuss how much

(a) All possible values of $(a_0)_i$ and $(a_1)_i$

$(a_0)_i$	0	0	1	1
$(a_1)_i$	0	1	0	1
Pr	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

(b) All possible values of $(b_0)_i$ and $(b_1)_i$

$(b_0)_i$	0	0	1	1
$(b_1)_i$	0	1	0	1
Pr	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Table D.4: All possible values of $(a_0)_i/(b_0)_i$ and $(a_1)_i/(b_1)_i$ for the *i-th* round where $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$.

information $\mathcal{P}^*$ can give to $\mathcal{L}^*$ in order to help her pass the current protocol run. On one hand, $\mathcal{P}^*$ can only give either $(b_0)_i$ or $(b_1)_i$ for each round, otherwise $\mathcal{L}^*$ can trivially recover $(x_{\mathcal{P}})_i$ by computing $(b_0)_i \oplus (b_1)_i$. On the other hand, $\mathcal{L}^*$ needs to know the actual response $r_i \in \{(b_0)_i, (b_1)_i\}$ to pass the *i-th* round. $\mathcal{L}^*$ can query for r_i by sending $\ell_i = (a_{c_i})_i$ to $\mathcal{P}^*$ who will return $r_i = (b_{\ell_i})_i$. Without knowing c_i, $\mathcal{L}^*$ has to query both $(a_0)_i$ and $(a_1)_i$, but this will reveal $\mathcal{L}^*$'s secret key $(x_{\mathcal{L}})_i$ to $\mathcal{P}^*$. Therefore, the best strategy for $\mathcal{P}^*$ is to send half of the two register values to $\mathcal{L}^*$. For example, $\mathcal{P}^*$ may give the first register b_0 to $\mathcal{L}^*$. Thus, this case falls in the same category as the one-hop TF and thus we get $\Pr(\mathcal{L}^*, \mathcal{P}^*) = (\frac{3}{4})^n$.

In the two-hop $\mathsf{SKI}_{\mathsf{extend}}$ protocol both $\mathcal{L}^*$ and $\mathcal{P}^*$ have three registers. The analysis is very similar with that in the two-hop Swiss-Knife, but with more registers. The adversaries have more flexible ways to query the responses. In particular, $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$ where $a_0/b_0, a_1/b_1$ are vectors consisting of n random numbers belonging to $\mathbb{F}_3$ generated by a PRF, $(a_2)_i = (a_0)_i + (a_1)_i + (x_{\mathcal{L}})_i \mod 3$ and $(b_2)_i = (b_0)_i + (b_1)_i + (x_{\mathcal{P}})_i \mod 3$. Table D.5 shows all possible values of ℓ_i and r_i. In the first case, (i.e., $(a_0)_i = (a_1)_i = (a_2)_i$), $\mathcal{L}^*$ can query any two values such as $(a_0)_i$ and $(a_1)_i$, and get the correct response r_i. In this way, $\mathcal{L}^*$ does not leak $(x_{\mathcal{L}})_i$, nor does $\mathcal{P}^*$ leak $(x_{\mathcal{P}})_i$. In case 2, (i.e., two registers have the same value in the *i-th* position), $\mathcal{L}^*$ can still query two values and can succeed. Suppose $(a_0)_i = (a_1)_i \neq (a_2)_i$, then $\mathcal{L}^*$ can query $(a_0)_i$ and $(a_2)_i$ to get all the possible responses that will be used to answer the real challenges sent by $\mathcal{V}$. In the last case, all the registers have different values in the *i-th* position, then $\mathcal{L}^*$ has to query for all the positions, but this will reveal $(x_{\mathcal{P}})_i$ to $\mathcal{L}^*$ if $\mathcal{P}^*$ gives all $(b_0)_i$, $(b_1)_i$ and $(b_2)_i$ to $\mathcal{L}^*$. In this case, $\mathcal{P}^*$ can give at most two register values to $\mathcal{L}^*$. Therefore, the success probability should be $\Pr(\mathcal{L}^*, \mathcal{P}^*) = (\frac{1}{9} * 1 + \frac{2}{3} * 1 + \frac{2}{9} * \frac{2}{3})^n = (\frac{25}{27})^n$. In fact, for an (m, m) secret sharing scheme, we can obtain the following equation in order to compute the two-hop TF probability, that is,

$$\Pr(\mathcal{L}^*, \mathcal{P}^*) = \left(1 - \frac{m!}{m^{m+1}}\right)^n \tag{D.36}$$

6.3 Experiments

To verify our theoretical security analysis, we simulated the two different attack scenarios in the two-hop versions of the five selected DB protocols in Matlab, namely the case of $\mathcal{L}^*\mathcal{P}$ (dishonest linker, honest prover) and the case of $\mathcal{L}^*\mathcal{P}^*$ (dishonest linker, dishonest prover). For each selected value n (i.e., the number of rounds in fast phase), we calculate the realistic success probability of the adversary by figuring out how many correct responses that the adversary has returned, and we repeat the simulation for 1000 times, as a consequence obtaining the final simulated success probability by averaging the 1000 values. The results are shown in Figure

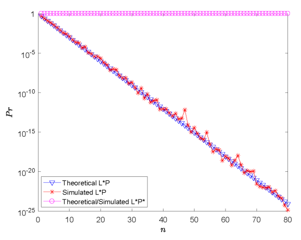
(a) All possible values of $(a_0)_i$, $(a_1)_i$ and $(a_2)_i$

$(a_0)_i$	y_0	y_0	y_0
$(a_1)_i$	y_0	y_0	y_1
$(a_2)_i$	y_0	y_1	y_2
Pr	$\frac{1}{9}$	$\frac{2}{3}$	$\frac{2}{9}$

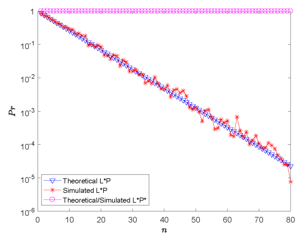
(b) All possible values of $(b_0)_i$, $(b_1)_i$ and $(b_2)_i$

$(b_0)_i$	y_0	y_0	y_0
$(b_1)_i$	y_0	y_0	y_1
$(b_2)_i$	y_0	y_1	y_2
Pr	$\frac{1}{9}$	$\frac{2}{3}$	$\frac{2}{9}$

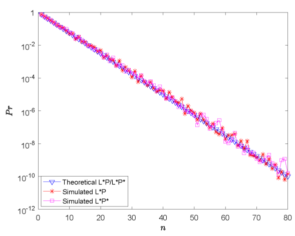
Table D.5: All possible values of $(a_0)_i/(b_0)_i$, $(a_1)_i/(b_1)_i$ and $(a_2)_i/(b_2)_i$ for the *i-th* round, $y_0, y_1, y_2 \in \{0, 1, 2\}$. $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$.



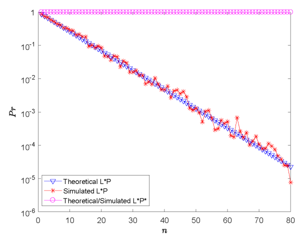
(a) Two-hop Brands and Chaum (BC) protocol



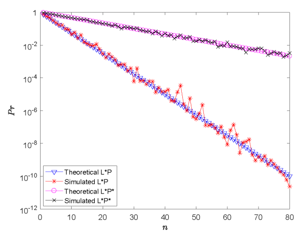
(b) Two-hop Hancke-Kuhn (HK) protocol



(c) Two-hop Swiss-Knife protocol



(d) Two-hop Reid *et al.* protocol



(e) Two-hop $\mathsf{SKI}_{\mathsf{extend}}$ protocol

Figure D.21: Theoretical and simulated success probabilities of the attackers in two attack scenarios for the selected two-hop DB protocols. In particular, $\mathcal{L}^*\mathcal{P}$ means the attack case of dishonest linker and honest prover, while $\mathcal{L}^*\mathcal{P}^*$ refers to the attack scenario of dishonest linker and dishonest prover. The x-axis shows the number of rounds in fast phase and the y-axis shows the adversary's success probability.

D.21. Each subfigure illustrates one protocol with regard to the two attacks and plots both the theoretical and the simulated success probabilities of the attacker. The x-axis shows the number of rounds in fast phase and the y-axis shows the adversary's success probability. According to Figure D.21, the evaluation results verify our theoretical analysis very well.

Moreover, we explore the relationship between a (m, m) secret-sharing scheme with the success probability against one-hop terrorist fraud and two-hop $\mathcal{L}^*\mathcal{P}^*$ attack, since secret-sharing scheme seems to be a good candidate to prevent these complicated attacks. Figure D.22 shows that when m increases, the success probabilities against both attacks also increase. In particular, when $m \geq 6$, the adversary has an overwhelming advantage to win the $\mathcal{L}^*\mathcal{P}^*$ attack. Intuitively, this is because, when m is larger, the linker can exchange more information with the dishonest prover without revealing any secret information to each other.

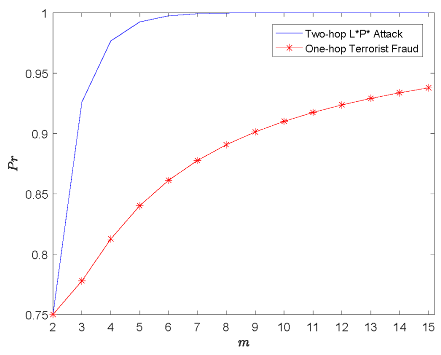


Figure D.22: Relationship between the success probability against one-hop terrorist fraud, two-hop $\mathcal{L}^*\mathcal{P}^*$ attack and m, $m \geq 2$.

7 Conclusions

In this paper, we investigated how to extend DB protocols to the two-hop case, i.e., when the prover and the verifier do not lie in each other's communication range and they need to rely on an in-between entity (linker) to perform authentication and distance-bounding. We defined two categories of DB protocols and provided a model that captures all the so-called register-based ones (which is the large majority of existing proposals). Using this model, we constructed a general method to derive the two-hop DB protocol from a one-hop register based one. A detailed security analysis for the two general constructions is then given, in particular we were the first to define attack scenarios for the two-hop case. Experiments were run on five different protocols to compare the values of the two main attacks against DB and their analogues in the two-hop case. We discussed the relation between the obtained results, and observed that (not surprisingly) the security of two-hop DB protocols is less or equal than that of the corresponding original DB protocols, which is consistent with our security analysis.

Bibliography

[1] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. "A Framework for Analyzing RFID Distance Bounding Protocols". In: *J. Comput. Secur.* 19.2 (2011), pp. 289–317.

[2] Gildas Avoine and Aslan Tchamkerten. "An efficient distance bounding RFID authentication protocol: balancing false-acceptance rate and memory requirement". In: *International Conference on Information Security.* Springer. 2009, pp. 250–261.

[3] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. "Secure localization algorithms for wireless sensor networks". In: *IEEE Communications Magazine* 46.4 (2008), pp. 96–101.

[4] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. "Practical and provably secure distance-bounding". In: *Journal of Computer Security* 23.2 (2015), pp. 229–257.

[5] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. "Towards secure distance bounding". In: *International Workshop on Fast Software Encryption.* Springer. 2013, pp. 55–67.

[6] Stefan Brands and David Chaum. "Distance-bounding protocols (extended abstract)". In: (1993), pp. 344–359.

[7] Laurent Bussard and Walid Bagga. "Distance-bounding proof of knowledge to avoid real-time attacks". In: *IFIP International Information Security Conference.* Springer. 2005, pp. 223–238.

[8] Srdjan Capkun, Karim El Defrawy, and Gene Tsudik. "Group distance bounding protocols". In: *International Conference on Trust and Trustworthy Computing.* Springer. 2011, pp. 302–312.

[9] Srdjan Capkun and J-P Hubaux. "Secure positioning in wireless networks". In: *IEEE Journal on Selected Areas in Communications* 24.2 (2006), pp. 221–232.

[10] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. "A formal approach to distance-bounding RFID protocols". In: *International Conference on Information Security.* Springer. 2011, pp. 47–62.

[11] Sébastien Gambs, Cristina Onete, and Jean-Marc Robert. "Prover anonymous and deniable distance-bounding authentication". In: *Proceedings of the 9th ACM symposium on Information, computer and communications security.* ACM. 2014, pp. 501–506.

[12] Gerhard P Hancke and Markus G Kuhn. “An RFID distance bounding protocol”. In: *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on.* IEEE. 2005, pp. 67–73.

[13] Jens Hermans, Roel Peeters, and Cristina Onete. “Efficient, secure, private distance bounding without key updates”. In: *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks.* ACM. 2013, pp. 207–218.

[14] Y-C Hu, Adrian Perrig, and David B Johnson. “Packet leashes: a defense against wormhole attacks in wireless networks”. In: *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies.* Vol. 3. IEEE. 2003, pp. 1976–1986.

[15] Handan Kılınç and Serge Vaudenay. “Efficient public-key distance bounding protocol”. In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer. 2016, pp. 873–901.

[16] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. “The swiss-knife RFID distance bounding protocol”. In: *International Conference on Information Security and Cryptology.* Springer. 2008, pp. 98–115.

[17] Zang Li, Wade Trappe, Yanyong Zhang, and Badri Nath. “Robust statistical methods for securing wireless localization in sensor networks”. In: *Proceedings of the 4th international symposium on Information processing in sensor networks.* IEEE Press. 2005, pp. 91–98.

[18] Sjouke Mauw, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. “A class of precomputation-based distance-bounding protocols”. In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on.* IEEE. 2016, pp. 97–111.

[19] Elena Pagnin, Gerhard P. Hancke, and Aikaterini Mitrokotsa. “Using Distance-Bounding Protocols to Securely Verify the Proximity of Two-hop Neighbours”. In: *IEEE Communications Letters* 19.7 (2015), pp. 1173–1176.

[20] Marcin Poturalski, Panos Papadimitratos, and Jean-Pierre Hubaux. “Secure neighbor discovery in wireless networks: formal investigation of possibility”. In: *Proceedings of the 2008 ACM symposium on Information, computer and communications security.* ACM. 2008, pp. 189–200.

[21] Kasper Bonne Rasmussen and Srdjan Capkun. “Realization of RF Distance Bounding.” In: *USENIX Security Symposium.* 2010, pp. 389–402.

[22] Jason Reid, Juan Manuel González Nieto, Tee Tang, and Bouchra Senadji. “Detecting Relay Attacks with Timing-Based Protocols”. In: *ASIACCS 07.* Ed. by Feng Bao and Steven Miller. Singapore: ACM Press, 2007, pp. 204–213.

[23] Rolando Trujillo-Rasua, Benjamin Martin, and Gildas Avoine. “Distance bounding facing both mafia and distance frauds”. In: *IEEE Transactions on Wireless Communications* 13.10 (2014), pp. 5690–5698.

[24] Anjia Yang, Yunhui Zhuang, and Duncan S Wong. “An efficient single-slow-phase mutually authenticated RFID distance bounding protocol with tag privacy”. In: *International Conference on Information and Communications Security.* Springer. 2012, pp. 285–292.

[25] Yunhui Zhuang, Anjia Yang, Duncan S Wong, Guomin Yang, and Qi Xie. "A highly efficient RFID distance bounding protocol without real-time PRF evaluation". In: *International Conference on Network and System Security*. Springer. 2013, pp. 451–464.

Paper E

On the Leakage of Information in Biometric Authentication

Elena Pagnin, Christos Dimitrakakis, Aysajan Abidin, and Aikaterini Mitrokotsa

Abstract. In biometric authentication protocols, a user is authenticated or granted access to a service if her fresh biometric trait *matches* the reference biometric template stored on the service provider. This matching process is usually based on a suitable *distance* which measures the similarities between the two biometric templates. In this paper, we prove that, when the matching process is performed using a specific family of distances (which includes distances such as the Hamming and the Euclidean distance), then information about the reference template is leaked. This leakage of information enables a *hill-climbing* attack that, given a sample that matches the template, could lead to the full recovery of the biometric template (*i.e.* centre search attack) even if it is stored encrypted. We formalise this "leakage of information" in a mathematical framework and we prove that centre search attacks are feasible for any biometric template defined in $\mathbb{Z}_q^n, (q \geq 2)$ after a number of authentication attempts linear in n. Furthermore, we investigate brute force attacks to find a biometric template that matches a reference template, and hence can be used to run a *centre search attack.* We do this in the binary case and identify connections with the *set-covering* problem and *sampling without replacement.*

Keywords. Biometric Authentication, Privacy-Preservation, Centre Search Attack, Hill-Climbing, Brute Force Attacks.

Proceedings of the 15th International Conference on Cryptology in India (INDOCRYPT), 2014.

On the Leakage of Information in Biometric Authentication

1 Introduction

While biometric authentication is becoming increasingly popular, the privacy and security risks related to their usage are raising severe concerns. The main threats associated to biometric authentication include profiling and tracking of individuals and identity theft. If successfully performed, any attack that recovers a biometric template may have serious impact since users cannot change their biometric features and biometric data may reveal very sensitive information (*e.g.* genetic [20] information and medical diseases [4]).

Biometric authentication protocols involve comparing fresh biometric data with a stored biometric template. The process is essentially performed by computing some distance or divergence between the fresh and the stored template. If the measured distance is less than a predefined threshold, then the user is authenticated; otherwise she is rejected. Many biometric authentication protocols use straightforward choices for the distance, such as the Hamming distance [8, 18], the normalised Hamming distance ([13] for iris recognition) and the Euclidean distance [2, 15, 22]. In these cases the matching process leaks information that could be exploited by an adversary to recover the stored template. More precisely, the adversary could run an iterative process where he progressively changes the components of an arbitrary biometric template until acceptance. This strategy is known as *hill-climbing* attack [23], due to similarity with the synonymous optimisation technique. When the initial template is an acceptable biometric trait (*e.g.* a fresh sample) this process is called *centre search* attack [23]. Recovering stored biometric templates has more severe impact than just finding an acceptable biometric template. Indeed, the same stored template might be used in multiple biometric authentication systems which may even employ different matching processes. Furthermore, a recovered stored template could be used to find a match in criminal biometric template databases or even compromise health records [16].

Hill climbing attacks involve making incremental changes to a potential solution, until one or more acceptable solutions are found. In our case, the adversary observes how the matcher responds to forged biometric templates. His goal is to recover the stored template from one matching template. Bringer *et al.* [10] presented a hill-climbing strategy that is successful even when a dedicated secure access module (*e.g.* smartcard) is used to perform the biometric authentication process. The matching process considered in [10] involves an adapted Hamming distance with erasures, nevertheless, the adversary is able to recover multiple encrypted biometric templates. Later on, Simoens *et al.* [23] describe multiple attacks (including the centre search attack) that can be mounted by each of the internal entities in a distributed biometric authentication systems.

In the past years, privacy-preserving distance computation has been investigated [7, 9, 17]. Although these protocols have direct applications to biometric identification and authentication they all suffer from leakage of information when a centre search attack is employed.

The problem of leakage of information due to the employment of distances has also been investigated in other areas not relevant to biometric authentication. For example, the Hamming weight model has been employed in order to successfully perform side channel attacks [3, 5] (*e.g.* differential power analysis). It has been shown [3, 5] that the power consumption of a device (*e.g.* a smart card) directly depends on the Hamming weight and on the number of changes $0 \leftrightarrow 1$ in the binary vector that is considered during the execution of the attack.

Our contribution: In this paper, we point out that all biometric authentication protocols that rely on certain distances (including the Hamming and the Euclidean distance) are susceptible to leakage of information and we provide a formal mathematical framework to analyse this. In particular, we generalise the centre search attack and prove that it is efficient and feasible in the binary case as well as when the biometric templates are defined in $\mathbb{Z}_q^n$. In both cases we show that the maximal number of authentication attempts in order to fully recover the stored biometrics corresponding to the given data is linear in n (the size of the biometric string). Our proofs hold also when the Euclidean distance is employed. Thus, we go beyond the Hamming distance case that was described in [23]. We furthermore investigate the preliminary step to the centre search attack: finding a biometric template that matches a reference one. For the binary case, we propose a new algorithm that exploits a tree structure and we compare its performance to standard brute force attacks and to the *optimal* but infeasible attack. Finally, we highlight how the *optimal* solution of finding a matching biometric template connects to the NP-complete *set-covering* problem and *sampling without replacement.* Our proofs are valid for standard as well as for privacy-preserving biometric authentication protocols since the output of the matching process is not affected by the employed protection mechanism (*e.g.* homomorphic encryption). This means that encryption alone cannot mitigate the leakage of information of the matching process. More precisely, this leakage of information leads to full recovery of the stored template for the centre search attack and to a matching template for the brute-force attack. An implication of our work is that achieving security and privacy of biometric templates using the known techniques is challenging.

Outline: The notations and the background material are introduced in Section 2 while Section 3 describes the adversarial model. We generalise the centre search attack in Section 4 in two ways: first to any leaking distance on $\mathbb{Z}_2^n$ and then to any leaking distance on $\mathbb{Z}_q^n$. In addition, we investigate the success probability of finding an acceptable fresh biometric template and compare the bounds for the success probability in different cases in Section 5. Finally, Section 6 summarizes our results.

2 Preliminaries

Notations: Let $q \in \mathbb{Z}$ be a positive integer, $q \geq 2$. The set of n-dimensional vectors with components in $\mathbb{Z}_q = \{0, 1, \cdots, q-1\}$ is denoted by $\mathbb{Z}_q^n$. The i-th component of a vector $x \in \mathbb{Z}_q^n$ is referred to as $x_i \in \mathbb{Z}_q$. Given a distance $d : \mathbb{Z}_q^n \times \mathbb{Z}_q^n \to \mathcal{R}_{\geq 0}$, a point $x \in \mathbb{Z}_q^n$ and a positive number $\tau \in \mathcal{R}_{>0}$, the d-ball of center x and radius τ is defined as $B_x(\tau) = \{z \in \mathbb{Z}_q^n : d(x, z) \leq \tau\}$. In the following,

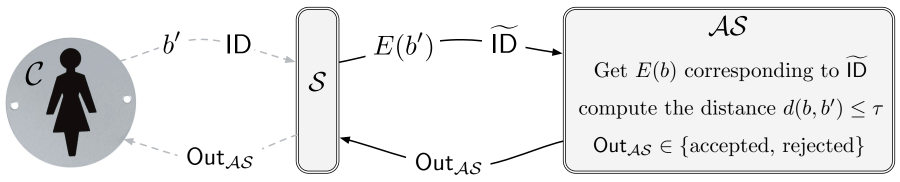

$\mathcal{C}$
b'
ID
$\mathcal{S}$
$E(b')$
$\widetilde{\mathsf{ID}}$
$\mathcal{AS}$
Get $E(b)$ corresponding to $\widetilde{\mathsf{ID}}$
compute the distance $d(b, b') \leq \tau$
$\mathsf{Out}_{\mathcal{AS}} \in$ {accepted, rejected}
$\mathsf{Out}_{\mathcal{AS}}$
$\mathsf{Out}_{\mathcal{AS}}$

Figure D.23: Authentication phase in a two-party biometric authentication system.

the binary case ($q = 2$) will always be explicitly written as $\mathbb{Z}_2^n$. If not otherwise specified, $\mathbb{Z}_q^n$ implies $q > 2$. We denote the bit-flip operation as $\bar{\ }: \mathbb{Z}_2 \to \mathbb{Z}_2$, namely $\bar{1} = 0$, $\bar{0} = 1$. The integer part of a real number τ, is denoted by $\lfloor \tau \rfloor$ (rounding to the closest integer $\leq \tau$).

2.1 Biometric authentication

A biometric authentication system consists of two main phases: the *enrolment phase* and the *authentication phase.*

The *enrolment phase* is a one-time step: a user (client) $\mathcal{C}$ registers to a trusted party her biometric templates (digital strings b) along with her identity ID. These two pieces of information are then stored in the database of the authentication server $\mathcal{AS}$. Once enrolled in the system, the client can authenticate herself an unlimited number of times.

In the *authentication phase*, the client is required to provide a fresh biometric trait b' as well as her identity ID. These two data are then communicated to the authentication server, which checks if matching templates (fresh b' and stored b) match. If the distance between the user's fresh biometric trait b' and the reference biometric template b is less or equal to a predefined threshold τ, then the client gets authenticated. Otherwise, the system rejects the user.

Without loss of generality we will consider only the two party setting (*i.e.* one client $\mathcal{C}$ and one authentication server $\mathcal{AS}$, as depicted in Figure D.23). However, our analysis naturally applies when more than two parties are involved in the biometric authentication process [1, 8, 24]. Due to privacy concerns, the biometric templates should be protected and not sent in the clear over the network. This implies that often the matching procedure is performed in the encrypted domain. For instance, in multiple privacy-preserving biometric authentication protocols, secure multi-party computation techniques are employed to preserve the privacy of the users. In those protocols usually the biometric data are protected using homomorphic encryption [19], garbled circuits [25] or oblivious transfer [21]. Figure D.23 depicts the authentication phase of a biometric authentication system in a two party setting, between a client $\mathcal{C}$ and an authentication server $\mathcal{AS}$. The client presents her fresh biometric and her ID to the authentication system. The sensor $\mathcal{S}$ gets the user's biometric vector b' and her identity. In the privacy-preserving case, $\mathcal{S}$ encrypts b' ($E(b')$) and ID ($\widetilde{\mathsf{ID}}$), otherwise this data is sent in the clear. Subsequently, the two data ($E(b')$, $\widetilde{\mathsf{ID}}$) are sent to the authentication server $\mathcal{AS}$, who retrieves the (possibly encrypted) stored template that corresponds to the user with identity ID. The matching process is then preformed by checking if the distance between the fresh and stored biometric templates is less than a predefined threshold τ (*i.e.* $d(b, b') \leq \tau$). Finally, depending on the outcome of the matching ($\mathsf{Out}_{\mathcal{AS}}$), the authentication server either accepts or rejects the client. Note that even in the privacy-preserving case, where the biometric data is encrypted, the output of the authentication server depends only on the value of $d(b, b')$, *i.e.* the distance between the fresh and the stored biometric vectors. Hence, encryption alone does not mitigate our attacks.

The main enablers of the attacks described in this paper are:

1. A return channel of the biometric authentication process, denoted as $\mathsf{Out}_{\mathcal{AS}}$ (*e.g.* access granted or not) that is sent by the authentication server to the user after each authentication attempt. In a real-life biometric authentication scenario this could be a door that opens denoting "access granted" when biometric authentication is used for access control in a building.

2. The fact that the matching process (and so the value of $\mathsf{Out}_{\mathcal{AS}}$) is based on a distance that is sensitive to single component variations (see leaking distance Definition 6.1).

In this paper, we demonstrate that even when secure-multi party computation techniques are employed, it is still possible to disclose the biometric templates as long as a certain family of distances is used to compare the raw (plaintext) biometric data. That is, an attacker can learn information about the value of b (plaintext of stored biometric template) by observing the authentication server's response $\mathsf{Out}_{\mathcal{AS}}$ to the client's authentication requests, if the response depends on the value of $d(b, b')$. More precisely, if d is a distance that detects component-variation (see Definition 6.1), and if there exists a function f that enables to retrieve information about the distance of the raw templates, given their possibly encrypted versions, *i.e.* $\exists f \; s.t. f(E(b), E(b')) = d(b, b')$, then the biometric authentication system leaks information (in the non privacy-preserving case $E = \mathrm{id}$, is the identity map and $f = d$ is the given distance). In particular, it is always possible to disclose the original b given a matching b'. For instance, consider the case [8] where $b, b' \in \mathbb{Z}_2^n$, $d = d_H$ is the Hamming distance and E and D are the Goldwasser-Micali [14] encryption and decryption functions, respectively. Then, $d_H(b, b') = \mathsf{HW}(b \oplus b)' = \mathsf{HW}\,(D(E(b \oplus b'))) = \mathsf{HW}\,(D(E(b) \times E(b')))$, where HW denotes the Hamming weight of a vector, *i.e.* $\mathsf{HW}(x) = \sum_{i=1}^{n} x_i$. In this case, we have $f = \mathsf{HW} \circ D \circ \times$.

3 Adversarial Model

The main threats in a privacy-preserving biometric authentication protocol are classified as follows [23]:

- *Biometric reference recovery:* the adversary tries to recover the reference (stored) biometric template b.

- *Biometric sample recovery:* the adversary tries to recover (or generate) a fresh biometric template b' that will be acceptable by the biometric authentication system.

- *Identity privacy:* the adversary tries to link a biometric template $b(i)$ of a user i to the user's identity $\mathsf{ID}(i)$.

- *Traceability and distinguishability of users:* the adversary's objective is to distinguish different users and/or trace one user in different authentication attempts.

In this paper, we focus on the two first threats only, as they apply to any biometric authentication system, privacy-preserving or not. We also consider that the adversary $\mathcal{A}$ has access to the output of the authentication process ($\mathsf{Out}_{\mathcal{AS}}$) as well as to the predefined threshold τ used in matching process. The settings for the two attacks are:

- *Biometric reference recovery:* the adversary $\mathcal{A}$ has an acceptable fresh biometric template b' at his disposal and tries to recover the stored template b (*centre search* attack).
- *Biometric sample recovery:* the adversary $\mathcal{A}$ does not have access to an acceptable fresh biometric b' but tries to find an accepted template anyway (brute force attack).

4 Generalisations of the Centre Search Attack

Let $b' \in \mathbb{Z}_q^n$ denote a fresh biometric template and $b \in \mathbb{Z}_q^n$ the reference (stored) template, for $q \geq 2$. The standard *centre search* attack aims at finding the point b in the centre of the *acceptance ball* $B_b(\tau) = \{\, z \in \mathbb{Z}_2^n \;:\; d(b,z) \leq \tau \,\}$. Simoens *et al.* [23] gave an informal description of this attack in the case d is the Hamming distance. Here, we extend this attack to a larger family of distances over $\mathbb{Z}_2^n$ (Theorem 6.2). In order to do so, we prove in Theorem 6.1 that any *leaking distance* (cf. Definition 6.1) over $\mathbb{Z}_2^n$ is *equivalent* to the Hamming distance. In addition, Theorem 6.3 proves that a centre search attack is feasible also for $b \in \mathbb{Z}_q^n$ when $q > 2$ if a *leaking distance* (e.g. the Euclidean distance) is employed in the matching process.

The family of distances we consider in this paper is defined as follows:

Definition 6.1 (Leaking distances). *Let $q \geq 2$, a distance $d : \mathbb{Z}_q^n \times \mathbb{Z}_q^n \to \mathcal{R}_{\geq 0}$, is said to be a leaking distance (to detect component variations) if it can be written as $d(x,y) = h\left(\sum_{i=1}^{n} |x_i - y_i|^k\right)$, for all $x, y \in \mathbb{Z}_2^n$, $k \in \mathbb{Q}_{>0}$ and $h : \mathcal{R} \to \mathcal{R}_{\geq 0}$ a monotonically strictly increasing positive function.*

The Hamming distance is an example of a leaking distance over $\mathbb{Z}_2$ (take h to be the identity map and $k = 1$). For a general $q \geq 2$, the Euclidean distance detects component variation (h is the square-root function and $k = 2$). Note that *leaking* distances are *reasonable* distances to be used for biometric authentication, as they enable to compare vectors (biometric data) component wise.

In order to simulate the query/access to an oracle, we introduce the following decision function.

Definition 6.2. *Let $q \geq 2$, $\tau \in \mathcal{R}_{>0}$ and let $d : \mathbb{Z}_q^n \times \mathbb{Z}_q^n \to \mathcal{R}_{\geq 0}$ be a distance metric. Then, for each $x \in \mathbb{Z}_q^n$, we define a decision function $\delta_x : \mathbb{Z}_q^n \to \{0,1\}$ as*

$$\delta_x(z) = \begin{cases} 0 & \text{if } d(x,z) > \tau \\ 1 & \text{if } d(x,z) \leq \tau \end{cases} .$$

It is easy to see that the decision function δ_x corresponds to the output of the authentication process denoted as $\mathsf{Out}_{\mathcal{AS}}$ in Sections 2 and 3. Firstly, we consider biometric templates as binary vectors. This is for instance the case for iris recognition based biometric authentication [6, 13]. We begin by proving that any binary leaking distance can be written in terms of the Hamming distance.

Theorem 6.1. *Let $d : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathcal{R}_{\geq 0}$ be a leaking distance on $\mathbb{Z}_2^n$. Then every d-ball corresponds to a d_H-ball, with d_H being the Hamming distance.*

We provide the proof of Theorem 6.1 in the appendix. Observe that Theorem 6.1 provides a *boardwalk* among all binary leaking distances. In particular, it enables us to extend all the results concerning Hamming distance to any other leaking distance (on $\mathbb{Z}_2^n$). For example, the *correction* factor for the Euclidean distance on $\mathbb{Z}_2^n$ is $\tau = \tilde{\tau}^2$.

Theorem 6.2. *Let $d_H : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathcal{R}_{\geq 0}$ be the Hamming distance and $\tau \in \mathcal{R}_{>0}$. Then, it is possible to determine the bit-values of a string x having access only to a vector $y \in B_x(\tau)$ and in at most $n + 2\tau$ calls to the decision function δ_x (cf. Definition 6.2).*

The proof of Theorem 6.2 is provided in the appendix. In light of Theorem 1, we have the natural extension of Theorem 2 to the case of any leaking distance on $\mathbb{Z}_2^n$.

Corollary 1. For any leaking distance d on $\mathbb{Z}_2^n$, Theorem 6.2 holds, with $\tau = h^{-1}(\tilde{\tau})$ being the corresponding threshold when $\tilde{\tau}$ is the given radius of the ball for the distance d.

As a side result, we have:

Corollary 2. If x is the stored biometric template b, and y is a matching fresh measurement b' satisfying $d(b, b') \leq \tau$, then Theorem 6.2 provides an algorithm to retrieve b being given b' in a number of authentication attempts linear in bit-length of the biometric templates.

In the protocol for iris recognition by Daugman [13], the matching process relies on a normalised Hamming distance, which is defined as $\mathsf{NHD}(b, b', X, Y) = \sum_{i=1}^{n}(b_i \oplus b'_i)X_iY_i \,/\, \sum_{i=1}^{n} X_iY_i$, for $b, b', X, Y \in \mathbb{Z}_2^n$. In the previous formula the vector X is the mask for the stored biometric template b, while Y masks the fresh trait b'. It is immediate to see that the normalised Hamming distance does not comply with Definition 6.1, nevertheless it is still possible, given b' and Y, to mount a centre search attack and recover the bits of b that are not blinded by the mask X,*i.e.* b_i such that $X_i = 1$.

Theorem 6.2 holds only for leaking distances on $\mathbb{Z}_2^n$ as in the proof we exploit the fact that $|x_i - y_i|$ can only assume two values 0 and 1, when $x_i = y_i$ and $x_i \neq y_i$ respectively. However, Theorem 6.3 generalises the reasoning in Theorem 6.2 to the non-binary case when any leaking distance is used (such as the Euclidean distance, often used in non-binary biometric authentication protocols).

Theorem 6.3. *Let $d : \mathbb{Z}_q^n \times \mathbb{Z}_q^n \to \mathcal{R}_{\geq 0}$ be any leaking distance on $\mathbb{Z}_q^n$ (cf. Definition 6.1) and $\tau \in \mathcal{R}_{>0}$, be a threshold such that $\tau < h(\lfloor \frac{q}{2} \rfloor^k)$, then it is possible to determine the value of the vector $x \in \mathbb{Z}_q^n$ having access only to a vector $y \in B_x(\tau)$ in at most mn calls to the decision function δ_x (as in Definition 6.2), where $m = \min\{\lfloor 2\tau \rfloor, 2\log q\}$.*

The proof of Theorem 6.3 is provided in the appendix. Also in this case, if we consider the vectors as biometric templates it holds:

Corollary 3. Considering x as the stored biometric template b, and y as the fresh matching trait b', then the proof of Theorem 6.3 provides an algorithm to mount centre search attacks against biometric authentication systems with templates in $\mathbb{Z}_q^n$. And the maximal number of authentication attempts is linear in length (dimension as vectors) of the biometric templates.

It is important to highlight that the results of this section imply that all biometric authentication protocols that employ a leaking distance in the matching process are vulnerable to the *centre search* attack, and this attack can be performed in an efficient way.

5 Biometric Sample Recovery Attacks in the Binary Case

One of the most severe threats to biometric authentication systems is recovering a stored raw biometric template b (maybe linked to the identity of the user). The knowledge of b provides more information than the knowledge of a fresh trait b', as the same b could be used in multiple biometric authentication systems possibly employing different matching processes (while b' might be rejected). In Section 4 we already presented efficient ways to recover the centre b of a ball, given a point b' *close* to it, namely $b' \in B_b(\tau)$. The question we address now is: *Is there a way to find a matching template b' given access only to δ_b?* The next subsections present four different answers to this question. We discuss the connection between this problem and the *set-covering* problem in Section 5.2.

In the following, we consider only the case in which the biometric traits are binary vectors, *i.e.* $b \in \mathbb{Z}_2^n$, and the employed distance is a leaking distance (cf. Definition 6.1).

5.1 Blind Brute Force

In the *blind brute force attack*, the attacker randomly chooses a point $b' \xleftarrow{R} \mathbb{Z}_2^n$, and checks the output of the function $\delta_b(b')$. If $\delta_b(b') = 1$, it means that $p \in B_b(\tau)$, so the attacker can easily recover b using this point b' (cf. Theorem 2). Otherwise (*i.e.*, if $\delta_b(b') = 0$), the attacker picks another point at random from $\mathbb{Z}_2^n$ as before. We call this attack *blind brute force* because in each attempt the adversary tries a random point until a point in $B_b(\tau)$ is found.

Let us compute the success probability of this attack after $t \in \mathbb{Z}_{>0}$ attempts. Suppose first that we pick $b' \in \mathbb{Z}_2^n$ uniformly at random. Then the probability of having b' accepted is $\omega := |B_b(\tau)|/|\mathbb{Z}_2^n| = \sum_{k=0}^{\tau} \binom{n}{k}/2^n$. In each attempt, if the trial point is chosen uniformly at random and independently from the previous attempts, then with probability ω this new trial point will be accepted. Let us now introduce binary random variables $X_i = 0$ or 1, for $i = 1, 2, \cdots, t$, and let $\Pr(X_i = 1) = \omega$ and $\Pr(X_i = 0) = 1 - \omega$. Obviously, X_i, $i = 1, 2, \cdots, t$, are i.i.d. Bernoulli random variables $X_i \sim \mathsf{Bern}(\omega)$. We are interested in computing $\Pr\left(\sum_{i=1}^{t} X_i = 1\right)$, the total probability of succeeding once in t attempts. It is not hard to see that $\Pr\left(\sum_{i=1}^{t} X_i = 1\right) = t\omega(1-\omega)^{t-1}$, as the random variable $\sum_{i=1}^{t} X_i \sim \mathsf{Binom}(t, \omega)$ has a binomial distribution.

5.2 Sampling without replacement

Brute Force without Point Replacement

In order to perform a brute force attack *without point replacement* the attacker has to define a set of potential candidates $C \subseteq \mathbb{Z}_2^n$. For the first trial, $C = \mathbb{Z}_2^n$ and the attacker chooses a point $b' \xleftarrow{R} C$ at random. If $\delta_b(b') = 1$, the selected point is inside the acceptance ball, $b' \in B_b(\tau)$, and so the attack is successful. Otherwise, the attacker updates the set of potential candidates $C = C \setminus \{b'\}$, deleting the one point that is not in the acceptance ball. The attack proceeds by randomly picking a point from the updated set C.

Let the random variables X_i, $i = 1, 2, \cdots, t$, be as in the case of the blind brute force attack. Note, however, that now $\Pr(X_i = 1)$ is different in each at-

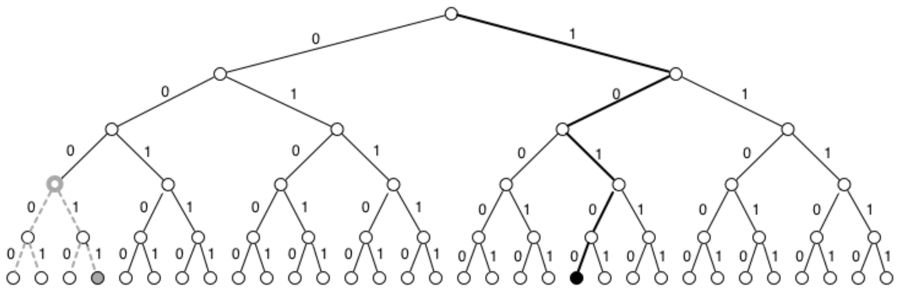


Figure D.24: The fundamental step of the Tree algorithm. Suppose the target biometric template is the vector $b = (10100) \in \mathbb{Z}_2^5$, the black bullet in the tree, and suppose the threshold is set to be $\tau = 2$. Let $a = (000)$ be the selected ancestor, highlighted as a grey circle in the picture. Let $b' = (00011)$ be the leaf randomly generated from a, then $d_H(b', b) > \tau$ and so $\delta_b\big((00011)\big) = 0$. In this case the points generated by a (*i.e.* that have a as common ancestor) will be deleted from the set of potential solutions.

tempt. In this case, $\sum_{i=1}^{t} X_i$ follows the Hypergeometric distribution. Therefore, $\Pr\left(\sum_{i=1}^{t} X_i = 1\right) = B\binom{2^n-B}{t-1}/\binom{2^n}{t}$, where $B = |B_x(\tau)| = \sum_{k=0}^{\tau}\binom{n}{k}$. This attack is intuitively *better* than the blind brute force, but of course the larger the n is, the less efficient it is.

The Tree Algorithm

We propose here a method (Algorithm 1) to find a point $b' \in \mathbb{Z}_2^n$ within distance τ from the unknown biometric template b, given access to the decision function δ_b (as in Definition 6.2). The central idea of Algorithm 1 is to consider the points of $\mathbb{Z}_2^n$ as leaves of a binary tree of depth n. The tree structure is then exploited to define relatives-relations among the points of $\mathbb{Z}_2^n$ and to ensure that at each unsuccessful trial one can delete non-overlapping portions of the space $\mathbb{Z}_2^n$. More precisely, if a point $p \in \mathbb{Z}_2^n$ is such that $\delta_b(p) = 0$, the algorithm removes from the set of potential centres not only the tried point p, but also its siblings-relatives generated by the τ common ancestor (see Figure D.24).

The main function called by the algorithm is generate. Its input is the threshold τ and a $(n-\tau)$-dimensional binary vector a. The output is a random leaf $b' \in \mathbb{Z}_2^n$ generated by a (the τ ancestor). That is, $\mathsf{generate}(a, \tau) = (a_1, \dots, a_{n-\tau}, r_1, \dots, r_\tau) = b'$, where $r_i \in \mathbb{Z}_2, i = 1, \dots, \tau$ are τ random bits. The set of potential ancestors C is updated at every unsuccessful round, by deleting the chosen ancestor. The tree algorithm uses the Hamming distance.

Algorithm 1 The Tree algorithm

```
Input: $(n, \tau, \delta_b,)$
Output: $b' = b'_1, \cdots, b'_n$ (a matching template)
$C = \mathbb{Z}_2^{n-\tau}$
for $i = 1$ to $2^{n-\tau}$: do
  $a \xleftarrow{R} \{C\}$
  $p = \mathsf{generate}(a, \tau)$
  if $\delta_b(b') = 1$ (accepted) then
    Return $b'$
  else
    $C = C \setminus \{a\}$
  end if
end for
```

For a practical implementation, we can store the paths of the tree that lead to the already rejected ancestors, and pick the new node a among the non-already-traversed paths. The running time of the attack is (of course) exponential, as it progressively constructs a binary tree of order $n-\tau$. Nevertheless, the probability to display the whole tree before finding a point that matches the reference template is very low (precisely: $2^{-n+\tau}$).

The *optimal* solution

The goal of the attacks described in this section is to find the ball $B_b(\tau) \subset \mathbb{Z}_2^n$ on which δ_b takes the value 1, without any additional information at hand. We have already investigated blind brute force (random tries), brute force without point replacement (remove one point at each unsuccessful trial), and the Tree algorithm (remove 2^τ points at each unsuccessful trial). The *optimal* brute force approach exploits the following idea: if a point $p \in \mathbb{Z}_2^n$ is rejected, *i.e.* $\delta_b(p) = 0$, it means that $b \notin B_p(\tau)$. Hence, the whole ball $B_p(\tau)$ can be removed by the set of potential centres. Intuitively, the *best* one can do to rapidly reduce the size of potential centres, is to use as trial points, points that lie at distance 2τ from each other. This corresponds to covering the space $\mathbb{Z}_2^n$ with the *smallest number* of balls of radius τ. This corresponds to an instance of the well-known *set-covering* problem in a space [11, 12].

More precisely, the optimal biometric sample recovery attack would involve the adversary covering $\mathbb{Z}_2^n$ with a family $\mathfrak{F}$ of balls of radius τ. At this point, the adversary needs to query the oracle (*i.e.* to use the decision function δ_b) at most $|\mathfrak{F}|$ times, one for each (centre of a) ball in $\mathfrak{F}$. Hence the *best* solution is for $\mathfrak{F}$ a *minimal* covering, *i.e.* $|\mathfrak{F}| = min_{\mathfrak{G} \in \mathscr{C}} |\mathfrak{G}|$, where $\mathscr{C}$ is the set of all possible covering of $\mathbb{Z}_2^n$ with balls of radius τ. This is exactly the set covering problem: to find the minimal number of balls needed to cover a space. It is proven that the set covering problem is NP-complete[12]. This result implies that also providing an *optimal* algorithm for the biometric sample recovery attack is an NP-complete problem. However, there exist some *greedy approximations* that are relatively efficient. In particular, for our case, Theorem 1 in [12] applies directly and hence the number of points that the adversary needs to query is only a factor of $O(\tau \ln(n+1))$ more than the optimal cover.

5.3 Comparisons and Bounds

In order to compare the performance of the four described methods we need to bound the probability that an attacker succeeds in finding a *matching* point, in each case. At the t-th trial, the attacker attempts point $x_t \in Z_2^n$ and observes $y_t \in 0, 1$, with $y_t \triangleq \mathbf{1}_{B_b(\tau)}(x_t) = \delta_b(x_t)$. Let $z_t \in \{0, 1\}$ denote whether or not the attacker has found an acceptable point after t trials and $s_t = \sum_{i=1}^t y_t$ be the number of points the attacker has found by time t.

To begin the analysis, we define $\mu_b(\tau) \triangleq |B_b(\tau)|/|\mathbb{Z}_2^n| \in [0, 1]$ to be the relative measure of the acceptance ball around b. In the binary case, dropping the dependence on b, τ, we have $\mu \in [2^{\tau-n}, (n+1)^\tau 2^{-n}]$. Of course, μ is also the probability of acceptance if sampling uniformly.

Blind brute force. In this case the points are selected uniformly without replacement, *i.e.* $x_t \sim \mathsf{U}(Z_2^n)$. It trivially follows that $\mathcal{E}(s_t) = \mu t$. It is also clear that the attack is successful whenever $s_t \geq 1$. For that reason, we shall attempt to bound the probability that this occurs while $\mu t < 1$. As a matter of fact, we can write:

$$\Pr(s_t \geq 1) = \Pr(\bigvee_{i=1}^{t} z_t = 1) \leq \sum_{i=1}^{t} \Pr(z_t = 1) = \mu t \leq (n+1)^{\tau} 2^{-n} t.$$

where the first inequality becomes an equality whenever $\mu t < 1$.

Sampling without replacement. All the other described approaches correspond to sampling without replacement. In either case, let $\alpha \in [0,1]$ denote the proportion of points removed at each step. Then, we obtain the following bound:

$$\begin{aligned}\Pr(s_t \geq 1) &\leq \sum_{i=1}^{t} \Pr(z_t = 1)\\ &\leq \sum_{i=1}^{t} \frac{\mu}{1-\alpha i}\\ &\leq \int_0^t \frac{\mu}{1-\alpha x}\,dx = \frac{\mu}{\alpha} \log \frac{1}{1-\alpha t}\ .\end{aligned}$$

For the point-wise replacement algorithm, $\alpha = q^{-n}$, hence there is little effect. For the binary case, we can employ the tree algorithm, $\alpha = 2^{\tau - n}$, which can be a substantial improvement. An unbounded adversary may use an optimal cover, in order to exclude as many points as possible whenever a point is rejected. In fact, in the best case, the adversary will be able to remove B points every time a point is rejected, giving a value of $\alpha = B2^{-n}$. To visualise the bounds, we choose some parameters such that there is a clear difference after a small number of iterations (depicted in Figure D.25). More precisely, Figure D.25 shows the performance of all four methods in terms of an upper bound on their success probability after a number of iterations. The four curves show sampling with replacement (*i.e.* brute force), and three different cases for sampling without replacement. Firstly, removing a single point. Secondly, removing 2^{τ} points using the tree construction. Finally, removing the maximum number of points B, which is computationally infeasible. There is a significant gain for the last choice, but only after a large portion of the space has already been covered. As when $\alpha \to 0$, $\ln \frac{1}{1-\alpha t} \to \alpha t$, the success probabilities of the first three methods are approximately linear in the size of the space, and hence exponential in the dimension.

The naive no replacement algorithm naturally does not improve significantly over brute force without replacement, since the volume that is excluded at every step is infinitesimal. Obviously, if we are able to remove a significant part of the volume, then we obtain a clear improvement in performance. Only an optimal adversary can do significantly better. However, this would assume either that *set-covering* is in P or that the adversary is computationally unbounded. Consequently, as there is no polynomial algorithm that is significantly better than brute force, biometric authentication schemes based on matching templates are secure against biometric sample recovery attacks.

6 Conclusions

In this paper, we prove that all biometric authentication protocols that employ distances between a template and an fresh biometric in the matching process suffer

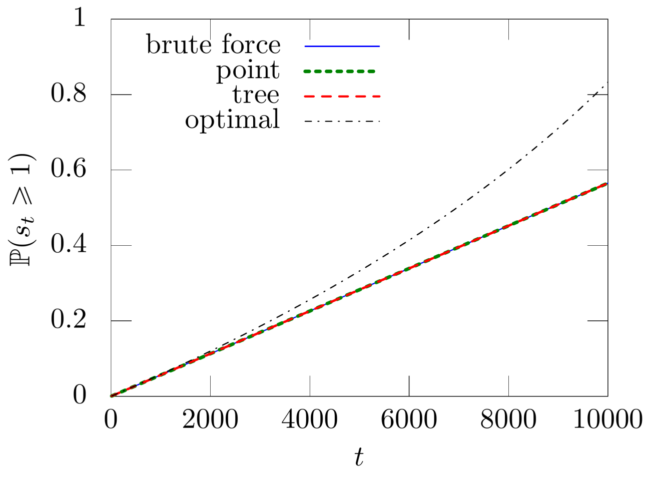


Figure D.25: Visualisation of the bounds for $q = 2$, $n = 32$, $\tau = 5$. In this case $\mu \approx 5.6 \times 10^{-5}$.

from leakage of information that could be exploited by an adversary to launch *centre search* attacks. In order to analyse this leakage of information, we provide a mathematical framework and prove that centre search attacks are feasible for any biometric template defined in $\mathbb{Z}_q^n$, $q \geq 2$, after a number of authentication attempts that is linear in n. Our results imply that it is possible to mount this attack on most existing biometric authentication protocols (including privacy-preserving ones) that rely on a Hamming, Euclidean, normalised Hamming distance or any distance that complies with Definition 6.1.

Furthermore, we investigate whether brute force attacks can be used to recover a matching biometric. We describe four strategies: blind brute force, brute force without replacement, a new algorithm based on a tree structure and the optimal case. Our results demonstrate that improving the success rate in these brute force attacks would imply finding a solution to the NP-complete *set-covering* problem. Thus, this provides some security guarantees of existing biometric authentication protocols as long as the attacker has not access to a matching biometric trait.

A possible countermeasure that could be employed in order to strengthen existing biometric authentication protocols against *centre search* attacks would be the employment of more sophisticated authentication methods. For example, simply using weighted distances in which the weights are secret and different for each user may provide sufficient security. Something similar is already employed in the normalised Hamming distance for which indeed the centre search attack is feasible but only for a subset of the components of the stored biometric template. An alternative and promising direction would be to rely on a mechanism that randomly selects a distance from a pool of distances at each authentication attempt. However, such measures should be incorporated carefully in order not to affect the accuracy of the biometric authentication system.

Bibliography

[1] Manuel Barbosa, Thierry Brouard, Stéphane Cauchie, and Sim$\mathcal{T}$ao Melo Sousa. “Secure Biometric Authentication with Improved Accuracy.” In: *ACISP 2008*. Ed. by Yi Mu, Willy Susilo, and Jennifer Seberry. Vol. 5107. LNCS. Springer, 2008, pp. 21–36.

[2] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Failla, Dario Fiore, Riccardo Lazzeretti, Vincenzo Piuri, Fabio Scotti, and Alessandro Piva. “Privacy-preserving fingercode authentication”. In: *Proceedings of the 12th ACM workshop on Multimedia and security*. 2010, pp. 231–240.

[3] E. Biham and A. Shamir. “Power analysis of the key scheduling of the AES candidates”. In: *Proceedings of the 2nd AES Candidate Conference*. 1999.

[4] J. Bolling. “A window to your health”. In: *Jacksonville Medicine, Special Issue: Retinal Diseases* 51 (2000).

[5] Eric Brier, Christophe Clavier, and Francis Olivier. “Correlation Power Analysis with a Leakage Model”. In: *CHES 2004*. Vol. 3156. LNCS. Springer Berlin Heidelberg, 2004, pp. 16–29.

[6] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, and G. Zémor. “Optimal Iris Fuzzy Sketches”. In: *Proceedings of the 1st IEEE International Conference on Biometrics: Theory, Applications, and Systems*. 2007.

[7] Julien Bringer, Herve Chabanne, Melanie Favre, Alain Patey, Thomas Schneider, and Michael Zohner. “GSHADE: Faster Privacy-preserving Distance Computation and Biometric Identification”. In: *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2014, pp. 187–198.

[8] Julien Bringer, Hervé Chabanne, Malika Izabachène, David Pointcheval, Qiang Tang, and Sébastien Zimmer. “An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication”. In: *ACISP 2007*. LNCS. Springer-Verlag, 2007, pp. 96–106.

[9] Julien Bringer, Hervé Chabanne, and Alain Patey. “SHADE: Secure HAmming DistancE Computation from Oblivious Transfer”. In: *Financial Cryptography Workshops*. 2013, pp. 164–176.

[10] Julien Bringer, Hervé Chabanne, and Koen Simoens. “Blackbox Security of Biometrics”. In: *Proceedings of the 6th International Conference on Intelligent InformationHiding and Multimenida Signal Processing*. 2010, pp. 337–340.

[11] Long Chen. “New analysis of the sphere covering problems and optimal polytope approximation of convex bodies”. In: *Journal of Approximation Theory* 133.1 (2005), pp. 134–145.

[12] V. Chvatal. “A Greedy Heuristic for the Set-Covering Problem”. In: *Mathematics of Operations Research* 4.3 (1979), pp. 233–235.

[13] J. Daugman. “How iris recognition works”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 14 (1 2004), pp. 21–30.

[14] Shafi Goldwasser and Silvio Micali. “Probabilistic encryption and how to play mental poker keeping secret all partial information”. In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing.* STOC 1982. ACM, 1982, pp. 365–377.

[15] Yan Huang, Lior Malka, David Evans, and Jonathan Katz. “Efficient Privacy-Preserving Biometric Identification”. In: *NDSS 2011.* 2011.

[16] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar. “Biometric template security”. In: *EURASIP J. Adv. Signal Process* 2008 (2008), 113:1–113:17.

[17] Ayman Jarrous and Benny Pinkas. “Secure Hamming Distance Based Computation and Its Applications”. In: *ACNS 2009.* Vol. 5536. LNCS. 2009, pp. 107–124.

[18] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. “SCiFI - A System for Secure Face Identification”. In: *Security and Privacy, 2010 IEEE Symposium on.* 2010, pp. 239–254.

[19] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *EUROCRYPT 1999.* Vol. 1592. LNCS. Springer, 1999, pp. 223–238.

[20] L.S. Penrose. “Dermatoglyphic Topology”. In: *Nature* 205 (1965), pp. 544–546.

[21] Michael O. Rabin. “How To Exchange Secrets with Oblivious Transfer”. In: *IACR Cryptology ePrint Archive* 2005 (2005), p. 187.

[22] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. “Efficient Privacy-Preserving Face Recognition”. In: *ICISC 2009.* LNCS. 2009, pp. 229–244.

[23] Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys. “A Framework for Analyzing Template Security and Privacy in Biometric Authentication Systems”. In: *IEEE Transactions on Information Forensics and Security* 7.2 (2012), pp. 833–841.

[24] A. Stoianov. “Security Issues of Biometric Encryption”. In: *Proceedings of the 2009 IEEE Toronto International Conference on Science and Technology for Humanity (TIC- STH).* 2009, pp. 34–39.

[25] Andrew Chi-Chih Yao. “How to generate and exchange secrets”. In: *Foundations of Computer Science, 1986., 27th Annual Symposium on.* IEEE. 1986, pp. 162–167.

Appendix

D.i Collected proofs

Theorem 6.1. By hypothesis d is a leaking distance, hence it is of the form $d(x,y) = h(\sum_{i=1}^{n} |x_i - y_i|^k)$, for all $x, y \in \mathbb{Z}_2^n$. Since $h : \mathcal{R} \to \mathcal{R}_{\geq 0}$ is monotonic, it is bijective on its image, in other words it has an inverse $h^{-1} : I \to \mathcal{R}$, where $I = Im(h) = \{w \in \mathcal{R}_{\geq 0} \ : \ w = h(z), \exists\, z \in \mathcal{R}\}$.
Consider the d ball of radius $\tilde{\tau}$ around a point $x \in \mathbb{Z}_2^n$, namely the set $\{y \in \mathbb{Z}_2^n \ : \ d(x,y) \leq \tilde{\tau}\}$. We want to prove this d-ball equals a *Hamming distance*-ball centred in x and of radius τ.
Indeed, $d(x,y) \leq \tilde{\tau} \iff h(\sum_{i=1}^{n} |x_i - y_i|^k) \leq \tilde{\tau}$. Noticing that h is increasing implies that h^{-1} is also increasing, one obtains: $\sum_{i=1}^{n} |x_i - y_i|^k \leq h^{-1}(\tilde{\tau})$. In addition, since $|x_i - y_i| \in \{0, 1\}$ we can *ignore* the exponent k in the expression (this is because $0^k = 0$ and $1^k = 1, \forall\, k \in \mathbb{Q}_{>0}$). Hence, $\sum_{i=1}^{n} |x_i - y_i| \leq h^{-1}(\tilde{\tau})$, but the left hand side of the inequality is exactly the Hamming distance between the points x and y. To summarise, we have $d(x,y) \leq \tilde{\tau} \iff d_H(x,y) \leq h^{-1}(\tilde{\tau})$. Let us put $\tau = h^{-1}(\tilde{\tau})$, then $\{y \in \mathbb{Z}_2^n \ : \ d(x,y) \leq \tilde{\tau}\} = \{y \in \mathbb{Z}_2^n \ : \ d_H(x,y) \leq \tau\}$. That is, any d-ball can be described as a d_H-ball (*Hamming distance*-ball) and vice versa. □ □

Theorem 6.2. STEP 1. Find a point w that lies just outside the boundary of $B_x(\tau)$. By hypothesis $\delta_x(y) = 1$. Let w be the vector obtained from y by flipping the first bit, *i.e.* $w_1 = \bar{y}_1$ and $w_i = y_i$, $\forall i \in \{2, \ldots, n\}$. If w is rejected, that is, if $\delta_x(w) = 0$, it means that y is already on the boundary of $B_x(\tau)$ and we are done by putting $v = y$. Otherwise, proceed by flipping one more bit of y until it exits $B_x(\tau)$. The general step after $k-1$ trials (flipping bits of y and being accepted) is: set $w = (\bar{y}_1, \ldots, \bar{y}_k, y_{k+1}, \ldots, y_n)$, if $\delta_x(w) = 0$ put $v = (\bar{y}_1, \ldots, \bar{y}_{k-1}, y_k, \ldots, y_n)$. If $\delta_x(w) = 1$, go on and flip the next component. It is quite intuitive that this procedure ends after at most $2\tau + 1$ steps (the worst case is when y is already on the boundary but we move it in the *wrong* direction and cross the ball along its diameter).

STEP 2. Determine the central point x of $B_x(\tau)$.
Note that by STEP 1, we already know the value of the k-th component of x, namely $x_k = v_k$. For $j \in \{1, 2, \ldots, n\} \setminus \{k\}$, consider the vector $v(j)$ defined as $v(j)_i = w_i$, $\forall\, i \in \{1, \ldots, n\} \setminus \{j\}$. If $\delta_x(v(j)) = 1$, it means that $v(j)$ *compensates* the error (in the k-th component) introduced by w with a *new* correct component (the j-th component). Hence $x_j = v(j)_j$. On the other hand, $\delta_x(v(j)) = 0$ implies that the j-th component of w was correct. Hence, in this case, $x_j = 1 - v(j)_j$. STEP 2 ends after $n-1$ queries. □ □

Theorem 6.3. Let $e(i) \in \mathbb{Z}_q^n$ denote the i-th vector of the canonical basis, *i.e.* for each $i = 1, \ldots, n$, $e(i)_i = 1$ and $e(i)_j = 0$, $\forall j \in \{1, \ldots, n\} \setminus \{i\}$. For each of the n components of a biometric template, determine two vectors $v(i), w(i) \in \mathbb{Z}_q^n, i = 1, \ldots, n$ such that: $v(i) = b' + \lambda_1 e(i)$ and $w(i) = b' + \lambda_2 e(i)$, with $\lambda_1 \in \{y_i, q-1-y_i\}$ and $\lambda_2 \in \{0, y_i - 1\}$. Moreover, $\delta_x\big(v(i)\big) = 1$ but $\delta_x\big(v(i) + e(i)\big) = 0$, and $\delta_x\big(w(i)\big) = 1$ but $\delta_x\big(w(i) - e(i)\big) = 0$. Such pair of vectors exists for each component, as $B_x(\tau)$ is a bounded subset of $\mathbb{Z}_q^n$ and $\tau < h(\lfloor \frac{q}{2} \rfloor^k)$. There are two possible situations:

- $v(i)$ and $w(i)$ are on the boundary of the ball $B_x(\tau)$. In this case the centre of the ball $x \in \mathbb{Z}_q^n$ will have the i-th component equal to the *middle point*

$x_i = (v(i)_i + w(i)_i)/2$, $\forall i \in \{1, \ldots, n\}$.

- $v(i)$ and $w(i)$ are not exactly *on* the boundary of the ball $B_b(\tau)$. Since it is $v(i), w(i), x \in \mathbb{Z}_q^n$ the respective distances from the boundary $\epsilon_{v(i)}$ and $\epsilon_{w(i)}$ must be equal (by symmetry). Thus, also in this case $b_i = (v(i)_i + w(i)_i)/2$, $\forall i \in \{1, \ldots, n\}$.

There are two efficient strategies to determine the vectors $v(i)$, $w(i)$:

- *Linear search:* in this case the worst case scenario is when $y = x$, and the adversary needs to try all the points (with components in $\mathbb{Z}_q$) that lie in the diameter of the ball $B_x(\tau)$, that is at most $\lfloor 2\tau \rfloor$ trials.

- *Binary search:* the adversary performs at most $2\log q$ trials to determine each *external* point, $v(i), w(i)$.

Thus, the maximum number of queries (access to the δ_x function) necessary in order to recover the centre x of a ball in $\mathbb{Z}_q^n$ is bounded by nm, with $m = \min\{\lfloor 2\tau \rfloor, 2\log q\}$.
$\square$

Paper F

Revisiting Yasuda et al.'s Biometric Authentication Protocol: Are you Private Enough?

Elena Pagnin, Jing Liu, and Aikaterini Mitrokotsa

Abstract. Biometric Authentication Protocols (BAPs) have increasingly been employed to guarantee reliable access control to places and services. However, it is well-known that biometric traits contain sensitive information of individuals and if compromised could lead to serious security and privacy breaches. Yasuda *et al.* [3] proposed a distributed privacy-preserving BAP which Abidin *et al.* [1] have shown to be vulnerable to biometric template recovery attacks under the presence of a malicious computational server. In this paper, we fix the weaknesses of Yasuda *et al.*'s BAP and present a detailed instantiation of a distributed privacy-preserving BAP which is resilient against the attack presented in [1]. Our solution employs Backes *et al.*'s [2] verifiable computation scheme to limit the possible misbehaviours of a malicious computational server.

Keywords. Biometric Authentication, Verifiable Delegation, Privacy-Preserving Authentication.

Proceedings of 16th International Conference on Cryptology and Network Security (CANS), 2017.

Revisiting Yasuda et al.'s Biometric Authentication Protocol: Are you Private Enough?

Biometric Authentication Protocols (BAPs) have increasingly been employed to guarantee reliable access control to places and services. However, it is well-known that biometric traits contain sensitive information of individuals and if compromised could lead to serious security and privacy breaches. Yasuda *et al.* [23] proposed a distributed privacy-preserving BAP which Abidin *et al.* [1] have shown to be vulnerable to biometric template recovery attacks under the presence of a malicious computational server. In this paper, we fix the weaknesses of Yasuda *et al.*'s BAP and present a detailed instantiation of a distributed privacy-preserving BAP which is resilient against the attack presented in [1]. Our solution employs Backes *et al.*'s [4] verifiable computation scheme to limit the possible misbehaviours of a malicious computational server.

1 Introduction

Biometric authentication has become increasingly popular as a fast and convenient method of authentication that does not require to remember and manage long and cumbersome passwords. However, the main advantage of biometrics, *i.e.,* their direct and inherent link with the identity of individuals, also rises serious security and privacy concerns. Since biometric characteristics can not be changed or revoked, unauthorised leakage of this information leads to irreparable security and privacy breaches such as identity fraud and individual profiling or tracking [18]. Thus, there is an urgent need for efficient and reliable privacy-preserving biometric authentication protocols (BAPs).

The design of privacy-preserving BAPs is by itself a very delicate procedure. It becomes even more challenging when one considers the distributed setting in which a resource-constrained client outsources the computationally heavy authentication process to more powerful external entities. In this paper, we focus on Yasuda *et al.*'s protocol for privacy-preserving BAPs in the distributed setting [23] and show how to mitigate the privacy attacks presented by Abidin *et al.* [1] by employing Backes *et al.*'s verifiable computation scheme [4].

1.1 Background & related work

Distributed privacy-preserving BAPs usually involve the following entities: (*i*) a client/user $\mathcal{C}$, (*ii*) a database $\mathcal{DB}$, (*iii*) a computational server $\mathcal{CS}$, and (*iv*) an authentication server $\mathcal{AS}$. The granularity of roles and entities in the biometric authentication process facilitates the privacy-preservation of the sensitive information. This distributed setting, indeed guarantees that no single entity has access to both the biometric templates (fresh and stored ones) and the identity of the querying user.

Several existing proposals of privacy-preserving BAPs use the distributed setting, *e.g.,* [5, 20, 22, 23], and make leverage on advanced cryptographic techniques such

as homomorphic encryption [7, 23], oblivious transfer [8] and garbled circuits [14]. In particular, Yasuda *et al.*'s protocol [23] was claimed to be privacy-preserving since it is based on the distributed setting and relies on a novel somewhat homomorphic encryption scheme based on ideal lattices. Abildin *et al.* [1] showed that Yasuda *et al.*'s BAP is privacy-preserving only in the honest-but-curious model and described an algorithm that enables a malicious $\mathcal{CS}$ to recover a user's biometric template. Intuitively, Abidin *et al.*'s attack succeeds because $\mathcal{AS}$ does not detect that the malicious $\mathcal{CS}$ returns a value different from the one corresponding to the output of the (honest) outsourced computation, leaving space for *hill-climbing strategies* [21] that may lead to the disclosure of the stored reference biometric template.
Verifiable delegation of computation (VC) is a cryptographic primitive that enables a client to securely and efficiently offload computations to an untrusted server [11]. Verification of arbitrary complex computations was initially achieved via interactive proofs [2, 13] and then moved towards more flexible and efficient schemes such as [3, 9, 10, 19]. The setting of VC schemes is by nature distributed and thus perfectly fits the basic requirement of privacy-preserving BAPs. For this reason, Bringer *et al.* [6] suggested to use VC techniques to detect malicious behaviours in BAP.

In this paper, we provide the first explicit instantiation of a distributed privacy-preserving BAP which achieves security against malicious $\mathcal{CS}$ thanks to the verifiability of the delegated computation.

1.2 Our contributions

In this paper, we mitigate Abidin *et al.*'s attack [1] against Yasuda *et al.*'s privacy-preserving biometric authentication protocol [23] by the means of the verifiable computation scheme by Backes *et al.* [4]. We combine the two schemes in an efficient and secure way, and obtain a modification of Yasuda *et al.*'s protocol with strong privacy guarantees. As a result, we obtain a new BAP which builds on top of Yasuda *et al.*'s and is truly privacy-preserving in the distributed setting.

From a general point of view, this paper offers a strategy to transform privacy-preserving BAPs that are secure in the honest-but-curious model into schemes that can tolerate a malicious $\mathcal{CS}$ by addressing the most significant challenges in privacy-preserving BAPs: to guarantee integrity and privacy of both the data and the computation. Despite the idea of combining VC and BAP is quite natural and intuitive [6], the actual combination needs to be done carefully in order to avoid flawed approaches.

Organisation. The paper is organized as follows. Section 2 describes the background notions used in the rest of the paper. Section 3 contains our modification of Backes *et al.*'s VC scheme to combine it with the somewhat homomorphic encryption scheme used in [23]. Section 4 presents an improved version of Yasuda *et al.*'s BAP together with a security and efficiency analysis. The proposed privacy-preserving BAP incorporates the new construction of VC on encrypted data of Section 3. Section 5 is an important side-note to our contributions, as it demonstrates how naïve and straight-forward compositions of VC and homomorphic encryption may lead to leakage of private information. Section 6 concludes the paper.

2 Preliminaries

Notations. We denote by $\mathbb{Z}$ and $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ the ring of integers and the integers modulo p, respectively. For two integers $x, d \in \mathbb{Z}$, $[x]_d$ denotes the reduction of x modulo d in the range of $[-d/2, d/2]$. We write vectors with capital letters, *e.g.,* A, and refer to the i-th component of A as A_i. The symbol $x \xleftarrow{\$} \mathcal{X}$ denotes selecting x

KeyGen$(\lambda) \rightarrow (ek, vk)$**:** Given the security parameter λ, the key generation algorithm outputs a secret verification key vk and a public evaluation key ek.

Auth$(vk, L, m) \rightarrow \sigma$: Given the secret verification key vk, a multi-label $L = (\Delta, \tau)$ and a valid message m, this algorithm outputs an authentication tag σ.

Ver$(vk, \mathcal{P}_\Delta, m, \sigma) \rightarrow \{0,1\}$: Given the secret key vk, a multi-label program $\mathcal{P} = ((f, \tau_1, \ldots, \tau_n), \Delta)$, a valid message m and a tag σ, the verification algorithm returns an acceptance bit acc: "0" for rejection and "1" for acceptance.

Eval$(ek, f, \boldsymbol{\sigma}) \rightarrow \sigma$**:** Given the public evaluation key ek, a circuit of a quadratic polynomial f and a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the evaluation algorithm produces a new tag $\sigma \leftarrow$ **GroupEval**$(f, \sigma_1, \ldots, \sigma_n)$. The evaluation of **GroupEval** proceeds gate-by-gate through the arithmetic circuit of f with the following rules:

Fan-in-2 addition gate:

i. $X_1, X_2 \in \mathbb{G}$, output $X = X_1 \cdot X_2 = g^{x_1} \cdot g^{x_2} = g^{x_1+x_2} \in \mathbb{G}$.

ii. $\hat{X}_1, \hat{X}_2 \in \mathbb{G}_T$, output $\hat{X} = \hat{X}_1 \cdot \hat{X}_2 = e(g,g)^{x_1} \cdot e(g,g)^{x_2} = e(g,g)^{x_1+x_2} \in \mathbb{G}_T$.

iii. $\hat{X}_1 \in \mathbb{G}_T$, $X_2 \in \mathbb{G}$, output $\hat{X} = \hat{X}_1 \cdot e(X_2, g) = e(g,g)^{x_1+x_2} \in \mathbb{G}_T$.

iv. $X_1 \in \mathbb{G}$, $\hat{X}_2 \in \mathbb{G}_T$, output $\hat{X} = e(X_1, g) \cdot \hat{X}_2 = e(g,g)^{x_1+x_2} \in \mathbb{G}_T$.

Fan-in-2 multiplication gate:

i. $X_1, X_2 \in \mathbb{G}$, output $\hat{X} = e(X_1, X_2) = e(g,g)^{x_1 x_2} \in \mathbb{G}_T$.

ii. $X_1 \in \mathbb{G} \cup \mathbb{G}_T$, $c \in \mathbb{Z}_p$ constant, output $X = (X_1)^c = e(g,g)^{x_1 c} \in \mathbb{G}_T$.

The output of **GroupEval** is the output of the last gate of the arithmetic circuit.

Figure H.26: The BFR verifiable delegation of computation scheme.

uniformly at random from the set $\mathcal{X}$. We denote the Hadamard product for binary vectors as $\diamond : \mathbb{Z}_2^n \diamond \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$, with $A \diamond B = C$, $C_i = A_i \cdot B_i \in \mathbb{Z}_2$ for $i = 1, 2, \ldots, n$. The Hadamard product is similar to the inner product of vectors except that the output is a vector rather than an integer.

Bilinear maps. A symmetric bilinear group is a tuple $(p, \mathbb{G}, \mathbb{G}_T, g, g_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p. The elements $g \in \mathbb{G}$ and $g_T \in \mathbb{G}_T$ are generators of the group they belong to, and $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is a bilinear map, *i.e.,* $\forall A, B \in \mathbb{G}$ and $x, y \in \mathbb{Z}p$ it holds that $e(xA, yB) = e(A,B)^{xy}$ and $e(g,g) \neq 1_{\mathbb{G}_T}$. In the setting of VC, the map e is cryptographically secure, *i.e.,* it should be defined over groups where the discrete logarithm problem is assumed to be hard or it should be hard to find inverses. In bilinear groups there exists a natural isomorphism between $\mathbb{G}$ and $(\mathbb{Z}_p, +)$ given by $\phi_g(x) = g^x$; similarly for $\mathbb{G}_T$. Since ϕ_g and ϕ_{g_T} are isomorphisms, there exist inverses $\phi_g^{-1} : \mathbb{G} \rightarrow \mathbb{Z}_p$ and $\phi_{g_T}^{-1} : \mathbb{G}_T \rightarrow \mathbb{Z}_p$, that can be used to homomorphically evaluate any arithmetic circuit $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$, from $\mathbb{G}$ to $\mathbb{G}_T$. More precisely, there exists a map **GroupEval** (as defined in [4]):

$$\mathbf{GroupEval}(f, X_1, \ldots, X_n) = \phi_{g_T}(f(\phi_g^{-1}(X_1), \ldots, \phi_g^{-1}(X_n))).$$

For security, we assume ϕ_g and ϕ_{g_T} are not efficiently computable.

Homomorphic MAC authenticators. In this paper, we make use of Backes, Fiore and Reischuk's verifiable computation scheme based on homomorphic MAC authenticators [4], which we refer to as BFR. The BFR scheme targets functions f that are quadratic polynomials over a large number of variables. Figure H.26 contains a succinct description of the BFR scheme. For further details we refer the reader to the main paper [4].

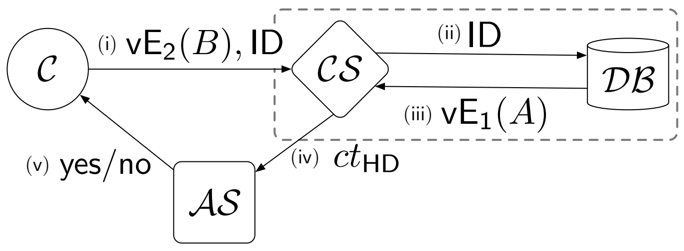


Figure H.27: Authentication phase in the Yasuda *et al.*'s BAP [23].

Homomorphic encryption. Let $\mathcal{M}$ denote the space of plaintexts that support an operation $\boxdot$, and $\mathcal{C}$ be the space of ciphertexts with $\odot$ as operation. An encryption scheme is said to be homomorphic if for any key, the encryption function **Enc** satisfies: $\mathbf{Enc}(m_1 \boxdot m_2) \leftarrow \mathbf{Enc}(m_1) \odot \mathbf{Enc}(m_2)$, for all $m_1, m_2 \in \mathcal{M}$, where $\leftarrow$ means computed without decryption. In this paper, we only use Somewhat Homomorphic Encryption schemes (SHE). As the name suggests these schemes only support a limited number of homomorphic operations, *e.g.,* indefinite number of homomorphic additions and finite number of multiplications. The choice to use SHE instead of Fully Homomorphic Encryption [12] is due to efficiency: SHE, if used appropriately, can be much faster and more compact [15].

The Yasuda *et al.* protocol. Yasuda *et al.* [23] proposed a privacy-preserving biometric authentication protocol that targets one-to-one authentication and relies on somewhat homomorphic encryption based on ideal lattices. Two packing methods facilitate efficient calculations of the secure Hamming distance, which is a common metric used for comparing biometric templates. The protocol uses a distributed setting with three parties: a client $\mathcal{C}$, a computation server $\mathcal{CS}$ (which contains the database $\mathcal{DB}$) and an authentication server $\mathcal{AS}$. The protocol is divided into three phrases.

Setup Phase: $\mathcal{AS}$ generates the public key pk and the secret key sk of the SHE scheme in [23]. $\mathcal{AS}$ gives pk to $\mathcal{C}$ and $\mathcal{CS}$ and keeps sk.

Enrollment phase: $\mathcal{C}$ provides a feature vector A from the client's biometric data (*e.g.,* fingerprints), runs the type-1 packing method and outputs the encrypted feature vector $\mathsf{vEnc}_1(A)$. The computation server stores $(\mathsf{ID}, \mathsf{vEnc}_1(A))$ in $\mathcal{DB}$ as the reference template for the client ID.

Authentication phase: upon an authentication request, $\mathcal{C}$ provides a fresh biometric feature vector B encrypted with the type-2 packing method and sends $(\mathsf{ID}, \mathsf{vEnc}_2(B))$ to the computational server. $\mathcal{CS}$ extracts from the database the tuple $(\mathsf{ID}, \mathsf{vEnc}_1(A))$ using ID as the search key. $\mathcal{CS}$ calculates the encrypted Hamming distance $\mathsf{ct}_{\mathsf{HD}}$ and sends it to the authentication server. $\mathcal{CS}$ decrypts $\mathsf{ct}_{\mathsf{HD}}$ and retrieves the actual Hamming distance $\mathsf{HD}(A, B) = \mathsf{Dec}(sk, \mathsf{ct}_{\mathsf{HD}})$. $\mathcal{AS}$ returns yes if $\mathsf{HD}(A, B) \leq \kappa$ or no if $HD(A, B) > \kappa$, where κ is the predefined accuracy threshold of the authentication system.

Figure H.27 depicts the authentication phase of Yasuda *et al.*'s BAP.
For additional details on biometric authentication protocols and systems we refer the reader to [16].

3 Combining the BFR and the SHE schemes

In this section, we describe how to efficiently combine the verifiable computation scheme BFR by Backes *et al.* [4] with the somewhat homomorphic scheme SHE by Yasuda *et al.* [23]. We call the resulting scheme BFR+SHE. Our motivation for defining this new scheme is to build a tailored version of BFR that we insert in Yasuda *et al.*'s biometric authentication protocol to mitigate the template recovery attack of [1].

As a preliminary step, we explain the most challenging point This problem rises because the elements and operations in BFR and SHE are defined over two different rings. This passage is quite mathematical, but it is necessary to guarantee the correctness of our composition BFR+SHE, presented later on in this section.

3.1 The ring range problem

The most significant challenge in combining the Backes et al. VC scheme with the Yasuda et al. SHE scheme is the different range of the base rings. While BFR handles all operations in $\mathbb{Z}_p$, where p is a prime, the operations in the SHE scheme [23] are handled in $\mathbb{Z}_d$, where d is the resultant of two polynomials. Therefore, in our BFR+SHE scheme, we need to tweak the input data if there is a mismatch in the ranges. In the calculation of the secure Hamming distance, there is a constant term equal to -2, which lives in $[-d/2, d/2)$ but not in $[0, p)$. In order to verify and generate proper tags, we can write -2 as $D = (d-2) \mod p$. Furthermore, we need to check the impact of the range difference to the verification carried out by the client. The first equation in the **Ver** algorithm of BFR is:

$$\mathsf{ct}_{\mathsf{HD}} = y_0^{(\mathsf{HD})}, \tag{H.37}$$

where $\mathsf{ct}_{\mathsf{HD}} \in \mathbb{Z}_d$ and $y_0^{(HD)} \in \mathbb{Z}_p$. In our instantiation, the term $\mathsf{ct}_{\mathsf{HD}}$ corresponds to the encrypted Hamming distance between the fresh and the reference templates, while $y_0^{(\mathsf{HD})}$ is a component of the final authentication tag. As long as $d \neq p$, Equation (H.37) is not satisfied even when the computation is carried out correctly.

We present a general solution to this problem. For simplicity, we assume $p < d$ (as the tag size should be ideally small), although the reasoning also applies when $p > d$ by swapping the place of p and d. Our solution relies on keeping track of the dividend. Given a stored template $\alpha \in \mathbb{Z}_d$ and a fresh template $\beta \in \mathbb{Z}_d$, both encrypted, we have that: $\alpha = \alpha' + mp, \alpha' = \alpha \mod p \in \mathbb{Z}_p$, $\beta = \beta' + kp$ and $\beta' = \beta \mod p \in \mathbb{Z}_p$.

Let $\mathsf{SWHD}(x, y)$ be the arithmetic circuit for calculating the encrypted Hamming distance without the final modulo d. Let $c = \mathsf{SWHD}(\alpha, \beta)$ and $c' = \mathsf{SWHD}(\alpha', \beta')$, we can derive: $\mathsf{SWHD}(\alpha, \beta) \mod p = \mathsf{SWHD}(\alpha', \beta') \mod p$; and $c = \ell \cdot p + c'$. The value ℓ is the dividend. In our case of study, we want to perform the comparison between the Hamming distance (of the biometric templates) and the threshold κ which determines if the templates match, *i.e.,* the client is authenticated, or not. To this end, we would track $\ell \mod d$ instead of ℓ directly. The reason is that ℓ contains more information and would lead to a privacy leak. Relating back to equation (H.37) we have: $\mathsf{ct}_{\mathsf{HD}} = c \mod d \in \mathbb{Z}_q$ and $y_0^{(\mathsf{HD})} = c' \in \mathbb{Z}_p$. Given $c = \ell \cdot p + c'$, it holds that:

$$\begin{aligned}\mathsf{ct}_{\mathsf{HD}} &= c \mod d = (\ell * p + c') \mod d \\ &= c' \mod d + (\ell \mod d) \cdot (p \mod d) \\ &= (y_0^{(\mathsf{HD})} \mod d) + (\ell \mod d) \cdot (p \mod d)\end{aligned} \tag{H.38}$$

Thus, if we define $\ell_d := (\ell \mod d) \cdot (p \mod d)$, the verification equation in (H.37) becomes $\mathsf{ct}_{\mathsf{HD}} = y_0^{(\mathsf{HD})}(\mod d + \ell_d)$, which is satisfied whenever $\mathsf{ct}_{\mathsf{HD}}$ is computed correctly (as we show in Section 3.3).

3.2 Our BFR+SHE scheme

To facilitate the intuition of how we incorporate BFR+SHE in Yasuda *et al.*'s BAP we describe the algorithms of BFR+SHE directly in the case the encrypted vectors are biometric templates:

BFR+SHE.**KeyGen**(λ)**:** The key generation algorithm runs SHE.**KeyGen**$(\lambda) \to (pk, sk)$ and BFR.**KeyGen**$(\lambda) \to (ek, vk)$. The output is the four-tuple (ek, pk, sk, vk).

BFR+SHE.**Enc**$(pk, A, phase)$**:** The encryption algorithm takes as input the (encryption) public key pk, a plaintext biometric template $A \in \{0,1\}^{2048}$ and a *phase* $\in \{1, 2\}$ to select the appropriate packing method. It outputs the ciphertext ct computed as $\mathsf{ct} = \mathsf{vEnc}_{phase}(A)$, using the type-*phase* packing method of the SHE scheme.

BFR+SHE.**Auth**$(vk, L, \mathbf{ct})$**:** on input the verification key vk, a ciphertext ct and a multi-label $L = (\Delta, \tau)$, with Δ the *data set identifier* (*e.g.,* the client's ID) and τ the *input identifier* (*e.g.,* "stored biometric template" or "fresh biometric template"); this algorithm outputs $\sigma \leftarrow \mathsf{BFR}.\mathbf{Auth}(vk, L, \mathsf{ct})$, with $\sigma = (y_0, Y_i, 1) = (\mathsf{ct}, F_K(\Delta, \tau) \cdot g^{-\mathsf{ct}})^{1/\theta})$, where the value θ and the function F_K are defined in vk.

BFR+SHE.**Comp**(pk, ct_1, ct_2)**:** The compute algorithm takes as input the encryption public key pk, and two ciphertexts ct_1, ct_2, which intuitively correspond to the encryptions $\mathsf{vEnc}_1(A)$ and $\mathsf{vEnc}_2(B)$ respectively. The output is the encrypted Hamming distance HD calculated as: $\mathsf{ct}_{\mathsf{HD}} = C_2 \cdot \mathsf{vEnc}_1(A) + C_1 \cdot \mathsf{vEnc}_2(B) + (-2 \cdot \mathsf{vEnc}_1(A) \cdot \mathsf{vEnc}_2(B)) \in \mathbb{Z}_d$, where $C_1 := \left[\sum_{i=0}^{n-1} r^i\right]_d$ and $C_2 := [-C_1 + 2]_d$ and r, d are extracted from pk. To solve the ring range problem described in Section 3.1 we compute ℓ_d as follows. Let c be the result of the (encrypted) Hamming distance computation without the final modulo d. Then $c' = c \mod p$ and $c = \ell p + c'$, where c' is a component in the authentication tag and ℓ is a dividend. We compute $\ell_d = \ell \mod d = (c - [c \mod p])/p \mod d$. The output is $(\mathsf{ct}_{\mathsf{HD}}, \ell_d)$

BFR+SHE.**Eval**$(ek, pk, \sigma_1, \sigma_2)$**:** The evaluation algorithm takes as input the evaluation key ek, the ecryption public key pk, and two tags, which intuitively correspond to the authenticators for the two biometric templates, A, B. In our case of study, the function to be evaluated is fixed to be $f = \mathsf{HD}$ the Hamming distance. This algorithm outputs $\sigma_{\mathsf{HD}} = (y_0, Y_1, \hat{Y}_2) \leftarrow \mathsf{BFR}.\mathbf{Eval}(ek, \mathsf{HD}, (\sigma_1, \sigma_2))$.

In details, every input gate accepts either two tags σ_A, $\sigma_B \in (\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T)^2$, or one tag and a constant σ, $c \in ((\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T) \times \mathbb{Z}_p)$. The output of a gate is a new tag $\sigma' \in (\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T)$, which will be fed into the next gate in the circuit as one of the two inputs. The operation stops when the final gate of f is reached and the resulting tag σ_{HD} is returned. A tag has the format $\sigma^{(i)} = (y_0^{(i)}, Y_1^{(i)}, \hat{Y}_2^{(i)}) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T$ for $i = 1, 2$ (indicating the two input tags), which corresponds respectively to the coefficients of (x^0, x^1, x^2) in a polynomial. If $\hat{Y}_2^{(i)}$ is not defined, it is assumed that it has value $1 \in \mathbb{G}_T$. Next we define the specific operations for different types of gates:

- **Addition.** The output tag $\sigma' = (y_0, Y_1, \hat{Y}_2)$ is calculated as:
$$y_0 = y_0^{(1)} + y_0^{(2)}, \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)}, \quad \hat{Y}_2 = \hat{Y}_2^{(1)} \cdot \hat{Y}_2^{(2)}.$$
- **Multiplication.** The output tag $\sigma' = (y_0, Y_1, \hat{Y}_2)$ is calculated as:
$$y_0 = y_0^{(1)} \cdot y_0^{(2)}, \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)}, \quad \hat{Y}_2 = e(\hat{Y}_1^{(1)}, \hat{Y}_1^{(2)}).$$
Since the circuit f has maximum degree 2, the input tags to a multiplication gate can only have maximum degree 1 each.
- **Multiplication with constant.** The two inputs are one tag σ and one constant $c \in \mathbb{Z}_p$. The output tag $\sigma' = (y_0, Y_1, \hat{Y}_2)$ is calculated as: $y_0 = c \cdot y_0^{(1)}, \quad Y_1 = (Y_1^{(1)})^c, \quad \hat{Y}_2 = (\hat{Y}_2^{(1)})^c$.

$\mathsf{BFR{+}SHE}.\mathbf{Ver}(vk, \mathcal{P}_\Delta, \mathbf{ct}_{\mathsf{HD}}, \sigma_{\mathbf{HD}}, \ell_d)$**:** The verification algorithm computes $b \leftarrow \mathsf{BFR}.\mathbf{Ver}(vk, sk, \mathcal{P}_\Delta, \mathsf{ct}_{\mathsf{HD}}, \sigma_{\mathsf{HD}}, \kappa)$ to verify the correctness of the outsourced computation. In our case of study, $\mathcal{P}_\Delta$ is a multi-labeled program [4] for the arithmetic circuit for calculating the encrypted HD. The $\mathsf{BFR}.\mathbf{Ver}$ algorithm essentially performs two integrity-checks:

$$\mathsf{ct}_{\mathsf{HD}} = y_0 \mod (d + \ell_d) \quad \text{(H.39)}$$
$$W = e(g, g)^{y_0} \cdot e(Y_1, g)^{\theta} \cdot (\hat{Y}_2)^{\theta^2} \quad \text{(H.40)}$$

If the verification output is $b = 0$ the algorithm returns

$$(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (0, 0).$$

Otherwise, if $b = 1$, it proceeds with the biometric authentication check: it computes $w \leftarrow \mathsf{SHE.Dec}(\mathsf{ct})$ to retrieve the actual Hamming distance $w = \mathsf{HD}(A, B)$. If $\mathsf{HD}(A, B) \leq \kappa$, here κ corresponds to the accuracy of the BAP, the algorithm returns

$$(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (1, 1).$$

If $\mathsf{HD}(A, B) > \kappa$, the output is

$$(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (1, 0).$$

3.3 Correctness analysis

In our BFR+SHE scheme the outsourced function is the Hamming distance HD, that can be represented by a bi-variate deterministic quadratic function. Thus, we can avoid using gate-by-gate induction proofs, as done in [4], and demonstrate the correctness in a direct way. In what follows, we adopt the notation in [4], and we prove the correctness of BFR+SHE by walking through the arithmetic circuit of HD step by step.

Figure H.28 depicts the arithmetic circuit for calculating the encrypted Hamming distance. A and B denote the encrypted stored and fresh biometric templates respectively. C_1 and C_2 are the constants in the function as defined in the BFR+SHE.**Comp** algorithm. The D letter indicates the -2 in the function, but since -2 is not in the valid range $\mathbb{Z}_p$ required by the original BFR scheme, we need to have an intermediate transformation of $D = d-2$. All A, B, C_1 and C_2 are in $\mathbb{Z}_d$. Finally, the σs are the outcome tags of the form $\sigma^{(i)} = (y_0^{(i)}, Y_1^{(i)}, \hat{Y}_2^{(i)}) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T$ after each gate operation, and the Rs are values in either $\mathbb{G}$ or $\mathbb{G}_{\mathbb{T}}$, which are used for homomorphic evaluation over bilinear groups (i.e., **GroupEval** in [4]). We let α

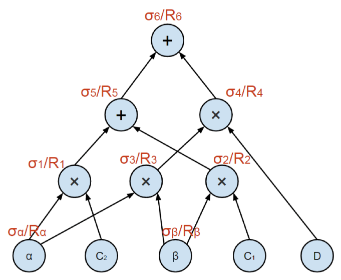


Figure H.28: The arithmetic circuit for calculating the encrypted Hamming distance.

and β be $\mathsf{vEnc}_1(A)$ and $\mathsf{vEnc}_2(B)$ and each of them has a tag: $\sigma_\alpha = (y_0^{(A)}, Y_1^{(A)}, 1)$ and $\sigma_\beta = (y_0^{(B)}, Y_1^{(B)}, 1)$. These two tags are generated by the BFR.**Auth** algorithm, which specifies that $y_0^{(A)} = \alpha$ and $Y_1^{(A)} = (R_\alpha \cdot g^{-\alpha})^{1/\theta}$. Similarly, we have $y_0^{(B)} = \beta$ and $Y_1^{(B)} = (R_\beta \cdot g^{-\beta})^{1/\theta}$. To verify the correctness of our BFR+SHE scheme, we need to check that the two equations specified in the BFR.**Ver** algorithm are satisfied if the computation is performed correctly. To this end, let $\sigma_{\mathsf{HD}} = (y_0^{(\mathsf{HD})}, Y_1^{(\mathsf{HD})}, \hat{Y}_2^{(\mathsf{HD})})$ be the final tag (which is equivalent to σ_6 in the arithmetic circuit depicted in Figure H.28).

The first step is to derive the tags for the intermediate calculation and eventually the final tag. If we run the SHE.**Eval** algorithm homomorphically through the circuit, we will get the outcome tags $\sigma_1, \dots, \sigma_6$ (for details see Appendix H.i). We thus derive σ_{HD} (equivalent to σ_6):

$$\begin{aligned}\sigma_{HD} &= (y_0^{(\mathsf{HD})}, Y_1^{(\mathsf{HD})}, \hat{Y}_2^{(\mathsf{HD})}) \\ &= (\quad C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)}, \\ &\qquad (Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1}, \quad e(Y_1^{(A)}, Y_1^{(B)})^D \quad).\end{aligned}$$

Now we show the proofs for the two verification equations. First we need to prove Equation (H.39), *i.e.,* $\mathsf{ct}_{\mathsf{HD}} = y_0 \mod (d + \ell_d)$. The equality holds as for Equation (H.38). The end result is: $\mathsf{ct}_{\mathsf{HD}} = y_0^{(\mathsf{HD})} \mod d + (\ell \mod d) \cdot (p \mod d)$. As we define $\ell_d = (\ell \mod d) \cdot (p \mod d)$, we can derive Equation (H.39). Secondly, we need to prove that Equation (H.40) holds, *i.e.,* $W = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D$. To this end, we run **GroupEval**(f, R_α, R_β) and execute the bilinear gate operations. Recall that R_α and R_β correspond to R_A and R_B in the notation used in the construction, Denote by R_6 the final result of running **GroupEval** over the circuit of HD. It holds that:

$$\begin{aligned} R_6 &= \textbf{GroupEval}(f, R_\alpha, R_\beta) = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D \\ &= \textbf{GroupEval}(f, R_\alpha, R_\beta) = e(g, g)^{y_0^{\mathsf{HD}}} \cdot e(Y_1^{\mathsf{HD}}, g)^\theta \cdot (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2} \end{aligned}$$

By expanding the last expression the desired result (see Appendix H.i for details). Thus, we have proved the correctness of the $\mathsf{BFR{+}SHE}$ scheme. which are the results of the pseudo-random function F_K in the $\mathsf{BFR}.\textbf{Ver}$ algorithm.

4 Improving the Yasuda *et al.* protocol

In this section, we describe a modified version of the Yasuda et al. [23] protocol that is secure against the recently identified *hill-climbing* attack that can be performed by a malicious computation server $\mathcal{CS}$. It is composed of four distributed parties: a client $\mathcal{C}$ (holding the keys pk, ek and vk), a computation server/database $\mathcal{CS}$ (holding the keys pk and ek), an authentication server $\mathcal{AS}$ (holding the keys pk, sk and vk). Figure In the proposed protocol, we preserve the assumption that $\mathcal{AS}$ is a trusted party and furthermore assume the client $\mathcal{C}$ and the database $\mathcal{DB}$ are also trusted parties. $\mathcal{C}$ is responsible to manage the secret key vk for the verifiable computation scheme and $\mathcal{DB}$ stores the encrypted reference biometric templates with the identities of the corresponding clients. However, $\mathcal{CS}$ can be malicious and cheat with flawed computations. We describe the three main phases of our proposed improvement of Yasuda *et al.*'s privacy-preserving biometric authentication protocol:

Setup Phase: In this phase the authentication server $\mathcal{AS}$ runs $\mathsf{SHE}.\textbf{KeyGen}(\lambda)$ to generate the public key pk and the secret key sk of the somewhat homomorphic encryption (SHE) scheme. $\mathcal{AS}$ keeps sk and distributes pk to both the client $\mathcal{C}$ and the computation server $\mathcal{CS}$.

Enrollment Phase: Upon client registration, the client $\mathcal{C}$ runs $\mathsf{BFR}.\textbf{KeyGen}(\lambda)$ to generate the public evaluation key ek and the secret verification key vk. $\mathcal{C}$ distributes ek to $\mathcal{CS}$ and vk to $\mathcal{AS}$. The client $\mathcal{C}$ generates a 2048-bit feature vector A from the client's biometric data, runs $\mathsf{BFR{+}SHE}.\textbf{Enc}(pk, A, 0)$ to obtain the ciphertext ct_A. $\mathcal{C}$ authenticates ct_A by running $\mathsf{BFR{+}SHE}.\textbf{Auth}(vk, L_A, \mathsf{ct}_A)$ and outputs a tag σ_A. Then $\mathcal{C}$ sends the three-tuple $(\mathsf{ID}, \mathsf{ct}_A, \sigma_A)$ to the database. This three-tuple serves as the reference biometric template for the specific client with identity ID.

Authentication Phase: The client provides fresh biometric data as a feature vector $B \in \{0,1\}^{2048}$. $\mathcal{C}$ runs $\mathsf{BFR{+}SHE}.\textbf{Enc}(pk, B, 1)$ to obtain the ciphertext ct_B and authenticates it by running $\sigma_B \leftarrow \mathsf{BFR} + \mathsf{SHE}.\textbf{Auth}(vk, L_B, \mathsf{ct}_B)$. $\mathcal{C}$ sends $(\mathsf{ID}, \mathsf{ct}_B, \sigma_B)$ to $\mathcal{CS}$, who extracts the tuple $(\mathsf{ID}, \mathsf{ct}_A, \sigma_A)$ corresponding to the client to be authenticated (using the ID as the search key). $\mathcal{CS}$ calculates the encrypted Hamming distance $ct_{\mathsf{HD}} \leftarrow \mathsf{BFR} + \mathsf{SHE}.\textbf{Comp}(pk, \mathsf{ct}_A, \mathsf{ct}_B)$ and generates a corresponding tag $\sigma_{\mathsf{HD}} \leftarrow \mathsf{BFR} + \mathsf{SHE}.\textbf{Eval}(ek, pk, \sigma_A, \sigma_B)$. Then, $\mathcal{CS}$ sends $(\mathsf{ID}, \mathsf{ct}_{\mathsf{HD}}, \sigma_{\mathsf{HD}})$ to the authentication server. $\mathcal{AS}$ runs $(acc_{\mathsf{VC}}, acc_{\mathsf{HD}})$

Input: the client's identity ID, the public key for SHE scheme pk, the public evaluation key for the BFR scheme ek, the stored reference template and tag $(\mathsf{vEnc}_1(A), \sigma_A)$, the encrypted fresh template $\mathsf{vEnc}_2(B)$ and its tag σ_B.

Output: the inner product $ct_P = \mathsf{vEnc}_1(A) \cdot \mathsf{vEnc}_2(A')$ and the tag σ'_{HD}.

Goals: make the authentication server accept the inner product computation and return yes, and use the hill-climbing strategy recover the reference template A.

Figure H.29: Setting for Abidin *et al.*'s template recovery attack in [1].

$\leftarrow \mathsf{BFR} + \mathsf{SHE}.\mathbf{Ver}(vk, sk, \mathcal{P}_\Delta, \mathsf{ct}_{\mathsf{HD}}, \sigma_{\mathsf{HD}}, \kappa)$, where κ is the desired accuracy level of the BAP. If either acc_{VC} or acc_{HD} is 0 $\mathcal{AS}$ outputs a no, for authentication rejection. Otherwise, $(acc_{\mathsf{VC}}, acc_{\mathsf{HD}}) = (1, 1)$ and $\mathcal{AS}$ outputs yes, for authentication success.

4.1 Security analysis of the proposed BAP

Our primary aim is to demonstrate that our privacy-preserving biometric authentication protocol is not vulnerable to Abidin *et al.*'s template recovery attack [1]. To this end, we sketch the attack setting in Figure H.29.
We recall that for this attack, the adversary is a malicious computational server who tries to recover the stored reference biometric template of a client with identity ID. All the other parties of the BAP, are trusted and behave honestly.

In what follows, we show that the malicious $\mathcal{CS}$ cannot forge a tag $\sigma_{\mathsf{HD}'}$ that passes the verification checks performed in BFR. It is possible for the adversary to cheat on the first equality (Equation (H.39)) as it only tests that the returned computation result ($\mathsf{ct}_{\mathsf{HD}}$ or ct_P) aligns with the arithmetic circuit used to generate the tag (σ_{HD} or $\sigma_{\mathsf{HD}'}$). In [1], $\mathcal{CS}$ succeeds by computing the arithmetic circuit for the inner product instead of HD. In this case, it is not possible for the malicious computational server to fool the second second integrity check (Equation (H.40)). In details, $\mathcal{AS}$ calculates $W = \mathbf{GroupEval}(f, R_\alpha, R_\beta)$, and since $\mathcal{AS}$ is honest, $f = \mathsf{HD}$ is the arithmetic circuit for the Hamming distance. If $\mathcal{CS}$ returns incorrect results, with overwhelming probability the second verification equation does not hold, thus the attack is mitigated.

Other threats.

In what follows, we consider attack scenarios in which one of the participating entities in the BAP is malicious.

Malicious client. $\mathcal{C}$ is responsible to capture the reference template and the fresh template as well as to perform the encryption. If the client is malicious, the knowledge of the encryption secret key and of the identity ID enables $\mathcal{C}$ to initiate a *center search attack* and recover the stored template A as explained in [17]. Unfortunately, Pagnin *et al.* [17] show that this class of attacks cannot be detected using verifiable computation techniques, since the attacker is not cheating with the computation.

A new concern with the modified Yasuda *et al.* protocol is the key generation for BFR. In the protocol, we let the client $\mathcal{C}$ generate the private key vk, the evaluation key ek and the authentication tags because we assume $\mathcal{C}$ is a trusted party. If $\mathcal{C}$

turns malicious, it could give a fake vk to the authentication server $\mathcal{AS}$ and initiate the template recovery attack with the inner product by simulating $\mathcal{CS}$. Since the adversary ($\mathcal{C}$) controls vk, the computation verification step becomes meaningless.

Malicious computation server. The main motivation to integrate VC in BAPs is indeed to prevent $\mathcal{CS}$ from behaving dishonestly. Unlike the client $\mathcal{C}$, $\mathcal{CS}$ only has access to the encrypted templates $\mathsf{vEnc}_1(A)$ and $\mathsf{vEnc}_2(B)$ and the user pseudonyms. $\mathcal{CS}$ cannot modify the secret key of the BFR scheme. We have analysed how the template recovery attack conducted by $\mathcal{CS}$ can be countered and hence we shorten the discussion here.

In contrast to the original protocol, $\mathcal{CS}$ needs to calculate an extra value ℓ_d to solve the range issue after integrating BFR. However, ℓ_d is still operated on the ciphertext level and is not involved in the second verification equation. Thus learning ℓ_d does not provide any significant advantage in recovering the templates.

Malicious authentication server. A malicious $\mathcal{AS}$ will completely break down the privacy of the BAP since it controls the secret key sk used by the SHE scheme. If $\mathcal{AS}$ successfully eavesdrops and obtains the ciphertext $\mathsf{vEnc}_1(A)$ or $\mathsf{vEnc}_2(B)$, it can recover the plaintext biometric templates.

4.2 Efficiency Analysis

The original BFR scheme in [4] allows alternative algorithms to improve the efficiency of the verifier. Although in our instantiation we did not use these algorithms, the current definition of the multi-labels in BFR+SHE is extensible. Given also that the function to be computed is $f = \mathsf{HD}$ and has a very simple description as arithmetic circuit, running the BFR+SHE.**Ver** algorithm requires $O(|f|)$ computational time. In addition, if the amortized closed-form efficiency functionality is adopted, the verification function will run in time $O(1)$. Nonetheless, the arithmetic circuit of HD has 6 gates only and the saved computation overhead would be relatively small.

5 A flawed approach

Privacy and integrity are the two significant properties desired in a privacy-preserving BAP. There are two possible ways to combine VC and homomorphic encryption (HE): running VC on top of HE, and viceversa, running HE on top of VC.

In the first case, the data (biometric template) is first encrypted and then encoded to generate an authentication proof. Our construction of BFR+SHE follows this principle. In this approach, $\mathcal{AS}$ can make the judgement whether the output of $\mathcal{CS}$ is from a correct computation of HD *before* decrypting the ciphertext.

In the second case, the data is first encoded for verifiable computation and then the encoded data is encrypted. This combination is not really straightforward and is prune to security breaches.

In this section, we demonstrate an attack strategy that may lead to information leakage in case the homomorphic encryption scheme (henceforth FHE)[30] is applied on top of a VC scheme. For the sake of generality, we define $\mathsf{FHE} = (KeyGen_{FHE}, Enc, Dec, Eval)$. For verifiable computation scheme we adopt the notation of Gennaro *et al.* [11] and define $\mathsf{VC} = (KeyGen_{VC}, ProbGen, Compute, Ver)$, where $KeyGen_{VC}$ outputs the private key sk_{vc} and public key pk_{vc}; $ProbGen$

[30]The same leakage of information could happen if a SHE scheme is used.

takes sk_{vc} and the plaintext x as input and outputs the encoded value σ_x; *Compute* takes the circuit f, the encrypted encoded input and outputs the encoded version of the output; Ver is performed to verify the correctness of the computation given the secret key sk_{vc} and the encoded output σ_y. The main idea of the flawed approach is to first encode the data in plaintext and then encrypt the encoded data. It can be represented by $\hat{x} = Enc(ProbGen(x))$, where $\hat{x}$ is what the malicious server gets access to.

5.1 The attack

We describe now a successful attack strategy to break the privacy-preservation property of a BAP built with the second composition method: HE on top of VC (or HE *after* VC). The adversary's goal is to recover σ_y, i.e., the encoded value of the computation result. The attack runs in different phases. We show that the privacy-preserving property is broken if $q \geq n$, where q is the number of queries in the learning phase and n is the length of the encoded result σ_y. For simplicity we collect the two entities $\mathcal{C}$ and $\mathcal{AS}$ into a single trusted party V that we refer to as the Verifier.

The attack is depicted in Figure H.30, a more detailed description follows.

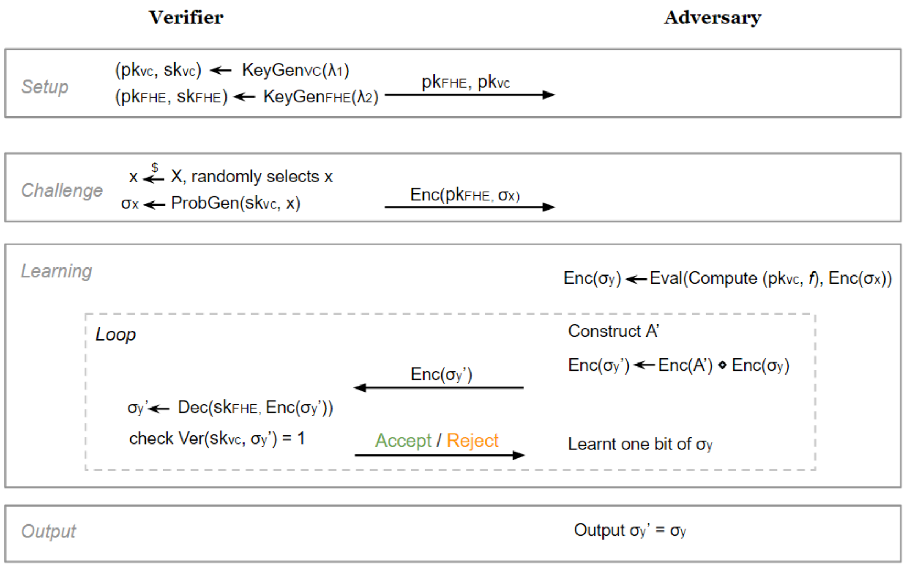


Figure H.30: An attack strategy against the naïve the integration of FHE on top of VC.

Setup phase: V generates the keys of the protocol and gives pk_{vc}, pk_{FHE} to Adv.

Challenge phase: V generates the encoded version σ_x for the input x. V encrypts the encoded input and sends $Enc(\sigma_x)$ to Adv.

Learning phase: Adv uses V as a decryption oracle by sending verification queries, which can be further divided into the following steps:

1. Adv performs honest computations and derives the $Enc(\sigma_y)$.
2. Adv constructs a vector $A' \in \mathbb{Z}_2^n$ equal in length to σ_y. A' is initialized with the last bit set to 0 and the rest of the bits set to 1. For the i^{th} trial, we set $A' = (1_1, \ldots, 0_i, 1_{i+1}, \ldots, 1_{n-1}, 1_n)$, i.e., set the i^{th} bit to 0 and the rest bits to 1.

3. *Adv* encrypts the tailored vector A' and reuses the honest result $Enc(\sigma_y)$ from step 1. Then she computes: $Enc(\sigma_{y'}) = Enc(A') \diamond Enc(\sigma_y)$, where $\diamond$ represents the Hadamard product for binary vectors and sends the result to V for verification.
4. V decrypts $Enc(\sigma'_y)$. Thanks to the homomorphic property of FHE, V can derive $\sigma_{y'} = A' \diamond \sigma_y$. V checks the computation based on the encoded result $\sigma_{y'}$ and returns either *accept* if $Ver(sk, \sigma_{y'}) = 1$ or *reject*, otherwise.
5. The attacker A' acts as a "mask": it copies all the bit values of σ_y into $\sigma_{y'}$ except for the i^{th} bit, which is always set to zero. Consequently, if the output of the verification is *accept*, *Adv* will learn that $\sigma_y = \sigma_{y'}$ as well as $Enc(\sigma_y) = Enc(\sigma_{y'})$, which reveals that the i^{th} bit of σ_y equals to 0. Similarly, if the output of the verification is *reject*, *Adv* learns that the i^{th} bit of σ_y is 1. In both cases, one bit of σ_y is leaked.

Output phase: After $q \geq n$ verification queries, where n equals the length of σ_y, *Adv* outputs $\sigma_{y'}$.

It is trivial to check that that $\sigma_{y'} = \sigma_y$ and thus $Ver(sk, \sigma'_y) = Ver(sk, \sigma'_y) = 1$ and attacker's goal is achieved.

The attack demonstrates that the order of combining a VC and a (F)HE is very crucial: the verifier must decrypt the ciphertext *before* it can determine whether it is the result of the correct outsourced computation. Adopting such a scheme in a BAP would make $\mathcal{AS}$ a decryption oracle. Leaking information on the Hamming distance may be exploited to perform further attacks that might lead to the full recovery of biometric templates as it has been recently shown [17]. Formally speaking, we can say that the HE on top of VC is not a chosen-ciphertext attack (CCA) secure scheme.

6 Conclusions

Biometric authentication protocols have gained considerable popularity for access control services. Preserving the privacy of the biometric templates is highly critical due to their irrevocable nature. Yasuda et al. proposed a biometric authentication protocol [23] using a SHE scheme. However, a hill-climbing attack [1] has been presented against this protocol that relies on a malicious internal computation server $\mathcal{CS}$ that performs erroneous computations and leads to the disclosure of the biometric reference template. We counter the aforementioned attack by constructing a new scheme named BFR+SHE which adds a verifiable computation layer to the SHE scheme. We then describe a modified version of the Yasuda et al. protocol that utilizes our BFR+SHE scheme, and demonstrate that the improved BAP provides higher privacy guarantees. Although employing VC to mitigate hill-climbing attack techniques seems a quite straight-forward step, we demonstrate that not all combinations of a VC scheme with a HE one are secure, and show how a naïve combination leads to a drastic private information leakage in BAP.

Bibliography

[1] Aysajan Abidin and Aikaterini Mitrokotsa. “Security aspects of privacy-preserving biometric authentication based on ideal lattices and ring-LWE”. In: *Proceedings of the IEEE Workshop on Information Forensics and Security 2014 (WIFS 2014).* 2014.

[2] L Babai. “Trading Group Theory for Randomness”. In: *Proceedings of STOC’85.* Providence, Rhode Island, USA: ACM, 1985, pp. 421–429.

[3] M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk. “ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data”. In: *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland).* 2015.

[4] Michael Backes, Dario Fiore, and Raphael M Reischuk. “Verifiable delegation of computation on outsourced data”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* ACM. 2013, pp. 863–874.

[5] Manuel Barbosa, Thierry Brouard, Stéphane Cauchie, and Sim$\mathcal{T}$ao Melo Sousa. “Secure Biometric Authentication with Improved Accuracy.” In: *ACISP 2008.* Ed. by Yi Mu, Willy Susilo, and Jennifer Seberry. Vol. 5107. LNCS. Springer, 2008, pp. 21–36.

[6] Julien Bringer, Hervé Chabanne, Firas Kraïem, Roch Lescuyer, and Eduardo Soria-Vázquez. “Some applications of verifiable computation to biometric verification”. In: *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on.* IEEE. 2015, pp. 1–6.

[7] Julien Bringer, Hervé Chabanne, and Alain Patey. “Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends”. In: *Signal Processing Magazine, IEEE* 30.2 (2013), pp. 42–52.

[8] Julien Bringer, Hervé Chabanne, and Alain Patey. “Shade: Secure Hamming Distance Computation from Oblivious Transfer”. In: *FC 13.* Springer. 2013, pp. 164–176.

[9] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. “Geppetto: Versatile verifiable computation”. In: *Security and Privacy (SP), 2015 IEEE Symposium on.* IEEE. 2015, pp. 253–270.

[10] Dario Fiore, Rosario Gennaro, and Valerio Pastro. “Efficiently verifiable computation on encrypted data”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security.* ACM. 2014, pp. 844–855.

[11] Rosario Gennaro, Craig Gentry, and Bryan Parno. "Non-interactive verifiable computing: Outsourcing computation to untrusted workers". In: *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 465–482.

[12] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. Bethesda, MD, USA: ACM Press, 2009, pp. 169–178.

[13] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM J. Comput.* 18.1 (Feb. 1989), pp. 186–208.

[14] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. "Improved garbled circuit building blocks and applications to auctions and computing minima". In: *Cryptology and Network Security*. Springer, 2009, pp. 1–20.

[15] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. "Can homomorphic encryption be practical?" In: *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM. 2011, pp. 113–124.

[16] Elena Pagnin. "Authentication under Constraints". Licentiate dissertation. Chalmers University of Technology, 2016.

[17] Elena Pagnin, Christos Dimitrakakis, Aysajan Abidin, and Aikaterini Mitrokotsa. "On the Leakage of Information in Biometric Authentication". In: *Progress in Cryptology–INDOCRYPT 2014*. Springer, 2014, pp. 265–280.

[18] Elena Pagnin and Aikaterini Mitrokotsa. "Privacy-preserving biometric authentication: challenges and directions". In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 450.

[19] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. "Pinocchio: Nearly Practical Verifiable Computation". In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 238–252.

[20] Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys. "A Framework for Analyzing Template Security and Privacy in Biometric Authentication Systems". In: *IEEE Transactions on Information Forensics and Security* 7.2 (2012), pp. 833–841.

[21] Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys. "A framework for analyzing template security and privacy in biometric authentication systems". In: *IEEE Transactions on Information Forensics and Security* 7.2 (2012), pp. 833–841.

[22] A. Stoianov. "Cryptographically secure biometrics". In: *SPIE 7667, Biometric Technology for Human Identification VII* (2010), pp. 76670C–12.

[23] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. "Practical packing method in somewhat homomorphic encryption". In: *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2014, pp. 34–50.

Appendix

H.i Details in the correctness analysis

In this section, we show the intermediate steps of the calculation.
The derived tags are:
$\sigma_1 = (C_2 \cdot y_0^{(A)}, (Y_1^{(A)})^{C_2}, 1)$; $\sigma_2 = (C_1 \cdot y_0^{(B)}, (Y_1^{(B)})^{C_1}, 1)$;
$\sigma_3 = (y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)}} \cdot (Y_1^{(B)})^{y_0^{(A)}}, e(Y_1^{(A)}, Y_1^{(B)}))$;
$\sigma_4 = (D \cdot y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)} \cdot D} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D}, e(Y_1^{(A)}, Y_1^{(B)}))^D$;
$\sigma_5 = (C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)}, (Y_1^{(A)})^{C_2} \cdot (Y_1^{(B)})^{C_1}, 1)$;

$$\sigma_6 = \begin{pmatrix} C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)} \\ (Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1} \\ e(Y_1^{(A)}, Y_1^{(B)})^D \end{pmatrix}$$

The homomorphic bilinear map calculation results are:
$R_1 = R_\alpha^{C_2}$; $R_2 = R_\beta^{C_1}$; $R_3 = e(R_\alpha, R_\beta)$; $R_4 = e(R_\alpha, R_\beta)^D$;
$R_5 = R_\alpha^{C_2} \cdot R_\beta^{C_1}$; $R_6 = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D$.

To prove that $W = \textbf{GroupEval}(f, R_\alpha, R_\beta)$ satisfies Equation (H.40), we start by analysing the three factors that made up the righthand of the equation, namely: $e(g,g)^{y_0^{\mathsf{HD}}} \cdot e(Y_1^{\mathsf{HD}}, g)^\theta \cdot (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2}$. We in turn expand each one of the factors and finally compute the product of the results, evaluating it against W.
The first factor can be expanded as:

$$e(g,g)^{y_0^{HD}} = e(g,g)^{C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)}} = e(g,g)^{C_2\alpha + C_1\beta + \alpha\beta D}.$$

The second factor is expanded as:

$$\begin{aligned} e(Y_1^{HD}, g)^\theta &= e((Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1}, g)^\theta \\ &= e((R_\alpha \cdot g^{-\alpha})^{(\beta D + C_2)/\theta} \cdot (R_\beta \cdot g^{-\beta})^{(\alpha D + C_1)/\theta}, g)^\theta \\ &= e(R_\alpha^{\beta D + C_2} \cdot R_\beta^{\alpha D + C_1} \cdot g^{-2\alpha\beta D - \alpha C_2 - \beta C_1}, g) \\ &= e(R_\alpha, g)^{\beta D + C_2} \cdot e(R_\beta, g)^{\alpha D + C_1} \cdot e(g,g)^{-2\alpha\beta D - \alpha C_2 - \beta C_1}. \end{aligned}$$

The third factor is expanded as:

$$\begin{aligned} (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2} &= e(Y_1^{(A)}, Y_1^{(B)})^{D\theta^2} = e((R_\alpha \cdot g^{-\alpha})^{1/\theta}, (R_\beta \cdot g^{-\beta})^{1/\theta})^{D\theta^2} \\ &= e(R_\alpha \cdot g^{-\alpha}, R_\beta \cdot g^{-\beta})^D = e(R_\alpha, R_\beta \cdot g^{-\beta})^D \cdot e(g^{-\alpha}, R_\beta \cdot g^{-\beta})^D \\ &= e(R_\alpha, R_\beta)^D \cdot e(R_\alpha, g)^{-\beta D} \cdot e(R_B, g)^{-\alpha D} \cdot e(g,g)^{\alpha\beta D}. \end{aligned}$$

Here we need to prove the right hand side is equal to W. We use a temporary variable $P = e(g,g)^{y_0^{\mathsf{HD}}} \cdot e(Y_1^{\mathsf{HD}}, g)^\theta \cdot (\hat{Y}_2^{(\mathsf{HD})})^{\theta^2}$ to denote the expansion result of the righthand-side. The expression below proves the correctness of the second verification equation (H.40).

$$\begin{aligned} P = {} & e(g,g)^{C_2 \cdot \alpha + C_1 \cdot \beta + D \cdot \alpha \cdot \beta} \cdot e(R_\alpha, g)^{\beta D + C_2} \cdot e(R_\beta, g)^{\alpha D + C_1} \cdot e(g,g)^{-2\alpha\beta D - \alpha C_2 - \beta C_1} \cdot \\ & \cdot e(R_\alpha, R_\beta)^D \cdot e(R_\alpha, g)^{-\beta D} \cdot e(R_B, g)^{-\alpha D} \cdot e(g,g)^{\alpha\beta D} \\ = {} & e(g,g)^0 \cdot e(R_\alpha, g)^{C_2} \cdot e(R_\beta, g)^{C_1} \cdot e(R_a, R_b)^D \\ = {} & e(R_\alpha^{C_2}, g) \cdot e(R_\beta^{C_1}, g) \cdot e(R_a, R_b)^D \\ = {} & e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_a, R_b)^D = W. \end{aligned}$$

It's not that I'm so smart,
it's just that I stay with problems longer.

Albert Einstein