Security aspects of privacy-preserving biometric authentication based on ideal lattices and ring-LWE

Aysajan Abidin and Aikaterini Mitrokotsa
(aysajan@chalmers.se and aikaterini.mitrokotsa@chalmers.se)
Chalmers University of Technology, Gothenburg, Sweden

Abstract—**In this paper, we study the security of two recently proposed privacy-preserving biometric authentication protocols that employ packed somewhat homomorphic encryption schemes based on ideal lattices and ring-LWE, respectively. These two schemes have the same structure and have distributed architecture consisting of three entities: a client server, a computation server, and an authentication server. We present a simple attack algorithm that enables a malicious computation server to learn the biometric templates in at most $2N - \tau$ queries, where N is the bit-length of a biometric template and τ the authentication threshold. The main enabler of the attack is that a malicious computation server can send an encryption of the inner product of the target biometric template with a bitstring of his own choice, instead of the securely computed Hamming distance between the fresh and stored biometric templates. We also discuss possible countermeasures to mitigate the attack using private information retrieval and signatures of correct computation.**

Index Terms—**Privacy-preserving biometric authentication, somewhat homomorphic encryption, lattices, ring-LWE, hill climbing attack**

I. Introduction

Privacy-preserving biometric authentication is attracting a lot of attention from the cryptographic community, due to the serious associated security and privacy implications. Indeed, compromised biometric templates may lead to serious threats (*i.e.* identity theft and fraud), while the inherent irrevocability of biometrics renders this risk even more serious. It has been shown that fingerprints may reveal genetic information, while retina scans may reveal diseases such as diabetes and strokes. Moreover, additional issues of linkability, profiling and tracking of individuals are raised by cross-matching biometric traits.

Many privacy-preserving biometric authentication protocols have been proposed [1]–[9], that rely on secure multi-party computation techniques, such as homomorphic encryption [10] and oblivious transfer [11], [12]. Recently, Yasuda *et al.* proposed two efficient privacy-preserving biometric authentication in [13], [14] using packed somewhat homomorphic encryption based on ideal lattices and ring learning with error (ring-LWE). In this paper, we analyse the security of these protocols and describe a simple attack algorithm that can be employed by a malicious internal entity to recover the biometric templates of arbitrary users.

The paper is organized as follows. Section 2 introduces the necessary background along with the biometric authentication protocols proposed by Yasuda *et al.* Section 3 presents the attack. After a brief discussion of possible countermeasures in Section 4, we conclude the paper in Section 5.

II. Preliminaries

Here we review the two packed *somewhat homomorphic encryption (SHE)* schemes proposed by Yasuda *et al.* in [13], [14], as well as the two privacy-preserving biometric authentication protocols that they proposed. Let us start by introducing the notations used.

When p is a prime number, the finite field with p elements is denoted by $\mathbb{F}_p$. For two distinct positive integers z and d with $d < z$, $z \mod d$ is denoted by $[z]_d$ when the operation maps the integers to the interval $[-d/2, d/2)$. Throughout this paper, we let $n = 2^i$, for a positive integer i. Then, $R := \mathbb{Z}[x]/(x^n + 1)$ is a ring. Finally, $N \leqslant n$ denotes the bit-length of a biometric template.

Let us now briefly review the packed SHE schemes proposed by Yasuda *et al.* in [13], [14] based on *i)* ideal lattices and *ii)* ring-LWE.

A. The SHE scheme based on ideal lattices

Here, we briefly describe the key parameters of a SHE scheme based on ideal lattices and how encryption and decryption are performed. For further details on these, we urge the reader to consult [13], [15], [16].

– ***Key Generation*** **-** $\mathsf{KeyGen}(1^\ell)$: Upon an input 1^ℓ, where ℓ is a security parameter, the key generation algorithm KeyGen outputs a public key $\mathsf{pk} = (d, r, n, s)$ and a secret key $\mathsf{sk} = \omega$ with $\gcd(\omega, s) = 1$. We note that n is the lattice dimensions, and that $\gcd(d, s) = 1$ and $r^n \equiv -1 \mod d$.

– ***Encryption*** **-** $\mathsf{E}(m, \mathsf{pk})$: A message $m \in \mathbb{Z}/s\mathbb{Z}$ is encrypted with the public key pk by first choosing a random noise vector $\mathbf{u} = (u_0, u_1, \cdots, u_{n-1})$ with $u_i \in \{0, \pm 1\}$ chosen as 0 with probability q and as ± 1 with probability $(1 - q)/2$, and then by computing $\mathsf{ct} := \mathsf{E}(m, \mathsf{pk}) = \left[m + s\sum_{i=0}^{n-1} u_i r^i\right]_d$.

– ***Decryption*** **-** $\mathsf{D}(\mathsf{ct}, \mathsf{sk})$: The message m is recovered from the ciphertext $\mathsf{ct} = \mathsf{E}(m, \mathsf{pk})$ by computing $[\mathsf{ct} \cdot \omega]_d \cdot \omega^{-1} \mod s$.

Let $A = (A_0, A_1, \cdots, A_{N-1})$ be a binary vector of length N. Yasuda *et al.* [13] proposed the following two packing methods.

– Packing Method Type 1:
Let $F_1 : A \mapsto \sum_{i=0}^{n-1} A_i x^i \in \mathbb{Z}[x]/(x^n + 1)$. Then the type 1 packed ciphertext of A is defined by the integer:

$$\mathsf{vE}_1(A) := [F_1(A)(r) + su_1(r)]_d = \left[\sum_{i=0}^{n-1} A_i r^i + su_1(r)\right]_d,$$

where (d, r, n, s) is the public key pk and $u_1(x) \in R$ denotes a noise polynomial.

– PACKING METHOD TYPE 2:
Let $F_2 : A \mapsto -\sum_{i=0}^{n-1} A_i x^{n-i} \in \mathbb{Z}[x]/(x^n + 1)$. Then the type 2 packed ciphertext of A is defined by the integer:

$$\mathsf{vE}_2(A) := [F_2(A)(r) + s u_2(r)]_d = \left[-\sum_{i=0}^{n-1} A_i r^{n-i} + s u_2(r) \right]_d,$$

where (d, r, n, s) is the public key pk and $u_2(x) \in R$ denotes a noise polynomial.

B. The SHE scheme based on ring-LWE

Let p be a prime number with $p \equiv 1 \mod 2n$, and let $t < p$ be a positive integer. Let the ring $R_t := \mathbb{Z}_t[x]/(x^n + 1)$ be the plaintext space, and the ring $R_p := R/pR = \mathbb{F}_p/(x^n + 1)$ the ciphertext space. A SHE scheme based on ring-LWE comprises of the following:

– ***Key Generation -*** $\mathsf{KeyGen}(1^\ell)$: Upon an input 1^ℓ, where ℓ is a security parameter, the key generation algorithm chooses $s \xleftarrow{\chi} R$ according to a discrete Gaussian error distribution $\chi := D_{\mathbb{Z}^n,\sigma}$ with standard deviation σ, samples a random element $a_1 \in R_p$ and an error $e \xleftarrow{\chi} R$. It outputs $\mathsf{pk} = (a_0, a_1)$ with $a_0 := (a_1 \cdot s + t \cdot e)$ and $\mathsf{sk} = s$ as the public and secret keys, respectively.
– ***Encryption -*** $\mathsf{E}(m, \mathsf{pk})$: A message $m \in R_t$ is encrypted with the public key pk by first sampling $u, f, g \xleftarrow{\chi} R$ and then computing $\mathsf{ct} := \mathsf{E}(m, \mathsf{pk}) = (c_0, c_1) = (a_0 u + tg + m, a_1 u + tf) \in R_p^2$.
– ***Decryption -*** $\mathsf{D}(\mathsf{ct}, \mathsf{sk})$: The message m is recovered from the ciphertext $\mathsf{ct} = (c_0, c_1)$ by computing $[c_0 + c_1 s]_p \mod t \in R_t$. In general, from a ciphertext $\mathsf{ct} = (c_0, \cdots, c_\delta)$, the corresponding plaintext m is recovered by computing $\left[\sum_{i=0}^{\delta} c_i s^i\right]_p \mod t$.

Let $A = (A_0, A_1, \cdots, A_{n-1})$ be a vector of length n as before. Yasuda *et al.* [14] proposed the following two packing methods:

– PACKING METHOD TYPE 1:
Set $\mathsf{pm}_1(A) := \sum_{i=0}^{n-1} A_i x^i$. For sufficiently large t, $\mathsf{pm}_1(A)$ can be considered as an element of R_t. Then define

$$\mathsf{vE}_1(A) := \mathsf{E}\big(\mathsf{pm}_1(A), \mathsf{pk}\big)$$

as the type 1 packed ciphertext.

– PACKING METHOD TYPE 2:
Set $\mathsf{pm}_2(A) := -\sum_{i=0}^{n-1} A_i x^{n-i}$. Again, for sufficiently large t, $\mathsf{pm}_2(A)$ can be considered as an element of R_t. Then define

$$\mathsf{vE}_2(A) := \mathsf{E}\big(\mathsf{pm}_2(A), \mathsf{pk}\big)$$

as the type 2 packet ciphertext.

C. Secure Hamming distance computation

Let $A = (A_0, A_1, \cdots, A_n)$ and $B = (B_0, B_1, \cdots, B_n)$ two binary vectors of length n. Then, the Hamming distance between them is

$$\mathsf{HD}(A, B) = \sum_{i=0}^{n-1} A_i \oplus B_i = \sum_{i=0}^{n-1} (A_i + B_i - 2A_i B_i).$$

So using the two packed SHE schemes presented above, the Hamming distance is securely computed as follows.

– **Using the ideal lattice based packed SHE:**
Let

$$C_1 = \Big[\sum_{i=0}^{n-1} r^i \Big]_d \text{ and } C_2 = [-C_1 + 2]_d.$$

Then, we can securely compute the Hamming distance between A and B from their packed encryptions of type 1 and 2, respectively. In particular,

$$\mathsf{ct}_H := \mathsf{vE}_1(A) C_2 + \mathsf{vE}_2(B) C_1 + (-2\mathsf{vE}_1(A)\mathsf{vE}_2(B)), \quad (1)$$

is an encryption of $\mathsf{HD}(A, B)$ [13].

– **Using the ring-LWE based packed SHE:**
Let

$$C_1 = -\sum_{i=0}^{n-1} x^{n-i} \text{ and } C_2 = \sum_{i=0}^{n-1} x^i.$$

Then, we can securely compute the Hamming distance between A and B from their packed encryptions of type 1 and 2, respectively. In particular,

$$\mathsf{ct}_H := \mathsf{vE}_1(A) C_1 + \mathsf{vE}_2(B) C_2 + (-2\mathsf{vE}_1(A)\mathsf{vE}_2(B)), \quad (2)$$

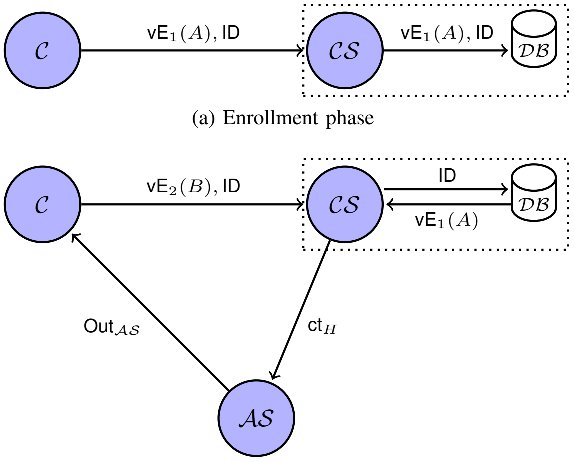
is an encryption of $\mathsf{HD}(A, B)$ [14].

We note that in both SHE schemes, $\mathsf{vE}_1(A)\mathsf{vE}_2(B)$ corresponds to an encryption of the inner product of A and B. Namely, the decryption of $\mathsf{vE}_1(A)\mathsf{vE}_2(B)$ results in the inner product of A and B [13], [14].

D. The protocols

Using the two packed SHE schemes presented in the previous section, Yasuda *et al.* proposed two biometric authentication protocols respectively in [13], [14]. The protocols employ a distributed model for internal entities, which consist of three entities: a client server $\mathcal{C}$, a computation server $\mathcal{CS}$ with a database $\mathcal{DB}$, and an authentication server $\mathcal{AS}$. Below, we give a description of the protocols, see also Fig. 1. In the description, vE_1 and vE_2 refer to the respective packed encryption of type 1 and type 2 in the above presented two packed SHE schemes.

- ***Setup Phase:*** The authentication server $\mathcal{AS}$ generates the public key pk and the secret key sk for the SHE schemes, and distributes only pk to both the client server $\mathcal{C}$ and the computation server $\mathcal{CS}$, while keeping sk secret.
- ***Enrollment Phase:*** The client server $\mathcal{C}$ generates a feature vector A from the client's biometric data (*e.g.* fingerprints), encrypts A into a packed ciphertext $\mathsf{ct}_1(A) = \mathsf{vE}_1(A)$ of the first type, and sends only $\mathsf{ct}_1(A)$ with client's ID to the computation server $\mathcal{CS}$, who then stores ct_1 and ID in $\mathcal{DB}$.
- ***Authentication Phase:*** The client server $\mathcal{C}$ generates a feature vector B from the client's biometric data, encrypts B into a packed ciphertext $\mathsf{ct}_2(B) = \mathsf{vE}_2(B)$ of the second type, and sends $\mathsf{ct}_2(B)$ with the client's ID to the computation server $\mathcal{CS}$. $\mathcal{CS}$ then retrieves the template $\mathsf{ct}_1(A)$ corresponding to ID from $\mathcal{DB}$, computes the Hamming distance from $\mathsf{ct}_1(A)$ and $\mathsf{ct}_2(B)$, following



(a) Enrollment phase

(b) Authentication phase

Fig. 1: The Yasuda *et al.* biometric authentication protocols.

equation(1) (resp. (2)) in the case of the ideal lattice based scheme (resp. in the case of ring-LWE based scheme), and sends only ct_H to $\mathcal{AS}$. The authentication server $\mathcal{AS}$ decrypts ct_H with the secret key sk to obtain the Hamming distance $\mathsf{HD}(A, B)$. Finally, $\mathcal{AS}$ returns the authentication result YES (resp. NO) to $\mathcal{C}$ if $\mathsf{HD}(A, B) \leqslant \tau$ (resp., otherwise), where τ is a pre-defined threshold.

III. The attack

In this section, we present an attack algorithm that can be used by the (malicious) computation server CS to recover the biometric templates of arbitrary users. Before we proceed, let us first discuss our threat model.

A. Threat model

In [17], Simoens *et al.* presented a framework for analysing security and privacy of biometric data in biometric authentication systems. Here, we follow their model and consider *malicious* (or *active*) internal adversaries. A malicious internal adversary $\mathcal{A}$ may deviate from the protocol specifications and may attempt to recover protected information, as long as such attempts are not discovered by the other protocol participants. In a biometric authentication protocol, the relevant protocol entities may pose the following threats to privacy:

- ***Biometric reference privacy:*** The adversary $\mathcal{A}$ should not be able to recover the stored reference biometric template.
- ***Biometric sample privacy:*** The adversary $\mathcal{A}$ should not be able to recover the fresh biometric sample.
- ***Unlinkability:*** The adversary $\mathcal{A}$ should not be able to link a biometric template to the user identity ID.
- ***Intractability or user indistinguishability:*** For a biometric authentication system to be privacy-preserving, the relevant entities must not be able to distinguish whether two authentication attempts are from the same user.

In the protocols under study, the protocol entities know who is trying to authenticate himself to the system. Therefore, from the intractability point of view, both protocols are not privacy-preserving.

Below, we analyse the security of the Yasuda *et al.* [13], [14] protocols against the other threats. Let us briefly discuss what each entity already knows and what can be done if the entity is malicious.

- **The client server $\mathcal{C}$:** $\mathcal{C}$ knows B as well as the identity ID of a user, and communicates with the computation server $\mathcal{CS}$. He can simulate $\mathcal{CS}$. Using B, he can find the stored biometric template A by mounting an attack known as the *center search attack* [17] (see Algorithm 2 below). In this attack, the adversary iteratively changes the fresh biometric template until it is rejected. Then, by changing the new template that is just rejected bit-by-bit, he can recover all bits of the stored template. We must mention here that in [14] Yasuda *et al.* state without giving any proof that their scheme is secure against *hill climbing attacks*, because all data is encrypted. Unfortunately, this is not true. As we have briefly explained here, the *center search attack*, which is a type of *hill climbing attack*, is possible in both protocols. In fact, this attack is possible regardless of the type of employed encryption scheme, as long as the Hamming distance is used to measure the closeness of a fresh biometric template to the reference template.
- **The computation server $\mathcal{CS}$:** $\mathcal{CS}$ knows the ID of a user and has access to the database $\mathcal{DB}$, where the encrypted reference biometric templates are stored. He communicates with the authentication server $\mathcal{AS}$. $\mathcal{CS}$ can simulate the client server $\mathcal{C}$. If he is malicious then his goal could be to recover the stored/fresh biometric templates corresponding to a user ID. If $\mathcal{CS}$ has access to a fresh biometric template corresponding to an ID, then he also can employ the above mentioned *center search attack* to find out the corresponding stored template. However, as we will show below, there is a simple yet powerful attack that can be employed by $\mathcal{CS}$ to recover both the stored and fresh biometric templates of arbitrary users.
- **The authentication server $\mathcal{AS}$:** $\mathcal{AS}$ has the secret key. If $\mathcal{AS}$ is compromised, the protocol's security and privacy are also compromised. A malicious authentication server $\mathcal{AS}$ can indeed intercept the communication from $\mathcal{C}$ to $\mathcal{CS}$, and recover the user ID and the biometric templates.

B. The attack algorithm

Note that in the biometric authentication protocols under study, the computation server $\mathcal{CS}$ computes from $\mathsf{vE}_1(A)$ and $\mathsf{vE}_2(B)$ the Hamming distance (*i.e.* masked version of it) ct_H between A and B following equation (1) in the case of the first SHE scheme and equation (2) in the case of the second SHE scheme, and sends ct_H to the authentication server $\mathcal{AS}$. We mentioned at the end of Section II-C that the decryption of $\mathsf{vE}_1(A)\mathsf{vE}_2(B)$ results in the inner product of A and B. Thus, a closer look at (1) and (2) would suggest that a dishonest $\mathcal{CS}$ can actually cheat in the computation of the Hamming distance to gain knowledge of the biometric templates.

More precisely, suppose that the malicious $\mathcal{CS}$ wants to find out A from $\mathsf{vE}_1(A)$ (a stored biometric template). Then, the malicious $\mathcal{CS}$ computes $\mathsf{vE}_1(A)\mathsf{vE}_2(D)$, where D is a suitably chosen bitstring of the same length as A, and sends the result to $\mathcal{AS}$, instead of sending the actual correct ct_H. For instance, first $\mathcal{CS}$ chooses a bitstring D consisting of τ consecutive 1's followed by all 0's, and computes $\mathsf{vE}_2(D)$. Then, he computes $\mathsf{vE}_1(A)\mathsf{vE}_2(D)$, which corresponds to an encryption of the inner product of A with D, and sends the result to $\mathcal{AS}$. If it gets accepted (i.e., $\mathcal{AS}$ outputs YES), then $\mathcal{CS}$ learns that $\sum_{i=1}^{\tau} A_i \leqslant \tau$ and in the next round, he flips the $\tau+1$-th bit of D and sends $\mathsf{vE}_1(A)\mathsf{vE}_2(D)$ to $\mathcal{AS}$. If it gets accepted, then $\mathcal{CS}$ flips the $\tau+2$-th bit of D and sends $\mathsf{vE}_1(A)\mathsf{vE}_2(D)$. This continues until $\mathcal{AS}$ responds NO. (This is the first loop of Algorithm 1). In this way, $\mathcal{CS}$ will recover a portion of A that has Hamming weight τ, and then bit-by-bit he can recover all bits of A; see the second and the third loops of Algorithm 1. In the decryption of Algorithm 1, the notation β_i, for $\beta \in \{0,1\}$, means that the i-th bit is β.

Algorithm 1 The attack algorithm

```
**Input:** $\mathsf{vE}_1(A)$ (an encrypted reference template)
**Output:** $A = A_1, \cdots, A_N$ (reference template)
**Initialise:** $A = 0_1 0_2 \cdots 0_N$
**For** $i = 0$ to $N - \tau$:
    Set $D = 1_1 \cdots 1_{\tau+i} 0_{\tau+i+1} \cdots 0_N$
    Compute $\mathsf{vE}_2(D)$
    Send $\mathsf{ct} = \mathsf{vE}_1(A)\mathsf{vE}_2(D)$ to $\mathcal{AS}$
    **If** rejected **Then**
        break
    **If** $i = N$ **Then**
        **Return centerSearch**$(A)$
Set $i' = \tau + i$
Set $A_{i'} = 1$
**For** $i = 1$ to $i' - 1$:
    Set $D = 1_1 \cdots 1_{i-1} 0_i 1_{i+1} \cdots 1_{i'} 0_{i'+1} \cdots 0_N$
    Compute $\mathsf{vE}_2(D)$
    Send $\mathsf{vE}_1(A)\mathsf{vE}_2(D)$ to $\mathcal{AS}$
    **If** accepted **Then**
        $A_i = 1$
**For** $i = i' + 1$ to $N$:
    Set $D = 1_1 \cdots 1_{i'-1} 0_{i'} \cdots 0_{i-1} 1_i 0_{i+1} \cdots 0_N$
    Compute $\mathsf{vE}_2(D)$
    Send $\mathsf{vE}_1(A)\mathsf{vE}_2(D)$ to $\mathcal{AS}$
    **If** rejected **Then**
        $A_i = 1$
**Return** $A$
```

By employing this algorithm, the malicious $\mathcal{CS}$ gains full knowledge of A in at most $2N-\tau$ trials, which can be seen as the sum of the total number of iterations in the three loops of the algorithm. We observe that if the first loop ends without being rejected, then this means that the reference template itself has a Hamming weight less than or equal to τ and thus, the adversary can mount the *center search attack* [17] using a bitstring of all zeros. That is why we have **centerSearch**(A) as a subroutine in the algorithm. In Algorithm 2, we describe the algorithm for the *center search attack*.

By using the same algorithm with $\mathsf{vE}_2(B)$ as an input and by changing $\mathsf{vE}_2(D)$ to $\mathsf{vE}_1(D)$ in the algorithm, $\mathcal{CS}$ can recover the fresh biometric template B of any user.

Algorithm 2 The center search attack

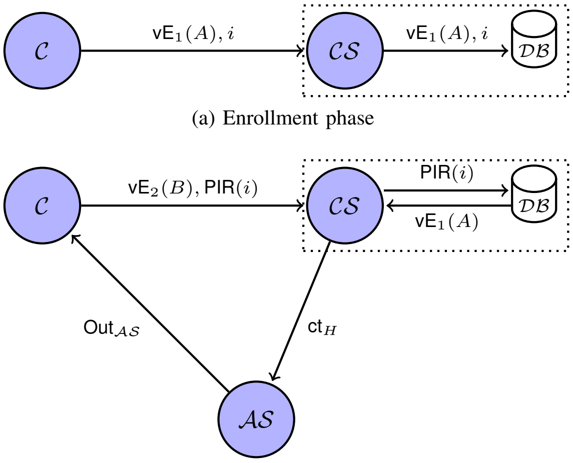
```
**Input:** $B = B_1, \cdots, B_N$ (a fresh but matching template)
**Output:** $A = A_1, \cdots, A_N$ (reference template)
**For** $i = 1$ to $N$:
    Set $D = \overline{B_1}, \ldots, \overline{B_i}, B_{i+1}, \ldots, B_N$
    Send $\mathsf{vE}_2(D)$ to $\mathcal{CS}$
    **If** rejected **Then**
        break
**For** $i = 1$ to $N$:
    Send $\mathsf{vE}_2(D_1, \ldots, \overline{D_i}, D_{i+1}, \ldots, D_N)$ to $\mathcal{CS}$
    **If** accepted **Then**
        $A_i = B_i$
    **Else**
        $A_i = \overline{B_i}$
**Return** $A$
```

As we can see, the attack is rather simple and straightforward, but powerful. Yasuda *et al.* [13], [14] claim that as long as $\mathcal{AS}$ manages the secret key sk (for the respective SHE schemes) properly, their protocols would be secure, without any threat model or proof. However, our attack is possible even if sk is securely managed, since $\mathcal{CS}$ can deviate from the protocol specifications without being detected. More specifically, instead of performing the computation (that is, (1) or (2)) that would result in the correct ct_H (a ciphertext encrypting the Hamming distance between the fresh and the stored templates), $\mathcal{CS}$ can replace it with a different ciphertext encrypting a the inner product of the target biometric template with a carefully chosen fake template. This new ciphertext allows $\mathcal{CS}$ to gain knowledge of the biometric template using $\mathcal{AS}$'s output. By repeating this procedure at most $2N-\tau$ times (or in at most $2N-\tau$ authentication attempts), $\mathcal{CS}$ can gain full knowledge of a biometric template. Note that in the attack $\mathcal{CS}$ only computes one encryption (of the fake template) and one homomorphic product, so the attack complexity is *low*.

It may seem obvious that $\mathcal{AS}$ might easily detect this attack due to the structure of the consecutive inputs. However, detecting such inputs is not always feasible if the attacker alternates between malicious and valid authentication attempts. Moreover, the structure in the inputs (to $\mathcal{AS}$) may be masked. A viable solution is to verify that $\mathcal{CS}$ performed the computations honestly by following the protocol specifications. Thus, the main question that needs to be answered is: *How exactly does $\mathcal{AS}$ verify that $\mathcal{CS}$ computed ct_H honestly?* We discuss this in the next section.

IV. Countermeasures

As mentioned earlier, the user ID is not protected in Yasuda *et al.* protocols, and this is not a desirable property from intractability point of view. Hence, we propose a slight modification to the protocols, so that during the enrolment $\mathcal{C}$ maps ID to an index and provides it along with the encrypted reference template to $\mathcal{CS}$ for storage. In the authentication phase, $\mathcal{C}$ provides a fresh encrypted template and a (possibly) encrypted query to be used by $\mathcal{CS}$ to privately retrieve the reference template from its database. In other words, a private information retrieval (PIR) [18], [19] technique is employed to retrieve the stored biometric template, and the client server $\mathcal{C}$ provides the PIR query. The rest of the protocols could remain the same, see Fig. 2. In this case, $\mathcal{CS}$ can still employ our attack to recover the templates, but he does not know to whom the templates belong.



(a) Enrollment phase

(b) Authentication phase

Fig. 2: Schematics of a possible modification to the Yasuda *et al.* biometric authentication protocols. Now, the computation server $\mathcal{CS}$ does not know the user ID.

In order to prevent the presented attack, we need something more than this. Intuitively, it is obvious that we need a (cryptographic) solution that allows $\mathcal{AS}$ to verify that $\mathcal{CS}$ performed the computations honestly. To be precise, if $\mathcal{AS}$ could check that the input (ct_H) he received from $\mathcal{CS}$ is the correct ciphertext resulting from equation (1) (or (2)), then the presented attack could be prevented. Since, in this case, $\mathcal{AS}$ would detect the suspicious input provided by $\mathcal{CS}$. However, such verifications may not be possible without introducing some new primitives to the protocol. Indeed, there are cryptographic schemes that allow delegation of computation in a verifiable manner so that the client can verify the correctness of some computations done by a cloud server [20]–[23]. Incorporating such schemes for verifiable delegation of computation seems to be a promising direction for improving the Yasuda *et al.* protocols. Furthermore, schemes for signatures of correct computation such as the one proposed by Papamanthou *et al.* [24] would also enable us to combat the described attack. Below we describe a modification to the Yasuda *et al.* protocols using the Papamanthou *et al.* [24] scheme for signatures of correct computation.

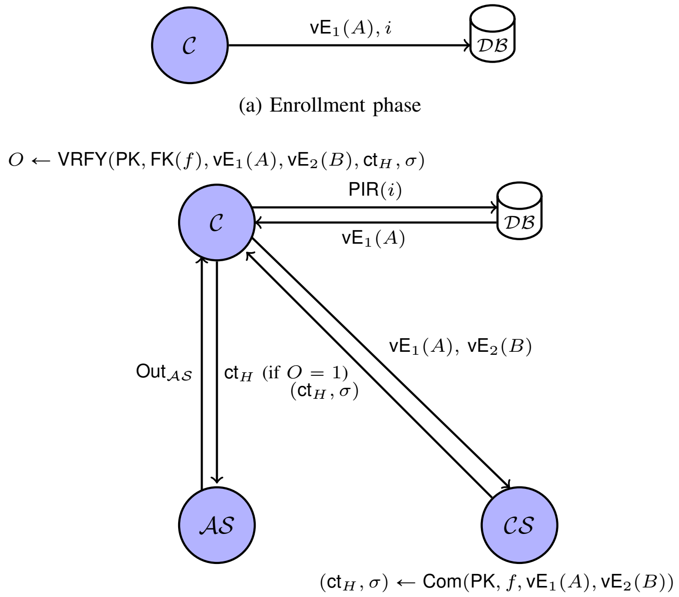
Signatures of correct computation (SCC) for a function family $\mathcal{F}$ consists of the following four algorithms $(\mathsf{KeyGen}, \mathsf{Setup}, \mathsf{Com}, \mathsf{VRFY})$ [24]. The key generation algorithm KeyGen takes a security parameter and a function family $\mathcal{F}$ as input, and outputs a public/secret key pair $(\mathsf{PK}, \mathsf{SK})$. The setup algorithm Setup takes the secret key SK, the public key PK and a function $f \in \mathcal{F}$ as input, and outputs the function public key $\mathsf{FK}(f)$ for the function f. These two algorithms are run by a trusted party. In our case, the authentication server $\mathcal{AS}$ can run these algorithms, since $\mathcal{AS}$ is regarded as the trusted party who manages the keys for the SHE schemes. The computation algorithm Com takes the public key PK, a function $f \in \mathcal{F}$ and a value $v \in \mathsf{domain}(f)$ as input, and outputs a pair (ω, σ), where $\omega = f(v)$ and σ is a signature. The verification algorithm VRFY takes the public key PK, the function public key $\mathsf{FK}(f)$ for the function f, a value $v \in \mathsf{domain}(f)$, a claimed result ω and a signature σ as input, and outputs 1 if ω is correct, otherwise outputs 0.

We note that in [24], there is an algorithm Update that is used to update the function public key for a new function. We omit it here, because in the Yasuda *et al.* protocols, the function to be computed does not change.

Hence, one way to employ signatures of correct computation in the Yasuda *et al.* protocols would be requiring $\mathcal{CS}$ to send his computation result ct_H with a signature back to $\mathcal{C}$, instead of sending ct_H to $\mathcal{AS}$. Then, $\mathcal{C}$ would verify the correctness of ct_H, and if the verification succeeds, he would send ct_H to $\mathcal{AS}$; otherwise, $\mathcal{C}$ would abort the protocol. However, in order to guarantee that this shall work, the inputs to the function computed by $\mathcal{CS}$ should be provided by the client server $\mathcal{C}$, that is, both $\mathsf{vE}_1(A)$ and $\mathsf{vE}_2(B)$ must be provided to the computation server $\mathcal{CS}$ by $\mathcal{C}$. However, in the Yasuda *et al.* protocols, $\mathsf{vE}_1(A)$ is stored on the computation server's side. We should note here, that the database $\mathcal{DB}$ is controlled by the computation server $\mathcal{CS}$. Therefore, we propose to split the database $\mathcal{DB}$ and the computation server $\mathcal{CS}$ and consider them as two separate entities. With this change, we can have the following modifications to the original Yasuda *et al.* protocols.

- ***Setup Phase:*** As in the original protocols, the authentication server $\mathcal{AS}$ generates the public key pk and the secret key sk for the SHE schemes, and distributes only pk to the client server $\mathcal{C}$ and the computation server $\mathcal{CS}$. $\mathcal{AS}$ also generates the public/secret $(\mathsf{PK}, \mathsf{SK})$ for the SCC scheme and the function public key $\mathsf{FK}(f)$ for the function f (running the algorithms KeyGen and Setup), which takes two inputs and is defined as in (1) (and/or (2)). He then distributes $(\mathsf{PK}, \mathsf{FK}(f))$ to $\mathcal{C}$ and PK to $\mathcal{CS}$.
- ***Enrollment Phase:*** The client server $\mathcal{C}$ generates a feature vector A from the client's biometric data, encrypts A into a packed ciphertext $\mathsf{ct}_1(A) = \mathsf{vE}_1(A)$ of the first type and maps ID to an index $i, 1 \leqslant i \leqslant N$, and sends only $\mathsf{ct}_1(A)$ with i to the database $\mathcal{DB}$ for storage. See Fig. 3(a).
- ***Authentication Phase:***
  – Phase 1 - Communication $\mathcal{C} \rightarrow \mathcal{DB}$: The client server $\mathcal{C}$ takes the client ID and extracts the corresponding index i. Then, $\mathcal{C}$ privately retrieves the reference template $\mathsf{ct}_1(A) = \mathsf{vE}_1(A)$ corresponding to the index i from the database $\mathcal{DB}$. See Fig. 3(b) for a schematic description.
  – Phase 2 - Communication $\mathcal{C} \rightarrow \mathcal{CS}$: The client server $\mathcal{C}$ generates a feature vector B from the client's biometric data, encrypts B into a packed ciphertext $\mathsf{ct}_2 = \mathsf{vE}_2(B)$ of the second type, and sends $\mathsf{ct}_1(A)$ and $\mathsf{ct}_2(B)$ to the computation server $\mathcal{CS}$. Upon receiving $\mathsf{ct}_1(A)$ and $\mathsf{ct}_2(B)$, $\mathcal{CS}$ runs the computation algorithm $\mathsf{Com}(\mathsf{PK}, f, \mathsf{ct}_1(A), \mathsf{ct}_2(B))$, and returns the output, which is ct_H and a signature σ, to $\mathcal{C}$.
  – Phase 3 - Communication $\mathcal{C} \rightarrow \mathcal{AS}$: The client server $\mathcal{C}$ first checks the correctness

of ct_H, by running the verification algorithm $\mathsf{VRFY}(\mathsf{PK}, \mathsf{FK}(f), \mathsf{ct}_1(A), \mathsf{ct}_2(B), \mathsf{ct}_H, \sigma)$. If the verification algorithm outputs 1, then $\mathcal{C}$ forwards ct_H to the authentication server $\mathcal{AS}$, otherwise, he aborts the protocol. The authentication server $\mathcal{AS}$ decrypts ct_H with the secret key sk to obtain the Hamming distance $\mathsf{HD}(A, B)$. Finally, $\mathcal{AS}$ returns the authentication result YES (resp. NO) to $\mathcal{C}$ if $\mathsf{HD}(A, B) \leqslant \tau$ (resp., otherwise).


$\mathcal{C}$
$\mathsf{vE}_1(A), i$
$\mathcal{DB}$
$O \leftarrow \mathsf{VRFY}(\mathsf{PK}, \mathsf{FK}(f), \mathsf{vE}_1(A), \mathsf{vE}_2(B), \mathsf{ct}_H, \sigma)$
$\mathcal{C}$
$\mathsf{PIR}(i)$
$\mathcal{DB}$
$\mathsf{vE}_1(A)$
$\mathsf{vE}_1(A),\ \mathsf{vE}_2(B)$
$\mathsf{Out}_{\mathcal{AS}}$
ct_H (if $O = 1$)
(ct_H, σ)
$\mathcal{AS}$
$\mathcal{CS}$
$(\mathsf{ct}_H, \sigma) \leftarrow \mathsf{Com}(\mathsf{PK}, f, \mathsf{vE}_1(A), \mathsf{vE}_2(B))$

(a) Enrollment phase

(b) Authentication phase

Fig. 3: Schematics of a proposed modification to the Yasuda *et al.* biometric authentication protocols.

V. Conclusions and Future Work

In this paper, two recently proposed privacy-preserving biometric authentication protocols that employ packed somewhat homomorphic encryption (SHE) based on ideal lattices and ring-LWE are reviewed. We presented a simple yet powerful attack that enables a malicious computation server to gain full knowledge of the stored and the fresh biometric templates of arbitrary users. The attack takes advantage of an homomorphic property of the employed encryption scheme, namely, the secure inner product computation. Another key factor that allows the attack to be launched undetected is the lack of a verification that the computation server performed the correct computation.

Furthermore, we discussed possible countermeasures. As a quick fix, we first proposed the use of private information retrieval in order to avoid storing the user ID in the database. This way, one can reduce the impact of the attack, since the adversary will not be able to link the recovered biometric template to a user identity. In order to prevent the attack, we discussed how a scheme for signatures of correct computation may offer secure solutions to the problem. In particular, we proposed to separate the database from the computation server in order to incorporate the scheme for signatures of correct computation, and changed the flow of information between the protocol entities. We would like to further investigate these areas in the future, and improve the protocol to achieve provable security and full privacy. For now, it is safe to say that SHE alone is not enough to build a fully privacy-preserving biometric authentication protocol. Great care must be taken when designing such a protocol based solely on SHE.

References

[1] J. Bringer, H. Chabanne, G. D. Cohen, B. Kindarji, and G. Zémor, "Optimal iris fuzzy sketches," *CoRR*, vol. abs/0705.3740, 2007.

[2] J. Bringer, H. Chabanne, M. Izabachène, D. Pointcheval, Q. Tang, and S. Zimmer, "An application of the Goldwasser-Micali cryptosystem to biometric authentication," in *ACISP*, ser. LNCS. Springer Verlag, 2007, pp. 96–106.

[3] A. Stoianov, "Security issues of biometric encryption," in *IEEE International Conference on Science and Technology for Humanity*, 2009, pp. 34–39.

[4] ——, "Cryptographically secure biometrics," *SPIE 7667, Biometric Technology for Human Identification VII*, pp. 76 670C–12, 2010.

[5] M. Barbosa, T. Brouard, S. Cauchie, and S. M. de Sousa, "Secure biometric authentication with improved accuracy." in *ACISP*, ser. LNCS, vol. 5107. Springer, 2008, pp. 21–36.

[6] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva, "Privacy-preserving fingercode authentication," in *Proceedings of the 12th ACM workshop on Multimedia and security*, 2010, pp. 231–240.

[7] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," in *NDSS*, 2011.

[8] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *PETs*, 2009, pp. 235–253.

[9] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *ICISC*, ser. LNCS, 2009, pp. 229–244.

[10] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*. Academic Press, 1978, pp. 165–179.

[11] M. O. Rabin, "How to exchange secrets with oblivious transfer," Aiken Computation Lab, Harvard University, Tech. Rep. TR-81, 1981.

[12] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[13] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, "Packed homomorphic encryption based on ideal lattices and its application to biometrics," in *Security Engineering and Intelligence Inf.*, ser. LNCS, vol. 8128, 2013, pp. 55–74.

[14] ——, "Practical packing method in somewhat homomorphic encryption," in *DPM/SETOP*, ser. LNCS, vol. 8147, 2013, pp. 34–50.

[15] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009, pp. 169–178.

[16] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *EUROCRYPT*, ser. LNCS, 2011, vol. 6632, pp. 129–148.

[17] K. Simoens *et al.*, "A framework for analyzing template security and privacy in biometric authentication systems," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 833–841, 2012.

[18] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[19] R. Ostrovsky and I. Willian E. Skeith, "A survey of single-database private information retrieval: techniques and applications," in *PKC*, ser. LNCS. Springer Verlag, 2007, pp. 393–411.

[20] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *CRYPTO*, ser. LNCS, vol. 6841, 2011, pp. 111–131.

[21] K.-M. Chung, Y. T. Kalai, and S. P. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *CRYPTO*, ser. LNCS, 2010, pp. 483–501.

[22] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *CRYPTO*, ser. LNCS, 2010, pp. 465–482.

[23] L. Zhang *et al.*, "Verifiable private multi-party computation: Ranging and ranking," in *INFOCOM*, 2013, pp. 605–609.

[24] C. Papamanthou, E. Shi, and R. Tamassia, "Signatures of correct computation," in *TCC*, ser. LNCS, 2013, pp. 222–242.