Grouping-Proof-Distance-Bounding Protocols: Keep All Your Friends Close

Christoffer Karlsson and Aikaterini Mitrokotsa

***Abstract*— The use of wireless communications has had tremendous expansion and has led to the development of wearable devices with limited resources. Often, to gain access to services/places, proving the physical proximity of a single device, may not be enough. Multiple wearable devices linked to function as a team may provide stronger guarantees on accurate authentication. Although distance-bounding (DB) protocols provide a reliable way to prove the physical proximity of a device and grouping-proof (GP) protocols can be employed to prove the presence of multiple provers, proving that multiple devices are present and in close proximity to a verifier is more challenging. In this letter, we introduce a new concept that extends traditional DB protocols to a multi-prover setting and we propose the first GPDB protocol that provides not only a proof of the presence of multiple provers at the same time but also assurance regarding the physical proximity of the provers, requiring limited computational effort. Furthermore, we discuss the effectiveness of this protocol, considering the main threats in DB and GP protocols.**

***Index Terms*— Grouping-proof protocols, distance-bounding, wearable devices.**

I. Introduction

WIRELESS communication technologies, have a transformative impact in our life and are increasingly deployed in a wide range of applications (e.g., supply chain management, NFC/WiFi payments, e-healthcare). Often, authentication is based on proximity which is guaranteed through the perceived operating range of the authenticating party (i.e., signal attenuation). This is the case for many access control applications (e.g., travel-cards, keyless automobile access). Relying on just one (wearable) device might not be enough to achieve accurate authentication. Multi-factor authentication, where multiple wearable devices are linked to function as a "team" can provide stronger authentication guarantees.

In many cases, in order to increase the security of the authentication process, we may need to employ both proximity-based authentication and multi-factor authentication, i.e., prove that multiple devices (provers) are present and close to the authentication access point (verifier). For instance, this might be the case in a multi-factor proximity-based authentication system: the prover may need to carry multiple devices as well as prove physical proximity to the access point. In access control for toll collection, a tag could be used to identify a car and another to identify the driver and both of these should be present and near the reader that grants access.

Manuscript received March 27, 2016; accepted April 26, 2016. Date of publication May 4, 2016; date of current version July 8, 2016. This work was partially supported by the SNF grant "SwissSenseSynergy" (CRSII2 154458) and from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no 608743 VR grant PRECIS and the STINT grant Secure, Private & Efficient Healthcare with wearable computing. The associate editor coordinating the review of this letter and approving it for publication was M. F. Flanagan.

The authors are with the Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg 4129, Sweden (e-mail: chrika@student.chalmers.se; aikmitr@chalmers.se).

Digital Object Identifier 10.1109/LCOMM.2016.2562618

Relying on the perceived operating range to provide a secure proximity proof is a significant security vulnerability. An adversary (*man-in-the-middle*) can perform a simple relay attack to circumvent this assumption on proximity. One of the main countermeasures against relay attacks are *distance-bounding* (DB) protocols [1]. DB protocols are challenge-response authentication protocols that rely on the round-trip-time of the transmitted messages to bound the distance between a trusted verifier (e.g., access point) and an untrusted prover (e.g., an RFID tag). *Grouping-proof* (GP) protocols can be employed to guarantee that multiple devices are present at the same time [2]. Although multi-factor authentication has been considered before, combining multi-factor authentication with proximity-based authentication is an unexplored area.

In this letter, we introduce a new concept that extends traditional DB protocols with one prover to a setting of multiple provers and we propose the first *grouping-proof-distance-bounding* (GPDB) protocol that can be employed for authentication and provides not only a proof of presence of multiple provers at the same time but also assurance regarding the physical proximity of the provers. Our protocol relies on the recently proposed by Hermans *et al.* [3] Elliptic Curve Cryptography (ECC)-based DB protocol which resists the main threats against GP protocols and DB protocols and, requires very limited computational effort (four scalar-EC point multiplications).

II. Background and Problem Statement

A. Preliminaries

We consider elliptic curves $\mathbb{E}$ and subgroups $\mathbb{G}_\ell$ of points on $\mathbb{E}$ of prime order ℓ over $\mathbb{F}_p$, generated by a point P. We denote the scalar multiplication of the point P by a scalar $a \in \mathbb{Z}_\ell^*$ by aP. We consider a public key encryption scheme that uses a pair of keys $(\mathsf{sk}, \mathsf{pk})$ such that $\mathsf{sk} \in_R \mathbb{Z}_\ell^*$ and $\mathsf{pk} = \mathsf{sk} \cdot P \in \mathbb{G}_\ell$. We denote by O the point at infinity of $\mathbb{E}$. We use the ECDSA conversion function [4] and we denote it as $\mathtt{xcoord}(\cdot)$, which returns the x-coordinate of a point. More precisely, for a point $Q = \{q_x, q_y\} \in \mathbb{G}_\ell$, where $q_x, q_y \in [0, \ldots, p-1]$, then $\mathtt{xcoord}(Q)$ maps Q to $q_x \pmod{\ell}$. We note that $\mathtt{xcoord}(O) = 0$. Below we describe the main assumptions on which our security analysis relies.

1) Discrete Logarithm (DL): Let Q be a given, arbitrarily chosen element in $\mathbb{G}_\ell$. The DL problem is to find the unique integer $k \in \mathbb{Z}_\ell^*$ such that $Q = kP$ where P is the base of the group. Solving the DL problem is computationally hard.

2) x-Logarithm (XL): The x-logarithm problem is given an elliptic curve point, to determine whether its discrete logarithm is congruent to the x-coordinate of an elliptic curve point [3], [4]. Solving the XL problem is computationally hard.

3) Decisional Diffie Hellman (DDH): Let $aP, bP \in \mathbb{G}_\ell$, where $a, b \in \mathbb{Z}_\ell^*$. The tuple $\langle P, aP, bp, abP\rangle$ is called a Diffie Hellman (DH) tuple. Given $cP \in \mathbb{G}_\ell$, the DDH problem is to determine if $\langle P, aP, bP, cP\rangle$ is a valid DH tuple or not. Solving the DDH problem is computationally hard.

B. Distance-Bounding Protocols

DB protocols, initially introduced to combat relay attacks in ATM systems [5], are challenge-response protocols that based on the round-trip-time of the exchanged messages can determine an upper bound on the physical distance between a verifier ($\mathcal{V}$) (e.g., an RFID reader) and an untrusted prover ($\mathcal{P}$). Numerous DB protocols have followed [6], [7], and [3]. The basic objective of a DB protocol is to protect against the following threats: *(i) Distance fraud:* In this fraud, a dishonest prover $\mathcal{P}$ tries to prove that it is located closer to $\mathcal{V}$, than it really is; *(ii) Mafia fraud:* An adversary $\mathcal{A}$, located close to $\mathcal{V}$ acts as a *man-in-the-middle*, between $\mathcal{V}$ and a far away $\mathcal{P}$. $\mathcal{A}$ tries 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 $\mathcal{V}$ communicate with $\mathcal{A}$; *(iii) Terrorist fraud:* This attack involves a verifier $\mathcal{V}$, a prover $\mathcal{P}$ located far away from $\mathcal{V}$ and a *man-in-the-middle* attacker $\mathcal{A}$. The goal of $\mathcal{A}$ is to shorten the distance between $\mathcal{P}$ and $\mathcal{V}$. In this case, $\mathcal{P}$ helps $\mathcal{A}$, but $\mathcal{A}$ should not be able get authenticated later on without the help of $\mathcal{P}$.

Most of the existing DB and in general authentication protocols for resource-constrained devices rely on secret key cryptography mainly due to the false belief that public key cryptography is too expensive. However, it has been shown [8], that public key cryptography and especially ECC can be implemented in resource-constrained devices. When authentication protocols rely on secret key cryptography, then a possible compromise of the prover's secret key may lead to traceability attacks. Recently, Hermans *et al.* [3] proposed a DB protocol that relies on public key cryptography and is resistant against the main two threats for DB protocols. Furthermore, it is "wide-forward-insider" private without key updates with a small computational cost. The notion of "wide-forward-insider" covers cases where an adversary uses the internal state from a corrupted prover to attack the privacy of other provers. Key updates are very costly for resource-constrained devices and may also be employed to launch de-syncronization attacks; thus, they should be avoided.

One interesting divergence from the two-party DB approach is performing DB with multiple parties [9] (i.e., multiple provers and verifiers). This group DB verifies that all the parties are in close proximity. However, in the special case of multiple provers interacting with one verifier what is vaguely proposed is that one party performs mutual DB with each of the remaining parties, while the last message of the interaction with one party P_k is used as the first challenge of the interaction with P_{k+1}. However, no analysis related to the generation of fake group-proofs by the untrusted provers is provided and no connections with the threats against GP protocols are identified. Our proposal, allows for a verifier to verify that multiple provers are in close proximity without requiring communication of the provers among themselves, while providing strong security and privacy guarantees against the main threats in DB and GP protocols.

C. Grouping-Proof Protocols

Grouping-proof (GP) protocols were initially introduced by Juels [2] to provide evidence that two RFID tags have been scanned simultaneously. Later on, this solution has been generalised to a group of tags and by GP protocols. Below we list the main security and privacy risks for GP protocols.

1) Replay Attack: In this attack the transcripts of previous runs of a GP protocol are used, to generate a valid proof.

2) Subset Replay Attack: In this case, $\mathcal{A}$ attempts to generate a fake proof that links a subset of simultaneously read legitimate provers to any other legitimate prover.

3) Multiple Impersonation Attack: This attack mainly targets GP protocols where the provers are divided into groups [10]. In this case, $\mathcal{A}$ãttempts to generate a fake proof of presence of two provers belonging to the same group.

4) Anonymity and Traceability Attacks: In this case, $\mathcal{A}$ attempts to identify/distinguish a prover $\mathcal{P}$ and subsequently this could be used to trace $\mathcal{P}$. $\mathcal{A}$ relies mainly on analysing the transcripts of previous runs of the GP protocol or even after $\mathcal{P}$ has been compromised (i.e., recovering its secret key).

5) Denial of Service: In this case, $\mathcal{A}$ attempts to render $\mathcal{P}$ in a state from which it can no longer function properly. This attack can be successfully performed against provers that have to be synchronised with $\mathcal{V}$. For instance, this is the case when keys are updated in each run of the protocol.

6) Denial of Proof (DoP): In this case $\mathcal{A}$ attempts to generate an invalid grouping-proof by employing some illegitimate provers among legitimate ones. We demote by m-DoP, a DoP attack launched by m malicious provers.

III. The Grouping-Proof-Distance-Bounding Protocol

All provers have a private, public key pair (x, $X = xP$, where $X \in \mathbb{G}_\ell$ and $x \in \mathbb{Z}_\ell^*$), while the provers' public keys are registered in $\mathcal{V}$'s database. Correspondingly, each $\mathcal{V}$ has a private/public key pair (y, $Y = yP$, where $Y \in \mathbb{G}_\ell$ and $y \in \mathbb{Z}_\ell^*$), while the public key Y is known to all provers.

The protocol is divided into two stages. In the first stage, the verifier $\mathcal{V}$ communicates with the first prover $\mathcal{P}_1$ (Fig. 1) and subsequently with the provers $\mathcal{P}_j$ for $j = \{2, \ldots, m\}$ (Fig. 2). Each stage is composed of the three following phases.

Initialization Phase: $\mathcal{V}$ generates a timestamp T_S and encrypts it getting the value ts. The value ts is used to allow $\mathcal{V}$ validate when the grouping-proof starts and ends. It chooses $r_\mathcal{V} \in_R \mathbb{Z}_\ell^*$ and computes $R_\mathcal{V} = r_\mathcal{V} P$. $\mathcal{V}$ sends ts to $\mathcal{P}_1$ and $\mathcal{P}_1$ generates $r_{1,1}, r_{1,2} \in_R \mathbb{Z}_\ell^*$ computes the $R_{1,1}$ and $R_{1,2}$ and transmits them to $\mathcal{V}$. Then, $\mathcal{V}$ checks that $R_{1,1}, R_{1,2} \neq O$, and consequently that $r_{1,1}, r_{1,2} \neq 0$, and, transmits $R_\mathcal{V}$ and Y to $\mathcal{P}_1$. Both $\mathcal{P}_1$ and $\mathcal{V}$ generate $a = [\texttt{xcoord}(r_u R_{1,1})]_{2n} = [\texttt{xcoord}(r_{1,1} R_u)]_{2n}$. Finally, $\mathcal{V}$ selects randomly $e \in_R \mathbb{Z}_\ell^*$ and takes the first n bits of it $c = [e]_n$.

Distance-Bounding Phase: In this phase, $\mathcal{V}$ starts the clock and sends to $\mathcal{P}_1$ at each round $i = \{1 \ldots, n\}$ one bit c_i. $\mathcal{P}_1$ computes the response $r_i = (a^0)_i$ if $c_i = 0$ and $r_i = (a^1)_i \oplus ([x_1]_n)_i$ and sends it to $\mathcal{V}$; the latter stores all the received responses r_i as well as the time Δt_i that corresponds on the time between sending c_i and receiving r_i.

Verification Phase: In this phase, $\mathcal{V}$ checks the correctness of the received r_i's and that it holds

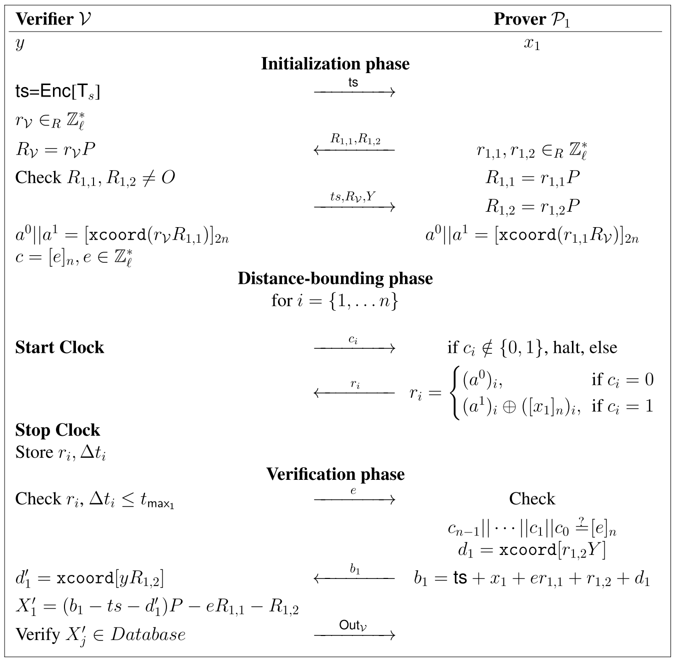
Verifier $\mathcal{V}$
Prover $\mathcal{P}_1$
y
x_1
Initialization phase
ts=Enc[T$_s$]
$\xrightarrow{\text{ts}}$
$r_{\mathcal{V}} \in_R \mathbb{Z}_\ell^*$
$R_{\mathcal{V}} = r_{\mathcal{V}}P$
$\xleftarrow{R_{1,1},R_{1,2}}$
$r_{1,1}, r_{1,2} \in_R \mathbb{Z}_\ell^*$
Check $R_{1,1}, R_{1,2} \neq O$
$R_{1,1} = r_{1,1}P$
$\xrightarrow{ts,R_{\mathcal{V}},Y}$
$R_{1,2} = r_{1,2}P$
$a^0||a^1 = [\texttt{xcoord}(r_{\mathcal{V}}R_{1,1})]_{2n}$
$a^0||a^1 = [\texttt{xcoord}(r_{1,1}R_{\mathcal{V}})]_{2n}$
$c = [e]_n, e \in \mathbb{Z}_\ell^*$
Distance-bounding phase
for $i = \{1, \dots n\}$
Start Clock
$\xrightarrow{c_i}$
if $c_i \notin \{0,1\}$, halt, else
$\xleftarrow{r_i}$
$r_i = \begin{cases} (a^0)_i, & \text{if } c_i = 0 \\ (a^1)_i \oplus ([x_1]_n)_i, & \text{if } c_i = 1 \end{cases}$
Stop Clock
Store $r_i, \Delta t_i$
Verification phase
Check $r_i, \Delta t_i \leq t_{\max_1}$
$\xrightarrow{e}$
Check
$c_{n-1}|| \cdots ||c_1||c_0 \stackrel{?}{=} [e]_n$
$d_1 = \texttt{xcoord}[r_{1,2}Y]$
$d_1' = \texttt{xcoord}[yR_{1,2}]$
$\xleftarrow{b_1}$
$b_1 = \text{ts} + x_1 + er_{1,1} + r_{1,2} + d_1$
$X_1' = (b_1 - ts - d_1')P - eR_{1,1} - R_{1,2}$
Verify $X_j' \in Database$
$\xrightarrow{\text{Out}_{\mathcal{V}}}$

Fig. 1. The GPDB protocol for the first prover $\mathcal{P}_1$.

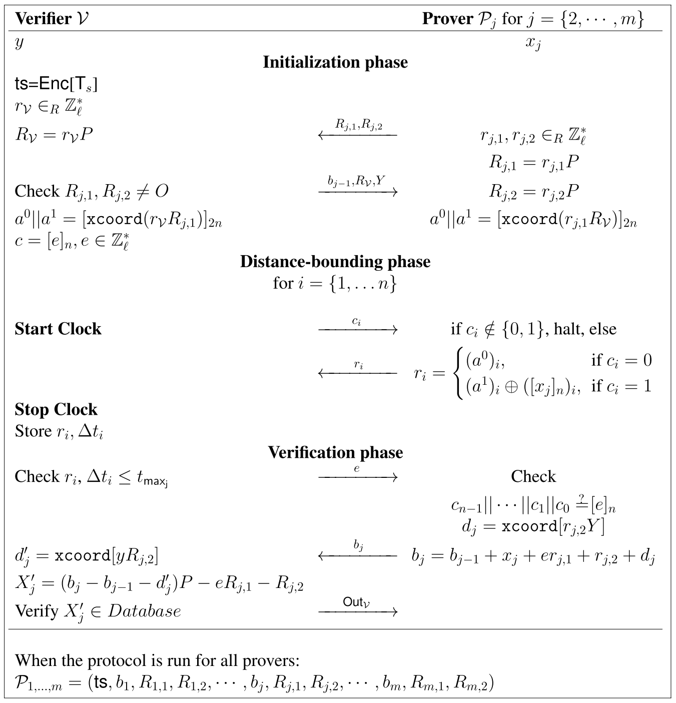
Verifier $\mathcal{V}$
Prover $\mathcal{P}_j$ for $j = \{2, \cdots, m\}$
y
x_j
Initialization phase
ts=Enc[T$_s$]
$r_{\mathcal{V}} \in_R \mathbb{Z}_\ell^*$
$R_{\mathcal{V}} = r_{\mathcal{V}}P$
$\xleftarrow{R_{j,1},R_{j,2}}$
$r_{j,1}, r_{j,2} \in_R \mathbb{Z}_\ell^*$
$R_{j,1} = r_{j,1}P$
Check $R_{j,1}, R_{j,2} \neq O$
$\xrightarrow{b_{j-1},R_{\mathcal{V}},Y}$
$R_{j,2} = r_{j,2}P$
$a^0||a^1 = [\texttt{xcoord}(r_{\mathcal{V}}R_{j,1})]_{2n}$
$a^0||a^1 = [\texttt{xcoord}(r_{j,1}R_{\mathcal{V}})]_{2n}$
$c = [e]_n, e \in \mathbb{Z}_\ell^*$
Distance-bounding phase
for $i = \{1, \dots n\}$
Start Clock
$\xrightarrow{c_i}$
if $c_i \notin \{0,1\}$, halt, else
$\xleftarrow{r_i}$
$r_i = \begin{cases} (a^0)_i, & \text{if } c_i = 0 \\ (a^1)_i \oplus ([x_j]_n)_i, & \text{if } c_i = 1 \end{cases}$
Stop Clock
Store $r_i, \Delta t_i$
Verification phase
Check $r_i, \Delta t_i \leq t_{\max_j}$
$\xrightarrow{e}$
Check
$c_{n-1}|| \cdots ||c_1||c_0 \stackrel{?}{=} [e]_n$
$d_j = \texttt{xcoord}[r_{j,2}Y]$
$d_j' = \texttt{xcoord}[yR_{j,2}]$
$\xleftarrow{b_j}$
$b_j = b_{j-1} + x_j + er_{j,1} + r_{j,2} + d_j$
$X_j' = (b_j - b_{j-1} - d_j')P - eR_{j,1} - R_{j,2}$
Verify $X_j' \in Database$
$\xrightarrow{\text{Out}_{\mathcal{V}}}$
When the protocol is run for all provers:
$\mathcal{P}_{1,\dots,m} = (\text{ts}, b_1, R_{1,1}, R_{1,2}, \cdots, b_j, R_{j,1}, R_{j,2}, \cdots, b_m, R_{m,1}, R_{m,2})$

Fig. 2. The GPDB protocol for the prover $\mathcal{P}_j$; $j = \{2, \dots, m\}$.

$\Delta t_i \leq t_{\max}$ where $t_{\max}$ is the maximum transmission time between $\mathcal{V}$ and $\mathcal{P}_1$. $\mathcal{V}$ sends to $\mathcal{P}_1$ the number e. $\mathcal{P}_1$ checks that $e \neq 0$ and the correctness of the received c_i's by comparing them to e. After computing $d_1 = \texttt{xcoord}[r_{1,2}Y]$, $\mathcal{P}_1$ computes the value $b_1 = \text{ts} + x_1 + er_{1,1} + d_1$ and sends it to $\mathcal{V}$, who computes the x-coordinate $d_1' = \texttt{xcoord}[yR_{1,1}]$. Then, $\mathcal{V}$ computes $X_1' = (b_1 - \text{ts} - d_1')P - eR_{1,1} - R_{1,2}$ and checks if X_1''s matches the one in its database. After this check $\mathcal{V}$ sends $\text{Out}_{\mathcal{V}} = 1$ if it accepts and $\text{Out}_{\mathcal{V}} = 0$ if it rejects.

Figure 2 depicts the protocol when run between $\mathcal{V}$ and $\mathcal{P}_j$, $\forall j = \{2, \dots, m\}$. It follows the same structure with the protocol run between $\mathcal{V}$ and $\mathcal{P}_1$. The main difference is how b_j is computed, which now depends on the value b_{j-1} generated by $\mathcal{P}_{j-1}$, i.e., $b_j = b_{j-1} + x_j + er_{j,1} + r_{j,2} + d_j$ where x_j is $\mathcal{P}_j$'s private key and d_j is generated similarly to d_1.

After the protocol is run between $\mathcal{V}$, $\mathcal{P}_1$ and $\mathcal{P}_j$ for $j = \{1, \dots, m\}$ the proof $\mathcal{P}_{1,\dots,m}$ is generated with: $\mathcal{P}_{1,\dots,m} = (\text{ts}, b_1, R_{1,1}, R_{1,2}, \cdots, b_j, R_{j,1}, R_{j,2}, \cdots, b_m, R_{m,1}, R_{m,2})$.

IV. Security Analysis

Below we describe the resistance of the GPDB protocols against the main attacks in DB and GP protocols. We mainly refer to the GPDB protocol run between $\mathcal{P}_1$ and $\mathcal{V}$ but the analysis applies for the GPDB run between $\mathcal{P}_j$ and $\mathcal{V}$.

Distance fraud: It is easy to see that $r_{\mathcal{V}} \cdot r_{1,1}$ can be replaced by a random integer. Indeed this holds, since we consider that $\mathcal{V}$ is honest and thus $r_{\mathcal{V}}$ is chosen at random after $\mathcal{P}_1$ has committed to $r_{1,1}$. Additionally, since $\mathcal{V}$ checks that $R_{1,1} \neq 0$, this implies that $r_1 \neq 0$. In most existing DB protocols, the prover and $\mathcal{V}$ generate a session secret using the long-term secret. To do so they often employ a pseudorandom function (PRF). In our case being based on ECC, the session secret is generated by employing the ECDSA conversion function of a shared point $r_{\mathcal{V}}r_{1,1}P$ on the elliptic curve. Although the x-coordinates are not uniformly distributed on $\mathbb{Z}_\ell^*$, Chevalier *et al.* [11] have shown that the binary truncation of the x-coordinate of the last element of an instance of the DDH problem is statistically indistinguishable from the uniform distribution, i.e. $\langle P, aP, bP, U_k \rangle \approx_S \langle P, aP, bP, [\texttt{xcoord}(cP)]_k \rangle$. Thus, for every i it holds that $\mathbb{P}[a_i^0 = a_i^1] = \frac{1}{2}$. We discriminate two cases depending on the values of a_i^0 and a_i^1. • Case 1: If $a_i^0 = a_i^1$, then r_i will be the same whether $c_i = 0$ or $c_i = 1$. Thus, the probability that a dishonest prover $\mathcal{P}^*$ is successful in this case is $\mathbb{P}[\text{case 1}] = \frac{1}{2} \cdot 1$. • Case 2: If $a_i^0 \neq a_i^1$ then the dishonest prover needs to guess the correct response. The probability to have a correct guess is $\frac{1}{2}$. The probability of $a_i^0 \neq a_i^1$ and $\mathcal{P}^*$ to make a correct guess is $\mathbb{P}[\text{case 2}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.

Thus, the overall probability of success for n rounds is $(\frac{3}{4})^n$.

Mafia fraud: In order to perform a successful mafia fraud attack $\mathcal{A}$ has to pass the time critical distance-bounding phase and thus compute successfully the session secrets $a^0||a^1$. This implies that $\mathcal{A}$ should be able to learn the values $r_{1,1}$ and $r_{\mathcal{V}}$ used for the calculation of a^0 and a^1. However, this would mean that the adversary is able to solve the DL problem with non-negligible advantage. In addition, based on the results of Chevalier *et al.* [11] $a^0||a^1$ is statistically indistinguishable from the uniform distribution. $\mathcal{A}$ may adopt two main strategies in order to pass the protocol:

• Case 1: $\mathcal{A}$ may guess early enough the c_i's and relay the corresponding responses from $\mathcal{P}_1$ to $\mathcal{V}$. If $\mathcal{A}$ guesses all c_i's to be successful, all her guesses should be correct since the c_i's are verified after the DB-phase by $\mathcal{P}_1$. $\mathcal{A}$ cannot change e to match her guesses, as the received e since this would invalidate the last authentication step of the protocol where the value s is computed based on the received c_i's. Thus, in this case $\mathcal{A}$ would be successful with probability $\frac{1}{2}$ per round.

• Case 2: If $\mathcal{A}$ guesses the r_i's, since there are two possible response values, the probability of giving a correct response is $\frac{1}{2}$ per round. To be successful $\mathcal{A}$ needs to guess all r_i's correctly since they are checked by $\mathcal{V}$ after the DB-phase.

TABLE I

COMPARISON TO SELECTED DB AND GP PROTOCOLS. (a) COMPARISON TO SELECTED DB PROTOCOLS. (b) COMPARISON TO SELECTED GP PROTOCOLS

(a)

Protocol	Distance	Mafia	Terrorist
HK [7]	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$	1
Reid *et al.* [6]	$(\frac{3}{4})^n$	1	$(\frac{3}{4})^n$
Swiss-Knife [12]	$(\frac{3}{4})^n$	$(\frac{1}{2})^n$	$(\frac{3}{4})^n$
Hermans et al. [3]	$(\frac{3}{4})^n$	$(\frac{1}{2})^n$	1
Our Protocol	$(\frac{3}{4})^n$	$(\frac{1}{2})^n$	1

(b)

Protocol	Privacy	Replay	Multiple Impersonation	DoP	DoS
Juels [2]	×	×	×	×	×
Saito & Sakurai [13]	×	×	×	×	×
Bolotnyy & Robins [14]	×	✓	✓	✓	×
Piramuthu [15]	×	✓	✓	×	×
Huang & Ku [16]	–	✓	✓	–	×
Our Protocol	✓	✓	✓	✓	✓

Thus, the success probability of the mafia fraud is $\frac{1}{2}$ per round and the overall success probability for n rounds $(\frac{1}{2})^n$.

TERRORIST FRAUD: In this case, if the prover $\mathcal{P}_1^*$ is dishonest and helps $\mathcal{A}$ by revealing the r_i's, $\forall i \in \{1, \ldots, n\}$ then $\mathcal{A}$ would be able to recover bits of the secret key x_1. However, if $\mathcal{P}_1^*$ would not reveal all responses to $\mathcal{A}$ but only a^0, $\mathcal{A}$ would be able to respond to all rounds for $c_i = 0$, while for $c_i = 1$ she will have to guess the response. In this case the attack's success probability would be $(\frac{3}{4})^n$ for n rounds.

ANONYMITY & TRACEABILITY ATTACKS: Since the underlying DB protocol is wide-forward-insider private, we merely have to ensure that the slightly modified messages reveal no information about the secret keys x_1 and x_j. One additional message transmitted is the encryption of the timestamp ts and the modified version of b_1 and b_j. The value ts leaks no information since it is encrypted and we consider a semantically secure encryption scheme. Similarly b_1 and b_j leak no information since they follow the same structure with the authentication messages sent in the original protocols and thus the secret key is blinded. Since the subgroup of x-coordinates in $\mathbb{Z}_\ell^*$ is non-uniform, an adversary against privacy could build a distinguisher for d_j. However, this cannot be performed against $d_j + r_{j,2}$ based on the XL assumption.

REPLAY ATTACK: Each session is started by $\mathcal{V}$ who generates an encrypted timestamp ts, which is then used by $\mathcal{P}_1$ to generate its partial proof. Sunsequently, each $\mathcal{P}_j$ builds its partial proof using the partial proof from the preceding prover, which makes all proofs chained back to ts. This makes all provers' proofs dependent on the timestamp, as well as on the preceding provers' proofs, and thus all the proofs are dependent on this unique value. Furthermore, random nonces are generated by both parties and are employed in each run of the protocol that guarantee freshness.

SUBSET REPLAY & MULTIPLE IMPERSONATION: To impersonate a prover and create a partial proof using the prover's data, $\mathcal{A}$ would need to calculate a valid proof b_j. To do this, $\mathcal{A}$ would need to recover the $\mathcal{P}_j$'s private key x_j, since everything else is selected randomly and dependent on the current session. However, recovering x_j implies solving the DL problem with non-negligible advantage.

DENIAL OF SERVICE (DOS): The provers are static and neither store any data between sessions, nor do they update any data before or after a session. Thus, there is nothing that could be exploited to launch a de-syncronization/DoS attack.

DENIAL OF PROOF (DOP): Any DoP attack against the proposed protocol would fail, since each prover is verified immediately after the verifier gets the prover's partial proof.

Table I describes how our protocol relates to existing DB and GP protocols.

V. CONCLUSION

We have introduced the concept of grouping-proof-distance-bounding (GPDB) protocols that can be employed to provide a proof of presence of multiple provers as well as assurance regarding the physical proximity of the provers to the access point. This new concept of GPDB protocols is very useful in settings where multiple provers are employed in multi-factor proximity-based authentication. The proposed protocol offers near optimal resistance against the main threats in GP and DB protocols and provide strong privacy guarantees.

REFERENCES

[1] C. Dimitrakakis and A. Mitrokotsa, "Distance-bounding protocols: Are you close enough?" *IEEE Security Privacy*, vol. 13, no. 4, pp. 47–51, Jul. 2015.

[2] A. Juels, "'Yoking-proofs' for RFID tags," in *Proc. IEEE PerCom Workshops*, Mar. 2004, pp. 138–143.

[3] J. Hermans, R. Peeters, and C. Onete, "Efficient, secure, private distance bounding without key updates," in *Proc. ACM WiSec*, 2013, pp. 207–218.

[4] D. R. L. Brown and K. Gjøsteen, "A security analysis of the NIST SP 800-90 elliptic curve random number generator," in *Proc. CRYPTO*, 2007, pp. 466–481.

[5] S. Brands and D. Chaum, "Distance-bounding protocols," in *Proc. EUROCRYPT*, 1993, pp. 344–359.

[6] J. Reid, J. M. G. Nieto, T. Tang, and B. Senadji, "Detecting relay attacks with timing-based protocols," in *Proc. ASIACCS*, 2007, pp. 204–213.

[7] G. P. Hancke and M. G. Kuhn, "An RFID distance bounding protocol," in *Proc. SECURECOMM*, 2005, pp. 67–73.

[8] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede, "Low-cost untraceable authentication protocols for RFID," in *Proc. ACM WiSec*, 2010, pp. 55–64.

[9] S. Čapkun, K. El Defrawy, and G. Tsudik, "Group distance bounding protocols," in *Trust and Trustworthy Computing* (Lecture Notes in Computer Science), vol. 6740. Springer, 2011, pp. 302–312.

[10] M. Burmester, B. de Medeiros, and R. Motta, "Provably secure grouping-proofs for RFID tags," in *Proc. CARDIS*, 2008, pp. 176–190.

[11] C. Chevalier, P.-A. Fouque, D. Pointcheval, and S. Zimmer, "Optimal randomness extraction from a Diffie–Hellman element," in *Proc. EUROCRYPT*, 2009, pp. 572–589.

[12] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira, "The swiss-knife RFID distance bounding protocol," in *Proc ICISC*, 2008, pp. 98–115.

[13] J. Saito and K. Sakurai, "Grouping proof for RFID tags," in *Proc. IEEE AINA*, vol. 2. Mar. 2005, pp. 621–624.

[14] L. Bolotnyy and G. Robins, "Generalized 'yoking-proofs' for a group of RFID tags," in *Proc. MOBIQUITOUS*, Jul. 2006, pp. 1–4.

[15] S. Piramuthu, "On existence proofs for multiple RFID tags," in *Proc. ACS/IEEE Int. Conf. Pervasive Service*, Jun. 2006, pp. 317–320.

[16] H.-H. Huang and C.-Y. Ku, "A RFID grouping proof protocol for medication safety of inpatient," *J. Med. Syst.*, vol. 33, no. 6, pp. 467–474, 2009.