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 analyze 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.

Index Terms—Distance-bounding, relay attacks, authentication

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 [1]. 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.

- *A. Yang is with the College of Information Science and Technology and the College of Cyber Security, Jinan University, Guangzhou 510632, China. E-mail: anjiayang@gmail.com.*
- *E. Pagnin and A. Mitrokotsa are with the Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg 412 58, Sweden. E-mail: {elenap, aikaterini.mitrokotsa}@chalmers.se.*
- *G.P. Hancke is with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: ghancke@ieee.org.*
- *D. S. Wong is with CryptoBLK Limited, Hong Kong. E-mail: duncanwong@cryptoblk.io.*

Manuscript received 31 Mar. 2017; revised 25 Aug. 2017; accepted 4 Nov. 2017. Date of publication 9 Nov. 2017; date of current version 1 June 2018. (Corresponding author: Elena Pagnin.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2017.2771769

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. [2], 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

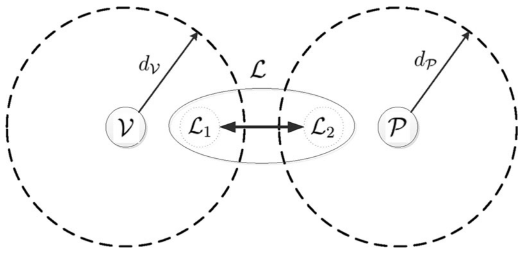


Fig. 1. A wormhole attack run by an external adversary that employs two malicious nodes $\mathcal{L}_1$ and $\mathcal{L}_2$. The communication ranges of $\mathcal{V}$ and $\mathcal{P}$ are indicated by $d_\mathcal{V}$ and $d_\mathcal{P}$, respectively.

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 $\mathcal{L}$ (Fig. 1). 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 ($\mathcal{L}$).

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 7 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 [3] 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 [4] 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., [5], [6], [7], [8], [9]). In the recent decade, the interest in formulating and evaluating DB protocols has grown considerably, and different approaches have been presented [5], [10], [11], [12], [13], [14]. 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 [15], [16], [17]. 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) [4]. 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 [3], where $\rho = 1$; Hancke-Kuhn [4], Swiss-knife [6], Bussard-Bagga [18] and Reid et al. [5], where $\rho = 2$; finally SKI [7] 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. [13] 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. Furthemore, a DB protocol where the responses are based on challenge reflection with channel selection (CRCS) were introduced by Rasmussen and Capkun [19].

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. [20] also constructed an

TABLE 1
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
$\mathsf{t}_{\mathsf{m}ax}$	:	the maximum round-trip-time allowed (as a bound on the distance)

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 resistent to mafia fraud attacks. As a comparison, we provide more comprehensive security analysis which captures all the three common attacks.

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 [21]. 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 [22]. This approach often involves location-based methods that require the physical location of each node to be known [2]. 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 [23]. There are several secure localization schemes that use DB protocols for the underlying distance estimation between nodes [24]. 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 [25] 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 1 refers to the main notations used throughout the paper. Let $\mathcal{K}$ denote the set of keys, $\mathcal{M}$ the set of messages and 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.[1] Then we can make the following claim:

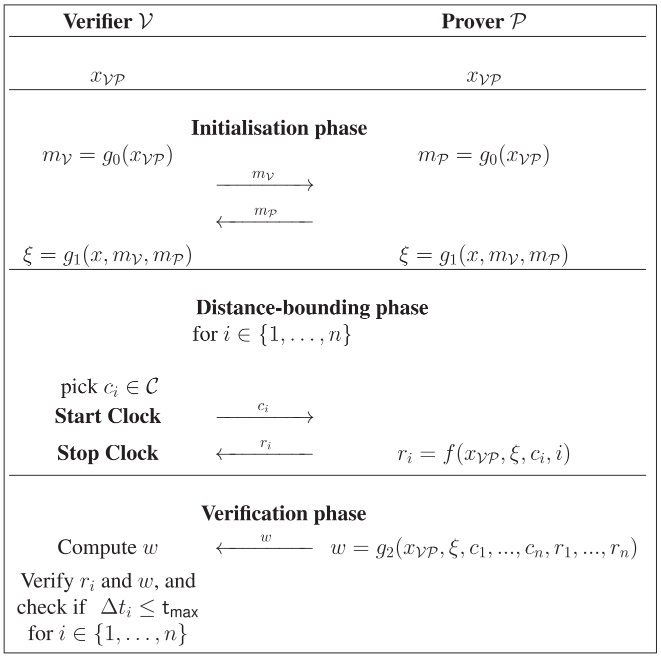
Claim: *All (existing) register-based* DB *protocols can be described using the seven (finite) sets* $\mathcal{K}, \mathcal{M}, \mathsf{X}, \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathsf{W}$ *and the four maps,* g_0, g_1, g_2, f *defined as follows:* $g_0 : \mathcal{K} \to \mathcal{M}$, $g_1 : \mathcal{K} \times \mathcal{M} \times \mathcal{M} \to \mathsf{X}$, $g_2 : \mathcal{K} \times \mathsf{X} \times \mathcal{C}^n \times \mathcal{R}^n \to \mathsf{W}$, $f : \mathcal{K} \times \mathsf{X} \times \mathcal{C} \times \mathcal{I} \to \mathcal{R}$.

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 $\mathcal{K}, \mathcal{M}, \mathsf{X}, \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathsf{W}$, one can obtain all register-based DB protocols. Specific instantiations for five existing DB protocols are given in Section 6.

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 Fig. 2 and is composed of the following phases:

a) *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) [4].

1. $\mathcal{I} = \{1, \ldots, n\}$ where n is the total number of rounds in the distance-bounding phase.


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 $i \in \{1, \ldots, n\}$

Fig. 2. The general description of a register-based DB protocol.

- $g_1 : \mathcal{K} \times \mathcal{M} \times \mathcal{M} \to \mathsf{X}$ 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 [3] 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.

b) *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 $i \in \{1, \ldots, n\}$. The main function in the whole DB protocol is the response function $f : \mathcal{K} \times \mathsf{X} \times \mathcal{C} \times \mathcal{I} \to \mathcal{R}$ 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.

c) *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 $\mathsf{t}_{\mathsf{max}}$ 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 [3] or those where there is no additional message, following the paradigm of Hancke and Kuhn's protocol [4]. 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 $g_2 : \mathcal{K} \times \mathsf{X} \times \mathcal{C}^n \times \mathcal{R}^n \to \mathsf{W}$. The exponent n is the number of rounds in the distance-bounding phase. The function g_2 computes the value (usually a vector) $w = g_2(x_{\mathcal{VP}}, \xi, c_1, \ldots, c_n, r_1, \ldots, r_n)$. 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 [3], a signature on the transcript [3], [6] or it might return a `void` value [5], [7].

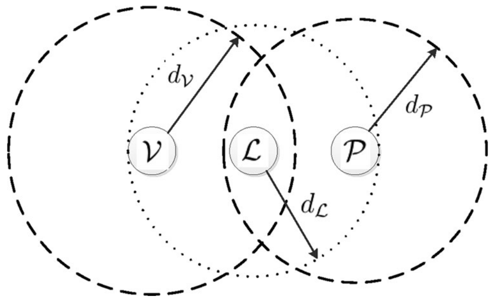


Fig. 3. The basic configuration of two-hop DB protocols: $\mathcal{P}$ lies at a two-hop distance from $\mathcal{V}$. Being outside $\mathcal{V}$'s communication rage, $\mathcal{P}$ relies on the in-between untrusted node $\mathcal{L}$, who is in the communication range of both $\mathcal{P}$ and $\mathcal{V}$. $d_{\mathcal{V}}$, $d_{\mathcal{L}}$, and $d_{\mathcal{P}}$ denote the communication ranges of $\mathcal{V}$, $\mathcal{L}$, and $\mathcal{P}$ correspondingly.

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 $\mathcal{K}, \mathcal{M}, \mathsf{X}, \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathsf{W}$ 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. Fig. 3 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.

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

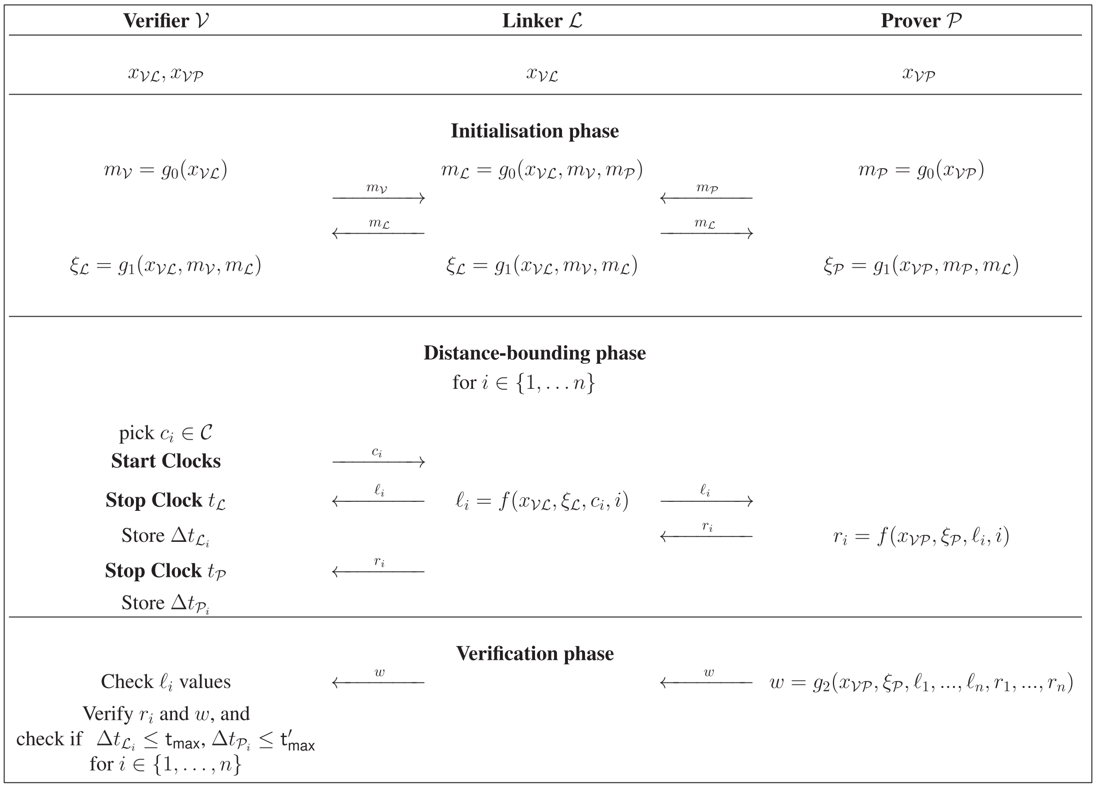

Verifier $\mathcal{V}$
Linker $\mathcal{L}$
Prover $\mathcal{P}$
$x_{\mathcal{VL}}, x_{\mathcal{VP}}$
$x_{\mathcal{VL}}$
$x_{\mathcal{VP}}$
Initialisation phase
$m_{\mathcal{V}} = g_0(x_{\mathcal{VL}})$
$m_{\mathcal{L}} = g_0(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{P}})$
$m_{\mathcal{P}} = g_0(x_{\mathcal{VP}})$
$m_{\mathcal{V}}$
$m_{\mathcal{P}}$
$m_{\mathcal{L}}$
$m_{\mathcal{L}}$
$\xi_{\mathcal{L}} = g_1(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{L}})$
$\xi_{\mathcal{L}} = g_1(x_{\mathcal{VL}}, m_{\mathcal{V}}, m_{\mathcal{L}})$
$\xi_{\mathcal{P}} = g_1(x_{\mathcal{VP}}, m_{\mathcal{P}}, m_{\mathcal{L}})$
Distance-bounding phase
for $i \in \{1, \dots n\}$
pick $c_i \in \mathcal{C}$
Start Clocks
c_i
Stop Clock $t_{\mathcal{L}}$
ℓ_i
$\ell_i = f(x_{\mathcal{VL}}, \xi_{\mathcal{L}}, c_i, i)$
ℓ_i
Store $\Delta t_{\mathcal{L}_i}$
r_i
$r_i = f(x_{\mathcal{VP}}, \xi_{\mathcal{P}}, \ell_i, i)$
Stop Clock $t_{\mathcal{P}}$
r_i
Store $\Delta t_{\mathcal{P}_i}$
Verification phase
Check ℓ_i values
w
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 $i \in \{1, \dots, n\}$

Fig. 4. The general structure of a two-hop DB protocol.

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 Fig. 4), 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 [3], [4], [5], [6], [18].

Our two-hop extension also applies to the register-based DB protocols for which $|\mathcal{C}| > |\mathcal{R}|$, e.g., [7]. 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 [7] 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 Fig. 4 and is composed of the following three phases:

a) *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 $m_{\mathcal{V}} = g_0(x_{\mathcal{VL}}, \texttt{void}, \texttt{void})$ and $m_{\mathcal{P}} = g_0(x_{\mathcal{VP}}, \texttt{void}, \texttt{void})$ 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 $g_0 : \mathcal{K} \times \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$ 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.

b) *Distance-Bounding Phase:* This phase consists of n time-critical rounds and it uses the response function $f : \mathcal{K} \times \mathsf{X} \times \mathcal{C} \times \mathcal{I} \rightarrow \mathcal{R}$. 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 ith 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 $\Delta t_{\mathcal{P}_i}$.

c) *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 $\Delta t_{\mathcal{L}_i} \leq \mathsf{t}_{\mathsf{max}}$, and $\Delta t_{\mathcal{P}_i} \leq \mathsf{t}'_{\mathsf{max}}$ where $\mathsf{t}_{\mathsf{max}} = c \cdot d_{\mathcal{V}}$ and $\mathsf{t}'_{\mathsf{max}} = c \cdot (d_{\mathcal{V}} + d_{\mathcal{L}})$; 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 $\mathsf{t}'_{\mathsf{max}} = 2t_{\mathsf{max}}$. 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[2] ($\mathcal{L}_2$ and $\mathcal{L}_1$ respectively) one near $\mathcal{P}$ and the other near $\mathcal{V}$ (as shown in Fig. 1). 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) [10], [14], 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) $\Delta t_i < \mathsf{t}_{\mathsf{max}}$ 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,[3] the following holds

2. Often, in order to ease the understanding the two adversarial-controlled nodes are *collapsed* into one single entity, the adversary $\mathcal{A}$.

3. 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.

TABLE 2
Comparison of the Best Case Success Probabilities of Attacks against Five DB Protocols

	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	$(\frac{3}{4})^n$	$(\frac{1}{2})^n$	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$	$(\frac{3}{4})^n$
SKI$_{\text{Extend}}$	$(\frac{17}{27})^n$	$(\frac{2}{3})^n$	$(\frac{7}{9})^n$	$(\frac{3}{4})^n$	$(\frac{25}{27})^n$

The values shown for the one-hop case are the ones provided by the authors in the corresponding papers.

$$\mathbb{P}_{\texttt{DF1}} = \prod_{i=1}^{n} p_{r_i} \tag{1}$$

Let c_i denote the actual challenge for round i, then $p_{r_i} = \mathbb{P}\left(r_i^* = f(x_{\mathcal{VP}}, \xi, c_i, i)\right) \geq \max\{\frac{1}{|\mathcal{R}|}, \frac{1}{|\mathcal{C}|}\}$. 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 ($\mathbb{P}_{\texttt{MF1}}$) 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:

$$\mathbb{P}_{\texttt{MF1}} = \sum_{k=0}^{n} \mathbb{P}(G_k)\,\mathbb{P}(E_f|G_k)\,\mathbb{P}(E_w|E_f \cap G_k) \tag{2}$$

  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 (2) into the $k = n$ term, and the $k < n$ term:

$$\begin{aligned}\mathbb{P}_{\texttt{MF1}} &= \mathbb{P}(G_n)\,\mathbb{P}(E_f|G_n)\cdot 1 + \sum_{k=0}^{n-1}\mathbb{P}(G_k)\,\mathbb{P}(E_f|G_k)\cdot\epsilon \\ &= p_{c_i}^n + \epsilon\sum_{k=0}^{n-1}\binom{n}{k}p_{c_i}^k\left(1-p_{c_i}\right)^{n-k}\left(\frac{1}{|\mathcal{R}|}\right)^{n-k}\end{aligned} \tag{3}$$

  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 (3) 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 (3) becomes: $\mathbb{P}_{\texttt{MF1}} = \frac{1}{|\mathcal{C}|}^n +$ negligible terms, for an unforgeable g_2; and $\mathbb{P}_{\texttt{MF1}} = \frac{1}{|\mathcal{C}|}^n\left(2 - \frac{1}{|\mathcal{C}|}\right)^n$, 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 [7], [11], [14]. 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 2 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.

a) *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 [4] DB protocol depicted in Fig. 6: 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 ith entry of the c_ith 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

$$\begin{aligned}\mathbb{P}(\mathcal{L}^*, \mathcal{P}) &= \sum_{k=0}^{n} \mathbb{P}(G_k)\, \mathbb{P}(E_f|G_i)\, \mathbb{P}(E_w|E_f \cap G_k) \\ &= \left(\prod_{i=1}^{n} p_{\ell_i}\right) + \epsilon \left[\sum_{\substack{k=0,\\ I \subsetneq \{1,\ldots,n\},\\ |I|=k}}^{n-1} \binom{n}{k} \left(\prod_{i \in I} p_{\ell_i}\right) \cdot \right. \\ &\quad \left. \cdot \left(\prod_{i \in \{1,\ldots,n\} \setminus I} (1 - p_{\ell_i}) \right) \left(\frac{1}{|\mathcal{R}|}\right)^{n-k} \right], \end{aligned} \quad (4)$$

  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 $\mathbb{P}_{\mathcal{L}^*\mathcal{P}} = \mathbb{P}_{\mathtt{MF1}}$ 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 $\mathbb{P}_{\mathcal{L}^*\mathcal{P}}$ is to $\mathbb{P}_{\mathtt{MF1}}$.

- 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 $\mathbb{P}(\mathcal{L}^*, \mathcal{P}^*) \leq 1$, 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 5.3 *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 5.3 *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 5.3. 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 Fig. 4. 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.

b) *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 $\mathbb{P}_{\mathtt{MF1}}$ in equation (3). 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 [4], Brands and Chaum [3], Reid et al. [5], Swiss-Knife [6] and $\mathsf{SKI}_{\mathsf{extend}}$ from SKI [7]. 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 [4] 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

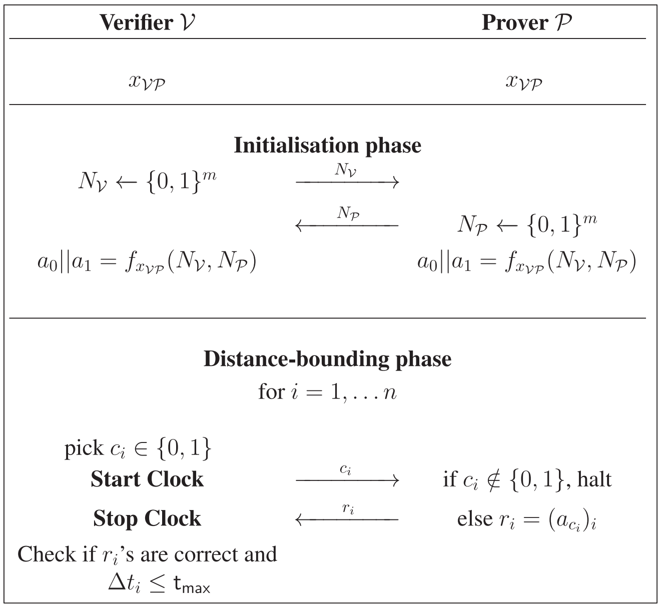

Verifier $\mathcal{V}$
Prover $\mathcal{P}$
$x_{\mathcal{VP}}$
$x_{\mathcal{VP}}$
Initialisation phase
$N_\mathcal{V} \leftarrow \{0,1\}^m$
$\xrightarrow{N_\mathcal{V}}$
$\xleftarrow{N_\mathcal{P}}$
$N_\mathcal{P} \leftarrow \{0,1\}^m$
$a_0||a_1 = f_{x_{\mathcal{VP}}}(N_\mathcal{V}, N_\mathcal{P})$
$a_0||a_1 = f_{x_{\mathcal{VP}}}(N_\mathcal{V}, N_\mathcal{P})$
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
$\Delta t_i \leq \mathsf{t_{max}}$

Fig. 5. The Hancke-Kuhn protocol.

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

1) Hancke and Kuhn (HK) *[4] Protocol:* In the Hancke-Kuhn protocol [4] (depicted in Fig. 5), 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`.

Fig. 6 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 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

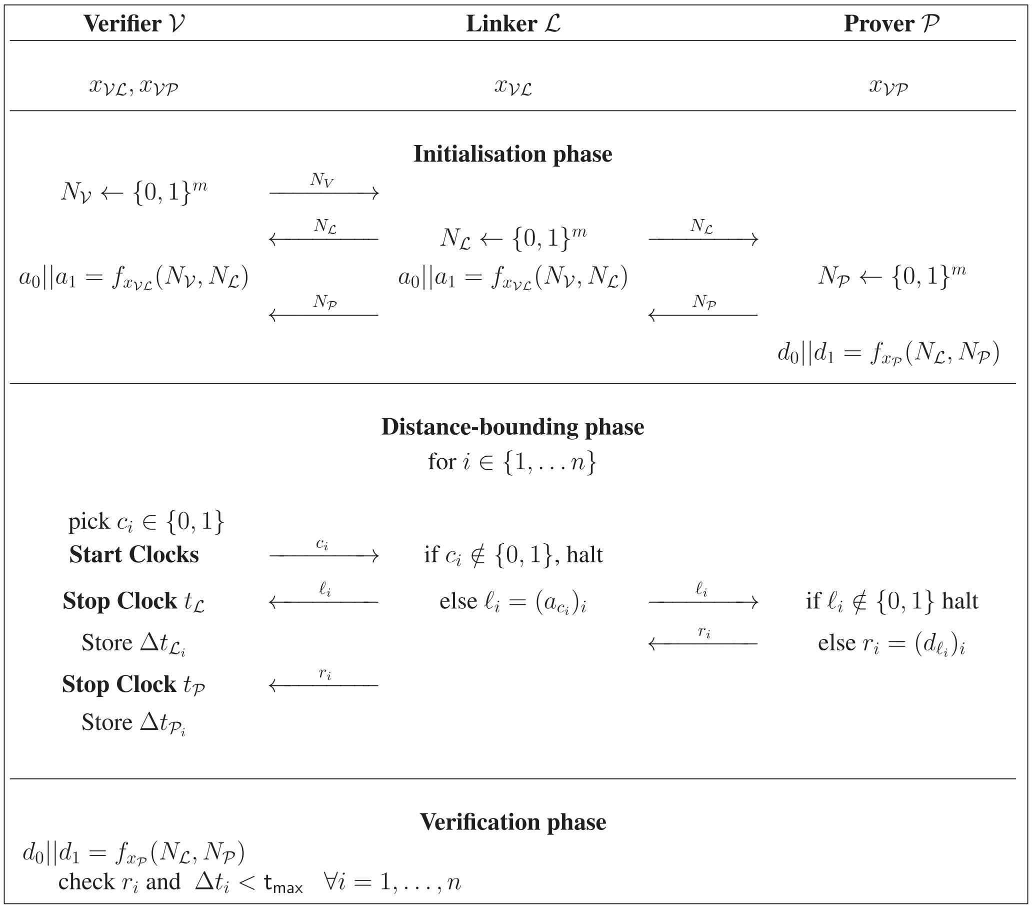

Verifier $\mathcal{V}$
Linker $\mathcal{L}$
Prover $\mathcal{P}$
$x_{\mathcal{VL}}, x_{\mathcal{VP}}$
$x_{\mathcal{VL}}$
$x_{\mathcal{VP}}$
Initialisation phase
$N_{\mathcal{V}} \leftarrow \{0,1\}^m$
N_V
$N_{\mathcal{L}}$
$N_{\mathcal{L}} \leftarrow \{0,1\}^m$
$N_{\mathcal{L}}$
$a_0||a_1 = f_{x_{\mathcal{VL}}}(N_{\mathcal{V}}, N_{\mathcal{L}})$
$a_0||a_1 = f_{x_{\mathcal{VL}}}(N_{\mathcal{V}}, N_{\mathcal{L}})$
$N_{\mathcal{P}} \leftarrow \{0,1\}^m$
$N_{\mathcal{P}}$
$N_{\mathcal{P}}$
$d_0||d_1 = f_{x_{\mathcal{P}}}(N_{\mathcal{L}}, N_{\mathcal{P}})$
Distance-bounding phase
for $i \in \{1, \dots n\}$
pick $c_i \in \{0,1\}$
Start Clocks
c_i
if $c_i \notin \{0,1\}$, halt
Stop Clock $t_{\mathcal{L}}$
ℓ_i
else $\ell_i = (a_{c_i})_i$
ℓ_i
if $\ell_i \notin \{0,1\}$ halt
Store $\Delta t_{\mathcal{L}_i}$
r_i
else $r_i = (d_{\ell_i})_i$
Stop Clock $t_{\mathcal{P}}$
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 $\Delta t_i < \mathsf{t}_{\max}$ $\forall i = 1, \dots, n$

Fig. 6. The two-hop Hancke-Kuhn DB protocol (extension to three participants).

whether the received input is acceptable, if so it reads the ith entry of the c_ith 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 $w = \texttt{void}$, and we omit to write it.

2) Brands and Chaum (BC) *[3]:* 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 $c_i \in \{0,1\}$.

3) Reid et al. *[5]:* 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 $a_1 = \mathsf{Enc}_{a_0}(x)$.

4) Swiss-Knife [6]: 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* $\mathsf{SKI}_{\mathsf{extend}}$ *[7]:* 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 $f = x'_i + (a_1)_i + (a_2)_i \bmod 2$ 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 $\texttt{void}$. 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 $f = x'_i + (a_0)_i + (a_1)_i \bmod 3$ 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.

1) One-Hop Distance Fraud: Using equation (1), we only need to calculate p_{r_i}. In the Brands and Chaum's protocol, the ith 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 2.

One-Hop Mafia Fraud: For the one-hop MF, we can apply equation (3), 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.

3) 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

$$\mathbb{P}_{\mathtt{TF1}} = \left(\frac{m-1}{m} \cdot 1 + \frac{1}{m} \cdot \frac{1}{m}\right)^n = \left(1 - \frac{1}{m} + \frac{1}{m^2}\right)^n \qquad (5)$$

According to equation (5), 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 (4), 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, $\mathbb{P}(\mathcal{L}^*, \mathcal{P}) = (\frac{1}{2})^n$. 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 $\mathbb{P}(\mathcal{L}^*, \mathcal{P}) = (\frac{1}{2} + (1 - \frac{1}{2}) * (\frac{1}{2} + (1 - \frac{1}{2}) * \frac{1}{2}))^n = (\frac{7}{8})^n$. The analysis of the other protocols is similar and the results are shown in Table 2.

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 $(1 \le i \le n)$ 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 $\mathsf{SKI}_{\mathsf{extend}}$.

TABLE 3
All Possible Values of $(a_0)_i/(b_0)_i$ and $(a_1)_i/(b_1)_i$ for the ith Round where $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$

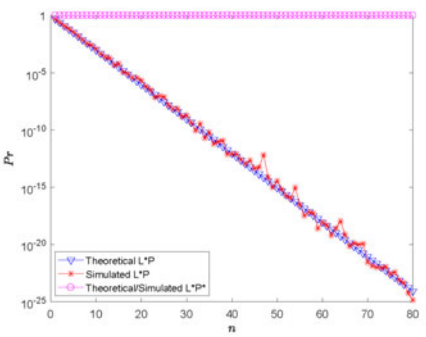
(a) All possible values of $(a_0)_i$ and $(a_1)_i$				
$(a_0)_i$	0	0	1	1
$(a_1)_i$	0	1	0	1
$\mathbb{P}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
(b) All possible values of $(b_0)_i$ and $(b_1)_i$				
$(b_0)_i$	0	0	1	1
$(b_1)_i$	0	1	0	1
$\mathbb{P}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

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 3 shows all the possible values of $(a_0)_i$ (resp. $(b_0)_i$) and $(a_1)_i$ (resp. $(b_1)_i$) in the ith round, as well as the probability of those cases. Now we discuss how much 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 ith 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 $\mathbb{P}(\mathcal{L}^*, \mathcal{P}^*) = (\frac{3}{4})^n$.

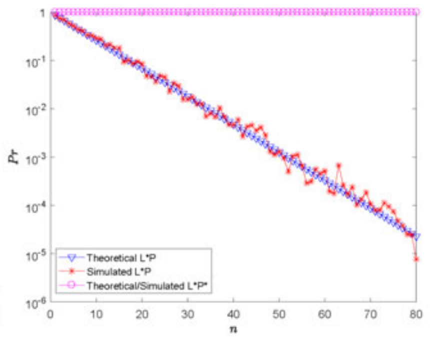
TABLE 4
All Possible Values of $(a_0)_i/(b_0)_i$, $(a_1)_i/(b_1)_i$ and $(a_2)_i/(b_2)_i$ for the ith Round, $y_0, y_1, y_2 \in \{0, 1, 2\}$. $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$

(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
$\mathbb{P}$	$\frac{1}{9}$	$\frac{2}{3}$	$\frac{2}{9}$
(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
$\mathbb{P}$	$\frac{1}{9}$	$\frac{2}{3}$	$\frac{2}{9}$

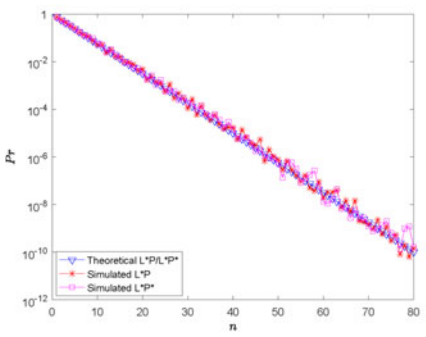
In the two-hop $\mathsf{SKI}_{\text{extend}}$ protocol both $\mathcal{L}^*$ and $\mathcal{P}^*$ have three registers. The analysis is very similar with that in the two-hop Swiss-Knife, but with more registers. The adversaries have more flexible ways to query the responses. In particular, $\ell_i = (a_{c_i})_i$, $r_i = (b_{\ell_i})_i$ 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, $(a_2)_i = (a_0)_i + (a_1)_i + (x_{\mathcal{L}})_i \bmod 3$ and $(b_2)_i = (b_0)_i + (b_1)_i + (x_{\mathcal{P}})_i \bmod 3$. Table 4 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 ith 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



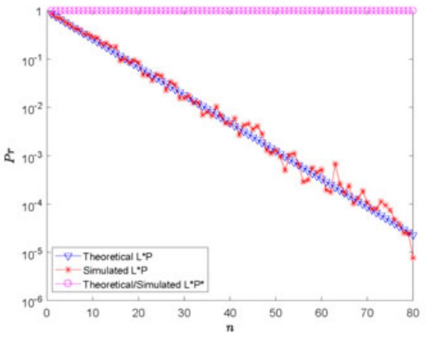
(a) Two-hop Brands and Chaum (BC) protocol



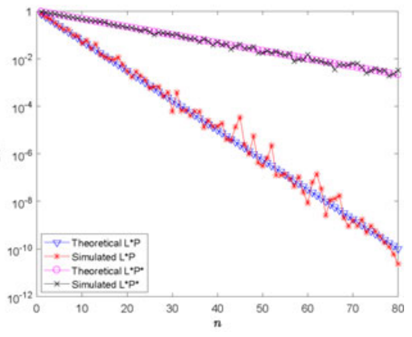
(b) Two-hop Hancke-Kuhn (HK) protocol



(c) Two-hop Swiss-Knife protocol



(d) Two-hop Reid *et al.* protocol



(e) Two-hop $\mathsf{SKI}_{\text{extend}}$ protocol

Fig. 7. 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.

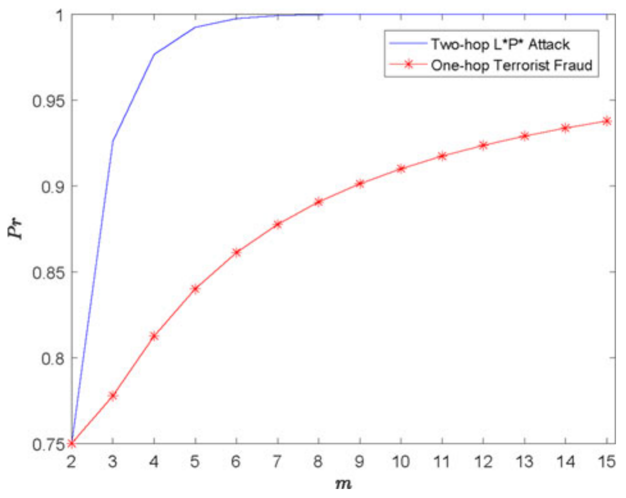


Fig. 8. Relationship between the success probability against one-hop terrorist fraud, two-hop $\mathcal{L}^*\mathcal{P}^*$ attack and m, $m \geq 2$.

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 ith 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 $\mathbb{P}(\mathcal{L}^*, \mathcal{P}^*) = (\frac{1}{9} * 1 + \frac{2}{3} * 1 + \frac{2}{9} * \frac{2}{3})^n = (\frac{25}{27})^n$. 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,

$$\mathbb{P}(\mathcal{L}^*, \mathcal{P}^*) = \left(1 - \frac{m!}{m^{m+1}}\right)^n \tag{6}$$

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 Fig. 7. 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 Fig. 7, the evaluation results verify our theoretical analysis very well.

Moveover, 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. Fig. 8 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.

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.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61702222, 61732021, 61472165, and 61373158), and China Postdoctoral Science Foundation funded project (Grant No. 2017M612842), a grant from the City University of Hong Kong (Project No. 9678131), the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement 608743, the STINT grant IB 2015-6001 and the SNSF SwissSenseSynergy project, Guangdong Provincial Engineering Technology Research Center on Network Security Detection and Defence (Grant No. 2014B090904067), Guangdong Provincial Special Funds for Applied Technology Research and Development and Transformation of Important Scientific and Technological Achievement (Grant No. 2016B010124009), the Zhuhai Top Discipline–Information Security, Guangzhou Key Laboratory of Data Security and Privacy Preserving, and Guangdong Key Laboratory of Data Security and Privacy Preserving.

References

[1] M. Potularski, P. Papadimitratos, and J.-P. Hubaux, "Secure neighbour discovery in wireless networks: Formal investigation of possibility," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2008, pp. 189–200.

[2] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun. IEEE Soc.*, 2003, vol. 3, pp. 1976–1986.

[3] S. Brands and D. Chaum, "Distance-bounding protocols (extended abstract)," in *EUROCRYPT'93, Lecture Notes in Computer Science 765*. Berlin, Germany: Springer-Verlag, 1993, pp. 344–359.

[4] G. P. Hancke and M. G. Kuhn, "An RFID distance bounding protocol," in *Proc. 1st Int. Conf. Security Privacy Emerging Areas Commun. Netw.*, 2005, pp. 67–73.

[5] J. Reid, J. M. Gonzalez Nieto , T. Tang, and B. Senadji, "Detecting relay attacks with timing-based protocols," in *Proc. 2nd ACM Symp. Inform., Comput. Commun. Security*, Mar. 2007, pp. 204–213.

[6] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira, "The swiss-knife RFID distance bounding protocol," in *Information Security and Cryptology*. Berlin, Heidelberg, Germany: Springer-Verlag, 2008.

[7] I. Boureanu, A. Mitrokotsa, and S. Vaudenay, "Practical and provably secure distance-bounding," Information Security, Chalmers Univ. of Technol., Gothenburg, Sweden, 2013.

[8] A. Yang, Y. Zhuang, and D. S. Wong, "An efficient single-slow-phase mutually authenticated RFID distance bounding protocol with tag privacy," in *Proc. Int. Conf. Intell. Circuits Syst.*, Oct. 2012, pp. 285–292.

[9] Y. Zhuang, A. Yang, D. Wong, G. Yang, and Q. Xie, "A highly efficient RFID distance bounding protocol without real-time PRF evaluation," in *Proc. Int. Conf. Netw. Syst. Security*, 2013, vol. 7873, pp. 451–464.

[10] I. Boureanu, A. Mitrokotsa, and S. Vaudenay, "Towards secure distance bounding," in *Proc. 20th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, pp. 55–67, 2013.

[11] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete, "A formal approach to distance-bounding RFID protocols," in *Proc. 14th Int. Conf. Inf. security*, 2011, pp. 47–62.

[12] R. Trujillo-Rasua , B. Martin, and G. Avoine, "Distance-bounding facing both mafia and distance frauds," *IEEE Trans. Wireless Commun.*, vol. 13, no. 10, pp. 5690–5698, Oct. 2014.

[13] G. Avoine and A. Tchamkerten, "An efficient distance bounding RFID authentication protocol: Balancing false-acceptance rate and memory requirement," in *Proc. Inform. Security Conf.*, Sep. 2009, pp. 250–261.

[14] G. Avoine, M. A. Bingöl, S. Kardas, C. Lauradoux, and B. Martin, "A framework for analyzing RFID distance bounding protocols," *J. Comput. Security*, vol. 19, no. 2, pp. 289–317, 2011.

[15] J. Hermans, R. Peeters, and C. Onete, "Efficient, secure, private distance bounding without key updates," in *Proc. 6th ACM Conf. Security Privacy Wireless Mobile Netw.*, 2013, pp. 207–218. [Online]. Available: http://doi.acm.org/10.1145/2462096.2462129

[16] S. Gambs, C. Onete, and J.-M. Robert, "Prover anonymous and deniable distance-bounding authentication," in *Proc. 9th ACM Symp. Inform. Comput. Commun. Security*, 2014, pp. 501–506.

[17] H. Kılınç and S. Vaudenay, "Efficient public-key distance bounding protocol," in *Advances in Cryptology – ASIACRYPT*. Berlin, Heidelberg, Germany: Springer, 2016, pp. 873–901.

[18] L. Bussard and W. Bagga, "Distance-bounding proof of knowledge to avoid real-time attacks," in *Proc. 20th IFIP Int. Inf. Security Conf.*, 2005, pp. 223–238.

[19] K. B. Rasmussen and S. Capkun, "Realization of RF distance bounding," in *USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2010, pp. 389–402.

[20] S. Mauw, J. Toro-Pozo, and R. Trujillo-Rasua , "A class of precomputation-based distance-bounding protocols," in *Proc. IEEE Eur. Symp. Security Privacy*, Mar. 2016, pp. 97–111.

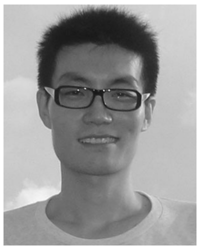
[21] S. Capkun, K. Defrawy, and G. Tsudik, "Group distance bounding protocols," *Proc. Trust Trustworthy Comput.*, vol. 6740, pp. 302–312, 2011.

[22] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proc. IEEE Symp. Inform. Process. Sensor Netw.*, 2005, pp. 91–98.

[23] A. Boukerche, H. Oliveira, E. Nakamura, and A. Loureiro, "Secure localization algorithms for wireless sensor networks," *Commun. Mag.*, vol. 46, no. 10, pp. 96–101, Oct. 2008.

[24] S. Capkun and J. Hubaux, "Secure positioning in wireless networks," *J. Sel. Areas Commun.*, vol. 44, no. 2, pp. 221–232, Oct. 2006.

[25] E. Pagnin, G. Hancke, and A. Mitrokotsa, "Using distance-bounding protocols to securely verify the proximity of two-hop neighbours," *IEEE Commun. Lett.*, vol. 19, no. 7, pp. 1173–1176, July 2015.



Anjia Yang received the BS degree from Jilin University, and the PhD degree from the City University of Hong Kong in 2011 and 2015, respectively. He is currently a postdoctoral researcher with Jinan University, Guangzhou. His research interests include RFID security and privacy, applied cryptography, blockchain security, and cloud computing.

Elena Pagnin received the bachelor's degree in theatrical mathematics from the Univeristy of Padova, and the master's degree in applied mathematics from the University of Trento in 2011 and 2013, respectively. She is currently working toward the PhD degree in network security, from the Department of Computer Science, Chalmers University of Technology, Sweden. Her research interests include lightweight authentication protocols, biometric authentication, and applied cryptography.

Aikaterini Mitrokotsa is an associate professor in the Department of Computer Science and Engineering, Chalmers University of Technology. Previously, she held positions as a visitor professor with ETHZ and the Tokyo Institute of Technology. Her main research interests include the information security, privacy-preservation, authentication protocols, and provable security. She has been awarded the Young Researcher Grant from the Swedish Research Council, the Rubicon Research Grant by NWO, and a Marie Curie Intra European Fellowship.

Gerhard P. Hancke received the BEng and MEng degrees from the University of Pretoria, Pretoria, South Africa, and the PhD degree from the Computer Laboratory, University of Cambridge, Cambridge, United Kingdom, in 2002, 2003, and 2008, respectively. He is an assistant professor in the Department of Computer Science, City University of Hong Kong. His main research interests revolve around advanced sensing and security, especially for RFID/contactless, mobile, and other pervasive technologies.

Duncan S. Wong is the founder and CEO of CryptoBLK Limited, Hong Kong. From 2016 to 2017, he was the vice president of the Financial Technologies (FinTech) Initiative of the largest R&D Center in Hong Kong, the Hong Kong Applied Science and Technology Research Institute (ASTRI). Before joining ASTRI in 2014, he was a professor with the Chinese University of Hong Kong, then the City University of Hong Kong for 12 years, and tenured in the Department of Computer Science, City University of Hong Kong, in 2009. His primary research interest include the information security, cryptography, and blockchain technologies, in particular, cryptographic systems, encryption and digital signature, and cryptocurrencies (e.g., Bitcoin). He has authored more than 200 publications in cryptography and information security, and has served as the chair of the program committee for more than 100 international conferences in cryptography and information security.