Using Distance-Bounding Protocols to Securely Verify the Proximity of Two-Hop Neighbours

Elena Pagnin, Gerhard Hancke, and Aikaterini Mitrokotsa

***Abstract*—Distance-bounding protocols allow devices to cryptographically verify the physical proximity of two parties and is a prominent secure neighbour detection method. We describe how existing distance-bounding protocols could be modified to verify the proximity of both next-hop and two-hop neighbours. This approach allows a node to verify that another node is a physical next-hop neighbour, and also detects legitimate neighbours who make dishonest claims as to who their neighbours are. This approach could prevent dishonest neighbours from hoarding traffic as the result of advertising false two-hop routes.**

***Index Terms*—Wireless sensor network, distance-bounding, secure neighbour discovery, wormhole attack.**

I. Introduction

Communication in ad-hoc wireless networks relies heavily on routing information provided by neighboring nodes. A neighbouring communication node is in a privileged position since it can directly influence the routing decisions of its immediate neighbors. Thus, it is important to verify these neighbours through *secure neighbour discovery* (SND) methods [1]. *Distance-bounding* (DB) protocols, is a prominent SND approach that determines an upper bound on the physical distance between two nodes. They provide a cryptographic proof of the neighbour's proximity but do not consider the proximity of nodes beyond the next-hop neighbour. When building a secure network, we should ideally not only have assurance regarding the neighbour's proximity but also regarding the neighbour's claims (i.e. authenticate the node or not).

In this letter, we introduce a new concept that extends traditional DB protocols to a two-hop setting. We propose a new approach for designing DB protocols that would provide some assurance regarding the physical proximity of both next-hop and two-hop neighbours. This will not only prevent external parties from making distant nodes appear as neighbours, but also prevent compromised or malicious legitimate nodes from advertising two-hop routes to nodes that are in reality much further away. We propose the general structure of a two-hop DB protocol and we discuss the effectiveness of this protocol considering dishonest actions by the untrusted immediate (next-hop) and the two-hop neighboring nodes (provers).

Manuscript received March 5, 2015; accepted May 7, 2015. Date of publication May 18, 2015; date of current version July 8, 2015. This work was supported in part by grants from STINT ("Cross-layer authentication for wireless networks"), SNSF Sinergia ("SwissSenseSynergy"), and City University of Hong Kong (Project No. 7200375). The associate editor coordinating the review of this paper and approving it for publication was P. Serrano.

E. Pagnin and A. Mitrokotsa are with Chalmers University of Technology, 412 96 Gothenburg, Sweden (e-mail: elenap@chalmers.se; aikmitr@chalmers.se).

G. Hancke is with City University of Hong Kong, Kowloon Hong Kong, (e-mail: ghancke@ieee.org).

Digital Object Identifier 10.1109/LCOMM.2015.2434373

II. Background and Problem Statement

In this section, we provide a brief introduction to DB protocols, their limitations against wormhole attacks and the need for two-hop DB. Furthermore, we provide the general structure of a DB protocol, which we later use to demonstrate our two-hop extension in Section III.

A. *Distance-Bounding Protocols*

DB protocols use the round-trip-time of multiple cryptographic challenge-response pairs to determine an upper bound on the physical distance between a verifier ($\mathcal{V}$) and an untrusted prover ($\mathcal{P}$). Brands and Chaum [2] have introduced the first DB protocol to combat relay attacks in ATM systems. Numerous DB protocols have followed, while the interest in formalizing and analyzing the security of these protocols has grown [3]–[6]. The basic objective of a DB protocol is to protect against the following three general threat scenarios:

1) Distance Fraud: In this fraud, a dishonest prover $\mathcal{P}$ tries to prove that it is located close to $\mathcal{V}$, while being far away.

2) Mafia Fraud: This attack involves an honest prover $\mathcal{P}$, an honest verifier $\mathcal{V}$ and an adversary $\mathcal{A}$ (a man-in-the-middle) located far from $\mathcal{V}$. $\mathcal{P}$ and $\mathcal{V}$ are not in close proximity, and $\mathcal{A}$ tries to shorten the distance by convincing $\mathcal{V}$ it communicates with $\mathcal{P}$, while in reality $\mathcal{P}$ and $\mathcal{V}$ are communicating with $\mathcal{A}$.

3) Terrorist Fraud: This attack involves a dishonest prover $\tilde{\mathcal{P}}$, an honest verifier $\mathcal{V}$ and an adversary $\mathcal{A}$ located far from $\mathcal{V}$. The prover $\tilde{\mathcal{P}}$ is far away from the verifier $\mathcal{V}$ but the adversary $\mathcal{A}$ is close to $\mathcal{V}$. The adversary's goal is to convince $\mathcal{V}$ that $\tilde{\mathcal{P}}$ is close; $\mathcal{A}$ achieves this by convincing $\mathcal{V}$ that it is communicating with $\tilde{\mathcal{P}}$ while in reality $\mathcal{V}$ communicates with $\mathcal{A}$. However, in this case $\tilde{\mathcal{P}}$ collaborates with $\mathcal{A}$ but without revealing any information about its long-term secret key to $\mathcal{A}$.

In Fig. 1, we present the general structure of a DB protocol that is resistant to all three attack scenarios. In the literature, protocols that are terrorist-fraud resistant are mainly based on a similar design approach [7]. Our illustrative protocol follows this approach, which is to implement a response function where the dishonest prover's key is revealed if $\mathcal{P}$ discloses the possible responses. The protocol can be broken down into three phases: an *initialization*, *a distance-bounding* and a *verification* phase. $\mathcal{V}$ and $\mathcal{P}$ share a secret key $x_{\mathcal{VP}}$. In the *initialization phase*, $\mathcal{V}$ and $\mathcal{P}$ exchange the randomly generated nonces $N_{\mathcal{V}}$ and $N_{\mathcal{P}}$ correspondingly. Both parties then calculate the response registers $a_0 = f_{x_{\mathcal{P}}}(N_{\mathcal{P}}, N_{\mathcal{V}})$ and $a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{P}})$, where f denotes a pseudorandom function (PRF) and Enc is any symmetric encryption function that would reveal the key $x_{\mathcal{P}}$ if both a_0 and a_1 are revealed, *e.g.* such an encryption function could give us $a_1 = a_0 \oplus x_{\mathcal{P}}$.

In the *distance-bounding phase* (composed of n time-critical rounds), $\mathcal{V}$ starts its clock and sends random single-bit challenges $c_i \in \{0, 1\}$ for $i \in \{1, \dots, n\}$ to $\mathcal{P}$, while $\mathcal{P}$ responds

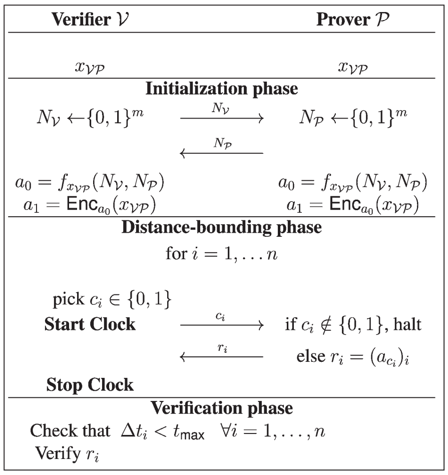
Verifier $\mathcal{V}$
Prover $\mathcal{P}$
$x_{\mathcal{VP}}$
$x_{\mathcal{VP}}$
Initialization phase
$N_{\mathcal{V}} \leftarrow \{0,1\}^m$
$\xrightarrow{N_{\mathcal{V}}}$
$N_{\mathcal{P}} \leftarrow \{0,1\}^m$
$\xleftarrow{N_{\mathcal{P}}}$
$a_0 = f_{x_{\mathcal{VP}}}(N_{\mathcal{V}}, N_{\mathcal{P}})$
$a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{VP}})$
$a_0 = f_{x_{\mathcal{VP}}}(N_{\mathcal{V}}, N_{\mathcal{P}})$
$a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{VP}})$
Distance-bounding phase
for $i = 1, \dots n$
pick $c_i \in \{0,1\}$
Start Clock
$\xrightarrow{c_i}$
if $c_i \notin \{0,1\}$, halt
$\xleftarrow{r_i}$
else $r_i = (a_{c_i})_i$
Stop Clock
Verification phase
Check that $\Delta t_i < t_{\mathsf{max}}$ $\forall i = 1, \dots, n$
Verify r_i

Fig. 1. One-hop distance bounding resistant to all three attacks.

with $r_i = (a_{c_i})_i$. As soon as $\mathcal{V}$ receives the response r_i it stops the clock. In the *verification phase*, $\mathcal{V}$ compares the received r_i with the expected one, determined by a_0 and a_1, and if these are correct it uses the round-trip-time Δt_i to check whether the response is within the maximum allowed time t_{max} to transmit a message between $\mathcal{P}$ and $\mathcal{V}$. This protocol protects against *distance fraud* with probability $\left(\frac{3}{4}\right)^n$, i.e., a dishonest prover knows half the responses (where $a_{0i} = a_{1i}$) and would need to guess each remaining challenge correctly to preemptively send a correct response. The *mafia fraud* attack succeeds with probability $\left(\frac{3}{4}\right)^n$, i.e., an attacker can pre-ask the prover for responses—if it guesses the pre-asked challenge correctly it wins the round, otherwise it needs to guess the response. The protocol also protects against *terrorist fraud*, as the attacker would learn the secret key if both a_0 and a_1 are shared with him. $\mathcal{P}$ can share one register (either a_0 or a_1) without revealing the key $x_{\mathcal{VP}}$. This would mean that $\mathcal{A}$ knows half the correct responses and needs to guess the rest. The resultant success probability for terrorist fraud is $\left(\frac{3}{4}\right)^n$.

B. Motivation Scenarios

In this paper, we introduce for the first time the concept of two-hop distance-bounding which extends the traditional setting of one-hop distance-bounding. In the two-hop setting, we consider three parties: a prover $\mathcal{P}$, a verifier $\mathcal{V}$ and an untrusted in between node (henceforth linker) $\mathcal{L}$. $\mathcal{P}$ and $\mathcal{V}$ are not in each other's communication range but $\mathcal{P}$ wants to be authenticated by $\mathcal{V}$. Two-hop distance-bounding can be employed to verify that $\mathcal{V}$ is close to $\mathcal{L}$ and $\mathcal{L}$ is close to $\mathcal{P}$, by measuring the time-of-flight of the messages exchanged, between $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{P}$. Thus, $\mathcal{V}$ is able to calculate an upper bound on an untrusted prover's ($\mathcal{P}$) distance that is not in its direct communication range. We need to point out here, that we are employing two-hop distance-bounding in order to verify that a prover $\mathcal{P}$ that is not in the communication range of $\mathcal{V}$ but is actually located in the communication range of $\mathcal{L}$ ($\mathcal{V}$'s one-hop neighbor). Two-hop distance-bounding can be useful in many different settings; such as the detection of *wormhole* attacks as well as access control scenarios when the prover is not in the range of the verifier.

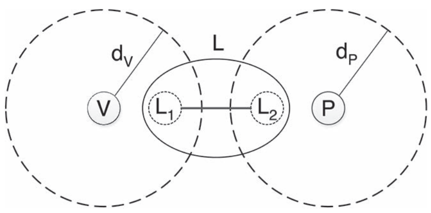

Fig. 2. A wormhole attack run by an adversary employing two malicious nodes $\mathcal{L}_1$ and $\mathcal{L}_2$. $d_{\mathcal{V}}$ and $d_{\mathcal{P}}$ indicate the communication ranges of $\mathcal{V}$ and $\mathcal{P}$.

A wormhole is an attack strategy for undermining routing protocols first described by Perrig *et al.*[8]. In this attack, an adversary wants to convince a network node that the most attractive route to another node is through it. This allows it to control the communication between the two nodes, e.g., it can modify or simply discard messages. In a wormhole attack scenario an adversary may have compromised a node $\mathcal{L}$ located in the communication range of two nodes $\mathcal{V}$ and $\mathcal{P}$, while $\mathcal{P}$ is outside the communication range of $\mathcal{V}$. $\mathcal{V}$ wants to transmit a message to $\mathcal{P}$ and verify that $\mathcal{P}$ is its two-hop neighbour. If $\mathcal{V}$ trusts $\mathcal{P}$ but both $\mathcal{P}$ and $\mathcal{V}$ do not trust $\mathcal{L}$ then by running twice a conventional (one-hop) DB protocol (once between $\mathcal{V}$ and $\mathcal{L}$ and once between $\mathcal{L}$ and $\mathcal{P}$) $\mathcal{V}$ could verify that indeed $\mathcal{P}$ is its two-hop neighbour. However, when $\mathcal{P}$ is not trusted, conventional (one-hop) DB protocols cannot solve this problem. The same problem applies when the adversary controls two nodes $\mathcal{L}_1$ and $\mathcal{L}_2$ instead of a single node $\mathcal{L}$ (Fig. 2). Thus, there is a need for a new mechanism to verify the two-hop proximity of $\mathcal{P}$ by relying on an untrusted one-hop neighbour ($\mathcal{L}$).

As additional motivation for the need of two-hop DB could be considered access control problems where the prover does not have direct access to the authenticator (access point, verifier) but has to rely in an untrusted node (in-between) node. For instance, this could be the case for many smart devices in ubiquitous computing environments, e.g. gaining access to a printer (printing service) if the prover can prove that it is a two-hop neighbour to the printer i.e., lie in a specified distance within the campus/building of a university), even without having direct access to the printer.

III. Two-Hop Distance Bounding

In this section we describe how DB protocols could be modified to allow a verifier $\mathcal{V}$ to compute a distance bound on the next-hop node (i.e., the linker $\mathcal{L}$) and the two-hop neighbouring node (i.e., the prover $\mathcal{P}$) even if both these nodes are untrusted. We base our modifications on simple assumptions regarding the communication model between these nodes, as shown in Fig. 3. If the linker $\mathcal{L}$ is within the communication range (one-hop) of the verifier $\mathcal{V}$, both the verifier $\mathcal{V}$ and the prover $\mathcal{P}$ are within the communication range of the linker $\mathcal{L}$, and $\mathcal{V}$ and $\mathcal{P}$ are beyond each other's communication range (two-hop neighbours). If all three nodes use the same communication channel, this means that only $\mathcal{L}$ can receive messages sent from $\mathcal{V}$, only $\mathcal{L}$ can receive messages sent from $\mathcal{P}$, and that anything $\mathcal{L}$ transmits is received by both $\mathcal{V}$ and $\mathcal{P}$.

Our extension of the general protocol described in Section II-A to the two-hop case is shown in Fig. 4. We assume that $\mathcal{V}$ and $\mathcal{P}$ share the secret key $x_{\mathcal{VP}}$ while $\mathcal{V}$ and $\mathcal{L}$ share the secret key $x_{\mathcal{VL}}$. During the *initialization phase* $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{P}$ respectively select randomly generated nonces $N_{\mathcal{V}}$, $N_{\mathcal{L}}$ and $N_{\mathcal{P}}$. $\mathcal{V}$ sends $N_{\mathcal{V}}$ to $\mathcal{L}$ and $\mathcal{L}$ transmits $N_{\mathcal{L}}$, which means it is

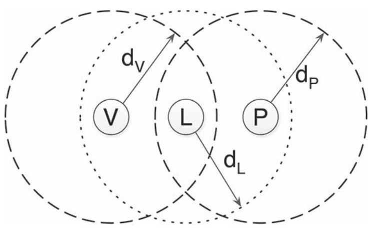

Fig. 3. The basic two-hop node configuration - verifier $\mathcal{V}$, prover $\mathcal{P}$ and linker $\mathcal{L}$.

received by both $\mathcal{V}$ and $\mathcal{P}$. $\mathcal{V}$ and $\mathcal{L}$ both calculate a_0 and a_1, while the prover calculates d_0 and d_1. The *distance-bounding phase* starts when $\mathcal{V}$ generates and sends a random challenge-bit c_i and starts two clocks $t_{\mathcal{L}}$ and $t_{\mathcal{P}}$. $\mathcal{L}$ receives the challenge and transmits $\ell_i = (a_{c_i})_i$, depending on the challenge c_i. $\mathcal{V}$ and $\mathcal{P}$ receive ℓ_i, with the former stopping clock $t_{\mathcal{L}}$ and the latter computing and transmitting his response $r_i = (d_{\ell_i})_i$ to $\mathcal{L}$. In the final step, $\mathcal{L}$ forwards r_i to $\mathcal{V}$ who stops the clock $t_{\mathcal{P}}$. These steps are repeated n times. In the *verification phase* $\mathcal{P}$ sends the nonce $N_{\mathcal{P}}$ along with all challenges received ℓ and the responses sent r (this message is authenticated with a message authentication code MAC) to $\mathcal{V}$. $\mathcal{V}$ computes d_0 and d_1 and verifies that all received ℓ_i and r_i, $\forall i \in \{1, \ldots, n\}$ are correct. If the verification is successful $\mathcal{V}$ uses $t_{\mathcal{L}}$ to bound the distance of $\mathcal{L}$ and $t_{\mathcal{P}}$ to bound the distance of $\mathcal{P}$.

A. Security Analysis

We describe the possible threats when one or both of the internal participants (prover $\mathcal{P}$ and linker $\mathcal{L}$) are dishonest. Due to space constraints we only deal with the main attack scenarios that we expect the two-hop DB protocol to detect.

Case A–Dishonest Prover $\tilde{\mathcal{P}}$, Honest Linker $\mathcal{L}$: To appear closer to $\mathcal{L}$, in the DB phase $\tilde{\mathcal{P}}$ has to send the fraudulent response $\tilde{r}_i$ before it has received the challenge-bit ℓ_i from $\mathcal{L}$. Since r_i is determined by two response registers $d_{0,1}$ $\tilde{\mathcal{P}}$ knows r_i if $d_{0i} = d_{1i}$. If $d_{0i} \neq d_{1i}$ then $\tilde{\mathcal{P}}$ has to guess the response r_i. The overall probability of success is $\left(\frac{3}{4}\right)^n$.

Case B–Honest Prover $\mathcal{P}$, Dishonest Linker $\tilde{\mathcal{L}}$: The dishonest linker $\tilde{\mathcal{L}}$ does not need c_i to determine ℓ_i when $a_{0i} = a_{1i}$. It can send ℓ_i to $\mathcal{P}$ earlier during these rounds, obtain the correct r_i earlier and then wait for c_i from $\mathcal{V}$, which means it wins these rounds. When $a_{0i} \neq a_{1i}$ it can follow two strategies to obtain the rest of r. The first one is to preemptively send a guessed response $\tilde{r}_i$, with a success probability of $\frac{1}{2}$ per exchange round. The second is to preemptively send a guessed bit $\tilde{\ell}_i$ to $\mathcal{P}$ before the challenge c_i is received, but wait until it receives c_i before sending ℓ_i to $\mathcal{V}$. All guesses of $\tilde{\mathcal{L}}$ sent to $\mathcal{P}$ must be correct (its chance of guessing all the bits right is $\left(\frac{1}{2}\right)^n$) or $\mathcal{V}$ will detect the fraudulent bit(s) during the verification of the MAC from $\mathcal{P}$ given that $\tilde{\ell}_i$ received by $\mathcal{P}$ is not the same as the ℓ_i received by $\mathcal{V}$. Thus, the adversary's round success probability is 1 when $a_{0i} = a_{1i}$ and $\frac{1}{2}$ when $a_{0i} \neq a_{1i}$, with an overall probability equal to $\left(\frac{3}{4}\right)^n$.

Case C–Dishonest Prover $\tilde{\mathcal{P}}$, Dishonest Linker $\tilde{\mathcal{L}}$: We may discriminate into two sub-cases.

— $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{L}}$ do not Collaborate: The probability of success is simply whether Case A or Case B succeeds when occurring at the same time. Neither attack strategy implemented by $\tilde{\mathcal{P}}$ or $\tilde{\mathcal{L}}$ effectively assists the other party and the success still depends on whether $\tilde{\mathcal{L}}$ can guess ℓ_i and r_i correctly. Even if $\tilde{\mathcal{L}}$ realises $\tilde{\mathcal{P}}$ is attempting to be dishonest and waits for its early replies their success depends on $\tilde{\mathcal{P}}$ guessing r_i correctly. The probability of either attack succeeding thus remains $\left(\frac{3}{4}\right)^n$.

— $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{L}}$ Collaborate: This sub-case is equivalent to a single-hop terrorist fraud. $\tilde{\mathcal{P}}$ assists $\tilde{\mathcal{L}}$ (located close to $\mathcal{V}$) to convince $\mathcal{V}$ that $\tilde{\mathcal{P}}$ is within the allowed distance bound. However, the attack is seen to be unsuccessful if $\tilde{\mathcal{P}}$ reveals any information about his secret key $x_{\mathcal{P}}$. During the initialisation phase $\tilde{\mathcal{P}}$ sends one of d_0 or d_1 to $\tilde{\mathcal{L}}$, thus not revealing any information about his key. $\tilde{\mathcal{L}}$ can now calculate half of the responses r_i correctly and send them in time to $\mathcal{V}$. $\tilde{\mathcal{L}}$ would need to correctly guess the responses generated from the other register. Thus, the success probability of the attack is $\left(\frac{3}{4}\right)^n$.

B. Discussion

The attacker's probability of success is $\left(\frac{3}{4}\right)^n$ in Case A, Case B, and in Case C. This is comparable to the success probabilities of the original one-hop DB protocol, and it appears as if the modification does not introduce any significant weakness. The additional effort required of each node is minimal. In the original one-hop protocol each entity has to send one conventional message, participate in a distance-bounding phase and calculate responses. In the extended protocol, the prover does not perform any additional actions, the linker needs to send one extra message to relay the prover's final message, and the verifier needs to calculate both $a_{0,1}$ and $d_{0,1}$.

IV. Related Work

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 [9]. 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 for 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 [10]. This approach often involves location-based methods that require the physical location of each node to be known [8]. Determining the location of a node requires additional network infrastructure and resources, especially indoors where Global Positioning Systems (GPS) are not as effective, while a system wide localization scheme still relies on accurate node-level neighbour detection to build secure connectivity maps [11]. There are several secure

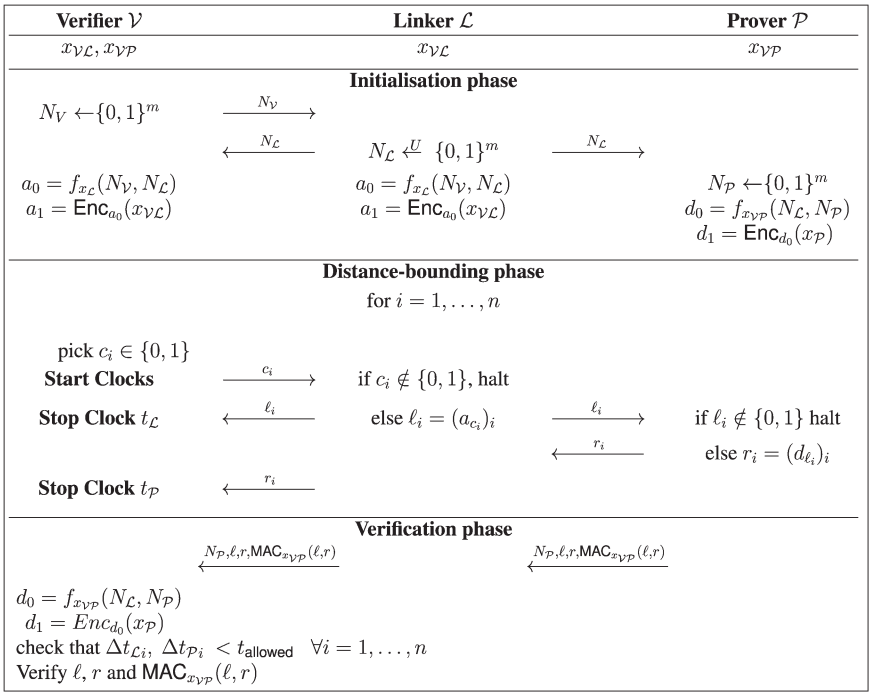
Verifier $\mathcal{V}$
Linker $\mathcal{L}$
Prover $\mathcal{P}$
$x_{\mathcal{VL}}, x_{\mathcal{VP}}$
$x_{\mathcal{VL}}$
$x_{\mathcal{VP}}$
Initialisation phase
$N_V \leftarrow \{0,1\}^m$
$N_\mathcal{V}$
$N_\mathcal{L}$
$N_\mathcal{L} \overset{U}{\leftarrow} \{0,1\}^m$
$N_\mathcal{L}$
$a_0 = f_{x_\mathcal{L}}(N_\mathcal{V}, N_\mathcal{L})$
$a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{VL}})$
$a_0 = f_{x_\mathcal{L}}(N_\mathcal{V}, N_\mathcal{L})$
$a_1 = \mathsf{Enc}_{a_0}(x_{\mathcal{VL}})$
$N_\mathcal{P} \leftarrow \{0,1\}^m$
$d_0 = f_{x_{\mathcal{VP}}}(N_\mathcal{L}, N_\mathcal{P})$
$d_1 = \mathsf{Enc}_{d_0}(x_\mathcal{P})$
Distance-bounding phase
for $i = 1, \ldots, n$
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
r_i
else $r_i = (d_{\ell_i})_i$
Stop Clock $t_\mathcal{P}$
r_i
Verification phase
$N_\mathcal{P}, \ell, r, \mathsf{MAC}_{x_{\mathcal{VP}}}(\ell, r)$
$N_\mathcal{P}, \ell, r, \mathsf{MAC}_{x_{\mathcal{VP}}}(\ell, r)$
$d_0 = f_{x_{\mathcal{VP}}}(N_\mathcal{L}, N_\mathcal{P})$
$d_1 = Enc_{d_0}(x_\mathcal{P})$
check that $\Delta t_{\mathcal{L}i},\ \Delta t_{\mathcal{P}i} < t_{\mathsf{allowed}}\ \ \forall i = 1, \ldots, n$
Verify ℓ, r and $\mathsf{MAC}_{x_{\mathcal{VP}}}(\ell, r)$

Fig. 4. Two-hop distance-bounding protocol.

localization schemes that use DB protocols for the underlying distance estimation between nodes [12]. 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.

V. Conclusion

In this paper, we introduce the concept of two-hop distance-bounding and propose a method based on which existing DB protocols could be modified to provide assurance regarding the physical proximity of both next-hop and two-hop neighbours. To illustrate our idea we presented the general structure of a DB protocol and extended it to the two-hop setting. We performed a security analysis of the introduced protocol when the internal parties ($\mathcal{P}$ and $\mathcal{L}$) are dishonest. Future work could consider whether the two-hop scenario introduces any new attack scenarios and perform a detailed study on the implications for security if the protocol needs to accommodate bit errors during the DB phase. More precisely, the proposed protocol could be rendered resistant to bit errors through the use of an acceptance threshold of erroneous responses, but it would be interesting to see if there are any implications if two channels are used (e.g., if ℓ_i should be transmitted twice). In addition, this analysis should take into consideration the recent point made by Hancke [13] that designing DB protocols resistant to terrorist fraud is significantly weakened by error resistance, as a prover is potentially able to keep its key secret or hinder an attacker from learning its key.

References

[1] M. Poturalski, P. Papadimitratos, and J.-P Hubaux, "Secure neighbour discovery in wireless networks: Formal investigation of possibility," in *Proc. ASIACCS*, 2008, pp. 189–200.
[2] S. Brands and D. Chaum, "Distance-bounding protocols (extended abstract)," in *Proc. EUROCRYPT*, 1993, pp. 344–359.
[3] 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, Apr. 2011.
[4] C. Dimitrakakis, A. Mitrokotsa, and S. Vaudenay, "Expected loss analysis for authentication in constrained settings," *J. Comput. Security*, DOI: 10.3233/JCS-140521, to be published.
[5] I. Boureanu, A. Mitrokotsa, and S. Vaudenay, "Practical & probably secure distance bounding," *J. Comput. Security*, DOI: 10.3231/JCS-140518, to be published.
[6] A. Mitrokotsa, P. Peris-Lopez, C. Dimitrakakis, and S. Vaudenay, "On selecting the nonce length in distance-bounding protocols," *Comput. J.*, vol. 56, no. 10, pp. 1216–1227, Oct. 2013.
[7] J. Reid, J. M. Nieto, T. Tang, and B. Senadji, "Detecting relay attacks with timing-based protocols," in *Proc. ASIACCS*, 2007, pp. 204–213.
[8] Y. -C Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *Proc. IEEE INFOCOM*, 2003, vol. 3, pp. 1976–1986.
[9] S. Čapkun, K. Defrawy, and G. Tsudik, "Group distance bounding protocols," in *Proc. TRUST*, 2011, vol. 6740, LNCS, pp. 302–312.
[10] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proc. IEEE ISPN*, 2005, pp. 91–98.
[11] A. Boukerche, H. Oliveira, E. Nakamura, and A. Loureiro, "Secure localization algorithms for wireless sensor networks," *IEEE Commun. Mag.*, vol. 46, no. 10, pp. 96–101, Oct. 2008.
[12] S. Čapkun and J. Hubaux, "Secure positioning in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 44, no. 2, pp. 221–232, Oct. 2006
[13] G. Hancke, "Distance-bounding for RFID: Effectiveness of terrorist fraud in the presence of bit errors," in *Proc. RFID-TA*, 2012, pp. 91–96.