Distance Lower Bounding

Xifan Zheng, Reihaneh Safavi-Naini, and Hadi Ahmadi

University of Calgary, Calgary, Canada
{xzheng,rei,hahmadi}@ucalgary.ca

Abstract. Distance (upper)-bounding (DUB) allows a verifier to know whether a proving party is located within a certain distance bound. DUB protocols have many applications in secure authentication and location based services. We consider the dual problem of distance lower bounding (DLB), where the prover proves it is outside a distance bound to the verifier. We motivate this problem through a number of application scenarios, and model security against distance fraud (DF), Man-in-the-Middle (MiM), and collusion fraud (CF) attacks. We prove impossibility of security against these attacks without making physical assumptions. We propose approaches to the construction of secure protocols under reasonable assumptions, and give detailed design of our DLB protocol and prove its security using the above model. This is the first treatment of the DLB problem in the untrusted prover setting, with a number of applications and raising new research questions. We discuss our results and propose directions for future research.

1 Introduction

Distance (upper) bounding (DUB) protocols have been widely studied in recent years: a *verifier* V interacts with a *prover* P to obtain assurance that the prover is at a distance at most B from the verifier. A DUB protocol was first proposed in [7] to thwart relay attacks in authentication protocols, by using the location as an unforgeable attribute of the prover. DUB protocols have been widely used for proximity based authentication (e.g., passive keyless entry and start system in modern cars [17]), proximity based control (e.g., implantable medical device [28]), and Radio-Frequency Identification (RFID) authentication [2, 32].

To estimate distance, secure DUB protocols measure the round-trip time of a *fast-exchange challenge-response protocol*, using electromagnetic (EM) communication and assuming constant speed for EM signals. We refer to protocols that use this method for distance estimation, as class $\mathcal{EF}$ (i.e., EM fast-exchange).

In this paper, we consider the dual problem of *distance lower bounding* (DLB), where a prover P wants to prove its distance from a verifier V is higher than a bound B. DLB problem naturally arises in application scenarios where privileges are given based on the distance of the requester to a verifier. For example a company offering unrestricted Internet access to games and entertainment software to employees, when they are outside of main office area of the company campus (e.g, Google campus), and restricted access when employees are within the main

office area. Here the requirement is for employees to prove they are outside the main working area. A second scenario is when the parking lot is divided into zones and parking charge depends on the distance of the car to the main point of interest (e.g. discounted rate will be given if users park their car at farer distance from the shopping mall entrance). In both cases once the privilege is granted based on the distance, one needs to use monitoring mechanisms such as continuous authentication to ensure that the user stays within the claimed area. Embedding such authentication in streaming services such as games or music is straightforward. For the latter scenario, one can use random scanning of the area to ensure correct claim. Although determined users may be able to bypass the authentication, but they will be inconvenient (e.g. move the car frequently) and also have to accept the risk of detection and penalty.

A first approach to solving DLB problem would be to use a DUB protocol. However although a successful run of a DUB protocol proves that the prover is close to the verifier, its failure does not say anything about the distance of the prover. This is because none of the DUB protocols protect against distance enlargement attack [10], where the malicious prover enlarges the distance by delaying the response. Other applications of DUB protocol, such as using DUB protocols with multiple verifiers for secure positioning[10], will also be vulnerable to distance enlargement attack. A second approach would be to use Global Positioning System (GPS)[21] to determine the location of the user. However one needs to trust the GPS measurements, which is known to be vulnerable to attacks, such as GPS spoofing attack [35] where fake satellite signals are used to modify the GPS location data. This solution also results in privacy loss and so one needs to consider privacy enhancing GPS solutions that require extra infrastructure.

Attacks on DLB protocols depend on the application scenario. In Section 2.1 we formalize attacks that are applicable in the above application scenarios, and show that they are parallel to attacks on DUB protocols. DUB protocols have been analyzed against three broad classes of attacks[34]: *distance fraud (DF)* where the prover is malicious and wants to shorten its distance to the verifier; *collusion fraud (CF)* where the prover is malicious and has a helper that would assist them to shorten its distance to the verifier ; and finally *Man-in-the-middle (MiM) attack* where the prover is honest and is the victim of an external attacker, who aims to shorten the distance between the honest prover and the verifier. These classes include attacks such as impersonation, Mafia fraud and Terrorist fraud, that are traditionally considered for DUB. We show that all above attacks are directly applicable to our DLB scenario above and capture important DLB attacks.

The solution to DLB depends on the trust assumption. The DLB problem in a setting that *both the prover and the verifier are trusted*, has been considered in [33]. In this paper, we consider a setting where *the prover is untrusted* and the verifier is trusted.

Here we unravel the main difference between DUB and DLB protocols: DUB protocols have been primarily designed in the setting that an untrusted prover

interacts with a trusted verifier. However, unlike DUB problem, one cannot completely remove trust assumption on the prover in DLB problem. In Section 2.2, we prove that it is impossible to have secure DLB protocol if provers are fully untrusted (have full control of the device hardware and software), which allows them to deviate arbitrarily from the protocol. One however can have secure protocols by making assumptions on the malicious prover's access to the device and/or communication channel. Table 1 summarizes trust assumptions in the two problems.

Table 1: Impossibility result of DB protocols with different trust assumptions

Trust	DLB problem	DUB problem
Trusted prover	Possible (e.g., secure ranging[33])	Possible[1] (e.g., DB [5])
Fully untrusted prover[2]	Impossible (Section 2.2)	
Partially trusted prover[3]	Possible (Section 3, Π_{DLB-BM})	

Our contribution. First, we initiate the study of distance lower bounding (DLB) problem in a setting where the prover is untrusted using motivating application scenarios. Second, we construct security model for DLB problem and define three broad classes of attacks: *distance fraud (DF)* where the prover is malicious and acts alone to enlarge its distance to the verifier; *collusion fraud (CF)* where the prover is malicious and has an helper to collaborate to enlarge its distance to the verifier ; and finally *Man-in-the-middle (MiM) attack* where the prover is honest and is the victim of an external attacker, who aims to enlarge the distance between the honest prover and the verifier. Third, we prove that security against any of these attacks without making *physical assumptions*[4] is impossible. In particular, a fully malicious prover can *always* succeed in the DF attack, and an external attacker (without the cryptographic credentials) can always jam-and-delay the signal between the verifier and the prover, and succeed in the MIM attack. This also implies that a malicious prover that has a helper (CF) will always succeed. Fourth, we study under which reasonable assumptions the problem is solvable and construct a secure DLB protocols under reasonable assumptions, and prove its security against DF, MiM and CF attacks. Finally, we estimate time, memory and energy requirements of our protocol and conclude with open questions and directions for future research.

[1] DUB protocols with fully untrusted prover are secure for all types of trust assumptions.

[2] The malicious prover who has unrestricted control of hardware *and* software of the prover device.

[3] The malicious prover who has restricted control of the hardware of the prover device, but can run malicious software on the device.

[4] Physical assumptions include limited access to the device hardware, and/or the communication channel.

Related work. There is a large body of research on secure positioning and distance estimation problem, including distance bounding protocols[5, 7, 20], positioning techniques[10, 21] and secure ranging protocols[33]. As we argued earlier, these approaches are not directly applicable to the DLB problem in the setting that the prover is not trusted. GPS systems use a set of satellites signals to determine the location and are designed for non-adversarial setting, and so GPS systems are vulnerable to signal spoofing attacks[35]; DUB protocols protect against malicious provers trying to shorten the distance, but are in general vulnerable to the distance enlargement attack[10]; secure positioning systems use a DUB protocol with multiple verifiers to triangulate the prover's location, but is also vulnerable to distance enlargement attack, making positioning an insecure approach for DLB; and secure ranging systems only consider non-adversarial setting as well.

To our knowledge this is the first paper to study DLB problem in a setting where the prover is not trusted. Our approach to defining attacks, distance estimation, and design of the protocol is inspired by the large body of literature on DUB, in particular, [34] for formalization of attacks, and [5, 20] for the design of the protocol. The use of bounded-memory assumption for the prover's device in the context of secure code update had been considered in [25]. Appendix A provides a more complete review of relevant works on distance estimation techniques and bounded memory model.

Paper organization. Section 2 introduces the model and impossibility results. Section 3 describes our approach and proposed protocol. Section 4 provides the security analysis. Section 5 presents relevant practical considerations and Section 6 concludes the paper.

2 DLB - Model and Impossibilities

We consider a multi-party system where a party U is modeled by a polynomially bounded interactive Turing machine (ITM) with a certain location loc_U, and some pre-shared key. Messages that are sent from one location to another, travel at the speed of light and the time taken for travel can be used to determine the distance between the two locations. We assume parties will receive privileges based on their pre-shared secrets and their location. A party can be a *prover* or a *verifier*. A *prover* P engages in a two-party protocol with a *verifier* V, to prove the claim that its distance to the verifier satisfies certain bound. Honest parties run predefined algorithms for their side of the interaction.

The verifier V is always honest. The prover however may be malicious, in which case it is denoted by P^*. A malicious prover deviates from the protocol to make incorrect distance claim and access privileges they are not entitled to. The experiment is expanded to include an external adversary $\mathcal{A}$ who interacts with the parties in the system according to its defined abilities as stated below.

A protocol instance defines an experiment denoted by $exp = (P(x; r_P) \leftrightarrow \mathcal{A}(r_A) \leftrightarrow V(y; r_V))$, where r_P and r_V are random coins, and x and y are secret keys, of the prover and the verifier, respectively. At the end of a protocol instance,

V has an output Out_V, which is 1 or 0, showing acceptance or rejection of the DLB, respectively. The prover does not have an output.

A participant in an experiment has a *view* consisting of all its inputs, coins, and messages that it can see. The external attacker $\mathcal{A}$ may interact with multiple Ps and Vs, and its view will include all these interactions. Throughout the paper $\Pr_r[event : experiment]$ denotes the probability of the *event* for the *experiment*, where r denotes that random coins used in the experiment.

Definition 1 (DLB Protocol). *A Distance Lower Bounding (DLB) protocol is a tuple (Gen, P, V, B), where $(x, y) \leftarrow Gen(1^s, r_k)$ is a randomized key-generation algorithm that takes security parameter s and randomness r_k and outputs keys x and y; $P(x; r_P)$ is the prover's ppt ITM that takes secret-key x and randomness r_P; $V(y; r_V)$ is the verifier's ppt ITM taking secret-key y and randomness r_V, and B is a distance-bound. It satisfies two properties:*

- *Termination: $(\forall s)(\forall R)(\forall r_k; r_V)(\forall loc_V)$ if $(.; y) \leftarrow Gen(1^s; r_k)$ and $(R \leftrightarrow V(y; r_V))$ is an execution of the protocol between the verifier and any (unbounded) prover algorithm, V halts in $Poly(s)$ computational steps;*
- *p-Completeness: $(\forall s)(\forall loc_V; loc_P$ such that $d(loc_V; loc_P) \geq B)$ we have*

$$Pr_{r_k;r_P;r_V}\left[Out_v = 1 : \begin{array}{c}(x; y) \leftarrow Gen(1^s; r_k)) \\ P(x; r_P) \leftrightarrow V(y; r_V)\end{array}\right] \geq p$$

2.1 Attacks on DLB Protocols

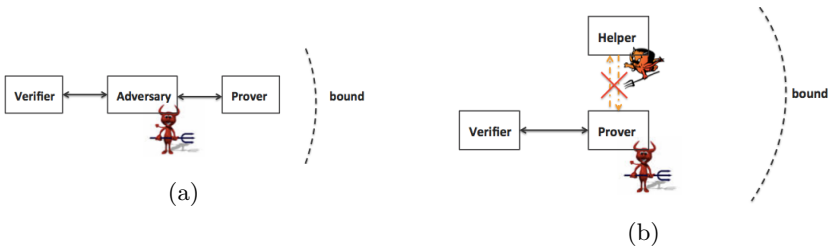

Fig. 1: MiM and CF attacks in DLB

We consider three classes of attacks: distance fraud (DF), man-in-the-middle attack (MiM), and collusion fraud (CF). In DF, P^*, with $d(P^*, V) < B$, wants to convince V that its distance is at least B. In MiM attack, an external attacker who does not have the secret key, interacts with multiple Ps and Vs, and finally succeeds in taking the role of a prover in a protocol instance (See Figure 1a). In CF, P^* colludes with a helper to claim a longer distance to V (See Figure 1b). The collusion should not leak the prover's secret key to the helper. The formal definitions of the attacks are below.

Distance fraud (DF) attack. In this attack, a dishonest prover who is closer than the distance (bound) B, wants to convince the verifier that it has a distance at least B.

Definition 2 (DF-resistance). *A DLB protocol Π is α-resistant to distance fraud if $(\forall s)(\forall P^*)(\forall loc_v$ such that $d(loc_v, loc_{p*}) \leq B)(\forall r_k)$, we have*

$$Pr_{r_v}\left[Out_v = 1 : \begin{array}{c}(x,y) \leftarrow Gen(1^s; r_k)\\ P^*(x) \leftrightarrow V(y; r_v)\end{array}\right] \leq \alpha$$

where P^* is any dishonest prover. Because of the concurrent setting we effectively allow polynomially bounded number of $P(x')$ and $V(y')$ close to $V(y)$ with independent (x', y').

Distance hijacking. Definition 2 captures *distance hijacking attack* [12] against DLB protocols. In this attack, P^* who is at distance $< B$, uses DLB communications of unaware honest provers at a distance $\geq B$, to claim a distance $\geq B$.

Man-in-the-middle (MiM) attack. In the MiM attack, there is an external adversary who does not have the secret-key of the protocol, interacts with honest provers and verifiers, and finally takes the role of a prover in a protocol instance with the verifier.

Definition 3 (MiM-resistance). *A DLB protocol Π is β-resistant to MiM attack if $(\forall s)$, $(\forall m, \ell, z)$ that are polynomially bounded, $(\forall A_1, A_2)$ polynomially bounded, for all locations such that $d(loc_{P_j}, loc_V) < B$, where $j \in \{m+1, \cdots, \ell\}$, we have*

$$Pr_{r_v}\left[Out_v = 1 : \begin{array}{c}(x,y) \leftarrow Gen(1^s)\\ P_1(x)...P_m(x) \leftrightarrow A_1 \leftrightarrow V_1(y)...V_z(y)\\ P_{m+1}(x)...P_\ell(x) \leftrightarrow A_2(View_{A_1}) \leftrightarrow V(y)\end{array}\right] \leq \beta$$

Here probability is over all random coins of the protocol, and $View_{A1}$ is the final view of A_1. The definition effectively allows polynomially bounded number of $P(x')$, $P^*(x')$, and $V(y')$ with independent (x', y'), anywhere.

In this definition the attacker can have a learning phase during which it interacts with m honest provers and z verifiers. It then uses its view in the attack phase, and engages in $\ell - m$ protocol instances between honest provers and the target verifier.

Mafia fraud and impersonation attack. Definition 3 is general and covers Mafia fraud and impersonation attack as special cases. In Mafia fraud, there is no learning phase. The attacker interacts with an honest prover and makes the verifier to output accept. That is, $m = z = 0$ and $\ell = 1$ in the attack phase. In impersonation attack the attacker uses multiple possibly concurrent interactions with the verifier to make the verifier output 1.

Definition 4 (CF-resistance). *A DLB protocol Π is (γ, η) resistant to collusion fraud if $(\forall s)(\forall P^*)(\forall loc_{v_0})$ such that $d(loc_{v_0}, loc_{P^*}) < D)$, $(\forall A^{CF} ppt.)$, if we have,*

$$Pr_{r_v}\left[Out_{v_0}=1:\begin{smallmatrix}(x,y)\leftarrow Gen(1^s)\\ P^*(x)\leftrightarrow A^{CF}\leftrightarrow V_0(y)\end{smallmatrix}\right]>\gamma$$

over all random coins of the protocol, there exists an extended[5] *MiM attack with* m,ℓ,z,A_1,A_2,P_i,
P_j,V_i *that uses interaction with* P *and* P^* *both, and* V*, in the learning phase and have*

$$Pr\left[\begin{smallmatrix}(x,y)\leftarrow Gen(1^s)\\ Out_v=1:\ P_1^{(*)}(x)...P_m^{(*)}(x)\leftrightarrow A_1\leftrightarrow V_1(y)...V_z(y)\\ P_{m+1}(x)...P_\ell(x)\leftrightarrow A_2(View_{A1})\leftrightarrow V(y)\end{smallmatrix}\right]>\eta$$

Here $P^{(*)}$ *is a prover that is either* P *or* P^**. We have* $d(loc_{P_j},loc_V)<B$*, for all* $j\in\{m+1\cdots\ell\}$*. We implicitly allow a polynomially bounded number of* $P(x')$*,* $P^*(x')$*, and* $V(y')$ *with independent* (x',y')*, anywhere but no honest participant is far from* V_0*.*

Collusion fraud implies that if a malicious prover P^* who is close to V_0 can help A^{CF} to succeed in the DLB verifier to output 1, then there is an adversary (A_1,A_2) who will succeed in an (extended) MiM attack, where multiple instances of $P^{(*)}(x)$, denoting honest or dishonest prover, is used during the learning phase. In other words colluding attack will not succeed unless some secret information of the malicious prover is leaked.

Terrorist fraud. In Terrorist fraud, P^*, with $d(P^*,V)<B$, gets aid from a helper who does not have the secret key, to succeed in an instance of the protocol with the verifier. Definition 4 captures terrorist fraud as a special case by letting $m=z=\ell=1$, by simply allowing A_1 to run A^{CF} and succeed in impersonation and making V to accept.

2.2 Impossibility results

We consider protocols in $\mathcal{EF}$. Let C denote speed of light, t_c and t_r denote the verifier's clock readings, when the challenge is sent and the response is received, respectively. Let T_{proc} denote the *processing time* of the prover. If the received response is correct, the verifier calculates $T_\Delta=t_r-t_c$ to estimate the distance of the prover. The verifier estimates the prover's distance D as

$$D=\frac{(T_\Delta-T_{proc})C}{2}. \tag{1}$$

In DF, considering a malicious prover with full (hardware and software) control over the proving device, the malicious prover at close distance D is a registered user of the system and knows the credential to calculate correct responses to the verifier challenges. To claim a longer distance, the prover simply modifies the execution to add appropriate delay by tampering with the hardware/software. This results in the impossibility of DF resistance. Without assuming any restrictions, a MiM adversary can easily succeed in DLB by sitting

[5] Because learning phase allows interaction with P^*.

between the verifier and an honest prover and launching a jam-and-delay attack to add appropriate delay. Hence, we have impossiblity of MiM resistance. The impossibility of resistance against CF follows immediately from the above impossibility results for DF and MiM resistance. We can state this even more generally: Any setting (set of assumptions) that makes DLB security against DF or MiM impossible will make it impossible to resist against CF. The reason is CF attackers can make prearrangements to simulate a DF attack (e.g., without helper being engaged in the DLB instance) or a MiM attack (e.g., by the dishonest prover acting as an honest one). These statements are formalized below

Theorem 1. *1. Any DLB protocol in $\mathcal{EF}$ is vulnerable to DF if P^* has full (hardware and software) control over the prover's device.*
2. No DLB protocols in $\mathcal{EF}$ can provide β-resistance with $\beta < 1$ to MiM attack launched by an external attacker who can jam and delay messages to (or from) the prover.
3. For any DLB protocol in $\mathcal{EF}$, P^* can succeed in CF with probability 1 and negligible key leakage to the helper, if the helper has full access to the communication channels with P^*, and P^* has full control over the prover's device. The result holds even if communication is only allowed in one direction between the prover and the helper.

The proof sketch of Theorem 1 can be found in Appendix C.

2.3 Restricted DF, MiM, and CF

Although it is impossible to protect DLB against unrestricted adversaries, security may become feasible if certain restrictive assumptions are made. To remove the above impossibility results, we use reasonable assumptions (restrictions) on the adversary's control of the device and/or the communication channel. We refer to attacks under these conditions as restricted DF, MiM and CF (rDF, rMiM and rCF), to emphasize extra assumptions are needed.

Table 2: DLB security against the three attacks in different settings.

Attacks	Assumptions			
	No Assumption	Prover's device	Communication	Combined
		[BM]	$[\overline{\mathsf{OC}}]$	$[BM + \overline{\mathsf{OC}}]$
DF-security	×	✓	×	✓
MiM-security	×	×	✓	✓
CF-security	×	×	×	✓

Notations. We use PD to denote the prover's device, and $\mathrm{rX}^{[Y]}$ to denote restricted version of attack X, where $X \in \{DF, MiM, CF\}$ and restrictions are

stated in Y. For example, rDF$^{[BM]}$ refers to the restricted DF attack, under the restriction that PD has bounded memory.

Table 2 summarizes our impossibility results and shows assumptions that are used in our construction in Section 3. The assumptions that we use for security against rDF are, (i) P^* cannot access (read or write) the PD 's read-only memory (ROM), and (ii) PD has *bounded memory (BM)*. Note that the first assumption still allows P^* to inject malicious codes into the device writable memory (RAM), and modify correct execution of the protocol. The bounded memory assumption is a well-established model in cryptography [8], and has also been used in the design of security systems [25]. To achieve the security against rMiM and rCF, in addition to the above assumptions, we require the helper to have no *Online Communication (OC)* with the prover during the fast-exchange phase. In Section 3, we present a DLB protocol that provides security against rDF, rMiM and rCF under the above assumptions. Note that one may achieve rDF, rMiM and rCF resistance using other assumptions that restrict the prover and the helper. For example instead of assuming a root of trust on the PD, one may establish a dynamic root of trust using software attestation. We give a software attestation-based DLB protocol in Appendix B. This protocol also requires no online-communication assumption for security against all attacks. We also provide an overview of security analysis and implementation challenges of the protocol.

3 DLB protocol Constructions

Assumptions and attack model. We assume that the PD is a bounded memory device, which has a protected memory (ROM), and a writable memory (RAM) with (fixed) L bit size. Without loss of generality, we consider RAM as an array indexed from 1 to L. The DLB protocol code is stored partly in ROM, denoted by DLB_{ROM}, and partly in RAM, denoted by DLB_{RAM}. We assume V has a shared key with the PD, and holds the same DLB code. The secret key of PD is stored in ROM and is accessible only to the code in ROM. We assume communication channel is noise free, although our results are extendable to noisy communication by applying similar methods as [31]. The adversary may store and run arbitrary malicious code on the RAM of the PD, but is not able to tamper the hardware of the device.

Approach. Using equation 1, P^* at distance D can always delay the response by $2D'/C$ second(s) to claim a longer distance $D + D'$. Let T_{max} denote the maximum expected response generation (processing) time by the verifier. (This can be estimated for example, by measuring the processing time of a set of functional devices, and choosing T_{max} larger than all the measured times.) Knowing that $0 \leq T_{proc} \leq T_{max}$, the verifier uses the round-trip time T_Δ to obtain the following distance bounds.

$$D \geq D_{lower} = \frac{(T_\Delta - T_{max})C}{2} \tag{2}$$

We propose a protocol that assumes *bounded memory for* PD and enables V *to force an upper bound on the delay introduced by* P^*.

DLB protocol outline. Π_{DLB-BM} consists of three phases: *(i) Initialization:* during which the prover and the verifier exchange nonce, and use them together with their pre-shared secrets to compute a shared response table. *(ii) Fast-exchange:* that consists of n challenge-response rounds, each round consisting of two consecutive challenges, followed by the two corresponding responses. *(iii) Verification:* during which the verifier checks the received responses to distance estimation and erasing sequence, and accepts if they satisfy the required conditions.

Our protocol referred to as Π_{DLB-BM}, is composed of two basic protocol blocks, Π_1 for distance estimation and Π_2 for secure memory erasure, shown in Figure 2. The combination effectively upper bounds the delay that the malicious prover can introduce. Π_1 is a pre-computation challenge-and-response distance estimation protocol [1] and Π_2 is a secure erasure protocol [25] (see Section A). A challenge-response round in Π_1 is used for distance estimation. A challenge-response round in Π_2 however is used to refresh part of the memory that will not be required for the future rounds.

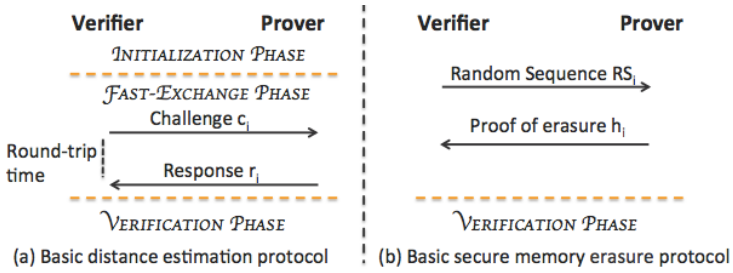

Fig. 2: Basic building blocks Π_1(a) and Π_2(b)

3.1 The protocol Π_{DLB-BM}

The secret key consists of two l-bits strings, x, and $\hat{x}$. When clear from context, we refer to each string as *key* also. The protocol uses a secure Pseudo Random Function (PRF) $f_x : \{0,1\}^{2k} \rightarrow \{0,1\}^{2n}, x \in \{0,1\}^{\ell}$, and a secure keyed-hash function $(H_{\hat{x}})_{\hat{x}\in\{0,1\}^{\ell}} : \{0,1\}^{*} \rightarrow \{0,1\}^{b}$. Figure 3 shows the messages communicated in the three phases of the protocol.

Phase 1: Initialization phase. The prover generates a k-bit nonce N_p and sends it to the verifier. The verifier selects a k-bit nonce N_v and a $2n$-bit random string A, calculates $M = A \oplus f_x(N_p, N_v)$, and sends (M, N_v) to the prover. With this information, the prover decrypts M to retrieve $A = M \oplus f_x(N_p, N_v)$ and stores it in memory. A is the *response table* that will be used by the prover to respond

to challenges in Phase 2. Considering $A = (a_{(1,j)}, a_{(2,j)})$, where $j = 1 \cdots n$, as a sequence of n bit pairs, we define a third string $a_{(3,j)} = a_{(1,j)} \oplus a_{(2,j)} \oplus x$. $a_{(3,j)}$ is computed at run time from the response table and so is not stored in memory.

Phase 2: Fast-exchange phase. This phase proceeds in n consecutive challenge-response rounds. In each round $1 \leq i \leq n$, the verifier chooses a random challenge $c_i \in \{1, 2, 3\}$ and sends it to the prover, immediately followed by a random erasing sequence RS_i of length z_i. In section 3.2, we will discuss how z_i is determined. The role of RS_i is to prevent P from delaying the response to extend its distance. On receiving the challenge c_i, the prover will retrieve the response $r_i = a_{(c_i,i)}$. When $z_i - 1$ bits of RS_i are received, the prover must send r_i to avoid it being overwritten by the final bit of RS_i. The prover must also send the response to the erasing sequence (also referred to as proof of erasure h_i). By correctly designing the computation of the hash, the correct proof of erasure will "prove" that the prover has received and stored the full RS_i and also has kept the code DLB_{RAM} intact (see Section 3.2 for details). In addition, the verifier records the time difference $T_{\Delta,i}$ between sending c_i and receiving r_i.

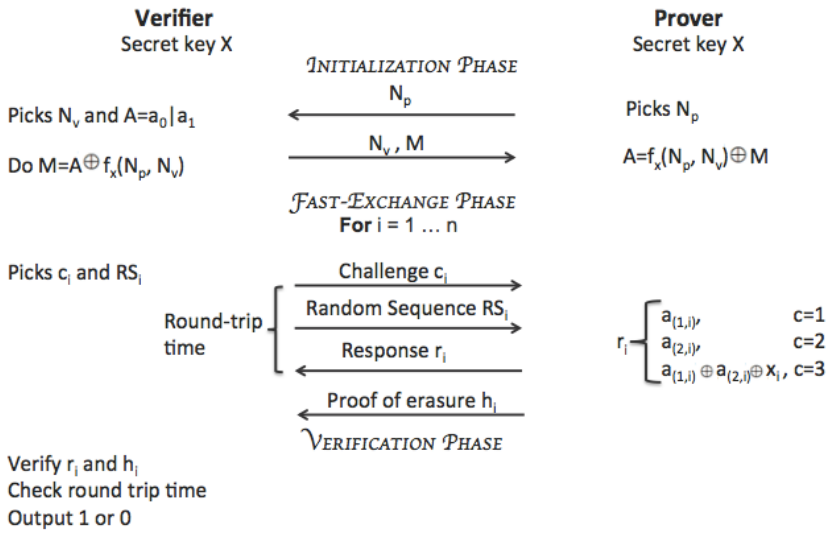

Fig. 3: Distance lower bounding protocol Π_{DLB-BM}

Phase 3: Verification phase. The verifier checks the correctness of response r_i and proof of erasure h_i for all rounds, $i = 1 \cdots n$. It also verifies whether all response times are higher than a threshold θ, determined as follows. Let B denote the distance-bound, and $T(z_i - 1)$ denote the time interval required by the prover to receive $z_i - 1$ bits. The acceptable round-trip time in round i must satisfy $T_{\Delta,i} \geq \theta = \frac{2B}{C} + T(z_i - 1)$, where $C = 3 * 10^8$ is the speed of light (see equation

2). The verifier outputs $Out_v = 1$, if and only if all verifications and time checks succeed. Here for simplicity, we have assumed the communication channel is noise free, and so a successful protocol requires all challenges to be correctly responded.

3.2 The design of erasure sequence and its response

In fast-exchange round i, an erasure sequence RS_i is sent to the P, and a correct response is required. RS_i is used to guarantee all the device memory is erased, except DLB code and the part of the memory that is required for future response. The length of the erasing sequence RS_i must be chosen as follows.

Sequence length. If one bit of memory in addition to what is required for future the responses is left out, the dishonest prover can exploit that bit to store the response bit of the current challenge and delay it. In order to erase all but the part of the memory that is required for future response, the length of the erasing sequence RS_i must be chosen as follows.

Let the sizes of the RAM and DLB_{RAM}, be L and λ, respectively. After the initialization phase, the $2n$-bit random table A is stored in the prover's device memory. In Round 1, the erasing sequence RS_1 must erase $L - \lambda - 2n$ unused memory, together with $(a_{(1,1)}, a_{(2,1)})$, the two response bits associated with round 1. In each subsequent challenge-response round i, two additional bits $(a_{(1,i)}, a_{(2,i)})$ of A will be used and so the length of the erasing sequence must be increased by two bits. By induction, the random sequence RS_i in round $1 \leq i \leq n$ must have length $L - \lambda - 2(n - i)$. Figure 4 shows the state of the prover's memory during protocol execution.

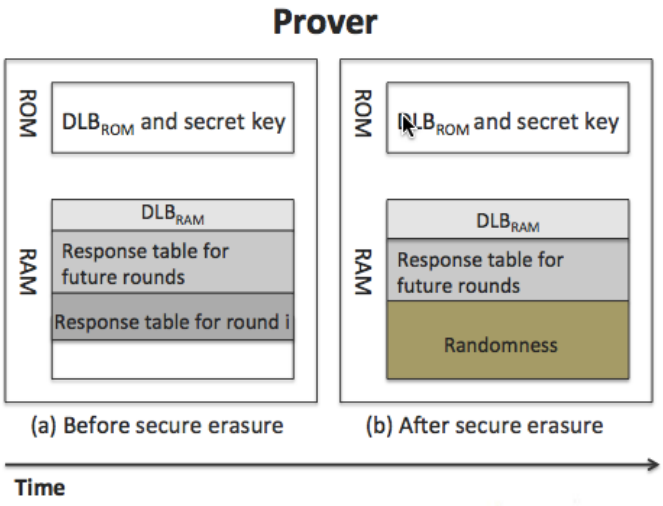

Fig. 4: Prover's memory during protocol execution

Response to the erasure sequence. The response in round i, denoted by h_i, must guarantee that the PD's memory, contains the sequence RS_i and DLB code DLB_{RAM} in full, and prove that the rest of the memory is erased. We refer to this response as *proof of erasure*, as it is inspired by [25]. To obtain assurance that RS_i is fully stored, it's sufficient to require the prover to return the exact RS_i, starting from the last received bit, to the first one. This however will be expensive from communication and power consumption view point. An efficient approach is to send the cryptographic hash of RS_i. To prevent the prover from calculating the hash value in real-time without storing the whole RS_i, we require the hash function to be applied to the received sequence in the reverse order of arrival. That is, assuming $RS_i = (u_1 \cdots u_{z_i})$, the hash will be applied to $\bar{RS}_i = (u_{z_i} \cdots u_1)$. This leaves the prover no choice other than waiting for the last bit to arrive, before starting the calculation. To prevent P^* to simply store the required hash value of the code, we use $h_i = H_{\hat{x}}(\bar{RS}_i \parallel DLB_{RAM})$. In this construction, $\bar{RS}_i$ serves as a random nonce, and rules out the possibility of P^* successfully passing the verification without storing the full DLB_{RAM}.

The response calculation must be such that it cannot be delegated to a helper. This requirement is for achieving security against rCF (Section 4). We thus use a *keyed-hash message authentication code,* that requires the prover's secret key. The keyed-hash message authentication code uses a suitable cryptographic hash function in a specific structure (e.g. HMAC), to construct a secure MAC, which ensures the keyed-hash value cannot be forged.

4 Security analysis of Π_{DLB-BM}

Π_{DLB-BM} protocol uses a PRF and HMAC, and we analyse its security against a computationally bounded adversary. We note that it is possible to construct an information theoretically secure version of this protocol, by replacing the PRF and HMAC with appropriate primitives.

rDF$^{[BM]}$ resistance. In DF, a malicious prover P^* with $d(loc_v, loc_{P^*}) \leq B$, wants to prove, its distance is higher than the bound. To achieve this goal, P^* must send the correct response bit r_i, and the correct proof of erasure h_i, both with sufficient delay, in all rounds $1 \leq i \leq n$, of the fast-exchange phase. Theorem 2 proves that the DF resistance of the DLB protocol Π_{DLB}, assuming that a malicious code of length at least g bits is required.

Theorem 2. *Π_{DLB-BM} is ϵ-resistant to rDF$^{[\mathtt{BM}]}$, with,*

$$\epsilon <= \max\left(2^{-(\frac{g}{2})^2}, 2^{-n(n+1)}\right),$$

against any rDF$^{[\mathtt{BM}]}$ attack that requires at least g-bit malicious code, assuming $H_x()$ is HMAC with a suitable cryptographic hash function.

rMiM$^{[\overline{\mathtt{OC}}]}$ resistance. In rMiM$^{[\overline{\mathtt{OC}}]}$, the adversary cannot send or receive signal to, or from the prover during the fast-exchange phase of the target instance. It

however has full communication power during other phases. We do allow the adversary to jam communications between the verifier and provers in all phases of the protocol (including fast exchange phase). The proof outline of the following Theorem 3 is given in the appendix E.

Theorem 3. *The DLB protocol Π_{DLB-BM} is β-resistant to rMiM$^{[\overline{\texttt{OC}}]}$ attack with $\beta = 2^{-l}$, by choosing $b > \frac{l}{n} - 1$.*

rCF$^{[\texttt{BM},\ \overline{\texttt{OC}}]}$ resistance.

Providing rCF security requires security against rDF and rMiM, and so their associated assumptions. We consider rCF$^{[\texttt{BM},\ \overline{\texttt{OC}}]}$, and show (i) this is a stronger attack than rDF$^{[\texttt{BM}]}$ (see Appendix F for details), and (ii) Π_{DLB-BM} is secure against this attack (see Theorem 4 and its formal proof is given in Appendix G).

Theorem 4. *The protocol Π_{DLB} is (γ, η)-resistant to rCF$^{[\texttt{BM},\ \overline{\texttt{OC}}]}$, with*

$$\gamma \leq \max\{2^{-\frac{g+0.5}{2}\frac{g+1}{2}}, 2^{-(n+0.25)(n+1)}\} \quad \textit{and} \quad \eta = 2^{-l},$$

assuming the malicious code is at least g bits.

5 Practical consideration

We noted earlier that although time is essential for distance estimation, it cannot be used as a single-handed resource to provide DLB guarantees because signals can be easily delayed by malicious provers to show they are farther. Security of DLB will thus rely on making physical assumptions about the device and the communication channel(s). Our basic DLB protocol relies on a memory-bounded device with known memory size, and assuming that attacks are software based and through codes that reside in memory. That is all other registers and memories (e.g. ROM, write protected flash memory, etc) are inaccessible to the malicious code. These assumptions are not unrealistic and have been made in previous work [25]. In the following, we discuss the parameter choices of the DLB protocol and analyze its performance in terms of time, energy, and memory consumption on a MicaZ sensor [22] with a TinyOS host. Below we give some specifications about the MicaZ sensor and explain our design of the DLB protocol. To be secure against real-time attacks, we seek for 10^{-6}-resistance to distance and restricted-collusion frauds. The security analysis of Section 4 shows resistance against malicous codes of length at least 1 byte, which quite reasonable.

The computation during the initialization and challenge-response distance estimation is similar to DUB protocols and so the excellent works [27] on the implementation of DUB protocols can be used to have performance estimates. However using erasing sequence is unique to Π_{DLB-BM}. We estimate memory, time, and energy consumption of Π_{DLB-BM} on a MicaZ sensor [22] using TinyOS. We assume the following parameters: $n = 10$ rounds and $k = 192$ bits of nonces.

We use HMAC-SHA1, denoted by $HMAC(.;.)$, for the PRF, f_x, and generation of response for RS_i, $H_{\hat{x}}$. That is, $f_x(N_p, N_v)$ equals the first $2n = 20$ bits of $HMAC(x; N_p||N_v)$ and $H_{\hat{x}}(\bar{RS}_i||DLB_{RAM})$ equals $HMAC(\hat{x}; \bar{RS}_i||DLB_{RAM})$ of length $b = 160$ bits.

MicaZ sensor specifications. The device is supplied by two AA batteries and includes an ATMEGA128 microcontroller and a TI-CC2420 radio transceiver. The micro-controller provides 4KB of writable memory (SRAM), with 4KB of EEPROM and 640 KB of write-protected flash memory. The radio transceiver chip works for an RF band of 2.4–2.48 GHz and has 250 Kbps data rate.

Memory consumption. HMAC-SHA1 takes code size of 4650 bytes [25] for implementation on ROM, and around 124 bytes of RAM to load data structures and stack. Considering $l = 128$ of secret key x, we have a reasonable estimation of code size to be 10KB. Although the EEPROM is only 4KB large, the ATMEGA128 architecture allows for ROM extension via the use of mask ROM, locked flash memory, and fuse bits. Using these extension methods, one can build read-only memory of size 10KB or more. Note that in order to obtain maximum energy consumption, we assume the size of DLB_{RAM} to be 0.

Energy and time consumption. The writable memory in ATMEGA128 (when flash memory is write-protected) is the 4KB SRAM. For a $n = 10$ rounds DLB protocol, the erasing sequence has length 32758 bits on average.

Communication costs. The protocol requires the prover to receive N_v, M in initialization phase and c_i, RS_i in each fast-exchange round, which gives a total of $l_{rx} = len(N_v) + len(M) + 2n + n \times len(RS_i) = 326912$ bits. There is also requirement for sending N_p, r_i's, and h_i's which sums to $l_{tx} = len(N_p) + n + n \times len(h_i) = 1802$ transmission bits. Sending (resp. receiving) a single bit requires $E_{rx} = 2.34\mu J/b$ (resp. $E_{tx} = 4.6\mu J/b$) by the radio transceiver with typical power adjustments [9]. The total communication energy is thus $E_{comm} = l_{tx}E_{tx} + l_{rx}E_{rx} = 773mJ$.

Computation costs. The main part belongs to computing h_i. The remaining computation is negligible. Extrapolating the figures for memory erasure phase in [25] to our 4KB-memory device, we require less than 600 milliseconds time for computing proof of erasure, which is quite practical. As for energy consumption, each HMAC-SHA1 computation uses $3.5\mu J$ per memory byte. Considering the memory size and the number of rounds, the required computation energy is obtained as $E_{comp} = 3.5 \times 10^{-6} \times 4 \times 2^{10} \times 10 = 140mJ$.

Each AA battery is capable of delivering 1.2 Amperes under an average voltage of 1.2 Volts for one hour [9], implying the power supply of $10,368J$ via the two batteries. This means the proving device can be used for approximately $\frac{10,368}{(773+140)10^{-3}} > 11,000$ runs of DLB protocol before the batteries die. This is quite a reasonable turn out for power consumption. Although we should consider idle/sleep mode energy consumption for more accurate analysis, this consideration will not cause a drastic change on the above result.

6 Concluding remarks

We motivated the novel security problem of DLB in the setting that the prover is not trusted using a number of application scenarios, and gave formal definition of security against three general classes of attacks (DF, MiM and CF). We proved that it is impossible to provide security against any of these attacks without making physical assumptions. Our results show that an adversary, even if it is computationally bounded, will *always succeed* in DF if it has unrestricted access to the prover's device (fully untrusted prover), and will succeed in MiM attacks, if it has unrestricted access to the communication channel. And security against CF requires restrictions on both types of accesses. These results show a fundamental difference between DLB and DUB problems. The only physical assumption in DUB protocols, is that the speed of EM signals is constant. In DLB protocols however, in addition to this assumption, one must assume other restrictions on the physical access of the adversary.

Our protocol considers a malicious prover that has restricted access to the prover device, and an external attacker that has restricted access to the communication channel to the prover. We provides security against $\text{rDF}^{[\texttt{BM}]}$, $\text{rMiM}^{[\overline{\texttt{OC}}]}$, and $\text{rCF}^{[\texttt{BM},\,\overline{\texttt{OC}}]}$, using reasonable assumptions that have been used in theoretical cryptography as well as security systems in practice, including systems for secure code update [25]. Enforcing assumptions in practice would need special technologies such as targeted jamming [18].

One can replace the above assumptions with other reasonable assumptions. The assumptions are necessary to prevent the adversary from introducing delay in the challenge and response time estimation phase. For example, instead of assuming bounded memory, one can use a software-based *externally verifiable code execution (EVCE)* system such as Pioneer [30], to guarantee that the *target executable* code associated with the distance measurement, is executed without modification by a malicious code that may reside on the device. Using this approach, in each challenge-response round, the device first proves that it has an untampered environment, and then provides the response. Note that EVCE also assumes a trusted network to eliminate proxy attacks allowing the construction to provide security against $\text{rMiM}^{[\overline{\texttt{OC}}]}$ and $\text{rCF}^{[\texttt{SA},\,\overline{\texttt{OC}}]}$. More details on an EVCE based DLB protocol is given in Appendix B. The important point to note is that *one must restrict the adversary's physical access to the environment to achieve any DLB security.*

Our primary application scenarios of DLB in this paper were examples of proximity-based access control. Other application scenarios in DLB may have different security requirements. Examining these requirements will be an important step in modelling security and designing secure protocols. Another interesting question is to efficiently incorporate DLB in DUB protocol to provide security against distance enlargement.

Bibliography

[1] G. Avoine, C. Lauradoux, and B. Martin. How secret-sharing can defeat terrorist fraud. In *ACM conference on Wireless network security*, 2011.

[2] G. Avoine and A. Tchamkerten. An efficient distance bounding rfid authentication protocol. In *Information Security*, pages 250–261. 2009.

[3] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. On the pseudorandom function assumption in (secure) distance-bounding protocols. In *LATINCRYPT*.

[4] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Practical & provably secure distance-bounding.

[5] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Secure & lightweight distance-bounding. In *LightSec 2013*.

[6] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Towards secure distance bounding. *FSE 2013*.

[7] S. Brands and D. Chaum. Distance-bounding protocols. In *EUROCRYPT 1993*, pages 344–359.

[8] C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In *CRYPTO'97*.

[9] M. Calle and J. Kabara. Measuring energy consumption in wireless sensor networks using gsp. In *Personal, Indoor and Mobile Radio Communications*.

[10] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, 2006.

[11] D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. Lipton, and S. Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In *Theory of Cryptography*. 2007.

[12] C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *S&P*, pages 113–127.

[13] G. Di Crescenzo, R. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In *Theory of Cryptography*.

[14] S. Dziembowski. Intrusion-resilience via the bounded-storage model. In *Theory of Cryptography*, pages 207–224. 2006.

[15] M. Fischlin and C. Onete. Subtle kinks in distance-bounding: an analysis of prominent protocols. In *S & P in wireless and mobile networks*.

[16] M. Fischlin and C. Onete. Terrorism in distance bounding: modeling terrorist-fraud resistance. In *Applied Cryptography and Network Security*.

[17] A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.

[18] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu. They can hear your heartbeats: non-invasive security for implantable medical devices. In *ACM SIGCOMM*, pages 2–13, 2011.

[19] G. P. Hancke. Distance-bounding for rfid: Effectiveness of terrorist fraud in the presence of bit errors. In *RFID-Technologies and Applications*, 2012.

[20] G. P. Hancke and M. G. Kuhn. An rfid distance bounding protocol. In *SecureComm*, pages 67–73, 2005.

[21] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. Global positioning system. theory and practice. 1993.
[22] C. T. Inc. Micaz datasheet.
[23] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The swiss-knife rfid distance bounding protocol. In *ICISC 2008*, pages 98–115.
[24] A. Mitrokotsa, P. Peris-Lopez, C. Dimitrakakis, and S. Vaudenay. On selecting the nonce length in distance-bounding protocols. *The Comp. Journal*.
[25] D. Perito and G. Tsudik. Secure code update for embedded devices via proofs of secure erasure. In *Computer Security–ESORICS 2010*. 2010.
[26] A. Ranganathan, N. O. Tippenhauer, B. Škorić, D. Singelée, and S. Čapkun. Design and implementation of a terrorist fraud resilient distance bounding system. In *ESORICS 2012*, pages 415–432.
[27] K. B. Rasmussen and S. Capkun. Realization of rf distance bounding. In *USENIX Security Symposium*, pages 389–402, 2010.
[28] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun. Proximity-based access control for implantable medical devices. In *Computer and communications security*, pages 410–419, 2009.
[29] J. Reid, J. M. G. Nieto, T. Tang, and B. Senadji. Detecting relay attacks with timing-based protocols. In *Information, comp. and comm. security*.
[30] A. Seshadri, M. Luk, A. Perrig, L. v. Doorn, and P. Khosla. Externally verifiable code execution. *Communications of the ACM*, pages 45–49, 2006.
[31] D. Singelée and B. Preneel. Distance bounding in noisy environments. In *Security and Privacy in Ad-hoc and Sensor Networks*, pages 101–115. 2007.
[32] B. Song and C. J. Mitchell. Rfid authentication protocol for low-cost tags. In *Wireless network security*, 2008.
[33] N. O. Tippenhauer, K. B. Rasmussen, and S. Capkun. Secure ranging with message temporal integrity. *IACR Cryptology ePrint Archive*, 2009.
[34] S. Vaudenay, I. Boureanu, A. Mitrokotsa, et al. Practical & provably secure distance-bounding. In *The 16th Information Security Conference*.
[35] J. S. Warner and R. G. Johnston. A simple demonstration that the global positioning system (gps) is vulnerable to spoofing. *Journal of Security Administration*, 25(2):19–27, 2002.

A Related Work

A.1 Secure DUB protocols

A distance upper bounding (DUB) protocol obtains an upper bound on the distance between a prover and a verifier. Secure DUB protocols [5, 20, 27, 29] estimate the distance through measuring the round-trip time of simple challenge-response messages. The three commonly considered attacks on DUB protocols are, *distance fraud, Mafia fraud and terrorist fraud*. In [12], the authors proposed a new type of attack called *distance hijacking*, as an extension to distance fraud.

Brands and Chaum proposed the first pre-commitment DUB protocol[7]. Hancke and Kuhn then proposed a pre-computation DUB protocol[20] that can

be easily implemented on RFID devices. Singelee et. al [31] considered distance bounding in noisy environment and proposed a protocol that can tolerate noise. None of the above protocols is secure against terrorist fraud.

Protocols with claimed security against terrorist fraud include [1, 23, 26, 29]. These protocols used different approaches to provide this security. Reid et. al [29] proposed to combine the response with the long-term secret key so that knowing the whole response table enables the helper to recover the secret key and impersonate the prover in future. Kim et. al [23] noted that Reid et. al's protocol will be insecure if the adversary can see the result of protocol. Avoine et. al [1] used threshold secret sharing to prevent terrorist fraud.

Recently, a number of new attacks against protocols that were believed to be secure, have been proposed[3, 12, 19, 24]. These attacks have generated further interest in formalizing security of distance upper bounding protocols. In [1] a semi-formal model for secure distance-bounding is given; [15] and [16] give a formal model capturing resistance to terrorist fraud; [12] proposes a detailed list of known attacks against distance bounding protocol and gives a formal security model in multi-party setting; and [4] provides a security model in terms of resistance to three general classes of attacks that include other known attacks as special cases. We will follow this last approach.

Boureanu et. al framework Boureanu et. al [4, 5, 6] showed security weaknesses of a number of protocols and proposed a security driven design for a class of protocols called SKI, which builds on the previous key papers including [1, 29]. Our distance estimation sub-protcol follows the design approach of SKI with security only for non-noisy environments. This allows a simplified design.

A.2 Bounded memory and bounded retrieval model

Memory bounded adversaries in cryptography were first considered in information theoretic setting [8]. A related model is Bounded-Retrieval Model (BRM) where the adversary is limited in the number of bits that it can access. BRM [11, 13, 14] is a well established leakage model where leakage parameter is an arbitrary and independent parameter in the system design showing the absolute amount of leakage (to the adversary) that the system can tolerate.

B Software attestation based DLB protocol

In the setting of $\text{rDF}^{[\text{SA}]}$, we do not assume a root of trust on the PD. Instead, we establish a dynamic root of trust, and guarantee correct execution of the fast-exchange phase, as part of the DLB protocol. In [30], a software-based *externally verifiable code execution (EVCE)* system (called Pioneer) is proposed to assure correct code execution over an untrusted device. It guarantees that a piece of code, referred to as*target executable*, executes untampered from possible malicious codes that may reside on the device, by dynamically establishing

a *root of trust* and secure execution environment, on the device. The protocol $\Pi_{DLB-EVCE}$ uses EVCE in each fast-exchange round.

The fast exchange phase uses the *externally verifiable code execution (EVCE)* system proposed in [30]. In EVCE, the goal is to ensure that the correct target code is invoked, and the execution is untampered in the sense that, other than performing denial of service attacks, no malware that may exist on the computing device can interfere with the execution of the target code. In this model, no root of trust on the device is assumed: the verifier establishes an isolated execution environment on the device using the verification function, and uses checksum and time to ensure the correct execution of the verification function; it then calculates the keyed-hash value of the target code and sends it to the verifier, assuring the integrity of the target executable. By giving the verification function and the target code the highest privilege level of the CPU, the execution of the verification function and the target executable will be atomic, and no untrusted code can execute before the above two codes complete their execution. EVCE requires the target executable code to be self-contained. To satisfy this requirement, we propose protocol $\Pi_{DLB-EVCE}$ that uses EVCE in each round of DLB.

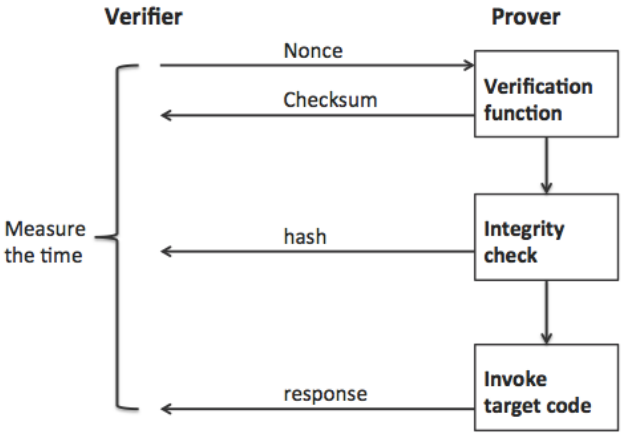

Fig. 5: software attestation based DLB protocol

$\Pi_{DLB-EVCE}$**.** The initialization and verification phases of this protocol are as in Section 3.1. Figure 5 shows a fast-exchange round of the protocol. In each round, the verifier sends a nonce that invokes the verification function of the prover. The verification function computes the checksum over itself, and sets up an isolated execution environment. The keyed-hash value of the target executable is calculated for the purpose of integrity check, and then the target code is invoked. The target code consists of the function used for finding the response

to the nonce, and sending the response to the verifier. The time T_Δ is measured from the time that the nonce is sent, until the time that the response is received. The round trip time is then calculated as $T_\Delta - T_{proc}$, where T_{proc} consists of the time used for, (i) finding the checksum of the verification function, (ii) integrity check of the target code, and (iii) finally its invocation. The main drawback of the protocol however, is the accuracy of the distance estimation that depends on correct estimation of T_{proc}. A correct estimation of T_{proc} in turn depends on the correct estimation of these three components, and so high accuracy would be hard to achieve.

Security analysis of $\Pi_{DLB-EVCE}$. Because of space, we only give an outline of the security proof. DF-resistance of $\Pi_{DLB-EVCE}$ follows from the correct execution of fast-exchange rounds, which is guaranteed because of the security of software-based attestation EVCE. This means that no delay will be introduced in the execution of the protocol, and at the end of the fast-exchange phase, the distance will be correctly estimated. EVCE also requires a trusted network to eliminate proxy attacks where the computing device asks a faster computer (proxy) to compute the checksum on its behalf, and so for this construction only rMiM$^{[\overline{\mathsf{OC}}]}$ and rCF$^{[\mathsf{SA},\,\overline{\mathsf{OC}}]}$ can be considered. That is, any other restrictive assumptions for rMiM and rCF must include no-online communication. Security against rMIM and rCF follows from secure generation and sharing of the response table that ensures that the response bits cannot be guessed by outsiders (See Section 4).

C Proof sketch of Theorem 1

For (1), assume a malicious prover (who can calculate correct responses to the verifier challenges) at $D < B$. To claim a longer distance $D + D'$, the prover modifies the execution to add appropriate delay by tampering with the hardware/software and responds after $2D'/C$ second(s). The attack succeeds with probability 1.
For (2), A MiM attacker can use the following strategy: upon receiving a message from one party, the adversary jams the signal to prevent it from being received by the other, and later forwards it with appropriate delay. The theorem holds irrespective of any hardware assumption (e.g. bounded memory) on the prover's device, and succeeds even if jam and delay can be applied in one direction only.
For (3), note that CF resistance requires both DF resistance and MiM resistance: A CF attacker can simply simulate a successful DF attacker by simply ignoring the helper; or it can also simulate a successful MiM attacker, by allowing P^* in the CF attack to run the algorithm of P, and the helper in CF to run the algorithm of the MiM adversary, A^{MiM}.

D Proof of Theorem 2

Proof. A dishonest prover P^* succeeds if it passes verification of all rounds. We show that for *all* P^*'s possible strategies, its success probability is bounded. A

P^*'s strategy σ, is defined by a sequence of actions that it will take over the n rounds. P^* needs a malicious code of size at least g to implement its strategy. The code must be stored in the PD's RAM. In each round, P^* must dedicate g bits of RAM for the malicious code MC, by either over-writing the response table $A^{[i]}$, or RS_i, or DLB_{RAM}, or part of each. Here $A^{[i]}$ is the un-used part of A at the start of round i. It is important to note that success probability of P^* in each round, depends on the action taken in the current round, and all actions taken in all the previous rounds. For example if P^* has overwritten $(a_{(1,i)}, a_{(2,i)})$, during an earlier round j, where $j < i$, then the success probability of producing the correct response to c_i, will be at most $1/2$.

Let $\Pr(Succ^{\sigma}_{DF})$ denote the prover's success probability for a strategy σ (n-round strategy, possibly adaptive) used by P^*. Let S_i denote the event associated with the success in round i, $1 \leq i \leq n$. We have the following:

$$Pr(Succ^{\sigma}_{DF}) = \Pr(\bigwedge_{i=1}^{n} S_i) = \prod_{i=1}^{n} \Pr(S_i | S_{i-1}, \ldots, S_1).$$

Because of the properties of probability, for all rounds i, we have

$$\Pr(S_i | S_{i-1}, \ldots, S_1) \leq 1$$

In round i, the P^*'s device receives a challenge symbol c_i, followed by $L - \lambda - 2(n-i)$ bits of RS_i. The response consists of r_i, and $h_i = H_{\hat{x}}(\bar{RS_i} || DLB_{RAM})$. Because of the unforgeability of HMAC, to calculate h_i, the string RS_i must be fully stored, and DLB_{RAM} must remain intact. If some of these bits, say ℓ, are overwritten, to generate the correct response, the ℓ missing bits can be guessed., with the success probability $2^{-\ell}$.

Let g be even and smaller than the original value of the response table, $g \leq 2n$. In each round, 2 bits of the this table is used and the erasing sequence will be lengthened by 2 bits to overwrite them. The reduction in the size of the table in each round finally reaches a round $n_0 \stackrel{\triangle}{=} n - \frac{g}{2}$, after which the size of $A^{[i]}, i > n_0$, is less than the malicious code. That is, $2(n-i) < g$ and the length of RS_i satisfies $L - 2(n-i) > L - g$. From round $i > n_0$, to keep the g bit malicious code, some bits from RS_i must be overwritten and this number equals,

$$g - 2(n-i) = g - 2(n_0 + \frac{g}{2} - i) = 2(i - n_0).$$

This leads to a success chance of $2^{-(2(i-n_0)+1)}$ in calculating h_i in round i. The overall success chance is given by,

$$\begin{aligned} \Pr(Succ^{\sigma}_{DF}) &\leq \prod_{1 \leq i \leq n_0} 1 \times \prod_{n_0+1 \leq i \leq n} 2^{-(2(i-n_0))} \\ &= 2^{-\left(\sum_{i=n_0+1}^{n} 2(i-n_0)\right)} = 2^{-\left(\sum_{i'=1}^{n-n_0} 2i'\right)} = 2^{-(\frac{g}{2})(\frac{g}{2}+1)} < 2^{-(\frac{g}{2})^2}. \end{aligned}$$

If g is odd:. An argument similar to the above, shows that the prover needs to drop $2(i-n_0)-1$ bits in rounds $i > n_0 =\stackrel{\triangle}{=} n - \frac{g+1}{2}$. The success probability is thus obtained as,

$$\begin{aligned}\Pr(Succ^{\sigma}_{DF}) &\leq 2^{-\left(\sum_{i=n_0+1}^{n} 2(i-n_0)-1\right)} = 2^{-\left(\sum_{i'=1}^{n-n_0} 2i'-1\right)} \\ &= 2^{-(\frac{g+1}{2})(\frac{g+1}{2}+1)}.2^{\frac{g+1}{2}} = 2^{-(\frac{g+1}{2})^2} < 2^{-(\frac{g}{2})^2}.\end{aligned}$$

If $g \geq 2n$. Here, the prover needs to drop some bits of the erasing string RS_i in all rounds $1 \leq i \leq n$; in other words, $n_0 = 0$ and the prover's success chance is,

$$\Pr(Succ^{\sigma}_{DF}) \leq \prod_{1 \leq i \leq n} 2^{-(2i)} = 2^{-\left(\sum_{i=1}^{n} 2i\right)} = 2^{-n(n+1)}.$$

This means that the success probability of P^* in *any strategy* is bounded, and the proof is complete.

E Proof outline of Theorem 3

- If the target DLB instance does not include a prover, the MiM attack is the same as an impersonation attack. To succeed in this attack, the attacker needs to guess the secret key or guess all responses correctly. It can be seen (and formally proved) that the initialization phase, and the prover's responses in the fast-exchange phase, do not leak prover's secret key, resulting in the success probability equals to 2^{-2l}, assuming $n(b+1) > 2l$.
- Consider the case that the target instance includes a prover. The adversary can succeed if it can jam the messages from prover to the verifier, and send its own response. The adversary can only see the initialization phase, and not the prover's messages in the fast exchange phase of the target DLB instance. Thus again, the adversary needs to either guess the key, or guess all challenge and proof-of-erasure values, for all rounds. The probability of the attack will thus be no more than 2^{-2l}.

F Argument for rCF is more powerful than rMiM

We show rCF$^{[\mathtt{BM},\ \overline{\mathtt{OC}}]}$ is more powerful than rDF$^{[\mathtt{BM}]}$by giving a protocol that is rDF$^{[\mathtt{BM}]}$ resistant, but not rCF$^{[\mathtt{BM},\ \overline{\mathtt{OC}}]}$ resistant. Consider protocol Π^*_{DLB-BM} which is the same as Π_{DLB-BM} in Section 3.1, but uses the hash function without the secret key. By using argument similar to Section 4, we prove Π^*_{DLB-BM} is rDF$^{[\mathtt{BM}]}$ resistant. However now the response can be constructed by anyone. The colluders can plan their responses as follows: P^* will delay the response to the challenge symbol at will, and the helper prepares and sends the response to RS_i. This shows that Π^*_{DLB-BM} is not rCF$^{[\mathtt{BM},\ \overline{\mathtt{OC}}]}$ resistant.

G Proof of Theorem 4

Proof. In the rCF$^{[\mathtt{BM},\ \overline{\mathtt{OC}}]}$, an L-bit-memory malicious prover P^* at distance $D < B$ wants to claim a distance $D \geq B$ by the help of an adversary A^{CF} who cannot communicate with the prover during the fast-exchange phase of DLB. The choice of $\eta = 2^{-l}$ in the theorem implies that the prover cannot leak any part of its key to the helper since any leakage would lead to a nOC-BM-MiM with probability $> 2^{-l}$. We thus study the best success probability ($\Pr(Succ_{CF})$) of nOC-BM-CF assuming no key leakage to the helper. We note that the prover still can leak part of its response table, which is build as a result of initialization: For each round of fast-exchange i, the prover's response bit to the challenge $c_i \in \{1, 2, 3\}$ is $r_i = a_{c_i,i}$ where $a_{1,i}$ and $a_{2,i}$ come from the PRF output and $a_{3,i} = a_{1,i} \oplus a_{2,i} \oplus x_i$, where x_i is a bit of the secret-key. So, as long as only two out of the three bits $(a_{1,i}, a_{2,i}, a_{3,i})$ are passed by the prover, there is no secret-key leakage and $\eta = 2^{-\ell}$ is satisfied. Hence, without loss of attack success optimality, we let the adversary have exactly two elements (e.g., the first two) from the triple $(a_{1,i}, a_{2,i}, a_{3,i})$ for each and every round $1 \leq i \leq n$. In analogy to distance fraud, we let P^* store a malicious code of $g > 0$ in RAM and define $n_0 = n - \frac{g-1}{2}$.

Let $\Pr(Succ^{\sigma}_{CF})$ denote the prover's success probability for a strategy σ (n-round strategy, possibly adaptive) used by P^* and A^{CF}. Let S_i denote the event associated with the success in round i, $1 \leq i \leq n$. Similar to the DF-security analysis (proof of Theorem 2), we have

$$Pr(Succ^{\sigma}_{CF}) = \prod_{i=1}^{n} \Pr(S_i | S_{i-1}, \ldots, S_1), \quad \text{where} \quad \Pr(S_i | S_{i-1}, \ldots, S_1) \leq 1.$$

To simplify the proof like that for DF security, we assume without loss of optimality all r_i's are calculated successfully (with probability 1) and only focus on the calculation of h_i's. Since there is no communication between the prover and the adversary during fast exchange, the should have made prearrangements about which of them is in charge of sending which information to the verifier. Although some part of the response table is known by the adversary, this cannot help the adversary calculate hash responses h_i with any better probability (since the secret-key used for MAC is not leaked). This follows that the success chance is maximized of the prover calculates and transmits h_i's. The CF success probability thus will be bounded in the same way as that of DF in Theorem 2:

$$Pr(Succ^{\sigma}_{CF}) \leq \max\left(2^{-(\frac{g}{2})^2}, 2^{-n(n+1)}\right).$$