Distance Lower Bounding

Xifan Zheng(✉), Reihaneh Safavi-Naini, and Hadi Ahmadi

University of Calgary, Calgary, AB, 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 [3] 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 [8]), proximity based control (e.g., implantable medical device [15]), and Radio-Frequency Identification (RFID) authentication [1,18].

Secure DUB protocols estimate distance by measuring the round-trip time between a challenge and its response, which are transmitted as light-speed electromagnetic (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

L.C.K. Hui et al. (Eds.): ICICS 2014, LNCS 8958, pp. 89–104, 2015.
DOI: 10.1007/978-3-319-21966-0_7

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.

Despite the relation between DUB and DLB and the fact that a successful DUB protocol run proves an upper bound on the prover's distance, its failure does not say anything about the distance of the prover. None of the DUB protocols protect against distance enlargement attack [6], 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 [6], will also be vulnerable to distance enlargement attack. A second approach would be to use Global Positioning System (GPS)[11] 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 [21] 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 Sect. 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 [20]: *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 problem depends on the trust assumption. DLB problem in a setting that *both the prover and the verifier are trusted*, has been considered in [19]. 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 in Sect. 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 [19])	Possible [a] (c.f., DB [2])
Fully untrusted prover[b]	Impossible (Sect. 2.2)	
Partially trusted prover[c]	Possible (Sect. 3, Π_{DLB-BM})	

[a] DUB protocols with fully untrusted prover are secure in other trust settings.
[b] Malicious prover with unrestricted control of the prover device hardware/software.
[c] Restricted Malicious prover who can run malicious software on the prover device.

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), Man-in-the-middle attack (MiM) and collusion fraud (CF). Third, we prove that security against any of these attacks without making *physical assumptions*[1] 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 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.

Related Work. There is a large body of research on secure positioning and distance estimation, including distance bounding protocols [2,3,10], positioning techniques [6,11] and secure ranging protocols [19]. 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 [21]; DUB protocols protect against malicious provers trying to shorten the distance, but are in general vulnerable to the distance enlargement attack [6]; 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.

[1] Including limited access to the device hardware, and/or the communication channel.

To our knowledge, this is the first paper to study DLB 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, [20] for formalization of attacks, and [2,10] 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 [13]. We refer to [22]. for a complete review of relevant works.

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. 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 protocol instance defines an experiment denoted by $exp = (P(x; r_P) \leftrightarrow \mathcal{A}(r_A) \leftrightarrow V(y; r_V))$. 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.

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*

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

2.1 Attacks on DLB Protocols

We consider three classes of attacks: distance fraud (DF), man-in-the-middle (MiM) attack, 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 Fig. 1a). In

Fig. 1. MiM and CF attack in DLB

CF, P^* colludes with a helper to claim a longer distance to V (See Fig. 1b). The collusion should not leak the prover's secret key to the helper. The formal definitions of the attacks are below.

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*

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

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* [7] 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 $\geq B$.

Definition 3 (MiM-resistance). *A DLB protocol* Π *is* β*-resistant to MiM attack if* $(\forall s)$*,* $(\forall m, \ell, z)$ *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*

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

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.*

Mafia fraud and impersonation attack. Definition 3 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. This attack is captured by letting $\ell = m$.

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* $d(loc_{v_0}, loc_{P^*}) < D)$*,* $(\forall A^{CF} ppt.)$*:*

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

implies existence of an extended[2] *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 learning and satisfies*

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

Here, prover $P^{(*)}$ *is either* P *or* P^* *and we have* $d(loc_{P_j}, loc_V) < B$, *for* $j \in \{m+1\cdots\ell\}$. *We implicitly allow a polynomial number of* $P(x')$, $P^*(x')$, *and* $V(y')$ *with independent* (x', y') *anywhere but honest participants are close to* V_0.

Terrorist fraud. In Terrorist fraud, P^*, with $d(P^*, V) < B$, gets aid from a helper who does not have the secret key. 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. If the received response is correct, the verifier calculates $T_\Delta = t_r - t_c$ to estimate the distance of the prover. Let T_{proc} denote the *processing time* of the prover. The verifier estimates the prover's distance D as

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

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 by an external attacker who can jam and delay messages to/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 A.

2.3 Restricted DF, MiM, and CF

To remove the above impossibility results, we must 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.

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

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, $\mathrm{rDF}^{[BM]}$ refers to the restricted DF attack, under the restriction that PD has bounded memory.

Table 2 summarizes our impossibility results and shows assumptions used in our construction in Sect. 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 [4], and has been used in the design of security systems [13]. To achieve the security against rMiM and rCF, in addition to the above assumptions, we require the helper to have no *On-line Communication (OC)* with the prover during the fast-exchange phase. In Sect. 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 full version [22]. 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 the PD is a bounded memory device with a protected memory (ROM), and a writable memory (RAM) of (fixed) L bit size. 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 [17]. 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 Eq. 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* PDand enables V *to force an upper bound on the delay introduced by* P^*.

3.1 The Protocol Π_{DLB-BM}

The secret key consists of two binary strings, x, and $\hat{x}$ in $\{0,1\}^\ell$ respectively. 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 2 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 Sect. 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 Sect. 3.2 for details). In addition, the verifier records the time difference $T_{\Delta,i}$ between sending c_i and receiving r_i.

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

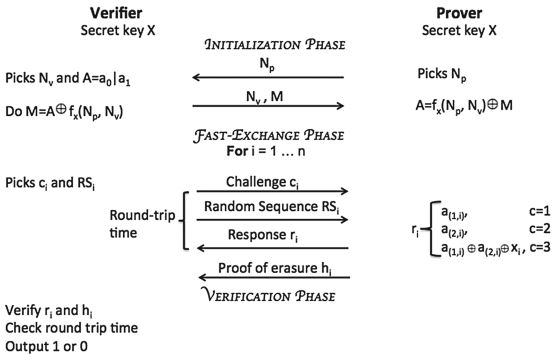

Fig. 2. Distance lower bounding protocol Π_{DLB-BM}

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 Eq. 2). The verifier outputs $Out_v = 1$, if and only if all verifications and time checks succeed. For simplicity, we assumed the communication channel is noiseless; thus, 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. 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 3 shows the state of the 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 [13]. 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

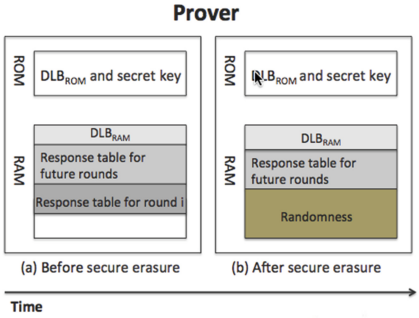

Fig. 3. Prover's memory during protocol execution

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 (Sect. 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$^{[BM]}$, with $\epsilon = \max\left(2^{-(\frac{g}{2})^2}, 2^{-n(n+1)}\right)$, against any rDF$^{[BM]}$ attack that requires at least g-bit malicious code, assuming $H_x()$ is HMAC with a suitable cryptographic hash function.*

The proof is given in Appendix B. The theorem implies that attacks with longer codes directly reduce the success probability of P^*. This substantially limits designing malicious codes. Note that for a 1-byte malicious code ($g = 8$) leads to a success chance of $\epsilon < 10^{-6}$, for protocols with at least 4 rounds, $n \geq 4$.

rMiM$^{[\overline{\texttt{OC}}]}$ resistance. In rMiM$^{[\overline{\texttt{OC}}]}$, 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 full version [22].

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$^{[\texttt{BM}]}$, and (ii) Π_{DLB-BM} is secure against this attack (see Theorem 4 and its formal proof is given in [22]).

Theorem 4. *The protocol Π_{DLB} is (γ, η)-resistant to rCF$^{[\texttt{BM},\ \overline{\texttt{OC}}]}$, with $\eta = 2^{-l}$ and $\gamma \leq \max\{2^{-\frac{g+0.5}{2}\frac{g+1}{2}}, 2^{-(n+0.25)(n+1)}\}$, assuming malicious code length $\geq g$.*

5 Practical Consideration

The computation during the initialization and challenge-response is similar to DUB protocols and so the excellent works [14] on the implementation of DUB protocols can be used for 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 [12] 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 4 KB of writable memory (SRAM), with 4 KB 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 [13] 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 10 KB. Although the EEPROM is only 4 KB 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 10 KB 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 [5]. The total communication energy is thus $E_{comm} = l_{tx}E_{tx} + l_{rx}E_{rx} = 773mJ$.

Computation Costs. The cost is highly due to computing h_i. The rest is negligible. Extrapolating memory erasure phase figures in [13] 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 [5], 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 provides security against rDF$^{[\mathtt{BM}]}$, rMiM$^{[\overline{\mathtt{OC}}]}$, and rCF$^{[\mathtt{BM},\ \overline{\mathtt{OC}}]}$, using reasonable assumptions that have been used in theoretical cryptography as well as security systems in practice, including systems for secure code update [13]. Enforcing assumptions in practice would need special technologies such as targeted jamming [9]. One can replace the above assumptions with other reasonable assumptions. For example, instead of assuming bounded memory, one can use a software-based *externally verifiable code execution (EVCE)* system such as Pioneer [16], 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. a trusted network to eliminate proxy attacks allowing the construction to provide security against rMiM$^{[\overline{\mathtt{OC}}]}$ and rCF$^{[\mathtt{SA},\ \overline{\mathtt{OC}}]}$. 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.

A 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. 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. 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}.

B Proof of Theorem 2

A dishonest prover P^* succeeds if it passes verification in all rounds. 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:

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

The properties of probability imply: $\forall i$, $\Pr(S_i|S_{i-1}, \ldots, S_1) \leq 1$. In round i, 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 success probability $2^{-\ell}$.

Let g be even and smaller than the response table original size, $g \leq 2n$. In each round, 2 bits of the table are used and the erasing sequence is lengthened by 2 bits. 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: A similar argument can show 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 equals

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

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,

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

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

References

1. Avoine, G., Tchamkerten, A.: An efficient distance bounding RFID authentication protocol: balancing false-acceptance rate and memory requirement. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 250–261. Springer, Heidelberg (2009)
2. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: Secure and lightweight distance-bounding. In: Avoine, G., Kara, O. (eds.) LightSec 2013. LNCS, vol. 8162, pp. 97–113. Springer, Heidelberg (2013)
3. Brands, S., Chaum, D.: Distance bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
4. Cachin, C., Maurer, U.M.: Unconditional security against memory-bounded adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 292–306. Springer, Heidelberg (1997)
5. Calle, M., Kabara, J.: Measuring energy consumption in wireless sensor networks using GSP. In: Personal, Indoor and Mobile Radio Communications (2006)
6. Capkun, S., Hubaux, J.-P.: Secure positioning in wireless networks. IEEE J. Sel. Areas Commun. **24**(2), 221–232 (2006)
7. Cremers, C., Rasmussen, K.B., Schmidt, B., Capkun, S.: Distance hijacking attacks on distance bounding protocols. In: S&P, pp. 113–127 (2012)
8. Francillon, A., Danev, B., Capkun, S.: Relay attacks on passive keyless entry and start systems in modern cars. In: NDSS (2011)
9. Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., Fu, K.: They can hear your heartbeats: non-invasive security for implantable medical devices. In: ACM SIGCOMM, pp. 2–13 (2011)
10. Hancke, G.P., Kuhn, M.G.: An rfid distance bounding protocol. In: SecureComm, pp. 67–73 (2005)
11. Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J.: Global Positioning System Theory and Practice. Springer, Wien (1993)
12. C. T. Inc., Micaz datasheet
13. Perito, D., Tsudik, G.: Secure code update for embedded devices via proofs of secure erasure. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 643–662. Springer, Heidelberg (2010)
14. Rasmussen, K.B., Capkun, S.: Realization of rf distance bounding. In: USENIX Security Symposium, pp. 389–402 (2010)
15. Rasmussen, K.B., Castelluccia, C., Heydt-Benjamin, T.S., Capkun, S.: Proximity-based access control for implantable medical devices. In: Computer and Communications Security, pp. 410–419 (2009)
16. Seshadri, A., Luk, M., Perrig, A., Doorn, L.v., Khosla, P.: Externally verifiable code execution. Commun. ACM **49**(9), 45–49 (2006)

17. Singelée, D., Preneel, B.: Distance bounding in noisy environments. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 101–115. Springer, Heidelberg (2007)
18. Song, B., Mitchell, C.J.: RFID authentication protocol for low-cost tags. In: Wireless Network Security (2008)
19. Tippenhauer, N.O., Rasmussen, K.B., Capkun, S.: Secure ranging with message temporal integrity. IACR Cryptology ePrint Archive (2009)
20. Vaudenay, S., Boureanu, I., Mitrokotsa, A. et al.: Practical & provably secure distance-bounding. In: the 16th Information Security Conference (2013)
21. Warner, J.S., Johnston, R.G.: A simple demonstration that the global positioning system (gps) is vulnerable to spoofing. J. Secur. Adm. **25**(2), 19–27 (2002)
22. X. Zheng, R. Safavi-Naini, and H. Ahmadi. Distance lower bounding. Cryptology ePrint Archive, Report 2014/xxx (2014). http://eprint.iacr.org/