Mafia Fraud Attack against the RČ Distance-Bounding Protocol

Aikaterini Mitrokotsa
EPFL, Lausanne
Email: katerina.mitrokotsa@epfl.ch

Cristina Onete
CASED & TU Darmstadt, Germany
Email: cristina.onete@cased.de

Serge Vaudenay
EPFL, Lausanne
Email: serge.vaudenay@epfl.ch

***Abstract*—At ACM CCS 2008, Rasmussen and Čapkun introduced a distance-bounding protocol [22] (henceforth RČ protocol) where the prover and verifier use simultaneous transmissions and the verifier counts the delay between sending a challenge (starting with a hidden marker) and receiving the response. Thus, the verifier is able to compute an upper bound on the distance separating it and the prover. Distance bounding protocols should resist to the most classical types of attacks such as distance fraud and mafia fraud. In mafia fraud, a man-in-the-middle adversary attempts to prove to a legitimate verifier that the prover is in the verifier's proximity, even though the prover is in reality far away and does not wish to run the protocol. The RČ protocol was only claiming to resist distance fraud attacks. In this paper, we show a concrete mafia fraud attack against the RČ protocol, which relies on replaying the prover nonce which was used in a previous session between a legitimate prover and the verifier. This attack has a large probability of success. We propose a new protocol called LPDB that is not vulnerable to the presented attack. It offers state-of-the-art security in addition to the notion of location privacy achieved by the RČ protocol.**

I. Introduction

Man-in-the-middle (MITM) attacks are powerful strategies towards breaking the security of authentication protocols. In authentication scenarios, a prover proves its legitimacy to a verifier, and the basic security requirement is resistance to impersonation attacks, i.e. the adversary (who is not legitimate) should not be able to impersonate a legitimate prover. However, by relaying messages between a legitimate prover (who is unaware of the attack) and a legitimate verifier, a MITM adversary always succeeds in impersonating the prover.

Distance-bounding protocols were introduced in 1993 by Brands and Chaum [6] as a countermeasure against MITM attacks. Their idea relies on the observation that, in having to process and then relay transmissions between the prover and the verifier, the adversary has some non-negligible delay in his responses. Thus, if a verifier is equipped with a clock and measures the time-of-flight for challenges and responses, it is able to detect relay attacks. Essentially thus, distance-bounding protocols are an extension of authentication protocols, where the verifier accepts the prover as legitimate if (1) the prover proves to the verifier that it is in the verifier's neighborhood, and (2) the transmission time of the prover's responses is upper-bounded by some threshold value $t_{\max}$. We should note here that distance-bounding works best in scenarios where transmissions are fast and can be assumed to take constant time (e.g. transmissions at light speed for radio waves).

Distance-bounding is a well-researched area of cryptography [1], [3]–[10], [12], [13], [15], [17]–[20], [23], [24], [26], though the approach in most works is mostly informal. The two formal frameworks due to Avoine et al. [3] and Dürholz et al. [15] both concur in defining distance-bounding as an authentication protocol where the verifier is convinced that the prover is in its neighborhood. Both frameworks describe the following four attacks for distance-bounding protocols:

Distance Fraud: The dishonest prover wants to cheat the verifier's clock and prove that it is within the threshold distance when it is in fact farther away.

Mafia Fraud: The MITM adversary can leech information from the honest prover $\mathcal{P}$ in order to prove to the honest verifier $\mathcal{V}$ that $\mathcal{P}$ is in proximity although he is far away.

Terrorist Fraud: Here the dishonest prover colludes with the adversary $\mathcal{A}$ in order to help $\mathcal{A}$ successfully pass the protocol. The restriction is that once the dishonest prover stops helping, $\mathcal{A}$ should not be able to prove that the prover is in the verifier's neighborhood. In particular, the dishonest prover is not allowed to disclose the secret key to $\mathcal{A}$.

Impersonation Security: This requirement is a recent idea, first introduced by Avoine and Tchamkerten [5]. Here, the adversary should not be able to impersonate the prover to the verifier, assuming that no pure relay is used.

It is easy to realize the significance of distance-bounding protocols if we take under consideration real-world applications such as bankcards and access control to cars. RFID protocols are frequently used by car manufacturers for the locking/unlocking system in cars. However, it has been shown that these protocols are susceptible to relay attacks [16]. Payment with bankcards [14] are also vulnerable to relay attacks. The main countermeasure against this type of attacks is distance-bounding protocols [14]. Thus, there is an increasing need to use secure distance bounding protocols in order to achieve security and reliability in real-world applications.

Cremers et al. [11] presented a new attack called distance hijacking, which involves an honest and a dishonest prover. The idea is that the dishonest prover uses the honest prover in order to commit distance fraud. Since the standard distance-bounding scenario only consists of a single prover and verifier, this attack is out of scope for models such as [3] and [15].

Distance-bounding protocols typically consist of an *initial-*

ization phase, in which the verifier's clock is not used, and of a *rapid bit exchange* (RBE) or *distance bounding* phase, where the verifier challenges the prover and measures the time-of-flight until it receives the prover's response. Typically this is done on a round-to-round basis [5]–[7], [19]–[21].

However, in 2008, Rasmussen and Čapkun introduced a distance-bounding protocol that aimed to also achieve location privacy. This protocol (the RČ protocol, in short) uses simultaneous transmissions between the prover and verifier, therefore we do not speak of round-based challenges and responses. During the initialization phase, both the prover and the verifier compute a hidden marker M. During the *distance bounding* phase, the prover and verifier begin by sending out random transmissions (in a continuous, simultaneous stream). At some point, the verifier will send the hidden marker and then a randomly-chosen challenge, and will count the delay (in bits) until it receives the response. Thus, the verifier is able to upper-bound its distance from the prover; since we assume constant, fast transmission speeds, the notion of time distance is well defined and corresponds to a physical distance between two parties. We note that the hidden marker is encrypted and signed using two shared secret keys: thus, some measure of authentication is automatically achieved. Rasmussen and Čapkun claim that this protocol is not to be used for authentication, even though it is based on a secret key. This obviously suggests that it offers little security.

Recently Aumasson et al. [2] showed an attack against this scheme, which relies on replaying the nonces chosen by the prover and verifier. This passive dictionary attack may lead to revealing the location of the prover and the verifier. Nevertheless, this attack has a complexity which is exponential in terms of the size of the used nonces and thus, it can be deployed only after eavesdropping a very high number of sessions.

Our Contribution: In this paper we show an efficient mafia fraud attack against the RČ protocol. The attacker first eavesdrops on an honest session between the legitimate prover and the legitimate verifier, then it tries to replay the same nonce as the one used by the prover in a subsequent attempt to pass the protocol. This attack can be run in a mafia fraud setting, to relay the credential of a far away prover to the verifier, which is what distance bounding was meant to avoid. The key vulnerability here is that the prover's and the verifier's messages during the initialization phase are independent of each other, and can thus be replayed. The success of the attack depends on the probability that the adversary guesses the length offsets of the challenge, resp. the response, in the adversary's and resp. the prover's attempts to pass the protocol. Guessing the time offset depends on an adversary's ability to guess the location of the prover.

II. Preliminaries

We consider distance-bounding protocols as outlined in [15], i.e. we consider a single prover $\mathcal{P}$ and a single verifier $\mathcal{V}$ which share a key K generated by some key generation algorithm Kg. The mafia adversary is a MITM type of adversary, which we denote by $\mathcal{A}$. The adversary may eavesdrop on distance-bounding executions between $\mathcal{P}$ and $\mathcal{V}$, and may interact with each of the two parties.

The RČ protocol aims to achieve location privacy. While we do not discuss the topic of location privacy, distance-bounding is automatically associated with some measurements of relative distance between parties. In general, for two parties A and B, we denote by $\Delta\mathsf{t}(A, B)$ the time distance between A and B, i.e. the time it takes for a bit to travel the distance between A and B. We note that, for messages consisting of multiple bits, the sending and receiving times of two different bits of this message may be distinct, depending on the communication protocol used between the two parties.

As far as the communication model goes, we simply note that when an adversary $\mathcal{A}$, placed at distance $\Delta\mathsf{t}(\mathcal{A}, S)$ from a sender S eavesdrops on the transmission of a single bit b between the sender S and a receiver R (in practice, the sender could be either the honest prover or the honest verifier), the bit is eavesdropped by the adversary with a delay corresponding to that time distance. Concretely, if a bit b is sent at time ts by S, it is received by the receiver R at time $\mathsf{tr} = \mathsf{ts} + \Delta\mathsf{t}(R, S)$, and eavesdropped by the adversary at time $\mathsf{tr}_{\mathcal{A}} = \mathsf{ts} + \Delta\mathsf{t}(\mathcal{A}, S)$. We use these notations in the description of our attack.

Furthermore, we note that the success of our attack depends on the adversary's ability to eavesdrop on (and fully reconstruct fragments of) an execution of the RČ protocol between an honest prover and an honest verifier. Rasmussen and Čapkun [22] suggest the use of frequency hopping as a countermeasure to eavesdropping attacks. However, we note that an adversary could simply eavesdrop on all possible frequencies at the same time using for instance a sniffer [25] and subsequently reconstruct the transmission. Thus, we do not consider frequency hopping as an impediment to our attack.

III. The RČ Protocol

In what follows, we outline the distance-bounding protocol due to Rasmussen and Čapkun [22]. This protocol runs in two consecutive phases, at the end of which the verifier $\mathcal{V}$ upper-bounds the distance between itself and the prover $\mathcal{P}$. The two parties share a secret key K for encryption and authentication. Initially, the prover and verifier run the so-called *initialization phase*, where the hidden marker M is generated by the verifier and sent (as part of an encryption) to the prover. Subsequently, the two parties run the *distance bounding* phase.

The protocol uses a symmetric encryption scheme (KGen, $\mathsf{Enc}, \mathsf{Dec}$), and an unforgeable MAC algorithm.

Initialization Phase:

Step 1: The prover $\mathcal{P}$ generates a random nonce $N_{\mathcal{P}}$ of length n. It then computes the encryption $\mathsf{Enc}_K(\mathcal{P}, \mathcal{V}, N_{\mathcal{P}})$ where $\mathcal{P}$ and $\mathcal{V}$ denotes the identities of $\mathcal{P}$ and $\mathcal{V}$ correspondingly. The prover also calculates the MAC of $N_{\mathcal{P}}$ (i.e. $\mathsf{MAC}(N_{\mathcal{P}})$) and sends the concatenation of the two values (i.e. $c_1 = \mathsf{Enc}_K(\mathcal{P}, \mathcal{V}, N_{\mathcal{P}}) \| \mathsf{MAC}(N_{\mathcal{P}})$) to the verifier $\mathcal{V}$.

Step 2: $\mathcal{V}$ receives the value c_1, generates a random variable M of length m that is called *hidden marker* and computes the encryption $\mathsf{Enc}_K(\mathcal{P}, \mathcal{V}, N_{\mathcal{P}})$ as well as the MAC of the concatenation of $N_{\mathcal{P}}$ and M (i.e. $\mathsf{MAC}(N_{\mathcal{P}} \| M)$.

Finally he sends the concatenation of the two results (i.e. $c_2 = \mathsf{Enc}_K(\mathcal{P}, \mathcal{V}, N_\mathcal{P}) \| \mathsf{MAC}(N_\mathcal{P} \| M)$). Finally the verifier $\mathcal{V}$ generates a random nonce $N_\mathcal{V}$ of length n.

Step 3: $\mathcal{P}$ decrypts the value $\hat{c}_2$ and checks that the MAC. If so, the distance bounding phase is run as below; else, the protocol aborts.

Distance Bounding (DB) Phase: In this phase $\mathcal{P}$ and $\mathcal{V}$ transmit simultaneously, in a constant bit stream. The verifier $\mathcal{V}$ transmits a stream $stream_\mathcal{V}$ as follows:

$stream_\mathcal{V} := Rand_{\mathcal{V}_1} \| M \| N_\mathcal{V} \| Rand_{\mathcal{V}_2}$.

The beginning time of this transmission is random. However, it seems that this bit stream is transmitted simultaneously with a stream $stream_\mathcal{P}$ generated by the prover $\mathcal{P}$ such that:

$$\begin{aligned} stream_\mathcal{P} &:= Rand_{\mathcal{V}_1} \oplus Rand_{\mathcal{P}_1} \| \hat{M} \oplus Rand_{\mathcal{P}_2} \| \\ &\quad \hat{N}_\mathcal{V} \oplus N_\mathcal{P} \| Rand_{\mathcal{V}_2} \oplus Rand_{\mathcal{P}_3} \\ &:= Rand_{\mathcal{P}_4} \| \hat{N}_\mathcal{V} \oplus N_\mathcal{P} \| Rand_{\mathcal{P}_5}, \end{aligned}$$

Here, it holds that $Rand_{\mathcal{P}_4} := Rand_{\mathcal{V}_1} \oplus Rand_{\mathcal{P}_1}$ and $Rand_{\mathcal{P}_5} := Rand_{\mathcal{V}_2} \oplus Rand_{\mathcal{P}_3}$. We note that the prover $\mathcal{P}$ parses the received bits from the stream $stream_\mathcal{V}$ and sends its own transmission of $stream_\mathcal{P}$ at the same time; however the two parties begin their simultaneous continuous bit stream exchange at a random time. The distance-bounding properties of the protocol rely on the fact that $\mathcal{P}$'s response is asynchronous. We describe this process in five steps:

1) $\mathcal{V}$ generates and sends random data ($Rand_{\mathcal{V}_1}$) to $\mathcal{P}$. As $\mathcal{P}$ receives this data, it XORs the received bits with random data generated by itself and responds with the resulting stream ($Rand_{\mathcal{V}_1} \oplus Rand_{\mathcal{P}_1}$). Depending on when $\mathcal{P}$ really starts, some leading bits of $Rand_{\mathcal{V}_1}$ may be ignored by $\mathcal{P}$ (if $\mathcal{P}$ starts later) or some extra leading 0's may be added by $\mathcal{P}$ (if $\mathcal{P}$ starts earlier).

2) At some randomly selected point, unspecified by [22], $\mathcal{V}$ starts transmitting the hidden marker M, which the $\mathcal{P}$ also XORs with random data ($\hat{M} \oplus Rand_{\mathcal{P}_2}$), sending this as part of its stream.

3) After M is fully transmitted, $\mathcal{V}$ starts sending $N_\mathcal{V}$ (i.e. the nonce it generated during the initialization phase) to $\mathcal{P}$. The prover, who has also computed the hidden marker M, will expect $N_\mathcal{V}$ to be transmitted after the transmission of M. When M stops, the prover XORs the subsequent received bits (which we denote $\hat{N}_\mathcal{V}$) with its own random nonce $N_\mathcal{P}$, sending in its continuous stream the value ($\hat{N}_\mathcal{V} \oplus N_\mathcal{P}$).

4) After finishing the transmission of $N_\mathcal{V}$, the verifier $\mathcal{V}$ restarts transmitting random data $Rand_{\mathcal{V}_2}$, to which the prover $\mathcal{P}$ responds by XORing this data with random values of its own as follows: $Rand_{\mathcal{V}_2} \oplus Rand_{\mathcal{P}_3}$. Both parties continue sending random data for a random interval, then stop transmitting. Again, depending on whether $\mathcal{P}$ halts before or after $\mathcal{V}$, some tailing bits are ignored or some extra are added by $\mathcal{P}$.

5) At the end of this phase, the verifier $\mathcal{V}$ counts the number of bits it received between sending the first bit of $N_\mathcal{V}$ and receiving the first bit of the value $N_\mathcal{V} \oplus \hat{N}_\mathcal{P}$. This delay can be translated into an upper bound on the distance $\Delta\mathsf{t}(\mathcal{P}, \mathcal{V})$ by using the bit rate and the process delay.

Notes on the protocol: Note that the XORing process is wholly unnecessary in this protocol: in fact, the prover could simply respond by transmitting random data and, after the hidden marker M is received, it could simply reply with the value $\hat{N}_\mathcal{V} \oplus N_\mathcal{P}$. If the messages should be both encrypted and MACed, we suggest using state-of-the-art symmetric authenticated encryption.

IV. Description of the attack

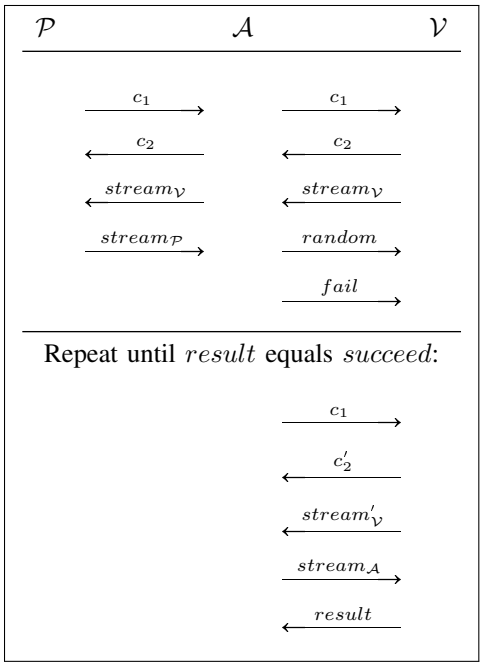


Fig. 1. Mafia fraud Attack.

The attack can be discriminated in two stages: the *man-in-the-middle stage* and the *guess stage*. More precisely:

Stage 1: Man-in-the-middle

In this stage, the adversary $\mathcal{A}$ acts as a man-in-the-middle when the protocol is run between a legitimate prover $\mathcal{P}$ and a legitimate verifier $\mathcal{V}$.

More precisely, the prover $\mathcal{P}$ sends some value c_1 (i.e. $\mathsf{Enc}_K(\mathcal{P}, \mathcal{V}, N_\mathcal{P}) \| \mathsf{MAC}(N_\mathcal{P})$). The adversary acting as a MITM relays c_1 to the verifier $\mathcal{V}$. The verifier $\mathcal{V}$ responds with a value c_2 (i.e. $\mathsf{Enc}_K(\mathcal{P}, \mathcal{V}, M, N_\mathcal{V}, N_\mathcal{P}) \| \mathsf{MAC}(M \| N_\mathcal{V} \| N_\mathcal{P})$) where M, $N_\mathcal{V}$, $N_\mathcal{P}$ are used for the hidden marker and the random nonces generated by the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ respectively and the adversary relays these messages.

During the distance bounding phase, $\mathcal{A}$ relays $stream_\mathcal{V}$ to $\mathcal{P}$ and replies to $\mathcal{V}$ with a stream of bits picked at random. $\mathcal{P}$ and $\mathcal{V}$ send $stream_\mathcal{P}$ and resp. $stream_\mathcal{V}$ such that:

$$stream_\mathcal{V} = Rand_1 \| M \| N_\mathcal{V} \| Rand_2, \quad \text{and}$$
$$stream_\mathcal{P} = Rand_3 \| N_\mathcal{V} \oplus N_\mathcal{P} \| Rand_4.$$

where $Rand_i$, for $i \in \{1, 2, 3, 4\}$, denotes random data sent either by the prover $\mathcal{P}$ or the verifier $\mathcal{V}$. It holds that:

$$(stream_\mathcal{V} \| \mathbf{0}) \oplus \mathsf{Shift}_p(stream_\mathcal{P} \| \mathbf{0}) = Rand_5 \| N_\mathcal{P} \| Rand_6 \| \mathbf{0} \quad (1)$$

where $\mathbf{0}$ denotes a bit stream of infinite length with only 0 bits, and p denotes a necessary offset which depends on when $\mathcal{P}$ and $\mathcal{V}$ start sending their transmission. $\mathsf{Shift}_k(s)$ denotes a function that performs a shift of a stream s for k bits; k can be positive or negative and thus, the shift is performed right or left correspondingly. More precisely, the offset p depends on the random time $t_{\mathcal{P}}$ at which the prover $\mathcal{P}$ starts its transmission, the time $t_{\mathcal{A}}$ at which the adversary $\mathcal{A}$ starts its transmission and the time $t_{\mathcal{VP}}$ depending on the time distance $\Delta\mathsf{t}(\mathcal{P},\mathcal{A})$ that is required for a message (consisting of possibly many bits) to be transmitted from the adversary $\mathcal{A}$ to the prover $\mathcal{P}$. Thus, the offset p is given by the following equation:

$$p = (t_{\mathcal{A}} + t_{\mathcal{AP}} - t_{\mathcal{P}}) * f$$

where f denotes the number of bits sent per second during the distance bounding phase between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. We assume in this paper that f is the same for $\mathcal{P}$ and $\mathcal{V}$. Note that $t_{\mathcal{P}}$ is random in the RC protocol.

If we assume that the adversary $\mathcal{A}$ can physically observe the location of $\mathcal{P}$, this means it can also deduce the time $t_{\mathcal{AP}}$ and then the time $t_{\mathcal{P}}$ from the reception time of $stream_{\mathcal{P}}$. Thus, it can calculate the value of p. If we assume an adversary $\mathcal{A}$ that only knows the location of the verifier, this adversary may just make a guess for p. Note that p is bounded by the length of $stream_{\mathcal{P}}$ and $stream_{\mathcal{V}}$, so it must be small in order for the protocol to be efficient.

The adversary $\mathcal{A}$ also makes a guess for the random position L of $N_{\mathcal{P}}$ in the stream $stream_{\mathcal{V}} \oplus \mathsf{Shift}_p(stream_{\mathcal{P}})$ (i.e. equation (1)) and deduces a value $N'_{\mathcal{P}}$ based on this guess. If the position L was guessed correctly, then the adversary can deduce $N_{\mathcal{P}}$ exactly.

At this point, the adversary has stored the value c_1 and a mapping $(p, L) \mapsto N'_{\mathcal{P}}$. If the values p and L are correct, then $N'_{\mathcal{P}} = N_{\mathcal{P}}$. Note that stronger adversaries, who are aware of the prover's position, know the correct offset p and thus do not need to guess it.

Stage 2: Guess

This second stage is depicted in Figure 1 and labeled as "repeat". Here, the adversary $\mathcal{A}$ starts a new session with the verifier $\mathcal{V}$ and sends as its first message the eavesdropped values c_1. Note that this allows the adversary to replay the same $N_{\mathcal{P}}$, regardless of how secure the encryption and the signature schemes are.

The verifier $\mathcal{V}$ will generate its own (fresh) nonce and hidden marker. Thus, the values used in this session are M', $N_{\mathcal{P}}$, and $N'_{\mathcal{V}}$. In the distance bounding phase of this session, the verifier $\mathcal{V}$ sends a new stream of bits $stream'_{\mathcal{V}}$ such that:

$$stream'_{\mathcal{V}} = Rand'_1 \| M' \| N'_{\mathcal{V}} \| Rand'_2.$$

In turn, the adversary $\mathcal{A}$ makes a guess for p and L and responds with its own bitstream $stream_{\mathcal{A}}$ computed as follows:

$$stream_{\mathcal{A}} = \mathsf{Shift}_q(stream'_{\mathcal{V}}) \oplus (N'_{\mathcal{P}} \| N'_{\mathcal{P}} \| \ldots \| N'_{\mathcal{P}})$$

where $N'_{\mathcal{P}}$ denotes the value that the adversary $\mathcal{A}$ has guessed for $N_{\mathcal{P}}$ in Stage 1 and q is a required alignment (compensating for the distance between the verifier and the adversary). The alignment q must be chosen such that $\mathcal{A}$ is sure that the value $N'_{\mathcal{V}} \oplus N'_{\mathcal{P}}$ is included in $stream_{\mathcal{A}}$. In other words, the length (L') of the $Rand'_1$, i.e. the number of random bits transmitted in the guess stage before the hidden marker M' is sent, is a multiple of the length of $N_{\mathcal{P}}$.

Thus, the verifier $\mathcal{V}$ should be able to calculate the value of the offset q such that it satisfies the following condition:

$$q = (|M'| + L') \bmod |N_{\mathcal{P}}|.$$

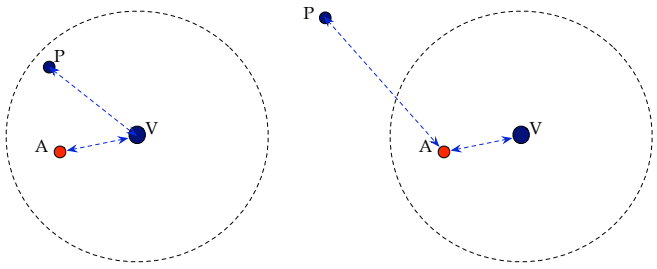
This stage repeats until the attack succeeds.

Insight: the offset q: The offset q need not be equal to the length (in bits) of $Rand'_1 || M'$. Instead, we write $|(Rand'_1 || M')| = k \cdot n + q$, where n is the length of the prover and verifier nonces. Since q is a remainder of the division by n, it follows that it can take values between 0 and $n-1$, and can be guessed with probability $\frac{1}{n}$. Consequently, if the offset q is guessed accurately, the verifier will receive the response $N'_{\mathcal{P}} \oplus N'_{\mathcal{V}}$ upon transmitting $N'_{\mathcal{V}}$. Thus, if the adversary's guess of $N_{\mathcal{P}}$ is accurate, the attack succeeds.

Attack scenarios: The attack can be launched in different scenarios depending on the location of the legitimate prover $\mathcal{P}$. One scenario might include an adversary $\mathcal{A}$ whose goal is to impersonate (stage 2) a legitimate prover $\mathcal{P}$ after having eavesdropped (stage 1) a session between $\mathcal{P}$ and $\mathcal{V}$. In such a scenario the prover $\mathcal{P}$ is in the range of the verifier $\mathcal{V}$ (depicted in Figure 2(a)).

Another scenario is our previous description of a mafia fraud where the legitimate prover $\mathcal{P}$ might be located very far from the verifier $\mathcal{V}$ while the adversary is in the communication range of $\mathcal{V}$ (depicted in Figure 2(b)). In this case the adversary $\mathcal{A}$ relays messages between $\mathcal{P}$ and $\mathcal{V}$ (i.e. corresponding to stage 1 of the described attack). Obviously, by simply relaying messages the distance bounding protocol will fail since the legitimate prover $\mathcal{P}$ is located quite far from the verifier $\mathcal{V}$. Nevertheless, the adversary $\mathcal{A}$ will be able to restart the protocol (i.e. stage 2 of the described attack) and use the information it received from Stage 1 to get authenticated.

Complexity of the attack: As described above, the success of



(a) Attack Scenario 1 (b) Attack Scenario 2

Fig. 2. Attack scenarios

the attack depends on calculating/guessing correctly the offsets p and q and the length L. If we denote by P_s the success probability of one iteration of the guess phase in the attack and by P_p, P_q and P_L the probability to successfully guess

the offsets p and q and the length L correspondingly then it holds that: $$P_s = P_p * P_q * P_L.$$

The number of repetitions is thus $1/P_s$ on average.

The value P_L depends on the min-entropy of the distribution of L in the protocol (a value unspecified in [22]). By guessing that $L = \arg\max_\ell \mathbb{P}[L = \ell]$, we have $P_L = 2^{-\mathcal{E}_{min}(L)}$, where $\mathcal{E}_{min}(L)$ is the min-entropy of L.

The distribution that minimizes the $\mathcal{E}_{min}(L)$ over a given set of values is the uniform distribution. The adversary knows that the verifier *must* send the nonce $N_\mathcal{V}$, preceded by the hidden marker. Thus, its guess of L ranges over the values $\{0, 1, \ldots, |stream_\mathcal{V}| - |N_\mathcal{V}|\}$. Thus, it holds that:

$$P_L \geqslant \frac{1}{|stream_\mathcal{V}| - n}.$$

Here, n is the length of the prover and verifier nonces, as specified in the protocol. If $\mathcal{A}$ knows the location of $\mathcal{P}$ and $\mathcal{V}$ that implies that $P_p = 1$, while P_q is given by:

$$P_q = \frac{1}{|N_\mathcal{P}|} = \frac{1}{n}.$$

Otherwise, if the adversary does not know the position of the prover and verifier, he must guess the offset p, thus:

$$P_p = \frac{1}{|stream_\mathcal{P}| - |N_\mathcal{P}|} = \frac{1}{|stream_\mathcal{P}| - n}.$$

The success probability if p is known is thus:

$$P_s \geqslant \frac{1}{n(|stream_\mathcal{V}| - n)}.$$

If p must be guessed, the probability is:

$$P_s \geqslant \frac{1}{n(|stream_\mathcal{V}| - n)(|stream_\mathcal{P}| - n)}.$$

Significance of this result. In the worst case scenario, that of an adversary who must guess the offset p as well as L and q, the complexity is upper bounded by the value $n(|stream_\mathcal{V}| - n)(|stream_\mathcal{P}| - n)$ as shown above. This is polynomial in n, as a minimal efficiency requirement is to demand that the prover and verifier run in polynomial time. To make the protocol usable in practice, both $stream_\mathcal{V}$ and $stream_\mathcal{P}$ must be small. Thus, in fact, the complexity of the adversary is quite small.

V. The LPDB Protocol

To combat the attack described in Section 4, we propose the following protocol (depicted in Figure 3). We call it the LPDB protocol, as for *Location-Privacy Distance-Bounding*.

We assume again that the prover $\mathcal{P}$ and verifier $\mathcal{V}$ share a secret key K. The protocol is again composed of two phases: the *initialization phase* and the *distance bounding phase*.

Initialization Phase: The prover $\mathcal{P}$ generates a random nonce $N_\mathcal{P}$ and sends it to the verifier $\mathcal{V}$. The verifier $\mathcal{V}$ generates a random nonce $N_\mathcal{V}$ and sends it to the prover $\mathcal{P}$. Both the prover and the verifier use as input the concatenation of the nonces $N_\mathcal{P}$ and $N_\mathcal{V}$ as input to a keyed pseudorandom function (f_K) and divide the output of the PRF into two parts, i.e.:

$$M \| R_\mathcal{P} \leftarrow f_K(N_\mathcal{P} \| N_\mathcal{V}).$$

Finally, $\mathcal{V}$ generates another random value $R_\mathcal{V}$ of length n.

Distance Bounding (DB) Phase: In this phase $\mathcal{V}$ computes a $stream_\mathcal{V}$ such that: $stream_\mathcal{V} := Rand_{\mathcal{V}_1} \| M \| R_\mathcal{V} \| Rand_{\mathcal{V}_2}$. $\mathcal{P}$ in a similar way to the RČ protocol parses the $stream_\mathcal{V}$; at the same time computes and sends the $stream_\mathcal{P}$ such that: $stream_\mathcal{P} := Rand_{\mathcal{P}_1} \| R_\mathcal{P} \oplus \hat{R}_\mathcal{V} \| Rand_{\mathcal{P}_2}$. $Rand_{\mathcal{V}_1}$, $Rand_{\mathcal{V}_2}$, $Rand_{\mathcal{P}_1}$, $Rand_{\mathcal{P}_2}$ denote random values generated by the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ respectively.

VI. Security of the LPDB Protocol

We briefly sketch here the security proof for our new protocol. Location privacy works just like in [22]. In addition to this, we can also formally prove resistance to distance fraud and mafia fraud, assuming that f is a PRF.

Theorem 1: Assuming that f is a PRF, that $R_\mathcal{V}$ is uniformly distributed in a set of exponential size, that $R_\mathcal{P}$ is in a set of exponential size, the LPDB protocol in Section V is a distance bounding protocol which provides location privacy, resistance to distance fraud, and resistance to mafia fraud.

Sketch: We don't redo the location-privacy part which is as discussed in [22].

a) Distance fraud: Assuming that a malicious prover $\mathcal{P}$ can pass the protocol with $\mathcal{V}$ although he is far away, we can transform it into an algorithm which guesses a random $R_\mathcal{V}$ which is selected by someone else at random. This is because no information about $R_\mathcal{V}$ leaks from $\mathcal{V}$ and the prover must send the correct value of $R_\mathcal{P} \oplus R_\mathcal{V}$ before receiving $R_\mathcal{V}$.

b) Mafia fraud: Let $\mathcal{A}$ be an adversary who runs a mafia fraud between a far away honest prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Following the game reduction technique, we transform the mafia fraud game into a game in which, for all $N_\mathcal{P}$ and $N_\mathcal{V}$, there is at most one session of the prover protocol $\mathcal{P}$ and one session of the verifier protocol $\mathcal{V}$ which use $N_\mathcal{P}$ and $N_\mathcal{V}$. So, the input to f does not collide in between two prover's sessions or two verifier's sessions. Then, we replace f by a truly random function in a bridging step based on the PRF assumption on f. We obtain that M and $R_\mathcal{P}$ are uniformly distributed for the pair of matching prover-verifier sessions. So, at the critical time when $\mathcal{A}$ must send $R_\mathcal{P} \oplus R_\mathcal{V}$ (assuming that he correctly guesses the position of M), he received no information about $R_\mathcal{P}$ yet from $\mathcal{P}$ who is far away. So, the probability to succeed is negligible. ■

VII. Conclusion

In this paper we present in detail a mafia fraud attack against the distance bounding protocol proposed by Rasmussen and Čapkun. We argue that the attack can be easily deployed and give the success probability of the attack which is quite high when the streams exchanged during the distance bounding phase have relatively short length; something that is implied if the protocol will be used in practice. Furthermore, we propose a new protocol that can be used to combat the attack and that is provably secure. We should note here that the selection of the random delays employed in the distance bounding phase is critical in order to guarantee that the locations of the prover and verifier remain private. We plan to investigate the selection of these random delays in future work.

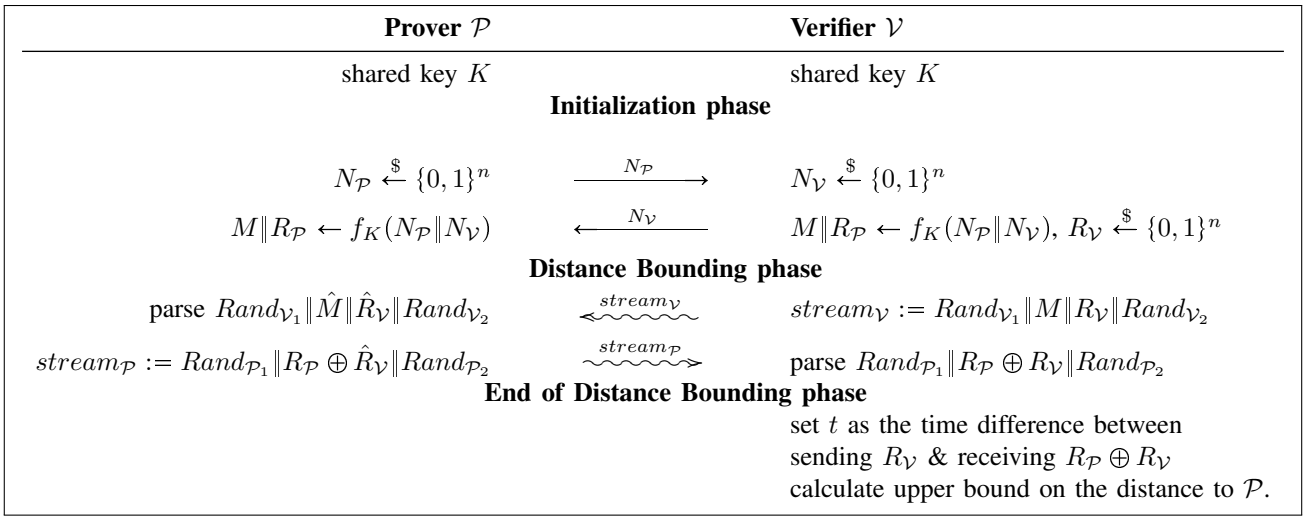


Fig. 3. The LPDB protocol.

ACKNOWLEDGMENT

This work was partially supported by the Marie Curie IEF Project "PPIDR: Privacy-Preserving IntrusionDetection and Response in Wireless Communications", Grant No. 252323.

REFERENCES

[1] M. R. S. Abyneh. Security Analysis of two Distance-Bounding Protocols. In *Proceedings of RFIDSec 2011*, Lecture Notes in Computer Science. Springer, 2011.

[2] J.-P. Aumasson, A. Mitrokotsa, and P. Peris-Lopez. A Note on a Privacy-preserving Distance Bounding Protocol. In *Proceedings of the 13th International Conference on Information and Communications Security (ICICS 2011)*, LNCS, pages 78–92. Springer, Beijing, China, 23–26 November.

[3] G. Avoine, M. A. Bingol, S. Karda, C. Lauradoux, and B. Martin. A Formal Framework for Analyzing RFID Distance Bounding Protocols. In *Journal of Computer Security - Special Issue on RFID System Security, 2010*, 2010.

[4] G. Avoine, B. Martin, and T. Martin. Optimal Security Limits of RFID Distance Bounding Protocols. In *RFIDSec 2010*, pages 220 – 238.

[5] G. Avoine and A. Tchamkerten. An Efficient Distance Bounding RFID Authentication Protocol: Balancing False-Acceptance Rate and Memory Requirement. In *Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 250–261. Springer-Verlag, 2009.

[6] S. Brands and D. Chaum. Distance-bounding protocols. In *Advances in Cryptology — Eurocrypt'93*, Lecture Notes in Computer Science, pages 344–359. Springer-Verlag, 1993.

[7] L. Bussard and W. Bagga. Distance-bounding Proof of Knowledge to Avoid Real-time Attacks. *Security and Privacy in the Age of Ubiquitous Computing*, 181:222–238, 2005.

[8] S. Capkun, L. Buttyán, and J.-P. Hubaux. SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks. In *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks - SASN*, pages 21 – 32. ACM Press, 2003.

[9] D. Carluccio, T. Kasper, and C. Paar. Implementation details of a multi purpose ISO 14443 RFID-tool. In *Printed handout of Workshop on RFID Security - RFIDSec 06*, July 2006.

[10] J. Clulow, G. P. Hancke, M. G. Kuhn, and T. Moore. So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks. In *Proceedings of the European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks*, volume 4357 of *Lecture Notes in Computer Science*, pages 83–97. Springer-Verlag, 2006.

[11] C. Cremers, K. B. Rasmussen, and S. Čapkun. Distance Hijacking Attacks on Distance Bounding Protocols. Cryptology ePrint Archive, Report 2011/129, 2011. EPRINTURL.

[12] Y. Desmedt. Major Security Problems with the Unforgeable' (Feige)-Fiat-Shamir Proofs of Identity and How to Overcome them. In *SecuriCom*, pages 15–17. SEDEP Paris, France, 1988.

[13] S. Drimer and S. J. Murdoch. Keep your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In *Proceedings of the 16-th USENIX Security Symposium on USENIX Security Symposium, article no. 7*. ACM Press, 2007.

[14] S. Drimer and S. J. Murdoch. Keep your enemies close: distance bounding against smartcard relay attacks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 7:1–7:16, Berkeley, CA, USA, 2007. USENIX Association.

[15] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A Formal Approach to Distance Bounding RFID Protocols. In *Proceedings of the* 14^{th} *Information Security Conference ISC 2011*, Lecture Notes in Computer Science, pages 47–62, 2011.

[16] A. Francillon, B. Danev, and S. Čapkun. Relay attacks on passive keyless entry and start systems in modern cars. Cryptology ePrint Archive, Report 2010/332, 2010.

[17] K. Haataja and P. Toivanen. Two Practical Man-In-The-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures. *Transactions on Wireless Communications*, 9(1):384–392, 2010.

[18] G. P. Hancke. A Practical Relay Attack on ISO 14443 Proximity Cards. http://www.cl.cam.ac.uk/gh275/relay.pdf, 2005.

[19] G. P. Hancke and M. G. Kuhn. An RFID Distance Bounding Protocol. In *SECURECOMM*, pages 67–73. ACM Press, 2005.

[20] C. H. Kim and G. Avoine. RFID Distance Bounding Protocol with Mixed Challenges to Prevent Relay Attacks. In *Proceedings of the 8th International Conference on Cryptology and Networks Security (CANS 2009)*, volume 5888 of *Lecture Notes in Computer Science*, pages 119–131. Springer-Verlag, 2009.

[21] C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The Swiss-Knife RFID Distance Bounding Protocol. In *Proceedings of the* 14^{th} *Information Security Conference ISC 2011*, Lecture Notes in Computer Science, pages 98–115. Springer-Verlag, 2009.

[22] K. Rasmussen and S. Čapkun. Location Privacy of Distance Bounding. In *Proceedings of the Annual Conference on Computer and Communications Security (CCS)*. ACM Press, 2008.

[23] J. Reid, J. M. G. Nieto, T. Tang, and B. Senadji. Detecting Relay Attacks with Timing-Based Protocols. In *ASIACCS*, pages 204–213. ACM Press, 2007.

[24] D. Singelée and B. Preneel. Distance Bounding in Noisy Environments. In *European Workshop on Security in Ad-hoc and Sensor Networks – ESAS*, volume 4572 of *Lecture Notes in Computer Science*, pages 101 – 115. IEEE Computer Society Press, 2007.

[25] D. Spill and A. Bittau. BlueSniff: Eve meets Alice and Bluetooth. In *Proceedings on the 1st USENIX workshop on Offensive Technologies (WOOT 07)*. USENIX Association Berkeley, CA, USA, 2007.

[26] R. Trujillo-Rasua, B. Martin, and G. Avoine. The Poulidor Distance-Bounding Protocol. In *RFIDSec 2010*, pages 239 – 257.