Yet Another Secure Distance-Bounding Protocol

Ventzislav Nikov and Marc Vauclair

NXP Semiconductors, Leuven, Belgium,
ventzislav.nikov@nxp.com, marc.vauclair@nxp.com

Abstract. Distance-bounding protocols have been proposed by Brands and Chaum in 1993 in order to detect *relay attacks*, also known as *mafia fraud*. Although the idea has been introduced fifteen years ago, only recently distance-bounding protocols attracted the attention of the researchers. Several new protocols have been proposed the last five years.
In this paper, a new secure distance-bounding protocol is presented. It is self-contained and composable with other protocols for example for authentication or key-negotiation. It allows periodically execution and achieves better use of the communication channels by exchanging authenticated nonces. The proposed protocol becomes suitable for wider class of devices, since the resource requirements to the prover are relaxed.

Keywords: Distance-Bounding protocols, Relay Attacks, Mafia-Fraud.

1 Introduction

Consider a model in which a party known as *verifier* V is interested in learning the proximity to a second party known as *prover* P. The prover can be a trusted device (e.g. the trust can be assured by the tamper-resistance of the device) or un-trusted. In both cases, the prover is surrounded by an un-trusted environment. Many practical situations motivate this model, e.g. RFID, content protection systems (digital rights management systems), ad-hoc wireless networks, sensor networks etc., see [1,2,3,4,5,6,7,8,9,10,11,12,13,14]. To motivate our research we will describe some of them.

Almost all existing RFID authentication schemes (tag/reader) are vulnerable to mafia attacks, because of their inability to estimate the distance to the tag. Such attacks are usually identified by the signal strength, but a resourceful adversary can easily thwarts this. Several positioning and distance measuring techniques for ad-hoc networks or wireless location-based access control has been proposed, but most of them consider just non-adversarial settings and rely on signal strength or (signal) noise analysis.

Location-based access control is based on the rule **bigger distance implies distrust**, i.e. a device which refuses to respond to the distance estimation request or which appears to be not close enough is simply denied access. So we assume that the goal of a malicious device is to be localized in a place other than its true location so it

participates in the protocol but tries to mislead the verifiers. For example in DRM the content owner (known as source) can refuse to deploy it to the sink (the receiver) if he is too far from the source.

Distance-bounding protocols provide secure proximity control, i.e. they prevent the so-called *distance fraud* attacks. Brands and Chaum [3] were the first who proposed a secure solution for this problem. The *distance-bounding* protocols measure the delays between the sending out of a challenge and the reception of the response. To be feasible, this approach requires nearly no computation during each challenge-response operation. Two types of attacks against such secure protocols are considered in the literature: *mafia fraud* or *mafia attack* [6] (or *Mig-in-the-middle* [1]) and *terrorist fraud* [6]. In the mafia fraud attack, the attacker does not perform any cryptographic operations based on the security protocol, and only forwards the challenges and the responses between the honest prover and the honest verifier. While in the terrorist fraud attack the prover is not honest and he collaborates with the attacker.

In this paper, we present an efficient secure distance-bounding protocol suitable for wide range of devices. Note that all protocols measure the round-trip time which is twice the one-way propagation time plus the processing delay at the prover. But for heterogeneous systems the processing delay can vary a lot, e.g. an RFID tag versus a PC. Our contribution can be summarized as follows:

Our distance-bounding protocol is self-contained and can be combined with other protocols (e.g. authentication protocols), i.e. it can be plugged into any protocol provided the assumptions are satisfied. Recall that Brands and Chaum introduced the rapid bit exchange techniques in order to measure the round-trip time. Most of the known protocols use the idea of rapid bit exchange. But this approach has a drawback as pointed out in [11] that most of the today used communication channels have a bandwidth much bigger than a bit. Having this in mind we extend the approach of Waters and Felten, so that the authenticated nonces are sent out and hence the overall efficiency of the protocol is improved. A combination of symmetric cryptographic algorithms with preprocessing reduces the resource requirements for the participants especially for the prover. In addition, our protocol does not require the prover to generate any random data which diminishes the requirements to the devices and hence makes the protocol suitable for wide range of devices. Because we measure the proximity to a mobile device the proposed protocol allows periodical execution. Because of this the preprocessing phase is non-interactive and lighter.

The paper is organized as follows: Section 2 describes all known distance-bounding protocols. A new distance-bounding protocol is proposed in Section 3. We conclude in Section 4.

2 Related Work

For all protocols described in this section a security parameter k is chosen, the security levels for the protocols can be described as a function of k. In all of these protocols it is assumed that the prover and the verifier have agreed in advance to use dedicated common primitives, e.g. asymmetric or symmetric encryption, signature scheme, commitment scheme, pseudo random function, one-way and collision resistant function, zero-knowledge protocol, message authentication codes, etc. and they both have access to secure random number generators. For each protocol we will indicate which keys the parties have in advance. All protocols have a phase of "rapid exchange", i.e. the verifier starts its timer and sends a challenge to which the prover replies, upon receiving the reply the verifier stops its timer. In most of the protocols the challenge and the reply are bits and then this rapid exchange is repeated k times. There are also few examples in which the challenge and the reply are strings of length derived from k and the rapid exchange is executed just ones.

Brands and Chaum [3] are the pioneers of distance-bounding protocols, they have designed several protocols secure against mafia fraud. In their settings it is assumed that the prover possesses a signature key-pair. K_P^{sign} denotes the prover's private signature key. At the beginning the prover and the verifier randomly generate $\beta_i \in_R \{0,1\}$ and $\alpha_i \in_R \{0,1\}$ for $i = 1, \ldots, k$. After this preparation a phase of k rapid bit exchanges starts. The verifier sends α_i to the prover, who replies with β_i. At last the prover concatenates the $2k$ bits α_i and β_i, signs the result and sends it to the verifier. The verifier verifies the received signature. Note that this basic protocol (see Fig. 1) prevents the mafia frauds, but the prover can still cheat by sending out the bits β_i before receiving α_i.

1. The prover randomly generates $\beta_i \in_R \{0,1\}$ for $i = 1, \ldots, k$, where k is determined by a chosen security parameter.
2. Analogously the verifier randomly generates $\alpha_i \in_R \{0,1\}$ for $i = 1, \ldots, k$.
3. After this preparation a phase of k rapid bit exchanges starts.
   - The verifier starts its timer and sends α_i to the prover.
   - The prover replies with β_i.
   - Upon receiving the bit the verifier stops its timer.
4. The prover concatenates the $2k$ bits α_i and β_i, signs the result i.e. $Sign(K_P^{sign}; \alpha_1|\beta_1|\ldots|\alpha_k|\beta_k)$ (where | denotes the bit string concatenation operator) and sends it to the verifier
5. The verifier concatenates the used $2k$ bits α_i and β_i and verifies the received signature.

Fig. 1. Brands and Chaum [3] - Basic Protocol

The authors propose also a modification which prevents the prover from sending out the bits β_i before receiving α_i (see Fig. 2). Again the prover and the verifier randomly generate $m_i \in_R \{0,1\}$ and $\alpha_i \in_R \{0,1\}$ for $i = 1, \ldots, k$. But the prover also commits to $m_1|\ldots|m_k$ and sends this commitment to the verifier. After this the phase of k rapid bit exchanges starts: the prover replies with $\beta_i = \alpha_i \oplus m_i$ to the received α_i. Again the prover signs the concatenation of α_i and β_i, but he also sends the opening of the commitment to $m_1|\ldots|m_k$. The verifier verifies the received signature and using the opened commitment checks whether $\beta_i = \alpha_i \oplus m_i$ for $i = 1, \ldots, k$.

1. The prover randomly generates $m_i \in_R \{0,1\}$ for $i = 1, \ldots, k$, where k is determined by a chosen security parameter.
2. Analogously the verifier randomly generates $\alpha_i \in_R \{0,1\}$ for $i = 1, \ldots, k$.
3. The prover commits to $m_1|\ldots|m_k$ and sends this commitment to the verifier.
4. After this preparation a phase of k rapid bit exchanges starts.
   - The verifier starts its timer and sends α_i to the prover.
   - The prover replies with $\beta_i = \alpha_i \oplus m_i$.
   - Upon receiving the bit the verifier stops its timer.
5. The prover concatenates the $2k$ bits α_i and β_i, signs the result i.e. $Sign(K_P^{sign}; \alpha_1|\beta_1|\ldots|\alpha_k|\beta_k)$ and sends it to the verifier together with the opening of the commitment to $m_1|\ldots|m_k$.
6. The verifier concatenates the used $2k$ bits α_i and β_i and verifies the received signature. Next he verifies the commitment and checks whether $\beta_i = \alpha_i \oplus m_i$ for $i = 1, \ldots, k$.

Fig. 2. Brands and Chaum [3]

Another solution in [3] which prevents the prover to send out bits too soon is to use random delays in the rapid bits exchange phase. This approach has the advantage from a practical point of view that it does not require the commitment scheme. The authors also have pointed out that their protocols are not secure against terrorist fraud.

Waters and Felten [14] have designed a proximity-proving protocol with the extension that it protects the privacy of the prover and that the measured latency is afterwards transmitted to a third party which can reveal the identity of the prover. We outline just a part of the protocol related to the proximity control (see Fig. 3). It is assumed that the verifier has an asymmetric encryption key-pair and that the prover possesses a signature key-pair. K_V^{enc} denotes the verifier's public encryption key and K_P^{sign} denotes the prover's private signature key.

The prover randomly generates two nonces *start* and *reply*. The length of the nonces is determined by a chosen security parameter. Next the prover sings its ID and then he encrypts $start, reply$ and the signature and sends it to the verifier. The verifier decrypts the received message, extracts the nonces and checks the signature. Next he generates a random nonce *echo* with length determined by the security parameter. Then the rapid exchange starts - he sends $(start, echo)$ to the prover, after verifying that the first part of the message is the nonce *start* the prover replies with $(reply, echo)$. The verifier verifies that the received message consists of two parts: first part is the nonce *reply* and second part is the nonce *echo*.

1. The prover randomly generates two nonces *start* and *reply*. The length of the nonces is determined by a chosen security parameter.
2. Next the prover sings its ID, i.e. $Sign(K_P^{sign}; ID)$. Then he sends $Enc(K_V^{enc}; start, reply, Sign(K_P^{sign}; ID))$ to the verifier.
3. The verifier decrypts the received message, extracts the nonces and checks the signature.
4. Next he generates a random nonce *echo* with length determined by the security parameter.
5. After this preparation the phase of rapid exchange starts:
   - The verifier starts its timer and sends $(start, echo)$ to the prover.
   - The prover verifies that the first part of the message is the nonce *start*, then he replies with $(reply, echo)$.
   - Upon receiving the message back the verifier stops its timer and records the round-trip latency.
6. The verifier checks that the received message consists of two parts: first part is the nonce *reply* and second part is the nonce *echo*.

Fig. 3. Waters and Felten [14]

This protocol is secure against mafia fraud but not against terrorist fraud, since after preparing the first message a dishonest prover can delegate the execution of the protocol to an attacker.

Probably the simplest distance bounding protocol was proposed by Sastry, Shankar and Wagner [10]. The verifier sends a randomly generated nonce (with length determined by a chosen security parameter) to the prover, who returns it back to the verifier. The use of random nonce prevents the prover to respond in advance, however without any authentication this protocol is vulnerable to the mafia attack. The authors have proposed a modification in which the parties share a common secret key s. Again the verifier sends a nonce r to the verifier who uses a pseudorandom function h (e.g. AES, HMAC-SHA1) to compute the reply $h(s; r)$. However, the time spend for computing the reply can be so large with respect to the travel time as to make it difficult to compute the distance except for relatively slow mediums like sound (for which Sastry *et al.* protocol is designed).

Hancke and Kuhn [7] proposed a modification to the challenge-response scheme which improves the overall efficiency of the distance-bounding protocol. In their protocol (see Fig. 4) both parties share a common secret key s.

The prover and the verifier randomly generate nonces r_P and r_V then exchange them. Both parties use (one-way and collision resistant) function h to derive two k-bit strings m and r from the shared key s and the exchanged nonces, i.e. $m|r = h(s; r_P|r_V)$. The verifier randomly generates α_i for $i = 1, \ldots, k$, after this preparation a phase of k rapid bit exchanges starts. The verifier sends α_i to the prover, who computes the reply β_i as follows: if $\alpha_i = 1$ then $\beta_i = m_i$ otherwise (i.e. if $\alpha_i = 0$) set $\beta_i = r_i$. The verifier verifies the value of β_i, if a value of β_i is wrong the protocol is stopped.

1. The prover randomly generates a nonce r_P and sends it to the verifier.
2. The verifier randomly generates a nonce r_V and sends it to the prover.
3. Both parties use (one-way and collision resistant) function h to derive two k-bit strings m and r from the shared key s and the exchanged nonces, i.e. $m|r = h(s; r_P|r_V)$.
4. The verifier randomly generates α_i for $i = 1, \ldots, k$.
5. After this preparation a phase of k rapid bit exchanges starts.
   - The verifier starts its timer and sends α_i to the prover.
   - The prover computes the reply β_i as follows: if $\alpha_i = 1$ then $\beta_i = m_i$ otherwise (i.e. if $\alpha_i = 0$) set $\beta_i = r_i$.
   - Upon receiving the bit the verifier stops its timer and verifies the value of β_i.
6. If a value of β_i is wrong the protocol is stopped.

Fig. 4. Hancke and Kuhn [7]

Bussard and Bagga [2] proposed the first, to our knowledge, protocol secure against terrorist fraud. Their protocol (see Fig. 5) is public-key based and uses zero-knowledge techniques, i.e. it is assumed that the prover owns a private key x and the verifier has the corresponding public key y.

The prover randomly generates a k-bit key s. Then he encrypts his private key using a symmetric key encryption method, i.e. $e = enc(s; x)$. Let x, y and e have length k. For $i = 1, \ldots, k$ the prover commits to s_i and e_i and sends these commitments to the verifier. As usual the verifier randomly generates α_i for $i = 1, \ldots, k$, after this the phase of k rapid bit exchanges starts. The verifier sends α_i to the prover, who computes the reply β_i as follows: if $\alpha_i = 1$ then $\beta_i = s_i$ otherwise (i.e. if $\alpha_i = 0$) set $\beta_i = e_i$. The prover opens the commitments which correspond to the values β_i, i.e. he opens either the commitment to s_i (if $\alpha_i = 1$) or the commitment to e_i (if $\alpha_i = 0$) for $i = 1, \ldots, k$. At last the prover executes a zero-knowledge protocol in order to prove that he possess a private-key x (corresponding to the public-key y) and that e is a ciphertext (with key s) of this private-key.

1. The prover randomly generates a k-bit key s, where k is determined by a chosen security parameter.
2. Then he encrypts his private key using a symmetric key encryption method (e.g. one-time pad), i.e. $e = enc(s; x)$. Let x, y and e have length k.
3. For $i = 1, \ldots, k$ the prover commits to s_i and e_i and sends these commitments to the verifier.
4. The verifier randomly generates α_i for $i = 1, \ldots, k$.
5. After this preparation a phase of k rapid bit exchanges starts.
   - The verifier starts its timer and sends α_i to the prover.
   - The prover computes the reply β_i as follows: if $\alpha_i = 1$ then $\beta_i = s_i$ otherwise (i.e. if $\alpha_i = 0$) set $\beta_i = e_i$.
   - Upon receiving the bit the verifier stops its timer.
6. The prover opens the commitments which correspond to the values β_i, i.e. he opens either the commitment to s_i (if $\alpha_i = 1$) or the commitment to e_i (if $\alpha_i = 0$) for $i = 1, \ldots, k$.
7. The prover executes a zero-knowledge protocol in order to prove that he possess a private-key x (corresponding to the public-key y) and that e is a ciphertext (with key s) of this private-key.

Fig. 5. Bussard and Bagga [2]

Thus the idea of Bussard and Bagga is to force the prover to give away his private key if he wants to mount a terrorist attack, since the attacker should own both e and s and hence x. On the other hand, the proposed protocol is not feasible because of the asymmetric techniques and especially the zero-knowledge protocol - both computationally demanding.

Recently Reid *et al.* [9] tried to combine the efficiency of the Hancke and Kuhn [7] protocol with the idea of Bussard and Bagga [2] for preventing terrorist attacks. In their protocol (see Fig. 6), it is assumed that both parties share a common secret key s.

Again the prover and the verifier randomly generate nonces r_P and r_V then exchange them. Both parties use (pseudo-random) function h to derive a k-bit strings m from the shared key s and the exchanged nonces, i.e. $m = h(s; r_P | r_V)$. Both parties encrypt the shared key s using a symmetric key encryption method (e.g. one-time pad), i.e. $e = enc(m; s)$. Let both s and e have length k. As usual the verifier randomly generates α_i for $i = 1, \ldots, k$, after this the phase of k rapid bit exchanges starts. The verifier sends α_i to the prover, who replies β_i as follows: if $\alpha_i = 1$ then $\beta_i = m_i$ otherwise (i.e. if $\alpha_i = 0$) set $\beta_i = e_i$. The verifier verifies the value of β_i, if a value of β_i is wrong the protocol is stopped.

1. The prover randomly generates a nonce r_P and sends it to the verifier.
2. The verifier randomly generates a nonce r_V and sends it to the prover.
3. Both parties use (pseudo-random) function h to derive a k-bit strings m from the shared key s and the exchanged nonces, i.e. $m = h(s; r_P|r_V)$.
4. Both parties encrypt the shared key s using a symmetric key encryption method (e.g. one-time pad), i.e. $e = enc(m; s)$. Let s and e have length k.
5. The verifier randomly generates α_i for $i = 1, \ldots, k$.
6. After this preparation a phase of k rapid bit exchanges starts.
   - The verifier starts its timer and sends α_i to the prover.
   - The prover computes the reply β_i as follows: if $\alpha_i = 1$ then $\beta_i = m_i$ otherwise (i.e. if $\alpha_i = 0$) set $\beta_i = e_i$.
   - Upon receiving the bit the verifier stops its timer and verifies the value of β_i.
7. If a value of β_i is wrong the protocol is stopped.

Fig. 6. Reid *et al.* [9]

Note that the knowledge of m and e is equivalent to revealing the shared secret s. The authors of [9] claim that this will prevent a terrorist attack, but since the key s is less valuable for the prover than his private key and since it is used only for distance-bounding, a corrupt prover may still mount a terrorist attack against the verifier.

Capkun and Hubaux [5] have proposed the following protocol (see Fig. 7). It is assumed that both parties share a common secret key s. The verifier and the prover simultaneously generate random nonces *start* and *reply*. Their length is determined by a chosen security parameter. Then the prover commits to his nonce and sends the commitment to the verifier. The verifier starts the rapid exchange by sending *start* to the prover, who replies with $start \oplus reply$. Then the prover computes a message authentication code (MAC) over both nonces, i.e. $MAC(s; start|reply)$. He sends the MAC together with the opening of the commitment to *reply* nonce. The verifier verifies the MAC and verify that the committed value indeed corresponds to the used one.

As it has been pointed out by Singelee and Preneel [11], all known protocols except Waters, Felten and Capkun, Hubaux are using the idea of Brands and Chaum to measure the proximity by a rapid bit exchange. In order to measure the round trip time with accuracy special hardware is required. Moreover most of the today used communication channels have a bandwidth much bigger than a bit. Another observation made in [11] is that any protocol secure against mafia attacks can be made secure against terrorist attacks when *trusted hardware* is used. The trusted hardware has the following properties: it is impossible to change the protocol that it has to perform and it is impossible to derive any value out of it. Thus it can only be used as a black-box.

1. The prover randomly generates a nonce *reply*, commits to it and sends both to the verifier.
2. The verifier randomly generates a nonce *start* and sends it to the prover.
3. After this preparation the rapid exchange starts.
   - The verifier starts its timer and sends *start* to the prover.
   - The prover replies with $start \oplus reply$.
   - Upon receiving it the verifier stops its timer.
4. The prover computes a MAC over both nonces, i.e. $MAC(s; start|reply)$. Then he sends the MAC together with the opening of the commitment to the *reply* nonce.
5. The verifier verifies: the MAC, the commitment and that the committed value indeed corresponds to the used one.

Fig. 7. Capkun and Hubaux [5]

3 The New Protocol

Design principles

There should be as few resource demands as possible on both parties but especially on the prover. Since our real goal is to enable proximity control for a large class of devices we would like to limit the computation power and hardware resources necessary to participate in such protocol to minimum. This requirement excludes many cryptographic solutions like public-key encryption and signature, commitment schemes, zero-knowledge protocols.

The setup requirement should be minimal. To participate in the protocol both parties must share one or two common secrets. How these secrets are set up is out of scope for the considered protocol, for example they could be derived from an authentication protocol executed beforehand or distributed via separate protocol or build in during the production phase of the devices.

As pointed out most of the today used communication channels have a bandwidth much bigger than a bit. Moreover even when the protocol specifies that a bit is sent because of the communication packeting this bit is encapsulated to much bigger packet which is then sent over the communication medium. This observation shows that the standard cryptographic approach of rapid bit exchange is not efficient in practice. Here we don't consider the time needed for the message to pass through the communication stack.

The protocol should be easily composable with other protocols and because of the nature of distance measuring protocol to evaluate proximity of a mobile device we may need periodical execution of the protocol. In this case certain phases of the protocol (except the rapid exchange phase) should be made lighter (e.g. avoiding any redundancy) and if possible non-interactive.

Threat Model

The protocol should be secure against both cheating verifier and mafia attacks. The reason why we don't consider terrorists attacks is twofold, first as noted by using

trusted hardware this attack is prevented (as this is the case for DRM). Secondly, as we discussed the only way a cryptographic proximity protocol to be secure against terrorists attacks is to force the prover to give away to the terrorists his private secrets (private-asymmetric or shared-symmetric key). In some cases this will prevent such an attack from a potential cheater. But depending on the application terrorists attacks are still a possible threat (e.g. DRM or sensor networks cases - if the device has been compromised by an attacker).

Since we would like the prover to belong to a large class of devices some of them with very limited resources, e.g. RFID, it is reasonable to assume that he either has no source of randomness or it is limited and thus insecure (predictable). Note that all existing protocols are subject to mafia attacks if the prover has insecure random number generator.

The Protocol

The purpose of the protocol is to prove to the verifier that the prover is within a given distance, without using any source of randomness. We assume that the prover P and the verifier V share some common secrets. Namely, a distance-authentication key and denoted by K and a seed by R, both with a fixed length $\tilde{k}$. These two secrets may be derived from the authentication protocol executed in advance by both parties, if the parties have already established a secure authenticated channel, or distributed via other means. We stress here that the distance-authentication key K must be different for different sessions, otherwise an attacker can use previously exchanged tags computed with K for the current session. Thus we separate the phase when the distance is measured from the phase when the authentication takes place. A pseudo-random function is used for the calculation of the verifier's challenge and prover's response. Since we don't consider the terrorist attacks the attacker has no access to the shared key. We will denote by $h(s; m)$ the pseudo-random function which has as inputs a secret key s and a public string m and outputs a string with a fixed length $\tilde{k}$, computationally indistinguishable from a uniformly random string. In practice, h can be HMAC or AES.

In order to make the distance-bounding protocol applicable for a wide range of environments (e.g. from low power RFID tags and embedded devices, to PCs) the protocol should not use "heavy" public key cryptographic schemes (signature, commitments, etc.). Indeed symmetric cryptographic algorithms are more suitable taking into account constraints like time, energy and computations.

Another goal when designing the protocol was to allow the periodical execution of it once a secure channel is established between the parties. In other words, the protocol can be executed several times at unspecified time intervals in order to ensure that the communicating parties are still in the same proximity. We assume that in case distance-bounding protocol fails then the secure authenticated channel is terminated. Note that all of the known protocols are not designed for periodical execution. Of course they can be executed periodically but using them in such a way will be not efficient. One way to improve the efficiency is to design a protocol which has suitable preparation (preprocessing) phase.

Brands and Chaum [3] have introduced the "classical" structure of distance bounding protocol: preprocessing phase in which some values are exchanged and the prover performs binding (like commitment) to some of them; then the phase of the rapid exchange; followed with a phase in which the bindings are opened and verified. Waters and Felten [14] slightly modified this design approach by making the last phase lighter and non-interactive. This approach was further used in [7,9], while the authors of [2,5] followed the classical model.

Since the proposed protocol should allow the periodical execution we adopted the approach of Waters and Felten by making the last phase lighter and non-interactive. But the proposed protocol differs from all known protocols on the fact that also the preprocessing phase is non-interactive and lighter. Moreover since all computations are performed in a preprocessing phase when the time is measured the processing delay is negligible.

Thus in the preprocessing phase both parties compute two sequences say $\{a_i\}$ and $\{b_i\}$ (for $i = j+1, ..., j+k$), where j is a counter known for both parties (initialized to zero when the protocol is executed for the first time). For example, by using dedicated h and the seed R, i.e. $a_i = h(R; i|V)$ and $b_i = h(R; i|P)$. Note that the sequences $\{a_i\}$ and $\{b_i\}$ may also be public and not pseudo-random. For example, a recurrence relation like the Fibonacci sequence (a_i is $i+5$-th Fibonacci number and $b_i = a_i + 1$) can be used. We stress here that the sequences $\{a_i\}$ and $\{b_i\}$ should satisfy the following condition: the probabilities $Prob(a_i = b_j), Prob(b_i = b_j)$ and $Prob(a_i = a_j)$ are negligible. In case the sequences are public the seed R is either made public (thus anybody can compute $a_i = h(R; i|V)$ and $b_i = h(R; i|P)$) or the seed is not used in the computation (in the recurrence relation case). By using these sequences we avoid the need of random source for the prover. Our protocol is described in Fig. 8.

Now we will show that the described protocol is secure against distance and mafia frauds. In order to mount a distance fraud attack the prover must respond to the challenge ma_i in advance (i.e. before getting the challenge). Hence he should choose at random one of his possible replies mb_i for $i = j+1, ..., j+k$ and send it out. The probability that prover's guess for i is correct is $1/k$ (since the verifier chooses i uniformly at random), but the repetition of the protocol will make the probability of the prover's correct guess negligible.

Consider an attacker in the maffia fraud setting. The attacker can run the distance-bounding protocol, pretending to be either prover or verifier, with a legitimate verifier or prover respectively. But then he should choose i and guess mb_i or respectively ma_i which he sends out. The probability that his guess of mb_i or ma_i is correct is negligible since these are the tags produced from a pseudo-random function and the attacker doesn't know the distance-authentication key. The probability that a random guess of mb_i or ma_i to be correct is $2^{-\tilde{k}}$, the same as guessing the distance-authentication key K. The attacker also can't use any of the previously exchanged authenticated nonces since by design the probabilities of $Prob(a_i = b_j), Prob(b_i = b_j)$ and $Prob(a_i = a_j)$ are negligible and hence the probabilities of $Prob(ma_i = mb_j), Prob(mb_i = mb_j)$ and

1. Let's assume that the prover P and verifier V share a common secret (distance-authentication key K) and another common secret (seed R) both with a fixed length $\tilde{k}$.
2. Let k be a security parameter and j be a counter known for both parties (initialized to zero when the protocol is executed for the first time). In the preprocessing phase both parties first compute fixed parts of the two sequences $\{a_i\}$ and $\{b_i\}$ (for $i = j+1, ..., j+k$).
3. The second step in the preprocessing phase for both parties is to compute the tags $ma_i = h(K; a_i)$ and $mb_i = (K; b_i)$ (for $i = j+1, ..., j+k$).
4. The interactive (rapid exchange) phase starts:
   - The verifier V chooses at random $i \in_R [j+1, j+k]$.
   - Then V starts his timer and sends the tuple i, ma_i to P.
   - The prover P compares the received value ma_i with his pre-computed one and if they are the same returns the tuple i, mb_i to V.
   - Upon receiving the reply the verifier stops his timer.
5. The verifier compares the received values i and mb_i with his pre-computed one and if the comparison is ok, computes the round-trip time.
6. Both parties P and V increase the counter j with k.

Fig. 8. The Proposed Protocol

$Prob(ma_i = ma_j)$ are negligible again because of the pseudo-randomness of the used function and the choice of the security parameter $\tilde{k}$.

We stress here that the protocol can be made secure against terrorist attacks when trusted hardware is used. Note that the trusted hardware also prevents the distance fraud. Hence from a practical point of view, the prevention from mafia fraud is more important than the prevention from the distance fraud.

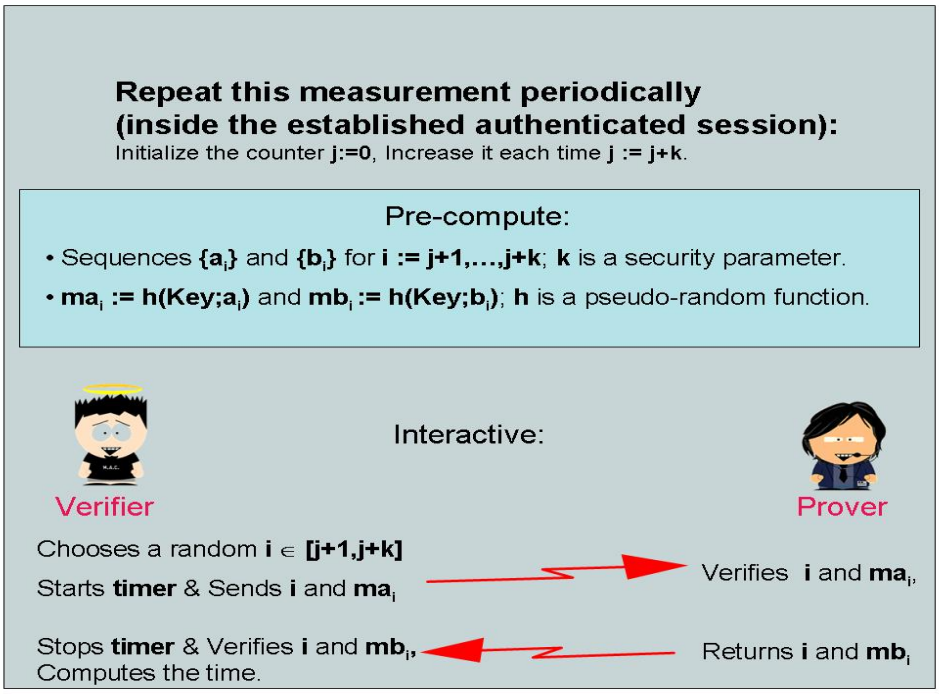
4 Conclusions

This paper presents a new secure distance-bounding protocol. It is self-contained and composable with other protocols for example authentication or key-negotiation. The protocol allows periodical execution, which is in accordance with the nature of the measuring proximity to mobile devices. Moreover the preprocessing phase of the protocol is non-interactive and lighter. Better usage of the communication channels is achieved by exchanging authenticated nonces, which also improves the overall efficiency of the protocol. Since the resource requirements to the prover are relaxed the proposed protocol is suitable for wider class of devices.

Acknowledgements

This work has been done during the design and development of Display Port Content Protection system. The authors would like to thank Michael Epstein and Raymond Krasinski for the valuable discussions.

Both derived: distance-authentication key **Key**, and seed **R** (optional).



References

1. R. Anderson, Security Engineering: A Guide to building dependable distributed systems, *John Wiley and Sons*, 2001.
2. L. Bussard, W. Bagga, Distance-bounding proof of knowledge to avoid real-time attacks, *IFIP/SEC*, 2005.
3. S. Brands, D. Chaum, Distance-Bounding Protocols, *EUROCRYPT'93*, LNCS 765, 1993, pp. 344–359.
4. S. Capkun, L. Buttyan, J.-P. Hubaux, SECTOR: Secure Tracking of Node Encounters in Multi-Hop Wireless Networks, *SASN* 2003, pp. 21–32.
5. S. Capkun, J.-P. Hubaux, Secure positioning in wireless networks, IEEE Selected Areas in Communications, vol.24, no.2, pp. 221-232, 2006.

6. Y. Desmedt, Major security problems with "unforgeable" (feige)- at- shamir proofs of identity and how to overcome them, *SecuriCom*, 1988.
7. G. Hancke, M. Kuhn, An RFID Distance Bounding Protocol, *IEEE SecureComm*, 2005, pp. 67–73.
8. J. Munilla, A. Peinado, Attacks on Singelee and Preneel's protocol, *Cryptology ePrint Archive*: Report 2008/283.
9. J. Reid, J. Neito, T. Tang, B. Senadji, Detecting Relay Attacks with Timing Based Protocols, *ACM ASIACCS*, 2007, pp. 204–213.
10. N. Sastry, U. Shankar, D. Wagner, Secure Verification of Location Claims, *ACM Workshop on Wireless Security*, 2003, pp. 48–61.
11. D. Singelee, B. Preneel, Location Verification using Secure Distance Bounding Protocols, *IEEE Computer Society*, 2005, pp. 834-840.
12. D. Singelee, B. Preneel, Distance Bounding in Noisy Environments, *ESAS* 2007, LNCS 4572, 2007, pp. 101–115.
13. D. Singelee, B. Preneel, Key Establishment Using Secure Distance Bounding Protocols, *IEEE-MobiQuitous* 2007.
14. B. Waters, E. Felten, Secure, Private Proofs of Location, *Princeton Computer Science* TR-667-03, 2003.