Towards More Secure and Reliable Access Control
Roel Peeters, Dave Singelée, Bart Preneel
K.U.Leuven ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10
3001 Heverlee (Leuven), Belgium
Tel: +32 16 32 10 50, Fax: +32 16 32 19 69
Email: firstname.lastname@esat.kuleuven.be

Abstract—Conventional access control mechanisms rely on the use of a single security token to authenticate a remote user. However, this introduces a single point of failure in the system, which results in various security and reliability issues. Even more, these systems are often vulnerable to relay attacks. To solve these problems, we propose a threshold-based distance bounding protocol, in which the user’s private key is distributed among a group of personal devices, including resource limited devices such as RFID tags. This significantly improves security and reliability of our system, and offers protection against any subset of \( t \) compromised devices, with \( t + 1 \) being an adjustable threshold number.

Keywords—Access Control; Threshold Cryptography; Distance Bounding Protocols; RFID;

1 INTRODUCTION

Conventional authentication mechanisms are typically based upon something you know (e.g., a password), something you have (e.g., a smartcard) or something you are (e.g., biometrical features). In mobile networks, location information can also be used to enhance mutual entity authentication protocols. For example, to open the door of a building, one needs to have the correct credentials (authorized to enter the building), but also needs to be close to the door. To cryptographically enforce the concept of “proximity”, distance bounding protocols can be employed. These protocols, introduced by Brands and Chaum [1], combine physical and cryptographic properties. These allow a remote prover to authenticate itself, and at the same time enable the verifying party to determine an upper bound \( d \) on the distance between itself and the prover. Employing distance bounding protocols avoids relay attacks, where an adversary being close to the verifier impersonates an authorized user, who is located elsewhere and is being unaware of the attack. Relay attacks are an important threat in access control systems. The practical feasibility of these attacks has been extensively demonstrated, among others in [2], [3].

Enhancing a mutual authentication protocol with location information is however not sufficient. Physical access control mechanisms usually rely on the use of a single security token, such as a contactless smartcard. Each legitimate person carries his own personal token containing identifying information and a secret key. When a user wants to enter the building, he puts his contactless smartcard close to a reader installed in the proximity of the door. Both devices will then carry out a mutual authentication protocol. Using a single security token introduces a single point of failure in the system. On the one hand, this is a security weakness. An adversary can steal a user’s security token and impersonate him. When an attack on a particular token or smartcard is detected, it must be revoked in order to patch the security breach. On the other hand, using a single security token does not provide user convenience. A legitimate user that does not carry around the security token is denied access.

One technique that can help to confront these challenges is threshold cryptography [4]. We observe that people already carry many personal devices with network capabilities, including mobile phones, RFID tags attached to clothing, mp3-players, car keys, etc. By employing threshold cryptography on them, we can exploit this potential.

1.1 Contributions

Our main contribution is a lightweight threshold-based distance bounding protocol that offers protection against any set of \( t \) compromised user’s devices, with \( t + 1 \) being an adjustable threshold number.

In our solution, we particularly envision the setting where some of the personal devices have a private key but do not have the capabilities update this private key or store additional data (e.g., a car key fob). By allowing these resource constrained devices to participate in the access control scheme, one can significantly improve the
security and reliability of the system. As an additional benefit, these devices can be used in several threshold-based distance bounding systems in parallel at no additional cost.

1.2 Organization

In the introduction, we have briefly discussed the concept of proximity based authentication, and put forward the argument that relying on a single security token to authenticate the prover introduces a single point of failure in the system. As a solution, we propose a threshold-based distance bounding protocol, which integrates various cryptographic techniques: distance bounding protocols, bilinear pairings and threshold cryptography. These techniques are briefly discussed in Sect. 2. In Sect. 3, we focus on a typical setting where access control can be enforced by carrying out a threshold-based distance bounding protocol, and give an overview of the necessary assumptions. Our threshold-based distance bounding protocol is described in Sect. 4. The security and implementation cost of our proposed solution is studied in Sect. 5. Related work is presented in Sect. 6. We conclude our work in Sect. 7.

2 BACKGROUND

2.1 Distance Bounding Protocols

Secure distance bounding protocols measure the time of flight to determine an upper bound on the distance between prover and verifier. This measurement is typically performed during a challenge-response protocol, the main building block of the distance bounding protocol. During each of the $m$ fast bit exchanges, the time between sending a challenge and receiving the corresponding response is measured. Multiplying the time of flight with the propagation speed of the communication medium (the speed of light for RF communication) gives an estimation on the distance between prover and verifier. During the run of the distance bounding protocol, the prover should also authenticate himself to the verifier. Without entity authentication, the verifier would not learn anything after having carried out the protocol.

Most distance bounding protocols want to preclude distance fraud and relay attacks. Some protocols also focus on terrorist fraud attacks, but these attacks will not be discussed further in the paper.

- **Distance fraud attacks:** One wants to prevent a dishonest prover claiming to be closer than he really is. This attack is called distance fraud attack and is conceptually shown in Fig. 1.

- **Relay attacks:** These attacks, which are also denoted by mafia fraud attacks, were first described by Desmedt [5]. In this attack scenario, both prover and verifier are honest, but a malicious intruder is performing the fraud. This is a man-in-the-middle attack where the intruder $I$ is modelled as a malicious prover $P$ and verifier $V$ that cooperate, as shown in Fig. 2. The malicious verifier $V$ interacts with the honest prover $P$ and the malicious prover $P$ interacts with the honest verifier $V$. The physical distance between the intruder and the verifier is small. This attack enables the intruder to identify himself to $V$ as $P$ being close to $V$, without either $P$ or $V$ noticing the attack.

Distance bounding protocols were introduced by Brands and Chaum [1]. Since distance fraud and relay attacks are also of major concern for RFID systems, Hancke and Kuhn [6] proposed a distance bounding protocol that is more suitable in this setting. This protocol has the major advantage that it does not require the computation of a signature after the $m$ fast bit exchanges. This comes with the cost of having a false-acceptance rate of $(3/4)^m$ instead of $(1/2)^m$ as in the Brands-Chaum protocol. Hancke and Kuhn also pointed out that distance bounding protocols should be designed to cope well with substantial bit error rates during the rapid single bit exchanges, as these are conducted over noisy wireless ad hoc channels.

The Hancke-Kuhn protocol works as follows. First, prover and verifier exchange a random nonce $(N_P$ and $N_V$ respectively). Both parties then compute a nonprobabilistic function on these exchanged nonces and a shared secret, known to both the prover and the verifier. Typically a pseudorandom function such as HMAC [7] is employed. The output of this function are two $m$-bit sequences $v^{(0)}$ and $v^{(1)}$. Next, a series of $m$ fast bit exchanges is performed. In each round, the verifier sends a random single bit challenge $c_i$ to the prover. If this challenge equals 0, then the prover responds with the $i$-th bit of $v^{(0)}$. If the challenge equals 1, then the prover sends the $i$-th bit of $v^{(1)}$. During this phase, $x$ bit errors are allowed. If at least $(m-x)$ responses are correct, the protocol succeeds.

Recently, several other distance bounding protocols have been proposed in the literature. An overview can be found in [8], [9]. These protocols often try to add new functionality, and/or to decrease the false-acceptance ratio without requiring a signature at the final stage of the protocol. For a long time, there was a lack of platforms that could implement RF distance bounding protocols, due to the strict processing that these protocols require.
in this setting, the private key should never exist in one place. Instead of reconstructing the private key, which would result in a security vulnerability, the participants will provide partial decryptions while only using their share of the private key. By interpolating these partial decryptions, one can compute a valid decryption.

We define a \((t, n)\) threshold cryptosystem, with \(t + 1\) an adjustable threshold number and \(n\) the total number of parties. Any combination of \(t + 1\) participants allows to decrypt ciphertexts. An adversary controlling up to \(t\) shares gains no information about the shared secret. For this reason the threshold number is usually chosen such that \(n \leq 2t + 1\).

The threshold cryptosystem first has to be setup before it can be actually used. A public key and shares of the corresponding private key need to be generated and distributed to the appropriate parties. This setup either involves a trusted dealer or a Distributed Key Generation (DKG) protocol. The setup using a trusted dealer is very efficient, but has as main disadvantage that one party, the trusted dealer, knows the private key that is shared among the parties. In a DKG protocol, a group of parties cooperates to jointly generate a public key and obtain shares of the corresponding private key, without any one entity knowing the private key. In most use case scenarios, including the distributed access control setting we envision in our article, the use of a trusted dealer is acceptable.

3 Distributed Access Control Setting

3.1 Use Case Scenario

Fig. 4 illustrates the setting where access control is enforced to enter a particular building. The user carries a group of personal devices that will jointly perform a proximity-based authentication protocol with the verifier, i.e., a reader placed next to the front door. These personal devices are denoted as end-devices in the rest of the article. One of the end-devices acts as a gateway device. Only this specific personal device will communicate directly with the verifier. The gateway device is also responsible for initiating the access control mechanism. Besides the necessary communication interfaces, the gateway device is required to have on-board or separate dedicated hardware to perform the fast bit exchanges of the distance bounding protocol, and should have sufficient processing power. We envision the user’s mobile
phone, equipped with RF communication technology, to act as the gateway device. Due to the widespread use of smartphones, which are rather powerful and have various communication mechanisms, this is a reasonable assumption to make. Note that it is possible for the user to have more than one gateway device. However, during each run of the protocol only one such device will act as the gateway device, all others will act as end-devices.

In our envisioned setting, we also want to use resource constrained devices that do not have (secure) writable memory (e.g., a car key). These low-cost devices will not be able to (securely) store and update their shares used in cryptographic threshold schemes. However, we assume that during fabrication each device is initialized with its own unique (fixed) private key, stored in secure memory. Secure memory offers protection against attackers that have physical access to the device and want to read out or tamper with the content of the memory. A low-cost technique to realize this feature is the use of Physically Unclonable Function (PUF) [15], [16], where small differences in the fabrication process allow to derive a unique private key for each device.

Simoens, Peeters and Preneel [17] showed how to construct a threshold group for this particular setting by employing bilinear pairings. During setup, shares of the private key are generated in a protected format. No information about the shared private key can be learned from the shares in protected format. As a result, these protected shares can be stored externally, i.e. in the gateway device. During a protocol run, each end-device only needs to use its own unique private key. As a consequence, end-devices can be part of multiple groups without any additional storage cost. For example, one could employ the same set of devices to gain access to the office building and one’s home.

### 3.2 Adversarial Model and Assumptions

We assume that all user’s devices communicate to each other over a dedicated broadcast channel. By dedicated we mean that if a device broadcasts a message, then it is received by all other active devices within communication range and recognized as coming from this device.

We assume the presence of a computationally bounded adversary who can corrupt up to \( t \) devices. The adversary has access to all information stored by the corrupted devices and can manipulate their behavior during the execution of the protocol. The goal of the adversary is to impersonate a genuine prover being close to the verifier. He can largely extend his communication range, and send/receive messages from a large distance. Since RF communication is used, the adversary cannot increase the propagation speed of the communication medium.

The main attack setting one needs to investigate, is an adversary that tries to get authorized access, where:

- the genuine prover is located at large distance from the verifier, and
- the adversary, not knowing the prover’s private key, is physically close to the verifier.

If the adversary would not be close to the verifier, an attack would not make much sense from a practical point of view. To get unauthorized access, the adversary needs to guess the prover’s key or carry out a relay attack where all the verifier’s messages are forwarded to a proxy device that is hidden in the neighborhood of the prover. An adversary can also compromise a set of the user’s end-devices and use these devices to perform partial decryptions. However, the set of compromised devices is assumed to be strictly smaller than \( t + 1 \). As we will show later in the article, our solution is resistant to such attacks.

### 4 Threshold-Based Distance Bounding Protocol

To remove the single point of failure in the system, we transformed the Hancke-Kuhn protocol into a lightweight threshold-based RFID distance bounding protocol, where a gateway device and several end-devices need to collaborate to successfully complete the protocol. The private key of the prover is distributed among all the user’s devices. The verifier (e.g., the office building’s system administrator) could act as a trusted dealer. Alternatively, the user can set up the group with a DKG protocol and register the group’s public key at the verifier. During a run of the distance bounding protocol, the presence of at least \( t + 1 \) user’s devices will result in a correct proximity based authentication claim of the user.

Instead of computing a symmetric pseudo-random function as in the Hancke-Kuhn protocol, we replaced it by an asymmetric decryption function which can be partially evaluated by each of the devices. Next, we integrated the modified Needham-Schroeder public-key protocol [18] into the protocol. It contains the encrypted exchange of random nonces and the demonstration of knowledge of these nonces. This last step is integrated in the fast bit exchange phase. The resulting threshold-based distance bounding protocol is depicted in Fig. 5.

We will now discuss the initial (one-time) threshold setup, and the various steps of the protocol more in detail.

#### 4.1 Threshold Initialization

Each device of the user has a unique private key \( s_i \). We define a non-degenerate bilinear pairing \( \hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T \) and let \( P, Q \) be generators of \( \mathbb{G}_1, \mathbb{G}_2 \) respectively. The devices’ public keys are defined as \( S_i = s_i^{-1}Q \). The shares \( x_i \) of the group’s private key \( x \), used in the threshold-based access control system, are masked with the public key \( S_i \) and stored externally in a protected format as \( C_i = x_iS_i \). The corresponding group’s public key is defined as \( y_P = \hat{e}(P, xQ) \).

Suppose there are \( n \) devices which need to receive a share \( x_i \). To set up these shares a trusted dealer can
setup a random polynomial of degree \( t \) and evaluate the polynomial in \( n \) points:

\[
f(z) = x + c_1 z + \ldots + c_t z^t , \quad x_i = f(i) .
\]

For scenarios in which it is not desired that one party knows the group’s private key, the DKG protocol can be used. For more details we refer the reader to [17].

### 4.2 Encrypted Exchange of Nonces

The distance bounding protocol is carried out as follows. Initially the user has to confirm that he wants to start the protocol, by performing a particular action (e.g., pressing a button) on the gateway device. After this approval by the user, the distance bounding protocol can start.

Next, the prover (i.e., the gateway device) and the verifier will construct an encrypted message as follows. It concatenates the nonces \( N_P \) and \( N_V \) and encrypts them using the public key \( y_P \) of the prover. This ciphertext is sent to the gateway device. To recover the corresponding plaintext \( (N_P || N_V) \), the gateway device performs a threshold-based ElGamal decryption in collaboration with at least \( t \) of the user’s end-devices and checks that the plaintext contains the correct nonce \( N_P \). The threshold-based ElGamal decryption works as follows:

- **Threshold Decryption:** To decrypt the given ciphertext \( (R, e) \) using the private key \( xQ \), compute the plaintext

\[
M = \frac{e}{\hat{e}(R, xQ)} \in \mathbb{G}_T .
\]

More in detail, the verifier constructs the encrypted message as follows. It concatenates the nonces \( N_P \) and \( N_V \) and encrypts them using the public key \( y_P \) of the prover. This ciphertext is sent to the gateway device. To recover the corresponding plaintext \( (N_P || N_V) \), the gateway device performs a threshold-based decryption in collaboration with at least \( t \) of the user’s end-devices and checks that the plaintext contains the correct nonce \( N_P \). The threshold-based ElGamal decryption works as follows:

- **Threshold Decryption:** To decrypt the given ciphertext \( (R, e) \) using the private key \( xQ \), compute the plaintext

\[
M = \frac{e}{\hat{e}(R, xQ)} \in \mathbb{G}_T .
\]

More in detail, the verifier constructs the encrypted message as follows. It concatenates the nonces \( N_P \) and \( N_V \) and encrypts them using the public key \( y_P \) of the prover. This ciphertext is sent to the gateway device. To recover the corresponding plaintext \( (N_P || N_V) \), the gateway device performs a threshold-based decryption in collaboration with at least \( t \) of the user’s end-devices and checks that the plaintext contains the correct nonce \( N_P \). The threshold-based ElGamal decryption works as follows:

<table>
<thead>
<tr>
<th><strong>Prover P</strong></th>
<th><strong>Verifier V</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_P \in_R {0, 1}^z )</td>
<td>( N_V \in_R {0, 1}^z )</td>
</tr>
<tr>
<td>( E_V(N_P</td>
<td></td>
</tr>
<tr>
<td><strong>Threshold decryption</strong></td>
<td>Decrypt, check ( P )</td>
</tr>
<tr>
<td>Check ( N_P )</td>
<td>( b = h(N_V</td>
</tr>
<tr>
<td>( b^{(0)} \leftarrow b_1</td>
<td></td>
</tr>
<tr>
<td>( b^{(1)} \leftarrow b_{m+1}</td>
<td></td>
</tr>
</tbody>
</table>

#### Start of fast bit exchanges

\[
c_i = c_i \in_R \{0, 1\}
\]

\[
\overline{c_i} \cdot b^{(0)}_i + c_i \cdot b^{(1)}_i
\]

#### End of fast bit exchanges

---

Each end-device combines its private key \( s_i \) with \( R \). The pairing of this partial decryption with the corresponding share in protected format is computed by the gateway device, which has stored these protected shares \( C_i \). Note that during this decryption process, the gateway device...
can use these protected shares without these ever having to be revealed in an unprotected format.

4.3 Rapid Bit Exchanges
After the encrypted exchange of the nonces, both the gateway device and the verifier apply a cryptographic hash function $h$, which should be at least collision resistant, on the concatenation of $N_P$ and $N_V$. The output is split in two $m$-bit sequences $b(0)$ and $b(1)$.

Last, a series of $m$ fast bit exchanges is performed. Note that this phase of the protocol should be carried out “as soon as possible” after the exchange of the encrypted nonces. If the delay between both phases is too large (can be chosen by the system integrator), a timeout occurs at the verifier and the protocol fails. In each round of the rapid bit exchanges, the verifier sends a random single bit challenge $c_i$ to the gateway device. If this challenge equals 0, then the gateway device responds with the $i$-th bit of $b(0)$. If the challenge equals 1, then the gateway device sends the $i$-th bit of $b(1)$. In each round, the verifier measures the time between sending $c_i$ and receiving the corresponding response. The maximum round trip time is selected and this measurement determines an upper bound on the estimation of the distance between prover and verifier. If at least $(m-x)$ of the responses sent by the gateway device are correct, the protocol succeeds. The security parameter $x$ denotes the number of allowed bit errors during the rapid bit exchanges and can be tuned to compensate for noisy environments.

5 DISCUSSION
5.1 Security
Our threshold-based distance bounding protocol consists of two main phases: an encrypted exchange of random nonces and the fast bit exchange phase. To provide entity authentication, we integrated the modified Needham-Schroeder public-key protocol into both phases of the protocol. During the fast bit exchanges, the prover demonstrates knowledge of both the nonces $N_P$ and $N_V$. Since the latter is encrypted with the public key of the prover, no one aside the prover could possibly recover this random nonce. The use of $N_P$ links both encrypted messages in the first phase of the protocol to each other. This is important, since the verifier has, at that time, no assurance about the source of the encrypted message. This also prevents replay attacks. A detailed security analysis of the modified Needham-Schroeder public-key protocol can be found in [18].

Our threshold-based distance bounding protocol is designed to prevent relay attacks. Recall that we assume that the genuine prover, the legitimate owner of a set of personal devices, is not located close to the verifier. An adversary that has not compromised the gateway device, has no knowledge of the shares in the protected format and can hence never decrypt the encrypted nonces exchanged in the first phase of the distance bounding protocol. In this scenario, his best attack strategy would be the “ask in advance” strategy [6]. The adversary interleaves the encrypted exchange of nonces and the rapid bit exchange phase, and uses the prover as an oracle to obtain the correct response for one of the challenges in each of the $m$ rounds. As a result, the success probability of the adversary is $(3/4)^m$. This probability slightly changes when one incorporates the effect of bit errors due to noise [9]. Note that the “ask in advance” strategy requires the adversary to trick the prover’s gateway device into starting the threshold-based distance bounding protocol. The latter should also have a threshold number of his personal devices within communication range (not necessarily present at the same location).

In contrast to conventional distance bounding solutions, the adversary cannot authenticate himself successfully by merely stealing one (gateway) device and being located close to the verifier. He would need to compromise or communicate to at least $t$ other end-devices to impersonate the prover. Note that no information about the prover’s private key can be learned from obtaining the protected shares, or the partial decryptions provided by the end-devices.

5.2 Reliability
Besides improving the overall security of the system, one of the main advantages of our threshold-based solution is that it offers enhanced reliability. To illustrate this property, let us compare our distributed access control setting with the solution where only one gateway device is used to enforce access control. When employing the appropriate password protection on this device, using tamper-resistant memory to store the private key, and taking the necessary countermeasures to ensure the device does not get compromised, one could achieve more or less the same security properties as our threshold-based solution. However, when the gateway device gets lost or stolen, the legitimate user loses his access privileges. The same problem arises when the user forgot the device at home, or when it breaks down (e.g., depleted batteries). All these issues are solved by employing our threshold-based solution. As long as the user carries $t+1$ personal devices, including one gateway device, access is granted. Reliability is hence significantly improved.

5.3 Implementation Cost
Since the end-devices are potentially resource constrained, it is important that the cost for these devices is minimized. This was an important design requirement of our threshold-based distance bounding protocol. During a protocol instance, an end-device only needs to perform one scalar-Elliptic Curve point multiplication. Contrary to common belief, Elliptic Curve Cryptography (ECC) can be realized on resource constrained devices such as passive RFID tags. According to [20], an architecture with this functionality can be made compact, while being
energy efficient and having a reasonable processing time. The architecture can be built with less than 15 K gates and the computation of one scalar-EC point multiplication takes $5ms$ and requires 1.18 $\mu J$ of energy. This is currently the most compact RFID implementation of a public-key cryptographic primitive. There is also no need for extra hardware to securely store a share, since we make use of the devices’ factory installed private-key (e.g., by using PUFs [15], [16]).

There is an ongoing debate on the number of gates that can be assigned to cryptographic building blocks in RFID tags. The exact budget that can be spent to security measures depends on the type of RFID tag used, and the application where the protocol is going to be deployed. One can however compare ECC to other approaches. There exist various symmetric-key based authentication protocols for RFID, for which the cryptographic building blocks can be realized with fewer gates than ECC. For example the most compact AES implementation requires 3400 gates and the most compact SHA-3 Round-Two candidate implementation requires about 5 K gates and additional memory. However, these building blocks do not have homomorphic properties. As a result, these symmetric-key based protocols cannot be transformed into a threshold-based variant, which is necessary to avoid having a single point of failure in the system.

Our protocol is particularly designed to be unbalanced, the complexity is shifted towards the gateway device. It needs to perform more computationally intensive operations, such as a pairing over an elliptic curve, and also needs to store the end-devices’ shares in the protected format. These operations are certainly feasible and have already been successfully implemented on a mobile phone [21]. This device also needs dedicated hardware to carry out the rapid bit exchanges of the distance bounding protocol. It has already been demonstrated that RF distance bounding protocols and particularly the rapid bit exchange phase can be realized in practice [11]–[13]. Such a dedicated distance bounding platform can then be connected to the gateway-device, which performs the other steps of the protocol.

6 RELATED WORK

Although the idea of combining distance bounding protocols and threshold cryptography was already proposed by the authors in [22], there are some important differences between both approaches. The focus of the previously proposed solution was to allow the legitimate user to control his security settings, by allowing him to dynamically change the composition of the group of devices and threshold number according to his needs. Due to this focus on updating shares, each device needed to store and update its share in additional secure memory. In our new solution, the focus is on increasing the security level and reliability in an efficient way. This is done by allowing resource constrained devices to participate to the protocol as end-devices.

Furthermore, in the previous work each device had to compute a partial RSA signature, which is an expensive operation [23]. The most compact commercial available RSA coprocessor is the TINY32 of Helion [24]. This architecture requires 8 K gates circuit area and 10 Kbits of ROM. However, to get an acceptable processing time (200 ms for a RSA signature) the clock frequency, and hence also the required energy, needs to be orders of magnitude higher than possible in a passive RFID tag. Another problem where RSA suffers from, in comparison with ECC, is that in order to maintain the same level of security over time, the key size needs to grow exponentially instead of linearly [25]. This means that the difference in required storage and computational effort, compared to ECC, will become larger over time.

7 CONCLUSION

Conventional access control mechanisms suffer from various security and reliability issues. These systems are often prone to relay attacks, in which an adversary being close to the verifier exploits a remote legitimate user to impersonate him. Furthermore, by relying on a single security token to authenticate a remote user, a single point of failure is introduced in the system. Not only does this allow an adversary to get unauthorized access by stealing a legitimate user’s security token, it also results in denial of access to this user. The same problem arises when the user forgets the security token, or when it breaks down.

The functionality of the devices one already carries can be extended by employing threshold cryptography on them. Relay attacks can be precluded by using distance bounding protocols. In this paper, we combine these observations, and present a threshold-based distance bounding protocol. The user’s private key is distributed among a group of personal devices. By including resource limited devices such as RFID tags, security and reliability can be even further improved. Our solution offers protection against any subset of $t$ compromised devices and the legitimate user is granted access as long as he carries $t + 1$ personal devices. Since each end-device only needs to perform one elliptic curve point multiplication using its own unique private key, our solution is well suited to be realized on resource constrained devices. As an additional benefit, these devices can be used in parallel in several threshold-based distance bounding systems at no additional storage cost.

ACKNOWLEDGMENT

This work is funded by the Katholieke Universiteit Leuven, and supported in part by the Concerted Research Action (GOA) Ambiotics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and by the Flemish IBBT projects. Roel Peeters is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).
REFERENCES


Roel Peeters obtained an Electrical Engineering degree at the Katholieke Universiteit Leuven (Belgium) in 2007. After his studies he started a PhD at the COSIC (Computer Security and Industrial Cryptography) research group at that same university. His research interests lie in the fields of threshold cryptography and mobile security.

Dave Singelée received the Master’s degree of Electrical Engineering and a PhD in Applied Sciences in 2002 and 2008 respectively, both from the Katholieke Universiteit Leuven (Belgium). He worked as an ICT security consultant at PricewaterhouseCoopers Belgium, and is currently a postdoctoral researcher at the research group COSIC. His main research interests are cryptography, security and privacy of wireless communication networks, cryptographic authentication protocols for RFID, and secure localization schemes. He has authored and co-authored more than 20 scientific publications, and participated in various research projects.

Bart Preneel received the Electrical Engineering degree and the Doctorate in Applied Sciences from the Katholieke Universiteit Leuven (Belgium), where he is currently full professor (gewoon hoogleraar) and head of the research group COSIC. He is president of the IACR (International Association for Cryptologic Research) and a member of the Editorial Board of the Journal of Cryptology, the IEEE Transactions on Information Forensics and Security, and the International Journal of Information and Computer Security. He is also a Member of the Accreditation Board of the Computer and Communications Security Reviews (ANBAR, UK). His main research interests are cryptography and information security.