Threshold-Based Location-Aware Access Control

Roel Peeters, Dave Singelée and Bart Preneel
K.U.LEUVEN - ESAT/COSIC & IBBT, Belgium

ABSTRACT

Designing a secure, resilient and user-friendly access control system is a challenging task. In this article, a threshold-based location-aware access control mechanism is proposed. In our design, we uniquely combine the concepts of secret sharing and distance bounding protocols to tackle various security vulnerabilities. The proposed solution makes use of the fact that the user carries around various personal devices. Our solution offers protection against any set of $(t - 1)$ or fewer compromised user's devices, with t being an adjustable threshold number. It removes the single point of failure in the system, as access is granted when one carries any set of t user's devices. Additionally it supports user-centered management, since users can alter the set of personal devices and can adjust the security parameters of the access control scheme towards their required level of security and reliability.

Keywords: Access Control, Authentication, Digital Signature, Mobile Technologies, Human/computer interaction

INTRODUCTION

Contactless smartcards are often used to enforce access control for secure facilities and buildings. These security tokens contain identifying information and a secret key, used to identify the user carrying the smartcard. When the user approaches 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 challenge-response protocol, in which the user's smartcard authenticates itself to the reader (in some scenarios, mutual authentication is required). If the protocol finishes successfully, the user is granted access. Besides access to building, similar mechanisms are employed to enter a car (Microchip KeeLoq), to use public transport (Octopus Cards, OV-chipkaart, Oyster Online), and even for payments with contactless credit cards (Mastercard PayPass, Visa Paywave).

Although widely used, this conventional access control solution has some important drawbacks, such as several security vulnerabilities. First, the use of a single security token introduces a single point of failure in the system. If this token gets stolen, an unauthorized adversary could get access to a secure building or resource. Security tokens and smartcards could also be compromised or cloned. A recent example of the latter was the MIFARE attack discovered by Gans *et al.* (2008).

A second security vulnerability is relay attacks, which are also known as mafia fraud attacks. These are man-in-the-middle attacks where a verifier (e.g., the reader next to the door of a building) is tricked in believing that a prover (e.g., the smartcard) is in its close vicinity by an adversary surreptitiously forwarding the signal between the verifier and an out-of-range prover

(Kim *et al.*, 2009). Such an attack is important in the setting of access control systems, particularly when challenge-response protocols are employed, and should definitely be avoided.

In addition, both reliability and user-friendliness could be improved in conventional access control systems. For each system the user is enrolled to, and this can be a relatively high number, he has to carry around a separate smartcard or security token. The legitimate user that does not carry around the security token automatically cannot get access. Furthermore, revocation of a particular token is often a cumbersome and relatively slow process. This is illustrated by the following plausible scenario. When initiating the revocation process, the user first informs the facility manager. Second, revocation lists are updated and distributed. Third, the user gets a new token or smartcard. Since such a revocation process is slow, it also poses a security risk: there is a grace period in which the adversary can still use the token before the revocation lists are updated.

Fortunately, both security vulnerabilities can be tackled by introducing several countermeasures. The single point of failure can be removed by sharing the secret over a set of user's personal devices. The vulnerability against relay attacks can be solved by using distance bounding protocols. In addition, secret sharing also provides reliability to the user and allows for, through the mechanism of resharing, user-centered access control. It hence automatically improves the user-friendliness of the system.

Secret sharing

The concept of secret sharing was first introduced by Shamir (1979). Instead of storing a secret on one device, the secret is divided into k pieces, each stored on different devices. Let t be a threshold number chosen by the user. This parameter directly relates to the security level of the scheme. The main characteristic of secret sharing is that the key can easily be reconstructed from any t pieces, but even complete knowledge of $(t - 1)$ pieces reveals absolutely no information about the secret. These shares are evaluations of a unique polynomial of degree $(t - 1)$. For $(t - 1)$ or fewer shares, one does not obtain any information at all about the secret, as one cannot reconstruct the unique underlying polynomial. For t or less stolen devices, there is hence no need for revocation as the adversary obtains no information about the secret. Secret sharing also allows for user-centered access control, since the user can decide which devices get a piece of the key, how large the threshold value t should be, and when the secret shares should be updated (i.e. when the resharing process should take place).

Outline

In the introduction, we showed that conventional access control mechanisms suffer from several security, reliability and usability issues. We put forward the idea of employing distance bounding protocols in combination with secret sharing on personal devices to solve several of these problems. In the next section, the general principles of distance bounding protocols are discussed more detail. Next we describe our threshold based access control scheme, which uses distance bounding protocols to tackle relay attacks. We also describe how a user can manage the access control configuration. To demonstrate that our proposed access control scheme solves the identified vulnerabilities, we discuss the security properties of our solution more in detail.

DISTANCE BOUNDING PROTOCOLS

To enhance entity authentication protocols, location information can be incorporated. Entities which are in a specific location or within a certain range of a particular device will be granted some privileges, in contrast to all other entities. Distance bounding protocols, which have been

introduced by Brands & Chaum (1994), can be used to cryptographically enforce the notion of “proximity”. The concept of proximity based authentication is depicted in Figure 1. Authentication requests originating from devices that are located within the range d of the verifier V are accepted, all other requests are rejected.

Figure 1: The concept of proximity based authentication: an authentication request from device A is accepted, and one from device B is rejected.

Distance bounding protocols combine physical and cryptographic properties to enable a verifying party (i.e. the *verifier*) to determine an upper bound on the distance between itself and a *prover*, who claims to be within a certain range.

The main building block of a distance bounding protocol is a challenge-response protocol which is executed n times. The number of rounds n is a security parameter. During each of the n rounds, which are also called *rapid bit exchanges*, the verifier measures the time between sending a challenge and receiving the response. The highest round trip time is then selected and multiplied with the propagation speed of the communication medium. This gives an estimate, i.e. an upper bound, on the distance between prover and verifier.

Let us now focus more on this distance estimation. Although the principle of measuring the round trip time during n rounds of the protocol is rather straightforward, one should take into account some practical details such as:

- Processing delay: It is important to notice that the round trip time is not equal to the propagation delay. For example, it takes some time to compute and transmit (or receive) the response. The variation on the processing delay should hence be as small as possible compared to the propagation time, because we are only interested in the latter. The uncertainty on the distance estimation directly depends on this variation. Unfortunately, in some settings it is not possible to determine or control the processing delay exactly. To solve this issue, one often tries to minimize the processing delay (e.g., by using very simple hardware operations in the protocol), such that its variation can be neglected compared to the time of flight. Another strategy is to increase the propagation delay by selecting an appropriate communication medium. One should however take into account that accuracy is typically not the main design requirement in the setting of proximity-based authentication. One wants to compute an upper bound on the distance, not the exact location of the prover.

- Communication medium: Two common communication technologies are ultra-sound and RF communication. Both have their specific (dis)advantages. The former is rather slow, and the processing delay can hence be neglected. The accuracy of the timing measurements is also not very critical in this scenario. The technology is however susceptible to *wormhole attacks*, where an adversary which is physically located between prover and verifier, forwards the signal between two distant points using a faster communication medium. RF communication is resistant to these attacks, but imposes more strict requirements on hardware, which could be a bottleneck. The round trip time should be measured very accurately, and the processing delay should be fixed and/or very small. These requirements can however be realized in practice. A prototype system has been implemented by Rasmussen and Capkun (2010) where the prover is able to receive, process and transmit signals in less than 1 ns.
- Transmission format: Clulow et al. (2006) have shown that one should also optimize the choice of transmission format when designing distance bounding protocols, to preclude certain physical attacks. They recommend using a communication format in which only a single bit is transmitted and the recipient can instantly react on its reception. In each of the n rounds, a single-bit challenge-response protocol should be carried out.
- Mobility: Since distance bounding protocols are carried out in mobile networks, prover and/or verifier could be mobile. This is however not a problem, because of two reasons. First, the execution time of the n fast bit exchanges is rather short. In realistic use case scenarios, the prover and/or verifier will have barely moved. This small execution time also limits the effect of a mobile adversary. Second, small variations in the round trip time, which could be caused by movement or by a cheating prover, are cancelled out by selecting the maximum round trip time. This maximum is not affected when prover and verifier are closer to each other in a few rounds.

In contrast to many distance estimation solutions, distance bounding protocols particularly focus on security. The verifier which carries out this protocol with a remote prover, wants to be able to check both the prover's identity, as the correctness of the latter's claim to be in the proximity of the verifier. To design a protocol that achieves these requirements, one should take into account some important cryptographic principles. It should be impossible for the prover to send the response before receiving the challenge from the verifier. Otherwise, the prover can pretend to be closer than he really is. This implies that the response should depend on the (random) challenge. A second requirement is that the prover cryptographically identifies itself during the execution of the distance bounding protocol (e.g., by using a secret key). Otherwise, the verifier does not know which entity is within a certain range.

The typical layout of a distance bounding protocol is depicted in Figure 2. Note that there are three distinct phases. In the first phase, prover and verifier can exchange some random nonces, create commitments, etc. The second and most important phase of the protocol are the n rounds of rapid bit exchanges. The protocol ends with a post-computation phase, in which signatures can be computed, commitments being opened, etc. In some protocols, the first and/or third phase is omitted.

Figure 2: Three phases of distance bounding protocols.

By employing the principles discussed above, one can design a distance bounding protocol which precludes one or more of the following attacks:

- **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 Figure 3. Distance bounding protocols are particularly designed to prevent this type of attack.

Figure 3: Distance fraud attack.

- **Mafia fraud attacks:** These were first described by Desmedt (1988). 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 modeled as a malicious prover $\bar{P}$ and verifier $\bar{V}$ that cooperate, as shown in Figure 4. The malicious verifier $\bar{V}$ interacts with the honest prover P and the malicious prover $\bar{P}$ 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. Drimer & Murdoch (2007) have presented a practical mafia fraud attack on the United Kingdom's EMV payment system Chip & Pin.

Figure 4: Mafia fraud attack.

- **Terrorist fraud attacks:** These are an extension of mafia fraud attacks (Desmedt, 1988). The intruder (being close to the verifier) and the prover will collaborate in this attack. It is important to note that the intruder must not know the private key of the prover, since the latter does not fully trust the former. If the intruder would know this private key, then it is impossible to make a distinction between the intruder and the prover, and as a result, terrorist fraud attacks can no longer be prevented. They would be the same party from a cryptographic point of view. The concept of a terrorist fraud attack is shown in Figure 5.

Figure 5: Terrorist fraud attack.

In this article, we will not focus on terrorist fraud attacks, and only concentrate on secure distance bounding protocols which prevent distance and mafia fraud attacks. Although the idea has been introduced more than fifteen years ago by Brands & Chaum (1994), it is only quite recently that distance bounding protocols attracted the attention of the research community. Hancke & Kuhn (2005) 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. They incorporated this important requirement in the design of their RFID distance bounding protocol. Singelée & Preneel (2007) have proposed a noise resilient extension of the Brands-Chaum protocol that provides mutual entity authentication. Recently, various other distance bounding protocols have been proposed in the literature. A short overview can be found in Avoine & Tchamkerten(2009), Rasmussen & Capkun (2009), and Kim *et al.*(2009).

LOCATION-AWARE ACCESS CONTROL

We focus on a location-aware access control mechanism where access is granted based on the combination of identity and proximity information. The setting is illustrated in Figure 6, where access control is enforced to enter a particular building. The user has a group of personal devices that will carry out a proximity-based authentication protocol with a verifying entity, e.g., a reader placed next to the door of the building. The former will be denoted by the prover, the latter by the verifier. If the protocol finished successfully, the user can enter the building. Only one of the user's devices will communicate directly with the verifying entity. This device is denoted by the gateway device in the rest of the article. The gateway device is also responsible for initiating the access control mechanism with the verifier. The other devices of the user are called end-devices. They

will also contribute in the access control scheme, but will not communicate directly with the verifier. We envision the user's mobile phone to act as gateway device, but the user can also choose to use one of his other devices. The only requirement for the gateway device is that it should have sufficient computational resources (to carry out cryptographic algorithms such as hash functions and digital signatures) and that it has an adequate user-interface. Note that it is not required to use the same gateway device in different instances of the protocol. However, in each run of the protocol, there is always exactly one gateway device; all other user's devices act as end-device. Since only the gateway device will communicate directly with the verifying entity, the end-devices do not need to be able to carry out a distance bounding protocol. The gateway device can use onboard or separate dedicated hardware (connected to the device) to perform the rapid bit exchanges during the run of the protocol.

Figure 6: Distributed access control setting.

As already discussed, by carrying out a distance bounding protocol, a verifying party can determine an upper bound on the distance between itself and a prover, who claims to be within a certain range. Cryptographic distance bounding protocols often require the prover and the verifier to compute a non-probabilistic function on a known input and a cryptographic key, known to both the prover and the verifier. Typically a pseudorandom function such as HMAC (Bellare *et al.*, 1996) or CBC-MAC (ISO/IEC 9797, 1994) is used. Such functions cannot be used in our access control mechanism, as we have chosen for a distributed solution where a gateway device and several end-devices need to collaborate to successfully complete the protocol. Instead, we will distribute the cryptographic key among the user's devices, and replace the pseudo-random function by a cryptographic function which can be partially evaluated by each of the user's devices¹. Each share of the cryptographic key is stored on a separate device. As a consequence, in contrast to the implementation of conventional access control systems, the adversary needs to compromise at least t devices instead of just one security token. Afterwards, the gateway device can then combine several of these partial evaluations to obtain the evaluation of the cryptographic function. An interesting function that has all these properties is the computation of an RSA signature (Rivest *et al.*, 1978).

¹One only needs a threshold number of cooperating devices to reconstruct the secret and hence evaluate the cryptographic function.

Through the mechanism of resharing, we also support user-centered management. Users can alter the set of personal devices. They can add or remove devices from this set, and can tune the security parameters of the access control scheme towards the required level of security and reliability.

Adversarial model and assumptions

Each device has its own public-private key pair. Public keys of potential gateway devices are known to all devices. All devices' public keys are known to devices that can contribute in resharing. How to register the public key with other devices is out of the scope of this article. One can use for example pairing protocols to authenticate the public key. These protocols are standardized in the ISO/IEC 9798-6 standard (2005).

The goal of the adversary is to get unauthorized access (e.g., enter a building). To achieve this objective, the adversary can perform passive and/or active attacks. The adversary, which is computationally bounded, can largely extend his communication range, and send/receive messages from a large distance. However, we assume that the adversary cannot increase the propagation speed of the communication medium. This implicitly implies that RF communication is used, or that the adversary cannot carry out a wormhole attack when a slower communication medium, such as ultra-sound, is used. The latter however requires additional countermeasures, which will not be further discussed. Also the influence of multipath propagation, interference with other wireless signals, etc. are not discussed in this article.

There are two scenarios that we need to investigate: the prover being located at a large distance from the verifier, and the prover being in the proximity of the verifier. The adversary (i.e. the person in control of the attacks) is assumed to be physically close to the verifier in both scenarios (otherwise, an attack would not make much sense from a practical point of view). Let us first focus on the first scenario. To get unauthorized access, the adversary needs to carry out a mafia fraud attack and forward all messages to a proxy device that is hidden in the neighborhood of the prover. An adversary can also compromise a subset of the user devices and use these devices to perform the mafia fraud attack (i.e. sign particular data). This subset is however assumed to be strictly smaller than the threshold number t . As we will show later in the article, our solution is resistant to such mafia fraud attacks. In the second scenario, where the user's devices are close to the verifier, we assume that the user can physically verify the presence of the adversary. In this setting, other (even non-technical) attacks are more likely to take place (e.g., such as trying to sneak into the building when the prover enters). We hence mainly need to focus on the first scenario.

Our threshold-based solution

We will now present our threshold-based access control solution that uses distance bounding protocols. When carrying out our scheme, several steps need to be performed. Initially, each new user needs to be registered during the enrolment phase. After this phase, which only needs to be done once for each new entity, the user is ready to actively use the access control mechanism. Access control is carried out by conducting a distance bounding protocol based on digital signatures. One of the main components of this protocol is a distributed RSA signature generation. For security and usability reasons, the private key used to compute the distributed RSA signature needs to be reshared at regular intervals.

Enrolment phase

Before a new user can actively use the access control mechanism, he first has to enrol. This enrolment phase is rather similar to the initialization phase of a conventional access control scheme, but there are some important differences. Instead of generating a shared secret key and storing it on the user's contactless smartcard, the verifier will generate a shared private RSA key that will be distributed among the personal devices of the user. There are two options. The verifier can send the private key to the gateway device, which will then share the key among the user's devices; each device will receive a share d_i . The device's shares are constructed by evaluating a polynomial of degree $(t - 1)$, with the polynomial's constant factor equal to the secret to be shared, in their corresponding identities (Shamir, 1979). Next the gateway device should delete the private key. Another solution is that the verifier itself acts as the trusted dealer, since it knows the private key.

The initial secret sharing phase goes as follows. The trusted dealer first broadcasts the public RSA parameters e, N . Next it computes and distributes a random number v and the initial verification keys $v_1 \dots v_k$ (with $v_i = v^{d_i}$). Each device i then gets its initial share d_i of the private key over a private channel. Techniques on how to construct a private channel are out of the scope of this article. The verification keys are stored by all the user's devices. These are used during resharing and to verify the gateway device's knowledge of its share, needed for the distributed signature generation.

Distance bounding using digital signatures

As already discussed, we have opted to use a threshold-based distance bounding protocol to enable a verifying party to check that a threshold number of particular devices are within a certain range. A group of personal devices will collaborate during the proximity-based authentication process. We started from the Hancke-Kuhn protocol (2005), as it is noise resilient and does not require the computation of a signature at the end of the protocol. We slightly modified the protocol in such way that instead of a pseudo-random function, prover and verifier need to compute an RSA signature S on a message M , which depends on the nonces exchanged between prover and verifier. We will use RSA signatures as defined in PKCS #1 version 2.1 (2002), but for simplicity reasons denote the encoded message as M . The resulting distance bounding protocol is depicted in Figure 7.

The protocol works as follows. Initially the user has to confirm that he wants to start the access control mechanism with a specific verifier, 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. One should note that in most use case scenarios, only one verifier will be in close proximity. There is hence no risk that a protocol run will be carried out with two verifiers in parallel, which could cause ping pong effects. If two verifiers are close by, the prover selects the verifier with whom he wants to carry out the protocol. After this initialization phase, both prover and verifier are ready to carry out the protocol. Only these two parties will execute the protocol, messages from other entities will be ignored.

When the protocol is initiated, the prover and verifier first exchange a random nonce, N_p and N_v respectively. Both parties then compute an RSA signature S , using the private key d which is known to both parties, on the message M , which is the result of applying a cryptographic hash function on the concatenation of N_v and N_p .

$$S = M^d \bmod N, \quad \text{for } M = h(N_v || N_p).$$

Figure 7: Threshold-based distance bounding protocol.

The verifier can directly compute the RSA signature on the message. However, at the prover's side the private key d is shared among the user's personal devices. The RSA signature on the message is generated in a distributed way. More details on the distributed RSA signature generation can be found in the next section.

After applying a deterministic reduction² function RED on the signature, the result is split in two n -bit sequences $r^{(0)}$ and $r^{(1)}$.

$$RED(S) = r^{(0)} || r^{(1)}.$$

Then a series of n rapid bit exchanges is performed. In each round, the verifier sends a random single bit challenge C_i to the prover. If this challenge equals zero, then the prover responds with the i -th bit of $r^{(0)}$. If the challenge equals one, then the prover sends the i -th bit of $r^{(1)}$. 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 $(n - x)$ of the responses sent by the prover are correct (the security parameter x denotes the number of allowed bit errors during the rapid bit exchange), the protocol succeeds.

Distributed RSA signature generation

Desmedt & Frankel (1991) proposed the first (non-robust) threshold RSA signature scheme. Later robust, but less practical threshold RSA signatures schemes were proposed by Frankel *et al.* (1997) and Rabin (1998). The first practical, robust threshold RSA signature scheme was presented by Shoup (2000). We will use this technique to generate the signature S on the message M .

The gateway device will initiate the distributed RSA signature generation. This device broadcasts the message M together with a zero knowledge proof (denoted by ZKP) of its share.

²The reduction function RED reduces the bitlength of the input to a fixed bit length $2n$.

The ZKP consists of a Schnorr signature (1989) (c, z) on the message and the public RSA key e . The gateway device chooses r at random in the interval $\{0 \dots 2^{L(N)+2L_1-1}\}$, with $L(N)$ the bitlength of N and L_1 a secondary security parameter.

$$v' = v^r ; \quad c = H(v', M, e, N) ; \quad z = d_i c + r.$$

The end-devices will first verify the ZKP, by computing v' , using the public verification key v_i , and checking c :

$$v' = v^z v_i^{-c} ; \quad c = H(v', M, e, N).$$

Note that by carrying out this ZKP, one has the guarantee that the distributed signature generation is initiated by a (trusted) gateway device, and not by a (hidden) proxy device controlled by the adversary. Next, the end-devices broadcast their partial signatures S_i encrypted with the public key of the gateway device.

$$S_i = M^{2\Delta d_i}.$$

The result is that only the gateway device can combine the partial signatures (using a subset of t devices, with t being the secret sharing threshold) into the signature on the message M . The Lagrange multipliers are used to interpolate through points on the polynomial of degree $(t-1)$. Let λ_i be the Lagrange multipliers multiplied with $\Delta = \prod_{i=1}^k i$. The latter is necessary to get integer values, since the order of the subgroup $\varphi(N)$ is unknown.

$$w = \prod_i S_i^{\lambda_i} , \quad \text{for } \lambda_i = \Delta \prod_{j \neq i} \frac{j-i}{j}.$$

We now have $w^e = M^{4\Delta^2}$. Since e and $4\Delta^2$ are coprime, we can find values α and β , for which $\alpha 4\Delta^2 + \beta e = 1$, using Euclid's algorithm.

$$S = w^\alpha M^\beta.$$

The gateway device can verify the correctness of the signature S using the public RSA parameters e, N .

$$S^e \bmod N = M.$$

If the verification of the signature does not hold, the subset contains one or more cheating devices. The gateway device then selects a different subset of t devices and recomputes the signature. This process is repeated until the verification of S succeeds.

Resharing

By carrying out secret resharing, new shares of the secret are generated and the old shares are rendered useless. This means that an adversary is forced to break the scheme within the time frame between two consecutive instances of resharing. Resharing also allows to go from a (t, k) -secret sharing to a (t', k') -secret sharing, with t' the new threshold number and k' the new number of participants. The threshold number t determines the level of security, i.e., the number of devices an adversary needs to compromise within the available timeframe. The number of devices k together with the threshold number t determines the level of reliability, since the legitimate user will still be able to use the scheme when the combined number of devices that are not present and compromised by an adversary is less than $(k-t)$. This is very important from a usability point of view.

Because of the resharing mechanism, our access control solution supports user-centered management. The set of personal devices, of which a subset of at least t devices is needed during the authentication phase, can be changed, hence partial signature rights can be revoked and/or

granted. Resharing is typically required when a device gets stolen, or when a user purchases a new device. From the moment a device is identified as compromised, it should be excluded from the set of devices that share the private RSA key. It will hence not receive a new share and its old share will be rendered useless.

Secret resharing, without reconstruction of the secret, was first described by Desmedt & Jajodia (1997) and Frankel *et al.* (1997). We will use the techniques proposed by Wong *et al.* (2002), where new participants can verify the validity of their shares.

The resharing mechanism works as follows. Basically, every contributing device constructs a polynomial of degree $(t' - 1)$, with t' the new threshold number. The number of contributing devices needs to be at least t , since this many shares are needed to reconstruct the original shared secret. Commitments to the coefficients c_{ij} of the constructed polynomial are broadcast.

$$f_i(x) = d_i + c_{i1}x + \dots + c_{i(k'-1)}x^{t'-1}, \text{ for } \forall j \in \{1, \dots, t'\}: C_{ij} = v^{c_{ij}}.$$

Subshares (evaluations of the polynomial $f_i(x)$) are handed out to the set of participating devices:

$$\forall j \in \{1, \dots, k'\}: d_{ij} = f_i(j).$$

The commitments to the coefficients of the polynomial and verifications keys allow validating these subshares.

$$v^{d_{ij}} = v_i \prod_{l=1}^{t'-1} C_{il}^{j^l}.$$

Contributing devices, for which all sent subshares validate, are added to the set of qualified dealers. The size of this set should be at least t to continue. The subshares from qualified dealers are combined into the new shares. Next, new verification keys are broadcasted.

$$d'_j = \frac{\sum d_{ij}\lambda_i}{\Delta}; \quad v_{j'} = v^{d'_j}.$$

The validity of these new verification keys, hence the new shares, can be tested by combining them:

$$v^\Delta = \left(\prod v_{i'}^{\lambda_i} \right)^e.$$

Please note that Δ and λ_i in the last equation need to be calculated for the new set of participants, if the set of participants is changed in this resharing instance.

DISCUSSION

Security analysis

Our threshold-based location-aware access control scheme improves the resilience to various important security vulnerabilities. We will now briefly discuss its security properties, although without giving formal proofs.

Please note that if an adversary succeeds in compromising the threshold number t of devices at one particular point in time, he can reconstruct the private RSA key, and hence completely break the access control scheme.

Relay attacks

One of the main components in our threshold-based location-aware access control scheme is the slightly modified distance bounding protocol of Hancke and Kuhn. This protocol is employed to prevent relay attacks. The adversary cannot conduct a protocol run simultaneously with the prover and the verifier, as this would increase the time of flight (and hence cause the distance bounding protocol to fail).

To analyze the resilience to relay attacks, we will now discuss the security properties of the distance bounding protocol more in detail. An adversary, who wants to authenticate himself successfully to the verifier, can follow several attack strategies. The first strategy is the *no-ask* strategy. The adversary just guesses all the responses during the rapid bit exchange phase. His success probability would then be $1/2^n$. However, the best strategy is the *ask-in-advance* strategy. Before discussing the attack more in detail, one should note that our distance bounding protocol starts by the prover and verifier exchanging some random nonces. Since the time is not measured during this stage of the protocol, the adversary does not yet interfere. The attack, which consists of two consecutive phases, will take place during the fast bit exchanges.

In the first phase, the adversary carries out a protocol run with the honest prover. During each of the n rounds, he guesses the challenges C_i in advance. To finish the protocol run, the adversary needs to trick the user in approving that the distance bounding protocol is being carried out. Let us for now assume that the adversary succeeds in this goal (we will come back to this issue later in this article). At the end of the protocol run, the adversary will hence have received n responses r_i .

In the second phase of the attack, the adversary is now located close to the verifier and performs a protocol run with the latter. In each round of the fast bit exchanges, the verifier sends a random challenge, and expects a correct response from the adversary. There are two scenarios. If the verifier sends a challenge C_i equal to the challenge used in the first phase of the attack (i.e. if the adversary has guessed the challenge correctly), then the adversary replies with the response r_i received from the prover. This response will always be correct. If the adversary has guessed the challenge C_i wrongly, he cannot use the response r_i . As a result, he will send a random response to the verifier. In this case, he has a $1/2$ probability of replying with a correct response. Since each scenario occurs with a probability of $1/2$, it is relatively easy to compute the overall success probability of the relay attack (Hancke & Kuhn, 2005). By following the *ask-in-advance* strategy, the adversary has in each of the n rounds a probability of $3/4$ to send a correct response. If the attack succeeds, the adversary is able to wrongfully convince the verifier that an entity in possession of the private RSA key is in the vicinity.

The attack probability slightly improves when one incorporates the effect of bit errors due to noise (Singelée & Preneel, 2007). This is however outside the scope of this article.

User interaction improves security

As explained above, an adversary needs to trick the user in carrying out a run of the distance bounding protocol without being in the vicinity of the verifier. This goal can be achieved by compromising at least one device with a user interface (i.e. a gateway device), since these devices can initiate a protocol instance.

A compromised gateway device could carry out mafia fraud attacks, where it requests the other devices in the network to compute a distributed RSA signature. To mitigate this security risk, one could require the user to approve this request on a predefined number of end-devices with a user interface. Analogue to the gateway device, these devices need to proof knowledge of their shares.

This means an adversary would have to compromise at least this predefined number of devices with a user interface to carry out a successful mafia fraud attacks.

By varying this predefined number, one can change the adversary's success probability (and hence the security level of the protocol). Peeters *et al.* (2009a) showed that by having a user verifying his request at only a small number of devices with a user-interface, the adversary's probability of success already reduces drastically. This is hence an interesting solution to further improve the security level of the system at a limited cost (in terms of user friendliness).

User-centered configuration

End users will be in control of the security of their system, and be able to adjust the security parameters of the access control scheme towards their preferences. Through the mechanism of resharing, they can alter the set of personal devices and/or change the threshold number t . We recommend that the default value of the threshold number should be equal to $t = \frac{k+1}{2}$. This parameter choice gives the best trade-off between security and reliability. A higher threshold number would make the system more secure, but the number of active devices that the user needs to carry would also increase, resulting in a higher probability to be denied access (i.e. more devices are needed to successfully authenticate).

Let us illustrate this by observing three concrete scenarios where resharing takes place: adding a device, removing a device and refreshing the shares of the private key. A preliminary usability study conducted by the authors (Peeters *et al.*, 2009b) showed that resharing in these three concrete scenarios is perceived by the end users as three different operations. From a security perspective, adding and removing devices always needs authorization by the end user, as an adversary could gain an advantage by performing these operations. For example, the adversary could try to add devices that are under his control, or remove devices that are not under his control. Furthermore, adding a new device provides an additional technical challenge as the device is not yet known to the group of other personal devices. The public key of the new device needs to be securely transferred to these personal devices and the public key needs to be authenticated. More details on how to perform resharing that is authorized by the end user can be found in (Peeters *et al.*, 2009a).

Compromised devices, e.g. stolen or lost devices, can be removed from the group without these needing to be present. This should be done as soon as possible after the detection of the compromise. Refreshing the shares helps to improve security when the end user is in possession of devices of which the adversary has obtained the share, but which are no longer controlled by the adversary. For example, if one leaves his mobile phone unattended during a lunch break, an adversary could extract the value of the share and afterwards could put the phone back in such a way that the legitimate owner will never notice it has been compromised. Refreshing the shares solves this issue. Note that refreshing does not help against actively compromised devices (e.g., on which root kits or other malware is installed). If the attacker is still in control of the device, he can obtain the new value of the share. However note that the access control system itself remains secure as long as fewer than t devices are compromised.

CONCLUSION

There is an evident need for the design of innovative secure and user-friendly access control systems. Conventional systems entail several security vulnerabilities. By using a single security token, one introduces a single point of failure in the system. Users cannot authenticate themselves without their token. Moreover, an adversary that steals or compromises a token will get the access

privileges of the corresponding user, until revocation has taken place. Challenge-response protocols conducted in wireless networks are also vulnerable to relay attacks.

In addition to these security vulnerabilities, conventional access control mechanisms often suffer from usability issues. Carrying out changes (such as revoking or updating keys) to centrally managed systems tends to be a rather slow and cumbersome process. Users can also not adjust the security properties of the access control scheme, as this is enforced by the system itself.

In this article, we proposed a threshold-based location-aware access control mechanism which combines the concepts of secret sharing and distance bounding protocols. Its main component is a distributed RSA signature generated by t user's devices: one gateway device and $(t - 1)$ end-devices. The gateway device interacts directly with the verifier and combines the partial signatures. We particularly envision the user's mobile phone to act as gateway device in our access control scheme.

We demonstrated that our solution solves the dependency on a single token, and is resistant to relay attacks. It offers protection against any set of $(t - 1)$ or fewer compromised user's devices. Compared to conventional access control mechanisms, our solution could offer an increased level of user-friendliness as it supports user-centered management. Users can vary the set of personal devices, of which at least t need to be present during authentication. This threshold number determines the security level of our access control scheme and can be freely adapted by the user.

To further support the claim that our proposed access control scheme improves user-friendliness, there is a need for a large usability study in which quantitative data is gathered. This will be part of future research.

ACKNOWLEDGEMENTS

This work is funded by the Katholieke Universiteit Leuven, and supported in part by the Concerted Research Action (GOA) Ambiorics 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

Avoine, G., & Tchamkerten, A. (2009). An Efficient Distance Bounding RFID Authentication Protocol: Balancing False-Acceptance Rate and Memory Requirement. In *LNCS: Vol. 5735. Proceedings of the 12th International Conference on Information Security (ISC '09)* (page 250-261). Springer Verlag.

Bellare, M., Canetti, R., & Krawczyk, H. (1996). Keying Hash Functions for Message Authentication. In *LNCS: Vol. 1109. Advances in Cryptology - CRYPTO '96* (page 1-15). Springer Verlag.

Brands, S., & Chaum, D. (1994). Distance-Bounding Protocols. In *LNCS: Vol. 765. Advances in Cryptology - CRYPTO '93* (page 344-359). Springer Verlag.

Clulow, J., Hancke, G.P., Kuhn, M.G., & Moore, T. (2006). So Near and Yet So Far: Distance Bounding Attacks in Wireless Networks. In *LNCS: Vol. 4357. Proceedings of the 3rd European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS '06)* (page 83-97). Springer Verlag.

Desmedt, Y. (1988). Major Security Problems with the "Unforgeable" (Feige)-Fiat-Shamir Proofs of Identity and how to overcome them. In *Proceedings of SecuriCom '88* (page 15-17).

Desmedt, Y., & Frankel, Y. (1991). Shared Generation of Authenticators and Signatures. In *LNCS: Vol. 576. Advances in Cryptology - CRYPTO '91* (page 457-469). Springer Verlag.

Desmedt, Y., & Jajodia, S. (1997). *Redistributing Secret Shares to New Access Structures and its Applications* (Tech. Rep. ISSE-TR-97-01). George Mason University.

Drimer, S., & Murdoch, S. (2007). Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In *Proceedings of the 16th USENIX Security Symposium* (page 87-102). USENIX.

Frankel, Y., & Gemmell, P., & MacKenzie, P. D., & Yung, M. (1997). Optimal Resilience Proactive Public-Key Cryptosystems. In *LNCS: Vol. 1294. Proceedings of IEEE Symposium on Foundations of Computer Science* (page 384-393). Springer Verlag.

Gans, G.K., Hoepman, J.H., & Garcia, F.D. (2008). A Practical Attack on the MIFARE Classic. In *LNCS: Vol. 5189. Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications (CARDIS '08)* (page 267-282). Springer Verlag.

Hancke, G., & Kuhn, M. (2005). An RFID Distance Bounding Protocol. In *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM '05)* (page 67-73). IEEE Computer Society.

ISO/IEC 9797 (1994). Information Technology - Security Techniques - Data Integrity Mechanisms Using a Cryptographic Check Function Employing a Block Cipher Algorithm.

ISO/IEC 9798-6 (2005). Information Technology - Security Techniques - Entity Authentication - Part 6: Mechanisms Using Manual Data Transfer.

Kim, C., Avoine, G., Koeune, F., Standaert, F., & Pereira, O. (2009). The Swiss-Knife RFID Distance Bounding Protocol. *Information Security and Cryptology - ICISC 2008: Revised Selected Papers*, 98-115.

Peeters, R., Kohlweiss, M. & Preneel, B. (2009a). Threshold Things That Think: Authorisation for Resharing. In *IFIP Advances in Information and Communication Technology: Vol. 309. Proceedings of iNetSec 2009 - Open Research Problems in Network Security* (page 111-124). Springer Verlag.

Peeters, R., Sulmon, N., Kohlweiss, M. & Preneel, B. (2009b). Threshold Things That Think: Usable Authorisation for Resharing," In *Symposium on Usable Privacy and Security - SOUPS 2009* (2 pages). ACM.

PKCS #1 v2.1 (2002). RSA Cryptography Standard.

Rabin, T. (1998). A Simplified Approach to Threshold and Proactive RSA. In *LNCS: Vol. 1462. Advances in Cryptology - CRYPTO '98* (page 349-369). Springer Verlag.

Rasmussen, K., & Capkun, S. (2009). Location Privacy of Distance Bounding Protocols. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security (CCS '08)* (page 149-160). ACM.

Rasmussen, K., & Capkun, S. (2010). Realization of RF Distance Bounding. To appear in *Proceedings of the 2010 Usenix Security Symposium*.

Rivest, R., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 210 (2), 120-126.

Schnorr, C.P. (1989). Efficient Identification and Signatures for Smart Cards. In *LNCS: Vol. 435. Advances in Cryptology - CRYPTO '89* (page 239-252). Springer Verlag.

Shamir, A. (1979). How to Share a Secret. *Communications of the ACM*, 220(11), 612-613.

Shoup, V. (2000). Practical Threshold Signatures. In *LNCS: Vol. 1807. Advances in Cryptology - EUROCRYPT 2000* (page 207-220). Springer Verlag.

Singelée, D., & Preneel, B. (2007). Distance Bounding in Noisy Environments. In *LNCS: Vol. 4572. Proceedings of the 4th European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS '07)* (page 101-115). Springer Verlag.

Wong, T.M., Wang, C., & Wing, J.M. (2002). *Verifiable Secret Redistribution for Threshold Sharing Schemes* (Tech. Rep. CMU-CS-02-114). Carnegie Mellon University.