Toward More Secure and Reliable Access Control

Conventional access control mechanisms, relying on a single security token to authenticate remote users, introduce a single point of failure and are vulnerable to relay attacks. A threshold-based distance-bounding protocol that distributes a user's private key among various personal devices improves system security and reliability.

Roel Peeters, Dave Singelée, and Bart Preneel
Katholieke Universiteit Leuven

Conventional authentication mechanisms are typically based on something you know (such as a password), something you have (such as a smartcard), or something you are (your biometrical features). Mobile networks can also use location information to enhance mutual entity authentication protocols. For example, to open your building's door, you might need the correct credentials (that is, authorization to enter the building), but you also must be close to the door.

Distance-bounding protocols can cryptographically enforce the concept of "proximity." They combine physical and cryptographic properties to let the user authenticate remotely and let the verifying party determine an upper bound on the distance between itself and the user (*prover*).[1] Employing distance-bounding protocols avoids relay attacks, where an adversary close to the verifier impersonates an authorized user. Such attacks are an important threat in access control systems.[2,3]

However, simply enhancing a mutual authentication protocol with location information isn't sufficient. Physical access control mechanisms usually rely on a single security token, such as a contactless smartcard. To enter a building, the user puts his or her smartcard close to a reader installed near the door, and both devices carry out a mutual authentication protocol. This introduces a single point of failure in the system, because an adversary could steal the security token and impersonate the user. Patching such a breach requires revoking the token or smartcard. Furthermore, a legitimate user without the security token can be denied access.

Integrating threshold cryptography into networked devices can help confront these challenges.[4] People already carry many personal devices with network capabilities, such as mobile phones, RFID tags attached to clothing, MP3 players, and car keys. Here, we present a lightweight threshold-based distance-bounding protocol for user devices. In particular, we envision personal devices that have a private key but can't update the key or store additional data (a car key fob, for example). Letting these resource-constrained devices participate in the access control scheme can significantly improve system security and reliability.

Cryptographic Techniques

Our protocol integrates the following cryptographic techniques.

Distance Bounding

Secure distance-bounding protocols measure the time of flight to determine an upper bound

Published by the IEEE CS ■ 1536-1268/12/$31.00 © 2012 IEEE

on the distance between the 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 verifier measures the time between sending a challenge and receiving the corresponding response. Multiplying the time of flight with the communication medium's propagation speed (the speed of light for RF communication) provides an estimate of the distance between the 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 wouldn't learn anything after having carried out the protocol.

Most distance-bounding protocols want to preclude distance-fraud and relay attacks. (Some protocols focus on terrorist fraud attacks, but we don't discuss them here.) In a distance-fraud attack, a dishonest prover claims to be closer than he or she really is (see Figure 1a).

In relay attacks (also denoted by mafia fraud attacks),[5] both the prover and verifier are honest, but there's a malicious intruder. This is a man-in-the-middle attack, where the intruder I is modeled as a malicious prover $\overline{P}$ and verifier $\overline{V}$ (see Figure 1b). The malicious verifier $\overline{V}$ interacts with the honest prover P, and the malicious prover $\overline{P}$ interacts with the honest verifier V. The physical distances between the intruder and prover and intruder and verifier are small, so neither P nor V notices the attack.

Stefan Brands and David Chaum first introduced distance-bounding protocols.[1] Because distance fraud and relay attacks are also a major concern for RFID systems, Gerhard Hancke and Marcus Kuhn later proposed a distance-bounding protocol that's more suitable in this setting.[6] Their protocol doesn't need to compute a signature after the m fast-bit exchanges, but this increases the false-acceptance rate to $(3/4)^m$ (up from $(1/2)^m$ in the Brands-Chaum protocol). Hancke and Kuhn also point out that distance-bounding protocols should be designed to cope well with substantial bit-error rates during the rapid single-bit exchanges, because these are conducted over noisy wireless ad hoc channels.

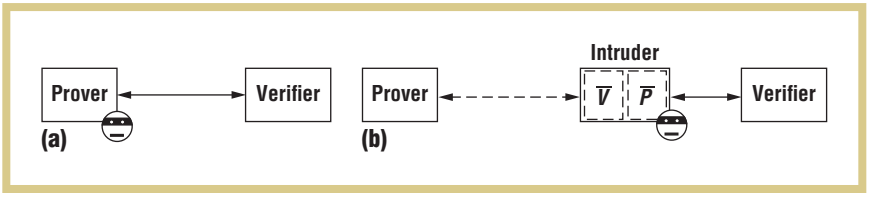


Figure 1. Most distance-bounding protocols want to preclude (a) distance-fraud and (b) relay attacks. These attacks are a major concern for systems where user proximity is important.

The Hancke-Kuhn protocol works as follows. First, the 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 verifier. Typically a pseudorandom function such as hash-based message authentication code (HMAC) is employed.[7] The output of this function is two m-bit sequences $b^{(0)}$ and $b^{(1)}$. 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 ith bit of $b^{(0)}$. If the challenge equals 1, then the prover sends the ith bit of $b^{(1)}$. During this phase, x bit errors are allowed. If at least $(m - x)$ responses are correct, the protocol succeeds.

Recently, researchers have proposed several other distance-bounding protocols.[8,9] These protocols often try to add new functionality or decrease the false-acceptance ratio without requiring a signature at the protocol's final stage. 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 (a processing delay in the order of nanoseconds or below). However, researchers have recently demonstrated that RF distance-bounding protocols can be realized in practice. Hancke designed a near-field, bit-exchange channel with minimal latency.[10] Other approaches use analogue hardware components[11] or Ultra Wide Band pulses[12,13] to minimize the processing delay and increase the accuracy on the distance measurements.

Pairings

In the threshold cryptography scheme discussed later, one of the mobile devices must compute pairings. These are essentially bilinear maps and are usually defined over elliptic curve groups.

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of order ℓ, and let $\hat{e}$ be a nondegenerate bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. A pairing is nondegenerate if for each element P in $\mathbb{G}_1$, there's a Q in $\mathbb{G}_2$ such that $\hat{e}(P, Q) \neq 1$, and vice versa for each element Q in $\mathbb{G}_2$. A pairing is bilinear if $\hat{e}(P + P', Q) = \hat{e}(P, Q) \times \hat{e}(P', Q)$; thus, $\hat{e}(aP, Q) = \hat{e}(P, Q)^a$, with $a \in Z_\ell$, and vice versa for elements in $\mathbb{G}_2$.

Threshold Cryptography

The notion of threshold cryptography builds upon the concept of Shamir's Secret Sharing.[14] The basic idea is to create a random polynomial $x = f(z)$ that intersects the x-axis in the secret value that is shared among the participants. They each get one share, which is an evaluation of the polynomial for a particular value of z (see Figure 2). The shared secret can be reconstructed by interpolation through a sufficient number of these shares. Less shares yield no information about the shared

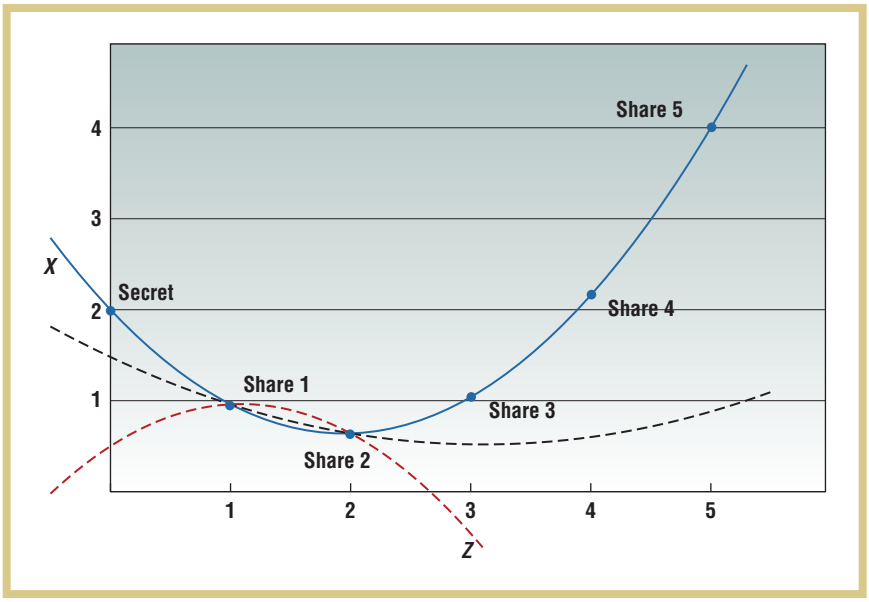


Figure 2. The concept of Shamir's Secret Sharing. A random polynomial $x = f(z)$ intersects the x-axis in the secret value, which is shared among the participants. They each get one share, which is an evaluation of the polynomial for a particular value of z. The shared secret can be reconstructed by interpolation through a sufficient number of these shares. Less shares yield no information about the shared secret (as denoted by the dashed lines).

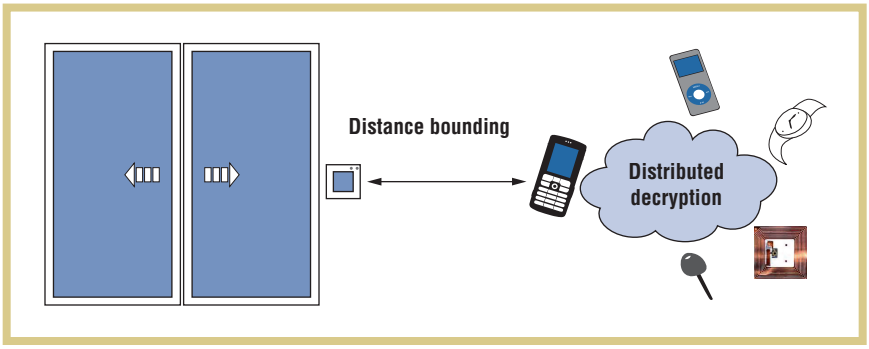


Figure 3. A distributed access control setting. Multiple personal devices cooperate in a proximity-based authentication protocol to gain entrance.

secret (as denoted by the dashed lines in Figure 2).

For a threshold cryptosystem, you define a public-private key pair for which the private key is the secret that will be shared. Everyone can encrypt messages using the public key. To decrypt a ciphertext, several participants of the system must cooperate. It's clear that, 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 provide partial decryptions while only using their share of the private key. By interpolating these partial decryptions, you can compute a valid decryption.

We define a (t, n) threshold cryptosystem, where $t + 1$ is an adjustable threshold number and n is the total number of parties. Any combination of $t + 1$ participants can decrypt the 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 must be initialized before it can be used. A public key, and shares of the corresponding private key, must 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 efficient, but one party (the trusted dealer) knows the private key that's shared among the parties. In a DKG protocol, a group of parties cooperates to jointly generate a public key and obtains 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 discuss here, a trusted dealer is acceptable.

Distributed Access-Control Setting

To get a better sense of how this setup actually works, let's consider an example scenario.

Use-Case Scenario

Figure 3 illustrates a building that enforces access control. The user carries a group of personal devices (*end devices*) that will jointly perform a proximity-based authentication protocol with the verifier (a reader near the front door).

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. In addition to the necessary communication interfaces, the gateway device must have on-board or separate dedicated hardware to perform the distance-bounding protocol's fast-bit exchanges and should have sufficient processing power. We envision the user's mobile phone, equipped with RF communication technology, acting as the gateway device. The user can have more than one gateway device, but

during each run of the protocol, only one such device acts as the gateway device; all others act as end devices.

In our envisioned setting, we also want to use resource-constrained devices that don't have (secure) writable memory (such as a car key). These low-cost devices can't (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. The secure memory offers protection against attackers who have physical access to the device and want to read or tamper with the memory's content. A low-cost technique to realize this feature is to use the Physically Unclonable Function (PUF),[15,16] where small differences in the fabrication process are used to derive a unique private key for each device.

In previous work, we showed how to construct a threshold group for this particular setting by employing bilinear pairings.[17] 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 the protected format. As a result, these protected shares can be stored externally—that is, in the gateway device. During a protocol run, each end device only needs to use its own unique private key, so these devices can be part of multiple groups without any additional storage cost. For example, you could employ the same set of devices to gain access to your office building and house.

Adversarial Model and Assumptions

We assume that all user devices communicate with each other over a *dedicated broadcast channel* (that is, if a device broadcasts a message, all other active devices within communication range receive that message and recognize the originating device).

We assume the presence of a computationally bounded adversary who can corrupt up to t devices. The adversary has access to all of the information that the corrupted devices store and can manipulate their behavior during protocol execution. The adversary's goal is to impersonate a genuine prover close to the verifier. The adversary can largely extend his or her communication range, and send and receive messages from far away. However, because RF communication is used, the adversary can't increase the communication medium's propagation speed.

In the main attack setting we investigate, the genuine prover is far from the verifier, and the adversary, which doesn't know the prover's private key, is physically close to the verifier. Attacks in which the adversary isn't close to the verifier aren't practical. To gain unauthorized access, the adversary must guess the prover's key or carry out a relay attack in which all the verifier's messages are forwarded to a proxy device that's hidden in the prover's neighborhood. An adversary can also compromise a set of the user's end devices and use them to perform partial decryptions. However, the set of compromised devices is assumed to be strictly smaller than $t + 1$. Our solution is resistant to such attacks.

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 must collaborate to successfully complete the protocol. The prover's private key is distributed among all of the user's devices. The verifier (for example, the office building's system administrator) could act as a trusted dealer. Alternatively, the user could 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 pseudorandom function, as in the Hancke-Kuhn protocol, we replace it with an asymmetric decryption function, which can be partially evaluated by each device. Next, we integrate 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. Figure 4 depicts the resulting 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.

Threshold Initialization

Each user device has a unique private key s_i. We define a nondegenerate 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. Please note that $\mathbb{G}_1$ and $\mathbb{G}_2$ are two distinct groups. 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

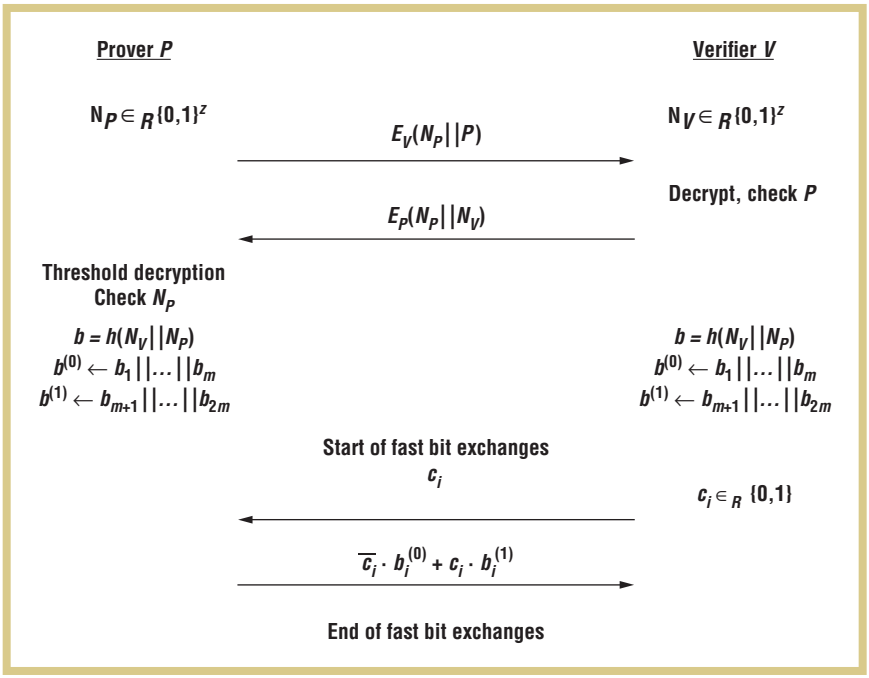


Figure 4. Threshold-based distance-bounding protocol. After an encrypted exchange of nonce values, multiple fast bit challenge-response protocols take place, resulting in an upper bound on the distance between prover and verifier.

access-control system, are masked with the public key S_i and stored externally in a protected format as $C_i = x_i S_i$. The corresponding group's public key is defined as $y_P = \hat{e}(P, xQ)$.

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

$$f(z) = x + c_1 z + \ldots + c_t z^t,$$

where $x_i = f(i)$. If you don't want one party to know the group's private key, you can use the DKG protocol. More details are available elsewhere.[17]

Encrypted Exchange of Nonces

To carry out the distance-bounding protocol, first the user must confirm that he or she wants to start the protocol by performing a particular action (such as pressing a button) on the gateway device. Next, the prover (the gateway device) and verifier generate a random z-bit nonce (N_P and N_V, respectively). N_P is concatenated with the identity of prover P. The result is encrypted with the verifier's public key and then sent to the verifier. After receiving this message, the verifier decrypts it using its private key and checks the prover's identity.

Next, the verifier constructs an encrypted message and sends it to the prover using a pairing-based variant of the ElGamal cryptosystem.[19] According to this cryptosystem, to encrypt a message $M \in \mathbb{G}_T$ under the public key $y = \hat{e}(P, xQ)$, choose a random $k \in_R \mathbb{Z}_\ell^*$ and output the ciphertext $(R, e) + (kP, My^k) \in \mathbb{G}_1 \times \mathbb{G}_T$. 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 specifically, the verifier encrypts the message as follows. It concatenates the nonces N_P and N_V and encrypts them using the prover's public key y_P. The verifier sends this ciphertext 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.

To decrypt the given ciphertext (R, e), the end devices provide partial decryptions

$$D_i = s_i R = s_i kP \in \mathbb{G}_1.$$

The gateway device receives the contributions D_i and verifies that $\hat{e}(D_i, S_i) = \hat{e}(R, Q)$. It then combines $t + 1$ valid contributions and computes the plaintext M as follows:

$$M = \frac{e}{d} \quad \text{with} \quad d = \prod \hat{e}(D_i, C_i)^{\lambda_i},$$

where λ_i are the appropriate Lagrange coefficients for interpolation.

Each end device combines its private key s_i with R. The gateway device computes the pairing of this partial decryption with the corresponding share in a protected format. The device stores these protected shares and can use them without revealing them in an unprotected format.

Rapid Bit Exchanges

After the nonces' encrypted exchange, both the gateway device and 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 should be carried out as soon as possible after the exchange of the encrypted nonces. If the delay between both phases is too large (the length of time 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.

Related Work in Threshold Distance-Bounding Protocols

We proposed the idea of combining distance-bounding protocols and threshold cryptography in earlier work,[1] but there are some important differences between that work and the approach we present here. The earlier solution let legitimate users control their security settings by letting them dynamically change the composition of the group of devices and threshold number according to their 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, we focus on efficiently increasing security and reliability, which we achieve by allowing resource-constrained devices to participate in the protocol as end devices.

Furthermore, in the previous work, each device had to compute a partial RSA signature, which is an expensive operation.[2] The most compact commercially available RSA coprocessor is the Tiny32 of Helion (www.heliontech.com/modexp.htm). This architecture requires 8 kilogates of circuit area and 10 kilobits of ROM. However, to get an acceptable processing time (200 ms for an RSA signature), the clock frequency (and thus the required energy) must be orders of magnitude higher than possible in a passive RFID tag. Another problem for RSA, in comparison with elliptic curve cryptography (ECC), is that to maintain the same level of security over time, the key size must grow exponentially instead of linearly.[3] This means that the difference in required storage and computational effort, compared to ECC, will become larger over time.

REFERENCES

1. R. Peeters, D. Singelée, and B. Preneel, "Threshold-Based Location-Aware Access Control," *Int'l J. Handheld Computing Research*, vol. 2, no. 3, 2011, pp. 22–37.
2. X. Zhao et al., "A 6.35Mbps 1024-Bit RSA Crypto Coprocessor in a 0.18um CMOS Technology," *Proc. 2006 IFIP Int'l Conf. Very Large Scale Integration*, IEEE, 2006, pp. 216–221.
3. "ECRYPT II: Yearly Report on Algorithms and Keysizes (2009–2010)," tech. report, European Network of Excellence in Cryptology II, 2010; www.ecrypt.eu.org/documents/D.SPA.13.pdf.

If this challenge equals 0, then the gateway device responds with the ith bit of $b^{(0)}$. If the challenge equals 1, then the gateway device sends the ith bit of $b^{(1)}$. In each round, the verifier measures the time between sending c_i and receiving the corresponding response. The maximum roundtrip time is selected and this measurement determines an upper bound on the estimation of the distance between the 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.

Discussion

Now that we've reviewed the basics of how our protocol works, let's discuss how the protocol improves security and reliability while minimizing implementation costs. In previous work, the idea of combining proximity-based authentication and threshold cryptography was introduced. (For more information, see the "Related Work in Threshold Distance-Bounding Protocols" sidebar.)

Security

Our threshold-based distance-bounding protocol comprises 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. During the fast-bit exchanges, the prover demonstrates knowledge of both the nonces N_p and N_V. Because the latter is encrypted with the prover's public key, no one aside from the prover could possibly recover this random nonce. Using N_p links both encrypted messages in the first phase of the protocol to each other. This is important, because at that time, the verifier has no assurances about the source of the encrypted message.

Our protocol is designed to prevent relay attacks. Recall that we assume that the genuine prover, the legitimate owner of a set of personal devices, isn't located close to the verifier. So, this is an adversary who hasn't compromised the gateway device, has no knowledge of the shares in the protected format, and can't decrypt the encrypted nonces exchanged in the first phase of the distance-bounding protocol. In this scenario, the adversary's 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 you incorporate the effect of bit errors caused by 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 or her personal devices within communication

range (not necessarily present at the same location).

In contrast to conventional distance-bounding solutions, the adversary can't 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.

Reliability

Let's compare our distributed access-control setting with the solution in which only one 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 that the device doesn't get compromised, you could achieve more or less the same security properties as our threshold-based solution.

However, when this one device gets lost or stolen, the legitimate user loses his or her access privileges. The same problem arises when the user forgets the device at home, or when it breaks down (owing to depleted batteries, for example). All of 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.

Implementation Cost

The end devices are potentially resource constrained, so it's important to minimize device costs. This was an important design requirement. 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.

Contrary to common belief, Elliptic Curve Cryptography (ECC) can be realized on resource-constrained devices such as passive RFID tags.

An architecture with this functionality can be made compact while being energy efficient and having a reasonable processing time.[20] The architecture can be built with less than 15 kilogates, and the computation of one scalar-EC point multiplication takes 85 ms and requires 1.18 microjoules of energy. This is currently the most compact RFID implementation of a public-key cryptographic primitive. There's also no need for extra hardware to securely store a share, because we use the devices' factory-installed private key.

There's 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 on security measures depends on the type of RFID tag used and the application where the protocol is going to be deployed. We can, however, compare ECC to other approaches. Various symmetric-key-based authentication protocols for RFID exist, for which the cryptographic building blocks can be realized with fewer gates than ECC.

For example, the most compact AES implementation requires 3,400 gates, and the most compact SHA-3 Round-Two candidate implementation requires about 5 Kgates and additional memory. However, these building blocks don't have homomorphic properties. As a result, these symmetric-key-based protocols can't be transformed into a threshold-based variant, which is necessary to avoid having a single point of failure in the system.

We designed our protocol to be unbalanced and shifted the complexity toward the gateway device. It needs to perform more computationally intensive operations, such as a pairing over an elliptic curve, and also must 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.

Although researchers have recently proposed hardware platforms for distance-bounding protocols, there is still a need for more general, lightweight platforms that support various types of cryptographic distance bounding solutions, including the modified Hancke-Kuhn protocol proposed here. Furthermore, research is needed to develop a practical, low-cost, dedicated broadcast communication channel, accessible to all devices.

The research results presented here could also be used in other application areas. In particular, threshold-based distance-bounding protocols can offer reliable user authentication in settings where the location or distance context is relevant to evaluate authentication claims.

ACKNOWLEDGMENTS

This work is funded by the Katholieke Universiteit Leuven and supported in part by the

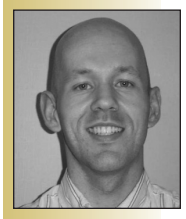
Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and by the Flemish IBBT projects. Roel Peeters is funded by a research grant of the Institute for the Promotion of Innovation through Science and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

REFERENCES

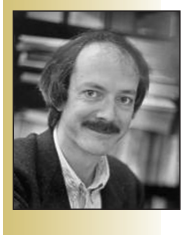
1. S. Brands and D. Chaum, "Distance-Bounding Protocols," *Advances in Cryptology—Eurocrypt 93*, LNCS 765, Springer, 1994, pp. 344–359.
2. L. Francis et al., "Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones," *Proc. 6th Int'l Workshop RFID Security and Privacy* (RFIDSec 10), LNCS 6370, Springer, 2010, pp. 35–49.
3. G. Hancke, K. Mayes, and K. Markantonakis, "Confidence in Smart Token Proximity: Relay Attacks Revisited," *Elsevier Computers and Security*, vol. 28, no. 7, 2009, pp. 615–627.
4. Y. Desmedt et al., "Threshold Things That Think (T4): Security Requirements to Cope with Theft of Handheld/Handless Internet Devices," *Proc. Symp. Requirements Engineering for Information Security*, 2001; web.science.mq.edu.au/~hwang/t4.ps.
5. Y. Desmedt, "Major Security Problems with the 'Unforgeable' (Feige)-Fiat-Shamir Proofs of Identity and How to Overcome Them," *Proc. 6th Worldwide Congress on Computer and Communications Security and Protection* (SecuriCom 88), SEDEP Paris France 1988, pp. 15–17.
6. G. Hancke and M. Kuhn, "An RFID Distance Bounding Protocol," *Proc. 1st Int'l Conf. Security and Privacy for Emerging Areas in Communications Networks* (SecureComm 05), IEEE CS, 2005, pp. 67–73.
7. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," *Advances in Cryptology—Crypto 96*, LNCS 1109, Springer, 1996, pp. 1–15.
8. G. Avoine and A. Tchamkerten, "An Efficient Distance Bounding RFID Authentication Protocol: Balancing False-Acceptance Rate and Memory Requirement," *Proc. 12th Int'l Conf. Information Security* (ISC 09), LNCS 5735, Springer, 2009, pp. 250–261.
9. D. Singelée and B. Preneel, "Distance Bounding in Noisy Environments," *Proc. 4th European Workshop on Security and Privacy in Ad Hoc and Sensor Networks* (ESAS 07), LNCS 4572, Springer, 2007, pp. 101–115.
10. G. Hancke, "Design of a Secure Distance-Bounding Channel for RFID," *J. Network and Computer Applications*, vol. 34, no. 3, 2011, pp. 877–887.
11. K. Rasmussen and S. Capkun, "Realization of RF Distance Bounding," *Proc. 19th Usenix Security Symp.*, Usenix, 2010, pp. 389–402.
12. M. Kuhn, H. Luecken, and N. Tippenhauer, "UWB Impulse Radio Based Distance Bounding," *Proc. 2010 Workshop on Positioning, Navigation and Communication* (WPNC 10), IEEE CS, 2010, pp. 28–37.
13. N. Tippenhauer and S. Capkun, "ID-Based Secure Distance Bounding and Localization," *Proc. 14th European Symp. Research in Computer Security* (ESORICS 09), LNCS 5789, Springer, 2009, pp. 621–636.
14. A. Shamir, "How to Share a Secret," *Comm. ACM*, vol. 22, no. 11, 1979, pp. 612–613.
15. J. Guajardo et al., "FPGA Intrinsic PUFs and Their Use for IP Protection," *Cryptographic Hardware and Embedded Systems Workshop*, LNCS 4727, Springer, 2007, pp. 63–80.
16. B. Gassend et al., "Silicon Physical Random Functions," *ACM Conf. Computer and Comm. Security*, ACM, 2002, pp. 148–160.
17. K. Simoens, R. Peeters, and B. Preneel, "Increased Resilience in Threshold Cryptography: Sharing a Secret with Devices That Cannot Store Shares," *Pairing-Based Cryptography—Pairing 2010*, LNCS 6487, 2010, pp. 116–135.
18. R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, vol. 21, no. 12, 1978, pp. 393–399.
19. T.E. Gamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *Advances in Cryptology—Crypto 84*, LNCS 196, G.R. Blakley and D. Chaum, eds., Springer, 1985, pp. 10–18.
20. Y.K. Lee et al., "Low-Cost Untraceable Authentication Protocols for RFID," *Proc. 3rd ACM Conf. Wireless Network Security* (WiSec 10), C. Nita-Rotaru and F. Stajano, eds., ACM, 2010, pp. 54–64.
21. M. Yoshitomi et al., "Efficient Implementation of the Pairing on Mobilephones Using BREW," *Information Security Applications*, LNCS 4867, S. Kim, M. Yung, and H.-W. Lee, eds., Springer, 2007, pp. 203–214.

the AUTHORS

Roel Peeters is a PhD student in the Computer Security and Industrial Cryptography (COSIC) research group at the Katholieke Universiteit Leuven, Belgium. His research interests include threshold cryptography and mobile security. Peeters has an MS in electrical engineering from Katholieke Universiteit Leuven. Contact him at roel.peeters@esat.kuleuven.be.



Dave Singelée is a postdoctoral researcher in the COSIC research group at the Katholieke Universiteit Leuven. His main research interests are cryptography, security and privacy of wireless communication networks, cryptographic authentication protocols for RFID, and secure localization schemes. Singelée has a PhD in applied sciences from the Katholieke Universiteit Leuven. He has authored and co-authored more than 20 scientific publications, and participated in various research projects. Contact him at dave.singelee@esat.kuleuven.be.



Bart Preneel is a full professor and head of the COSIC research group at the Katholieke Universiteit Leuven. His main research interests are cryptology and information security. Preneel has a PhD in applied sciences from the Katholieke Universiteit Leuven. He is the president of the International Association for Cryptologic Research (IACR) and a member of the editorial boards of the *Journal of Cryptology, IEEE Transactions on Information Forensics and Security,* and the *International Journal of Information and Computer Security.* He's also a member of the Accreditation Board of the Computer and Communications Security Reviews. Contact him at bart.preneel@esat.kuleuven.be.