

Contactless Access Control based on Distance Bounding

Handan Kılınc and Serge Vaudenay

EPFL, Lausanne, Switzerland

Abstract. Contactless access control systems are critical for security but often vulnerable to relay attacks. In this paper, we define an integrated security and privacy model for access control using distance bounding (DB) which is the most robust solution to prevent relay attacks. We show how a secure DB protocol can be converted to a secure contactless access control protocol. Regarding privacy (i.e., keeping anonymity in strong sense to an active adversary), we show that the conversion does not always preserve privacy but it is possible to study it on a case by case basis. Finally, we provide two example protocols and prove their security and privacy according to our new models.

Keywords: access control, distance bounding, RFID, NFC, relay attack, mafia fraud, distance hijacking, privacy

1 Introduction

Access control (AC) is a mechanism assuring that a system or a place can be accessed only by authorized users. AC is in the center of our daily lives. We use it to unlock smartphones, unlock and start cars, enter buildings or databases. Authentication in the AC systems based on two factors: The first one is a password, PIN code or biometric information such as fingerprints and retinal scans. The second one is a (contactless) card where authentication is done without contact via this card. With the development of the technology, the usage of contactless AC is becoming common because it is more convenient than carrying various keys, using PIN codes or using biometric information. However, the full security model for contactless AC has not been studied adequately. In this paper, we focus on contactless AC. So, whenever we use AC, we refer to contactless one.

A report from Smart Card Alliance [1] lists the main components of an access control system (tags, readers, controllers, database) and their security requirements which are however informal. Wongsen et al. [33] proposed an access control protocol between doors and mobile units (e.g. smartphone), but the protocol lacks any security proof. Some access control systems such as OPACITY [2] and PLAIN [14] mutually authenticate and establish a shared key between the terminal and card. The security analysis of PLAIN in [14] is far from being formal. OPACITY [2] was partly analyzed by Dagdelen et al. [7] where their security model is based on the key agreement security model of Bellare and Rogaway [4]. Hence, most of the previous works do not have a comprehensive security analysis. Moreover, **none of them consider relay attacks in their**

security analysis. Figure 1 and Figure 2 show real world relay attack scenarios. Unfortunately, these type of attacks are easily implementable [16, 17, 12, 28, 13, 24], so they violate access control.

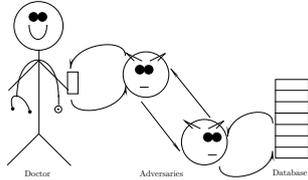


Fig. 1. The adversaries retrieve information from a hospital database by relaying the messages between the database reader and the doctor's card. Here, the doctor is far-away from the database. Arrows show that receiving or sending messages.

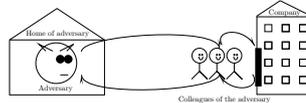


Fig. 2. The adversary who is an employee of the company accesses to the door of the company which shows that he arrived his job although he is at home. Here, the adversary can use one of his colleagues who is just next to the door. Arrows show that receiving or sending messages.

The other problem in contactless AC is to address privacy. Informally, if an AC protocol is private then it is hard for an outside observer to identify or recognize a party who wants to access a system. Some previous works [14, 8, 7] touched on privacy. PLAID [14] claims to be private (with an informal definition) but Degabriele et al. [8] show that it is weaker than what it claims. Dagdelen et al. [7] give two privacy related definitions: identity hiding and untraceability. The problem in their privacy model is that it only considers the interaction between the card and the reader. In reality, this may not be enough because the other interactions or outputs of the other components (i.e., controller, database) of an AC system can violate the privacy.

As a result, a formal security model which covers relay attacks has not been designed for AC. In addition to this, a formal privacy model which considers whole AC system is missing. In the literature, a powerful solution for relay attacks is distance bounding (DB) [6]. It relies on the limited celerity of communication signals. DB is typically an authentication protocol with the condition that a user who authenticates is close enough to a reader. Privacy has also been extensively studied in DB [29, 18, 15, 26, 34, 23, 3, 20].

By considering these critical issues, we design the first security and privacy model of an access control system which encompasses the propagation time of communication. Intuitively, in our definitions, we mix DB and access control based on a database of privileges. However, mixing both is not so straightforward when it comes to prove the security in a generic composition. Current AC protocols [2, 14] do not consider malicious users in their security models while DB considers malicious users (e.g., as in Figure 2). Therefore, the natural composition of them does not necessarily achieve the security level we need for AC protocols¹. In addition, we can show that an AC protocol which is constructed based on a private DB protocol does not achieve privacy in AC. All these reasons obviously show the need for complete security and privacy models in AC.

¹ A malicious user can behave maliciously in an AC protocol and retrieve some information which may help him to attack the DB protocol which is composed with this AC protocol.

Our Contribution:

- We first define **an integrated security model for AC** including identification, access control, and distance bounding by using the same components as defined in [1].
- We define **a new privacy model for AC** which includes the time of the communication. To the best of our knowledge, the time of the communication has not been considered for defining a privacy model before. Our new model covers all the previously defined privacy related definitions for access control such as identity hiding and untraceability.
- We give **a framework that clarifies how to use a secure DB** to construct a secure AC in our new security model. Basically, we show how to transform a man-in-the-middle (MiM), distance fraud (DF) and distance hijacking (DH) secure DB protocol into a secure AC scheme with proximity check. We also formally prove the security of this transformation.
- We show that the same framework can be used to achieve privacy in AC with restrictions on the database of AC system: The framework achieves privacy if the database is trivial meaning it is empty, or it includes all possible relations. We give a counterexample protocol that clearly shows why the framework does not work for non-trivial databases. This shows that **privacy in distance bounding is not always preserved when transformed into an access control system** which unfolds the need for a new model for AC.
- We construct a specific AC scheme by using a secure and private DB protocol Eff-pkDB [21] and prove its security and privacy with database.

2 Definitions from Previous work

In this section, we give some definitions and results about public-key DB which we integrate into our new security and privacy model for AC. This section is helpful to understand the DB related notions that we use in the next section.

Definition 1 (Public key DB Protocol [31]). *A public key distance bounding protocol is a two-party probabilistic polynomial-time (PPT) protocol and it consists of a tuple $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$. Here, $(\mathcal{K}_P, \mathcal{K}_V)$ are the key generation algorithms of P and V , respectively. The output of \mathcal{K}_P is a secret/public key pair $(\text{sk}_P, \text{pk}_P)$ and similarly the output of \mathcal{K}_V is a secret/public key pair $(\text{sk}_V, \text{pk}_V)$. P is the proving algorithm, V is the verifying algorithm where the inputs of P and V are from \mathcal{K}_P and \mathcal{K}_V . B is the distance bound. $P(\text{sk}_P, \text{pk}_P, \text{pk}_V)$ and $V(\text{sk}_V, \text{pk}_V)$ interact with each other. At the end of the protocol, $V(\text{sk}_V, \text{pk}_V)$ outputs a final message Out_V and have pk_P as a private output. If $\text{Out}_V = 1$, then V accepts. If $\text{Out}_V = 0$, then V rejects.*

A public-key DB protocol is correct if and only if under honest execution, whenever the distance between \mathcal{V} and P is less than B , then \mathcal{V} always outputs $\text{Out}_V = 1$ and pk_P .

In symmetric DB, we have one key generation algorithm \mathcal{K} and the input of P and V is a secret key generated by \mathcal{K} .

Now, we explain the security games which are designed for the threats of DB: mafia fraud and distance hijacking from [31]. These games address security in concurrent settings. So, they consist of multi-party settings which informally means that the parties run multiple times their algorithms during the games. An **instance** of a party is each new execution of its algorithm.

In mafia fraud, a man-in-the-middle (MiM) adversary between a verifier and a far-away honest prover tries to make the verifier accept. Formally, it is defined as follows:

Definition 2 (Mafia fraud (MiM security) [31]). *The game begins by running the key setup algorithms \mathcal{K}_V and \mathcal{K}_P which output $(\text{sk}_V, \text{pk}_V)$ and $(\text{sk}_P, \text{pk}_P)$, respectively. The adversary receives pk_V and pk_P . The game consists of several verifier instances including a distinguished one \mathcal{V} , honest prover's instances and adversary's instances. The adversary wins if \mathcal{V} outputs $\text{Out}_V = 1$ and pk_P when no close prover instance to \mathcal{V} exists. A DB protocol is MiM-secure if, for any such game, the probability of an adversary to win is negligible.*

In a nutshell, the adversary interacts or sees multiple new executions of P and V at any location to make only one of the verifier instances (\mathcal{V}) accept when no instance of P is close.

In distance hijacking (DH), a far-away malicious prover uses some honest and active provers who are close to the verifier to make the verifier grant privileges to the far-away prover. The distance hijacking security implies also the **distance fraud (DF)** security which provides security against a malicious and far-away prover who wants to authenticate himself (without using any other close party).

Definition 3 (Distance hijacking [31]). *The game consists of several verifier instances including a distinguished one \mathcal{V} , instances of honest prover P' and instances of malicious prover P . The game begins by running the key setup algorithms $\mathcal{K}_V, \mathcal{K}_P$ and malicious setup $\mathcal{K}_P^*(\text{pk}_V, \text{pk}_{P'})$ of P . P lets one of the instance of P' run the time critical phase of DB with \mathcal{V} . The malicious prover P wins if \mathcal{V} outputs $\text{Out}_V = 1$ and pk_P when P 's instance is far away from \mathcal{V} . A DB protocol is DH-secure if, for any such game, the probability of an adversary to win is negligible.*

The above definition is specific to a class of protocols which have a clearly identified time critical phase. Here, the time critical phase corresponds to a challenge/response exchange phase where the verifier calculates the round trip time of sending challenge and receiving response. Essentially, by letting an honest P' run this phase in the game, P tries to succeed to show himself close to the verifier. Again, we have many instances in this game.

The another security model in DB is for terrorist fraud (TF) [9]. Informally, TF adversary tries to authenticate himself while he is far-away from the verifier by getting help from his close accomplice. However, a trivial attack of TF-adversary could consist of giving his secret key to his accomplice who would execute DB with this key. So, usual definitions for TF [10, 11, 22, 5] exclude this particular attack explicitly. We do not integrate TF-security in our AC security

model because we think that this exclusion is arbitrary. In practice, we do not see why this attack would be excluded or how it would be prevented.

In the next definition, we give the privacy model by Hermans et al. [18] which has been used in many DB protocols [21, 31, 30, 19]. In this model, the adversary tries to distinguish provers. It can corrupt provers and learn their secret keys. The model is also called strong private. The details are given below:

Definition 4 (Privacy in DB [18]). *The privacy game is the following: Pick $b \in \{0, 1\}$ and let the adversary \mathcal{A} play with the following oracles:*

- **CreateP**(ID) $\rightarrow P_i$: *It creates a new prover identity of ID and returns its identifier P_i .*
- **Launch**() $\rightarrow \pi$: *It launches a new protocol with the verifier V_j and returns the session identifier π .*
- **Corrupt**(P_i) : *It returns the current state of P_i . Current state means the all the values in P_i 's current memory. It does not include volatile memory (i.e., the short term state in an interactive session).*
- **DrawP**(P_i, P_j) $\rightarrow vtag$: *It draws either P_i (if $b = 0$) or draws P_j (if $b = 1$) and returns the virtual tag reference $vtag$. If one of the provers was already an input of **DrawP** $\rightarrow vtag'$ query and $vtag'$ has not been released, then it outputs \emptyset .*
- **Free**($vtag$) : *It releases $vtag$ which means $vtag$ can no longer be accessed.*
- **SendP**($vtag, m$) $\rightarrow m'$: *It sends the message m to the drawn prover and returns the response m' of the prover. If $vtag$ was not drawn or was released, nothing happens.*
- **SendV**(π, m) $\rightarrow m'$: *It sends the message m to the verifier in the session π and returns the response m' of the verifier. If π was not launched, nothing happens.*
- **Result**(π) $\rightarrow b'$: *It returns a bit that shows if the session π is accepted by the verifier (i.e the message Out_V).*

In the end of the game, the adversary outputs a bit b'' . If $b'' = b$, then \mathcal{A} wins. Otherwise, it loses.

A DB protocol is strong private if for all PPT adversaries, the advantage of winning the privacy game is negligible.

3 Security and Privacy Model of AC

We first introduce the components of an access control system (ACS). In our definitions, for simplicity, we do not consider the user who may give PIN code or a biometric data to authenticate himself (this would be a parallel protocol). The components of an access control system are tag, reader, database and controller. Controller and database are in the secure area of ACS where it is not possible to tamper or access.

Tags (Access Cards): They hold personalized data which is used for identification and authentication. In ACS, each tag T generates a secret/public key pair

$(\text{sk}_T, \text{pk}_T)$. They also store the public key of the controllers that are responsible for the doors² that T can access.

Reader: A reader is an interface between a tag and a door. We can consider them as transmitters. They communicate with the tags. Each reader R has a location loc_R which is important as the tag can be granted if the tag proves that it is close enough to the reader.

Database: It contains information about tags and their rights. It stores a list of $(\text{pk}_T, loc_R, req)$ triplets meaning that the tag with pk_T is allowed to make the service request req on a reader at location loc_R . For instance, a service request can be the opening of a door. The database is in the secure area.

The database is not necessarily a list of triplets. It can also be a predicate deciding if a triplet belongs to it or not. A database is *trivial* if it is empty or if it contains all possible triplets.

For simplicity, we consider that the content of the database is static in what follows.

Controller: It controls access authentication. All controllers can be connected with multiple readers. Depending on the data they receive from its one of readers and the database, they give the final decision for the authorization.

More generally, the access control is relative to a service (such as opening a door) in a given location. The tag T of public key pk_T requests a service req to a reader at location loc_R and its corresponding controller checks if the privilege $(\text{pk}_T, loc_R, req)$ exists in the database. T stores req and it can change req later on. All controllers stay in the secure area.

Definition 5 (Access Control (AC)). *AC consists of a distance bound B , a database $DataB$, a controller C , a reader R , and a tag T , the key generation algorithms: Gen_C generating $(\text{sk}_C, \text{pk}_C)$ for a controller C and Gen_T generating $(\text{sk}_T, \text{pk}_T)$ for a tag T . C, R , and T run the algorithms $\mathcal{C}(\text{sk}_C, \text{pk}_C, DataB, B), \mathcal{R}(loc_R)$ and $\mathcal{T}(\text{sk}_T, \text{pk}_T, \text{pk}_C, req)$, respectively. In the end of the protocol, C outputs either $\text{Out}_C = 1$ and private output $\text{POut}_C = (\text{pk}_T, loc_R, req)$ if the authentication succeeds or $\text{Out}_C = 0$ if it fails. R also publicly outputs $\text{Out}_R = \text{Out}_C$.*

Definition 6 (Correctness of AC). *An AC is correct, if for all loc_R, req and for all sets of keys generated by Gen_C and Gen_T , if*

- T requests service req to R at location loc_R ,
- T is within a distance at most B from loc_R and
- $(\text{pk}_T, loc_R, req)$ is in $DataB$,

then

$$\Pr[\text{Out}_C = 1 \wedge \text{POut}_C = (\text{pk}_T, loc_R, req)] = 1$$

² Door is a representation of the system or service that a user desires to access.

3.1 Security

In this section, we give the formal security model for an access control system.

Adversarial and Communication Model: Each party (readers, controllers, tags, adversaries) has polynomially many instances. An instance of a party corresponds to a protocol execution with this party at a given location and time. Each instance of our model is as follows:

- All parties in AC are limited by the speed limit (speed of light) for communication, which simply says that a message sent at time t by a party X cannot arrive to a party Y at time t' which is less than $t + d(X, Y)$ (d is a metric which shows the time of flight distance between X and Y).
- Readers are all honest. They are connected to their corresponding controllers with a secure and an authenticated channel.
- Controllers are all honest. They are the only components of the ACS which can access the database.
- **Tags are all honest.** However, they can receive special signals [32]. There can be only one activatable instance of each tag at a time. The special signal $\text{Activate}(T, req)$ activates the only activatable instance of T with a specified input req ³. After receiving this signal, further activation signals are ignored by this instance. An instance can be terminated by one of the following signals: $\text{Terminate}(T)$ and $\text{Move}(T, loc')$. $\text{Terminate}(T)$ terminates the instance execution, but it remains “active”. The special signal $\text{Move}(T, loc')$ orders to terminate and move the tag to loc' . It means that the instance becomes inactive and that only one unused instance of T at location loc' can be activated. The terminated instance sends a special signal Go which, when received by this unused instance at location loc' , will make it activatable (Go signals cannot be sent by malicious participants; they are here only to enforce that a tag cannot move faster than a signal propagation). After, it may receive another $\text{Activate}(T, req')$ as a new instance of the same tag at location loc' . **This models the tags being at a single location and moving (as influenced by the adversary) to run other instances.** Besides, it models that instances of the same tag cannot be run concurrently.
- **Adversaries create the database.** So, they can generate fake relations $(\tilde{pk}_T, ..)$ where \tilde{pk}_T and its corresponding secret key \tilde{sk}_T are generated by an adversary. Instances which could hold some \tilde{sk}_T are called **fake tags**. In addition, adversaries can change the destination of messages (except for special signals) between a tag and a reader. We assume that they have very special hardware which can intercept a message and change its destination without any delay. Similarly, they can update a message and send it to the same destination with this hardware without any delay. So, if a party X sends a message at time t_1 , and the adversary reads or updates the message at time t_2 and sends it to a party Y at time t_3 , then the arrival of the message to Y is still bounded by $t_1 + d(X, Y)$ because $t_3 - t_2 \geq 0$.

³ This can also correspond to a user who is the owner of T to input whatever requests he wants into his tag.

Except for the communication between readers and controllers, the adversary instances see all communication.

Definition 7 (AC-Security). *The game begins by setting up the components of the ACS. The security game is as follows given the security parameter n :*

- Run $\text{Gen}_C(1^n) \rightarrow (\text{sk}_C, \text{pk}_C)$ for the controller and run $\text{Gen}_T \rightarrow (\text{sk}_{T_i}, \text{pk}_{T_i})$ for each tag T_i and give the public key pk_C and pk_{T_i} 's to the adversary.
- The adversary creates instances of T_i at chosen locations. Each instance can start after activation and run $\mathcal{T}(\text{sk}_{T_i}, \text{pk}_{T_i}, \text{pk}_C, \text{req})$ only once.
- The adversary creates instances of readers at chosen locations loc_{R_k} . They run $\mathcal{R}(\text{loc}_{R_k})$ once activated by an incoming message. They communicate with an instance of C over a secure channel⁴. There is a distinguished instance of a reader R . We denote by loc_R its location.
- The adversary sets DataB .
- The adversary creates instances of himself (fake tags). These instances run independently and communicate.

All messages follow our communication model. The game ends when the distinguished instance R (and its corresponding instance C) outputs some value Out_R . An AC protocol is secure, if for any such game, the adversary wins with a negligible probability. A wins the game if $\text{Out}_R = 1$ and $\text{POut}_C = (\text{pk}_T, \text{loc}_R, \text{req})$ for some pk_T and req satisfying at least one of the following conditions:

1. $(\text{pk}_T, \text{loc}_R, \text{req}) \notin \text{DataB}$,
2. $\text{pk}_T \in \{\text{pk}_{T_i}\}_{i=1}^t$ and no active instance of the honest tag holding pk_T is close to loc_R during the execution of the AC protocol with C and R ,
3. $\text{pk}_T \notin \{\text{pk}_{T_i}\}_{i=1}^t$ and no fake tag is close to loc_R during the execution with C and R .

where t is the number of public keys generated by Gen_T in setup.

Remarks:

- In the third condition, we need that no fake tag is close to loc_R to prevent the trivial attacks where a far away fake tag can give its secret key to a close by fake tag. Without this condition, the adversary would always win. This would however exclude all TF-attacks as well.
- If $\text{pk}_T \notin \{\text{pk}_{T_i}\}_{i=1}^t$, the security definition includes DH (and also DF).
- If $\text{pk}_T \in \{\text{pk}_{T_i}\}_{i=1}^t$, then the security definition corresponds to MF. It includes impersonation attacks, relay attacks and other forms of man-in-the-middle attacks as well since MF covers all of them.

⁴ For simplicity, we assume that the instance C of the controller is at the same location as R_k but the time of communication between R_k and C should have no influence on the result. The difference between C and R_k only makes sense for practical reasons.

In practice, the controllers are connected to multiple readers. So, it is not possible for them to check if a tag is close. Therefore, readers are the components that can give this decision.

Before proceeding the next part, we show that the natural composition of access control and distance bounding does not always achieve the security in Definition 7. Assume that we have a MiM, DF and DH secure symmetric DB protocol $DB = (\mathcal{K}, P, V, B)$. As an AC protocol, we have an AC protocol OPACITY [2]⁵. In the natural composition, first the parties run OPACITY with a minor change and then DB (the reader runs V , the tag runs P with the secret key K). The change in OPACITY is as follows: the reader sends K at the end of the OPACITY protocol. Clearly, the modified version of OPACITY is still secure AC in the security model of Dagdelen et al. [7] since K is completely independent parameter. Unfortunately, this composition is not secure in Definition 7 since an adversary can win AC-game with satisfying the second condition. However, when we look the modified OPACITY and DB separately in their own security models, they are secure. Therefore, the generic composition of AC and DB is not straightforward.

3.2 Privacy

Privacy is also important in access control protocols. The definition of privacy we provide uses the same adversarial and communication model that we use for security. It also covers the identity hiding and untraceability with the corruption of tags. Informally, identity hiding means given an execution of protocol the adversary should not output the public key of the tag and untraceability means the adversary should not decide if two executions belong to the same tag or not.

Definition 8 (AC-Privacy). *The privacy game has the same setting as the game in Definition 7. We first decide to play the right r or the left ℓ game. Differently than the security model, each active tag instance can be paired with an another tag instance by an adversary. The pairing happens with the signal $\text{Draw}(T_i, T_j, k)$ which pairs T_i and T_j by giving an index k , if the conditions below are satisfied:*

- T_i and T_j are at the same location,
- T_i and T_j have the same access privileges,
- neither T_i nor T_j is already paired and
- k is greater than the index of previous Draw signal to both T_i and T_j .

A tag instance can be paired to itself as well. The adversary lets $vtag = (T_i, T_j, k)$ be a virtual tag. All messages (and special signals) can only have a virtual tag as a destinator. If we are in game ℓ , then $vtag$ simulates T_i and if we are in game r , $vtag$ simulates T_j . The signal $\text{Free}(T_i, T_j, k)$ breaks the pair if

⁵ OPACITY is basically a key agreement protocol where the authentication of a tag is done with this key.

it exists. The adversary can corrupt a tag T_i (and actually all tags) by receiving sk_{T_i} during the setup.

In the end, the adversary decides if $vtag$ simulates game r or game ℓ . If the decision of the adversary is correct, then the adversary wins.

If an AC protocol is private, the advantage of a polynomial time adversary in this game is bounded by a negligible probability.

The most important distinction of our definition is that we consider “communication time which leaks the proximity of a party” in our privacy definition contrarily previous work related to privacy [29, 18]. To the best of our knowledge, it has not been taken into account before for a privacy model. It is reasonable to consider the location of a user as a privacy leakage for the protocols where the communication time influences the output such as DB.

Since Mitrokotsa et al. [25] showed that location privacy is nearly impossible to achieve, we cannot prevent this leakage. So, our privacy game has the condition of being at the same location which is necessary to avoid the adversary to trivially distinguish the left or right game by checking the communication time.

Besides, the condition of having the same access privileges is necessary to prevent the adversary to determine the left or right game by seeing the accepting or the rejecting message by a controller.

4 Distance Bounding in Access Control

In this section, instead of designing a new AC protocol, we give a conceivable framework that converts a DB protocol into an AC protocol. We prove in Theorem 1 that, after conversion, the AC protocol achieves AC-security (in Definition 7) assuming that the DB protocol is MiM and DH secure. However, we show that we cannot always achieve AC-Privacy with this framework, even though the DB protocol is (strong) private according to Definition 4. Therefore, we prove in Theorem 2 that the AC protocol which is converted from a private DB achieves privacy, if $DataB$ is trivial. The details are in the following subsections.

4.1 Secure AC with Secure DB

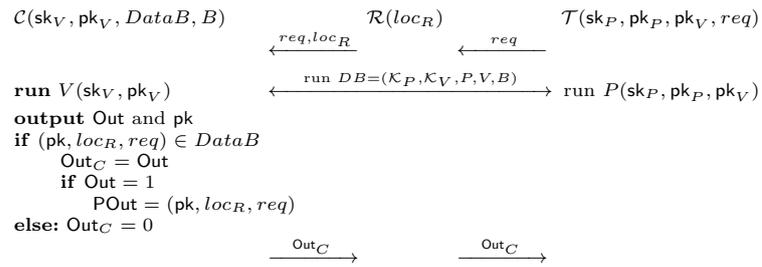


Fig. 3. The framework to convert a DB protocol to an AC protocol

If we have a public-key DB protocol $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$, we can construct an AC protocol with $(Gen_C, Gen_T, \mathcal{C}, \mathcal{T}, DataB, B)$ with the framework below:

- We match the key generation algorithms: $\text{Gen}_C = \mathcal{K}_V$, $\text{Gen}_T = \mathcal{K}_P$. So, $(\text{sk}_C, \text{pk}_C) = (\text{sk}_V, \text{pk}_V)$ and $(\text{sk}_T, \text{pk}_T) = (\text{sk}_P, \text{pk}_P)$.
 - We create DataB according to the access privileges of tags using the keys.
 - $\mathcal{T}(\text{sk}_P, \text{pk}_P, \text{pk}_V, \text{req})$ uses $P(\text{sk}_P, \text{pk}_P, \text{pk}_V)$ as a subroutine. \mathcal{T} outputs req and then run $P(\text{sk}_P, \text{pk}_P, \text{pk}_V)$.
 - Whenever $\mathcal{R}(\text{loc}_R)$ is activated with req , it sends req and loc_R to \mathcal{C} .
 - $\mathcal{C}(\text{sk}_V, \text{pk}_V, \text{DataB}, B)$ runs $V(\text{sk}_V, \text{pk}_V)$ as a subroutine jointly with $\mathcal{R}(\text{loc}_R)$. When V reaches the part where challenge/response is necessary to determine the distance to loc_R , \mathcal{R} steps in to check if the responses arrive on time and are correct.
- Here, \mathcal{C} may give all necessary input(s) to \mathcal{R} so that \mathcal{R} can check the responses. Alternatively, \mathcal{C} may only give the challenges, and \mathcal{R} only determines if the responses arrive on time. Then, if they arrive on time, \mathcal{R} can send the responses to \mathcal{C} so that \mathcal{C} can check if the responses are correct. The only restriction is that **\mathcal{R} has to decide if the responses arrive on time.**
- When $V(\text{sk}_V, \text{pk}_V)$ outputs Out and the private output pk_P : If $(\text{pk}_P, \text{loc}_R, \text{req}) \in \text{DataB}$ and $\text{Out} = 1$, it publicly outputs $\text{Out}_C = 1$ and privately outputs $\text{POut}_C = (\text{pk}_P, \text{loc}_R, \text{req})$. Otherwise, it outputs $\text{Out}_C = 0$. In both cases, \mathcal{R} outputs $\text{Out}_R = \text{Out}_C$. The framework is in Figure 3.

An example protocol in Figure 4 is constructed using this framework. Before, we prove that the framework achieves AC security if DB is MiM and DH secure.

Theorem 1. *Assuming that a DB protocol with $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$ is MiM-secure and DH-secure, then an AC protocol with using this DB protocol with the framework as described in Figure 3 is secure according to Definition 7.*

Proof. Assume that there exists an adversary \mathcal{A} which wins the game in Definition 7 where the output of the game is $\text{Out}_R = 1$ and $\text{POut}_C = (\text{pk}_{T_i}, \text{loc}_R, \text{req})$, then we can construct an adversary which wins MiM-game or DH-game.

Apparently, \mathcal{A} can win the AC-game with either second or third condition because \mathcal{C} outputs $\text{Out}_C = 0$ if given $(\text{pk}_{T_i}, \text{loc}_R, \text{req}) \notin \text{DataB}$ (the first winning condition) which makes impossible to win with the first condition.

Winning with the second condition: If $\text{pk}_{T_i} \in \{\text{pk}_{T_k}\}_{k=1}^t$ and no instance of the tag with pk_{T_i} is close to loc_R during the execution of the AC protocol with \mathcal{C} and R , then we can construct an adversary \mathcal{B} which wins MiM-game (Definition 2) of DB protocol with $(\mathcal{K}_P, \mathcal{K}_V, P, V)$.

\mathcal{B} receives pk_V and pk_P from MiM-game. Then, it randomly picks $i \in \{1, \dots, t\}$ where t is the number of (honest) tags needing to be simulated. The public key pk_{T_i} which will be used to simulate the i^{th} tag T_i is pk_P . Here, T_i will have a role as a prover on MiM-game. For the rest of the tags, \mathcal{B} generates $t - 1$ secret/public key pairs $(\text{sk}_{T_j}, \text{pk}_{T_j})$ with using $\text{Gen}_T(1^n)$ which are the secret/public keys of T_j 's. After, it sends pk_V as the controller's public key and $\text{pk}_{T_1}, \dots, \text{pk}_{T_{i-1}}, \text{pk}_P, \text{pk}_{T_{i+1}}, \dots, \text{pk}_{T_t}$ as the tags' public-keys in AC-game to \mathcal{A} . Remark that pk_V and pk_P are indistinguishable since they are generated with the same key generation algorithms of controllers and tags, respectively.

At some moment, \mathcal{B} receives $DataB$ from \mathcal{A} . If $(pk_P, \dots) \notin DataB$, then \mathcal{B} loses MiM-game since in this case, there will be no chance that \mathcal{A} wins the AC-game with this tag. Otherwise, it locates instances of T_i (which corresponds to P 's instances in MiM-game) on the locations that \mathcal{A} decides. \mathcal{B} simulates the instances of AC-game as follows:

- Instances of T_j 's where $T_j \neq T_i$: For the signals $Move(T_j, loc)$ and $Terminate(T_j)$, \mathcal{B} just simulates. When it receives the signal $Activate(T_j, req)$, it simulates by running the algorithm $\mathcal{T}(sk_{T_j}, pk_{T_j}, pk_V, req)$. Remark since \mathcal{B} knows each sk_{T_j} , it can run \mathcal{T} .
- Instances of T_i : For the signals $Move(T_i, loc)$ and $Terminate(T_i)$, \mathcal{B} moves the corresponding instance of P in the MiM-game to loc and halts the corresponding instance of P in the MiM-game, respectively. Whenever it receives the signal $Activate(T_i, req)$, it first outputs req and then runs (activates) the corresponding instance of P in the MiM-game. Whatever the instance of P in MiM-game outputs, \mathcal{B} outputs the same.
- Instances of controller and reader: Whenever \mathcal{A} activates \mathcal{R} (via sending req) so that \mathcal{C} , \mathcal{B} runs an instance of V .

In the end, if \mathcal{A} picks a reader instance R which sees $pk_{T_j} = pk_P$ as a distinguished one, \mathcal{B} wins with the success probability below. Otherwise, \mathcal{B} loses MiM-game since V has to output $Out_V = 1$ and pk_P in MiM-game.

$$\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins} \wedge \text{Condition 2}] \times \frac{1}{t}$$

Winning with the third condition: If $pk_T \notin \{pk_{T_i}\}_{i=1}^t$ and no instance of the adversary is close to loc_R during the execution with R , then we can construct an adversary \mathcal{B}' which wins DH-game. The reduction is very similar to the previous one except we replace P with an honest prover P' .

$$\Pr[\mathcal{B}' \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins} \wedge \text{Condition 3}] \times \frac{1}{t}$$

In the end, we have

$$\Pr[\mathcal{B} \text{ wins}] + \Pr[\mathcal{B}' \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins}] \times \frac{1}{t}.$$

Since we know that the success probability of \mathcal{B} in MiM and \mathcal{B}' in DH game is negligible, then the success probability of \mathcal{A} is negligible as well. \square

Now, we give an example AC protocol (Eff-AC) in our framework by converting the public-key DB protocol Eff-pkDB [21] which is one of the most efficient public-key distance bounding protocols.

Eff-AC: We use Eff-pkDB with its variant. Its variant uses a key agreement protocol Nonce-DH [21] (based on random oracle and Gap Diffie Hellman (GDH) [27]) to agree on a secret S and a symmetric-key DB OTDB [31] to run with S . We stress that this is only an example of the generic construction of Eff-pkDB. In particular, we could replace NonceDH by another key agreement protocol which is D-AKA secure [21] and possibly eliminate the random oracle assumption.

The public parameters for the key generation algorithms $\text{Gen}_C (\mathcal{K}_V)$ and $\text{Gen}_T (\mathcal{K}_P)$ are a group G of prime order q and its generator g . Gen_C and Gen_T pick sk_C and sk_T from \mathbb{Z}_q , and set $\text{pk}_C = g^{\text{sk}_C}$ and $\text{pk}_T = g^{\text{sk}_T}$, respectively. Eff-AC works as follows:

The tag has the input $\text{sk}_T, \text{pk}_T, \text{pk}_C, \text{req}$, the controller \mathcal{C} has the input $\text{sk}_C, \text{pk}_C, B, \text{DataB}$ and the reader \mathcal{R} has the input loc_R . \mathcal{T} sends req to \mathcal{R} and \mathcal{R} sends it along with loc_R to \mathcal{C} . Then, \mathcal{C}, \mathcal{R} and \mathcal{T} run Eff-pkDB. Here, \mathcal{T} runs the proving algorithm of Eff-pkDB, and \mathcal{C} and \mathcal{R} run the verifying algorithm of Eff-pkDB, jointly. The details of these algorithms are as follows: First, \mathcal{T} picks a random value N from $\{0, 1\}^n$ and sends N and pk_T . After \mathcal{C} receives them, it computes $S = H(g, \text{pk}_T, \text{pk}_C, \text{pk}_T^{\text{sk}_C}, N)$. Meanwhile, \mathcal{T} also computes $S = H(g, \text{pk}_T, \text{pk}_C, \text{pk}_C^{\text{sk}_T}, N)$. After, \mathcal{C} gives S and B to \mathcal{R} so that \mathcal{R} runs the challenge phase. Until this part corresponds to the Nonce-DH protocol. Then, OTDB [31] is run by \mathcal{R} and \mathcal{T} as follows:

\mathcal{R} picks a value $N_R \in \{0, 1\}^{2n}$ and sends it to \mathcal{T} . Then, \mathcal{R} and \mathcal{T} compute $X = N_R \oplus S$ before n -round challenge phase begins. In each round i , \mathcal{R} picks a challenge Q_i and starts the timer. In response, \mathcal{T} sends W_i which is the $2i + Q_i^{\text{th}}$ bit of X . When \mathcal{R} receives it, it stops the timer. After the challenge phase, if all responses are correct and arrive on time (i.e. with in less than $2B$), then \mathcal{R} sets $\text{Out} = 1$. Then, \mathcal{R} sends Out to \mathcal{C} . This is the end of of Eff-pkDB.

\mathcal{C} sets $\text{Out}_C = \text{Out}$. If $\text{Out} = 1$, \mathcal{C} checks if \mathcal{C} has the access privilege by checking if $(\text{pk}_T, \text{loc}_R, \text{req}) \in \text{DataB}$. If it is in DataB , it privately outputs $\text{POut}_C = (\text{pk}_T, \text{loc}_R, \text{req})$. Otherwise, it sets $\text{Out}_C = 0$. Finally, \mathcal{C} sends Out_C to \mathcal{R} and \mathcal{R} outputs it as Out_R .

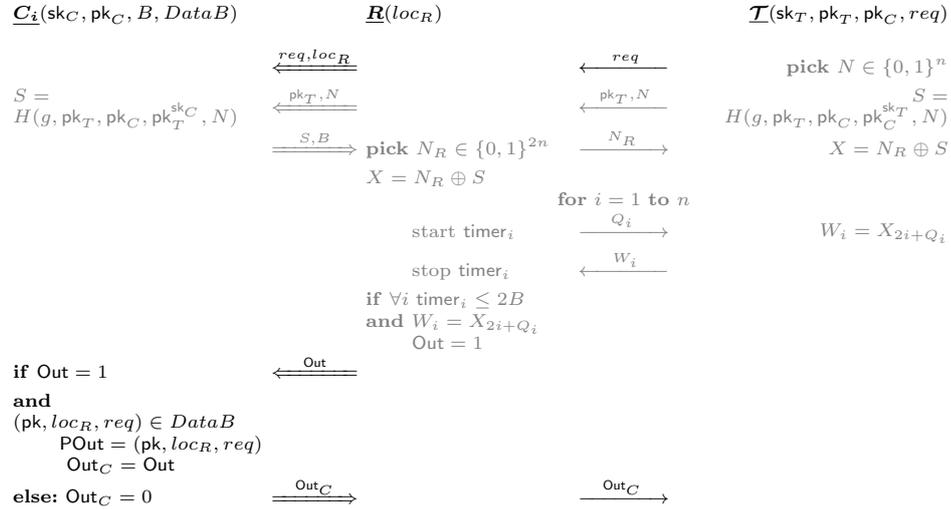


Fig. 4. Eff-AC. Double arrow shows that the communication is secure and authenticated while sending the message above it. The gray colored parts are Eff-pkDB.

Since Eff-pkDB is MiM and DH-secure [21], **Eff-AC which uses Eff-pkDB with the framework in Figure 3 is AC-secure thanks to Theorem 1.**

Remark: The security proof of Eff-pkDB [21] is also valid for a variant where the verifier generates an ephemeral $(\text{sk}_C, \text{pk}_C)$ pair and sends pk_C to the prover. So, tags do not even need to store pk_C in this variant of Eff-pkDB. Therefore, a variant of Eff-AC with an ephemeral key is secure thanks to Theorem 1. This variant is very desirable for practical reasons because we can allow many controllers and the tag does not need to store all the corresponding keys.

4.2 Private AC with Private DB

The difficulty in proving privacy in an AC protocol which uses a private DB protocol comes from the fact that *DataB* must discriminate tags. This fact may leak information about identities. In DB, the output of V does not depend on pk_P . Hence, the private output of the verifier (pk_P) plays no role in the privacy game of Definition 4. We show here a generic privacy preservation result with our framework, but only for a trivial *DataB*. Trivial *DataB* makes POut_C play no role in AC. We cannot prove the same result for an arbitrary database. Remember, a database is *trivial* if it is empty or if it contains all possible triplets.

Theorem 2. *Assuming that DB protocol with $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$ is private according to Definition 4, then an AC protocol with using this DB protocol with the framework as described in Figure 3 is private **when DataB is trivial** based on Definition 8.*

Proof. Assuming that there exists an adversary \mathcal{A} breaking the privacy in AC with a trivial *DataB*, then we can construct an adversary \mathcal{B} that breaks the privacy of DB.

\mathcal{B} simulates the communication model of AC for \mathcal{A} , except the subroutines P and V for honest participants. For each message and signal that \mathcal{B} receives for tags, it works as follows:

- *Receiving a signal Draw* (T_i, T_j, k) : It checks the necessary conditions to be paired. If they are satisfied, it calls the *Draw* oracle in the privacy game of DB with the inputs T_i, T_j . In respond, the *Draw* oracle sends *vtag*. \mathcal{B} stores the information that *vtag* corresponds to (T_i, T_j, k) .
- *Receiving a signal Free* (T_i, T_j, k) : It retrieves the corresponding *vtag* to (T_i, T_j, k) . If it exists, it calls the oracle *Free* with the input *vtag* in the privacy game of DB.
- *Receiving a signal Activate or Move*: It simulates them.
- *Receiving a message m*: It retrieves *vtag* and calls the oracle *SendP* in the privacy game of DB with the input (vtag, m) . Then, it receives a respond m' from the *SendP* oracle and sends m' to \mathcal{A} .

To simulate a reader receiving m , \mathcal{B} behaves as follows:

- If it is the first time and $m = \text{req}$, \mathcal{B} calls the *Launch* oracle to get a session identifier π . Then, it calls *SendV* with π and receives an empty message m' .
- Otherwise, it calls the oracle *SendV* with the input (π, m) and receives m' .

If m' is not the final message, it sends m' to \mathcal{A} . Otherwise, $m' = \text{Out}_V$. In this case, \mathcal{B} assigns $b = 0$ if DataB is empty and $b = 1$ if it is not empty (meaning that it has all possible relations). In the end, it sends $\text{Out}_C = \text{Out}_V \wedge b$ to \mathcal{A} . The simulation is perfect. So, \mathcal{A} and \mathcal{B} have the same advantage. \square

Why only for trivial DataB: We can show that Theorem 2 does not work for all DataB with the following counterexample.

Assume that we have a private DB $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$. From DB, we can construct another private protocol DB' $(\mathcal{K}_P, \mathcal{K}_V, P', V', B)$ where P' and V' work as defined below:

$\frac{P'(\text{sk}_P, \text{pk}_P, \text{pk}_V) :}{\text{receive } flag}$	$\frac{V'(\text{sk}_V, \text{pk}_V)}{\text{send } 0}$
<p>if $flag = 1$ and pk_P is odd</p>	<p>run $V(\text{sk}_V, \text{pk}_V)$</p>
<p style="padding-left: 20px;">$\mathcal{K}_P \rightarrow (\text{sk}'_P, \text{pk}'_P)$</p>	
<p style="padding-left: 20px;">$(\text{sk}_P, \text{pk}_P) \leftarrow (\text{sk}'_P, \text{pk}'_P)$</p>	
<p>run $P(\text{sk}_P, \text{pk}_P, \text{pk}_C)$</p>	

Clearly, DB' is still private because the only change is to remove the identity of the prover by replacing the secret and public keys with some random keys. (We recall that pk_P as a private output of V plays no role in Definition 4.)

Now, let's consider the conversion of DB' to an AC protocol with the framework. The adversary can break the privacy of the AC protocol as follows: He first picks two tags T_1 and T_2 which have public keys with different parities and moves them at the same location. It also creates a $\text{DataB} = \{(\text{pk}_{T_1}, \text{loc}_R, \text{req}), (\text{pk}_{T_2}, \text{loc}_R, \text{req})\}$. Then, it pairs (T_1, T_2) with the signal $\text{Draw}(T_1, T_2, 0)$ and activates the pair. It sends a message $flag = 1$ to $vtag = (T_1, T_2, 0)$ (by replacing the message $flag = 0$ which comes from a reader R). Then, it lets C, R and $vtag$ execute the protocol. In the end, R outputs Out_R . Depending on the parity, the adversary can find out the left or right game with probability 1 (e.g., if pk_{T_1} is odd and $\text{Out}_R = 1$, it means right game (T_2) is simulated).

In addition, even by weakening Definition 8 such that the adversary does not create a database and it is not allowed to pair tags (instead, the game does), we achieve **no** privacy. In this case, the advantage of the adversary with this attack would be $\frac{1}{2}$: If the paired parities' public keys have the same parity, then the attack does not give any more advantage than the privacy game of DB' gives. If they have different parity, the adversary wins with probability 1.

Even though we cannot use our framework to achieve privacy with **all** private DB protocols, we can still have private AC using our framework with **some** DB protocols where one of them is Eff-pkDB^p [21]. Now, we describe Eff-AC^p which is converted from Eff-pkDB^p.

Eff-AC^p (See Figure 5): It is very similar to Eff-AC. Differently here, the secret/public key pair of C consists of two parts: $(\text{sk}_C, \text{pk}_C) = ((\text{sk}_{C_1}, \text{sk}_{C_2}), (\text{pk}_{C_1}, \text{pk}_{C_2}))$ where $(\text{sk}_{C_1}, \text{pk}_{C_1})$ is used for the encryption and $(\text{sk}_{C_2}, \text{pk}_{C_2})$ is used for Nonce-DH (key agreement protocol). The only change on

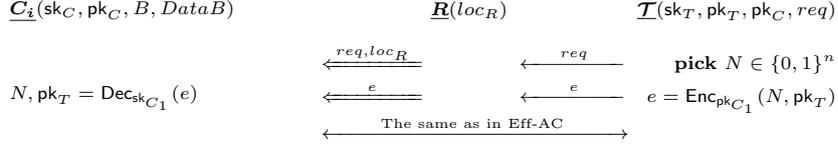


Fig. 5. Eff-AC^p

\mathcal{T} is that it sends the encryption of (pk_T, N) and on \mathcal{C} is that it retrieves pk_T, N by decrypting the encryption with sk_{C_1} . The rest is the same with Eff-AC.

Theorem 3. *Eff-AC^p is a private access protocol in the random oracle model according to Definition 8, assuming that the cryptosystem is IND-CCA secure and Gap Diffie-Hellman (GDH) problem [27] is hard.*

Note that the same result applies to the generic construction of Eff-pkDB^p [21], i.e., not only the one based on GDH and the random oracle. We could indeed replace Nonce-DH by another key agreement protocol which is D-AKA^p secure [21].

Proof (sketch): We adapt the proof from the privacy proof of Eff-pkDB^p [21].

We define games Γ_i^b below and the success probability of an adversary is p_i^b .

Γ_0^b : It is the same game that we defined in Definition 8 where $b = \ell$ meaning we are in the left-game or $b = r$ meaning we are in the right-game.

Γ_1^b : We reduce Γ_0^b to Γ_1^b where we simulate the controller instances without decrypting the ciphertext that is sent by a *vtag*. Because of the correctness of the cryptosystem, $p_1^b = p_0^b$.

Γ_2^b : We reduce Γ_1^b to Γ_2^b where *vtag* is simulated by encrypting a random value instead of (pk_T, N) . We can easily show $p_2^b - p_1^b$ is negligible by using the IND-CCA security of the cryptosystem.

We reduce Γ_2^b to Γ_2^r where we replace all secret/public keys $(\text{sk}_\ell, \text{pk}_\ell)$ which are the keys of the tag in the left-side in *vtag* by replacing secret/public keys $(\text{sk}_r, \text{pk}_r)$ of its paired tag. Using D-AKA^p security of Nonce-DH (Theorem 7 in [21]), we can show that $p_2^\ell - p_2^r$ is negligible.

Remark that if pk_ℓ and pk_r are kept in a plaintext and used by the controller, the replacing pk_ℓ with pk_r make the same Out_C result due to our assumption which says the paired tags have the same access privileges.

So, $p_0^\ell - p_0^r$ is negligible. □

5 Conclusion

In this paper, we designed a security model for AC which considers the whole interaction between components. The security model integrates the model of DB since the distance of the tag is important to detect the relay attacks. In our model, we preserve the security against adversaries which can be a tag or not. We also let the adversaries construct the database. We constructed a privacy model for AC which includes time of communication as well.

We gave a simple framework which securely transforms a DB to an AC. We proved a similar result for privacy assuming that *DataB* is trivial. We showed

why the theorem does not work for other types of database. Finally, we constructed two AC protocols Eff-AC and Eff-AC^p which are adapted from existing public-key distance bounding protocols Eff-pkDB and Eff-pkDB^p [21], respectively. We proved their security and privacy in our security and privacy models.

References

1. S. C. Alliance. Using smart cards for secure physical access. *Smart Card Alliance Report*, 54, 2003.
2. S. C. Alliance. Industry technical contributions: Opacity, 2013.
3. G. Avoine, E. Dysli, P. Oechslin, et al. Reducing time complexity in RFID systems. In *Selected Areas in Cryptography*, volume 3897, pages 291–306. Springer, 2005.
4. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Annual International Cryptology Conference*, LNCS 773, pages 232–249. Springer, 1993.
5. I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Inscrypt*, LNCS 8957, pages 170–190. Springer, 2014.
6. S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT*, LNCS 765, pages 344–359. Springer-Verlag, 1993.
7. Ö. Dagdelen, M. Fischlin, T. Gagliardoni, G. A. Marson, A. Mittelbach, and C. Onete. A cryptographic analysis of opacity. In *European Symposium on Research in Computer Security*, LNCS 8134, pages 345–362. Springer, 2013.
8. J. P. Degabriele, V. Fehr, M. Fischlin, T. Gagliardoni, F. Günther, G. A. Marson, A. Mittelbach, and K. G. Paterson. Unpicking PLAID. In *International Conference on Research in Security Standardisation*, LNCS 8893, pages 1–25. Springer, 2014.
9. Y. Desmedt. Major security problems with the “unforgeable” (Feige-) Fiat-Shamir proofs of identity and how to overcome them. In *Congress on Computer and Communication Security and Protection Securicom*, pages 147–159. SEDEP Paris France, 1988.
10. U. Dürholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance-bounding RFID protocols. In *Information Security*, LNCS 7001, pages 47–62. Springer, 2011.
11. M. Fischlin and C. Onete. Terrorism in distance bounding: modeling terrorist-fraud resistance. In *Applied Cryptography and Network Security*, LNCS 7954, pages 414–431. Springer, 2013.
12. A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.
13. L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, LNCS 6370, pages 35–49. Springer, 2010.
14. C. A. governments Department of Human Services (DHS). Protocol for lightweight authentication of identity (PLAID), 2010.
15. J. Ha, S. Moon, J. Zhou, and J. Ha. A new formal proof model for RFID location privacy. In *Computer Security-ESORICS 2008*, pages 267–281. Springer, 2008.
16. G. P. Hancke. A practical relay attack on iso 14443 proximity cards. *Technical report*, University of Cambridge Computer Laboratory, 59:382–385, 2005.
17. G. P. Hancke. Practical attacks on proximity identification systems. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6–pp. IEEE, 2006.
18. J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A new RFID privacy model. In *ESORICS*, LNCS 6879, pages 568–587. Springer, 2011.

19. J. Hermans, R. Peeters, and C. Onete. Efficient, secure, private distance bounding without key updates. In *WiSec*, Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pages 207–218, 2013.
20. A. Juels and S. A. Weis. Defining strong privacy for RFID. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):7, 2009.
21. H. Kılınc and S. Vaudenay. Efficient public-key distance bounding protocol. In *ASIACRYPT*, 2016.
22. C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The swiss-knife RFID distance bounding protocol. In *Information Security and Cryptology-ICISC 2008*, pages 98–115. Springer, 2008.
23. Y. Li, R. H. Deng, J. Lai, and C. Ma. On two RFID privacy notions and their relations. *ACM Transactions on Information and System Security (TISSEC)*, 14(4):30, 2011.
24. K. Markantonakis. Practical relay attack on contactless transactions by using NFC mobile phones. *Radio Frequency Identification System Security: RFIDsec*, 12:21, 2012.
25. A. Mitrokotsa, C. Onete, and S. Vaudenay. Location leakage in distance bounding: Why location privacy does not work. *Computers & Security*, 45:199–209, 2014.
26. C. Y. Ng, W. Susilo, Y. Mu, and R. Safavi-Naini. RFID privacy models revisited. In *European Symposium on Research in Computer Security*, LNCS 5283, pages 251–266. Springer, 2008.
27. T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, pages 104–118. Springer, 2001.
28. M. Roland, J. Langer, and J. Scharinger. Applying relay attacks to Google Wallet. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6. IEEE, 2013.
29. S. Vaudenay. On privacy models for RFID. In *ASIACRYPT*, LNCS 4833, pages 68–87. Springer, 2007.
30. S. Vaudenay. On privacy for RFID. In *Provable Security*, pages 3–20. Springer, 2015.
31. S. Vaudenay. Private and secure public-key distance bounding application to NFC payment. In *Financial Cryptography*, LNCS 8975, pages 207–216, 2015.
32. S. Vaudenay. Sound proof of proximity of knowledge. In *Provable Security*, LNCS 9451, pages 105–126. Springer, 2015.
33. E. R. Wognsen, H. S. Karlsen, M. Calverley, M. N. Follin, B. Thomsen, and H. Huttel. A secure relay protocol for door access control. In *Proceedings of the Xii Brazilian Symposium on Information and Computer System Security*. SBC-Sociedade Brasileira de Computação, 2012.
34. A. Yang, Y. Zhuang, D. S. Wong, and G. Yang. A new unpredictability-based RFID privacy model. In *Network and System Security*, pages 479–492. Springer, 2013.