Attribute-based Access Control Architectures with the eIDAS Protocols

Frank Morgner[1] Paul Bastian[1] Marc Fischlin[2]

[1]Bundesdruckerei GmbH
[2]Technische Universität Darmstadt, Germany

Abstract. The extended access control protocol has been used for the German identity card since November 2010, primarily to establish a cryptographic key between a card and a service provider and to authenticate the partners. The protocol is also referenced by the International Civil Aviation Organization for machine readable travel documents (Document 9303) as an option, and it is a candidate for the future European eIDAS identity system. Here we show that the system can be used to build a secure access system which operates in various settings (e.g., integrated, distributed, or authentication-service based architectures), and where access can be granted based on card's attributes. In particular we prove the protocols to provide strong cryptographic guarantees, including privacy of the attributes against outsiders.

1 Introduction

The extended access control (EAC) protocol has originally been proposed by the German Federal Office for Information Security (BSI) for identity cards and machine readable travel documents [1]. Indeed, it is listed as an option in Document 9303 of the International Civil Aviation Organization for protecting machine readable travel documents [20]. In the latest version of the BSI document [2] it has also been proposed as a part of the candidate for the European electronic identities, authentication, and trust services (eIDAS) system. Technically, the protocol establishes a cryptographic key between an eID card, connected to a local reader, and a remote service provider, via the so-called terminal authentication (TA) step and the chip authentication (CA) step. The protocol also mutually authenticates the parties. See Figure 1.

1.1 EAC for Attribute-based Access Control

Here we discuss how the EAC protocol can be adapted for more general (physical) access system architectures. Furthermore, using the established cryptographic key in the EAC protocol one can use its channel protocol, called secure messaging, to have the card send further attributes on which the access decision can be based, too. The advantage is that only mild changes to the existing infrastructure of the German identity card and candidate eIDAS system are necessary.

Ample architecture scenarios. The first extension refers to broader architectures in which the verifying party can be distributed across various entities. The common settings, also displayed in Figure 2, include:

- In the *integrated terminal architecture*, as in the border control scenario for travel documents, the reader implements the service provider functionality, and only forwards the attributes

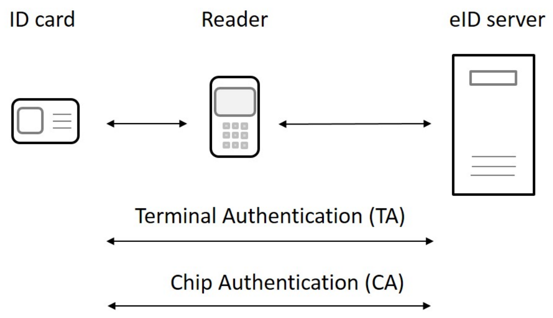


Figure 1: Extended Access Control (EAC), consisting of the terminal authentication (TA) step and the chip authentication (CA) step.

(sent over the secure messaging channel) for verification to the back-end management (via a secure connection like TLS). The reader then potentially grants access.

- In the *distributed terminal architecture*, as in the eID service scenario, the reader mainly connects the card to a controller which executes the EAC protocol with the card. The controller again calls the back-end management about the attributes.
- In the *eID-service architecture* an external service provider takes care of the cryptographic operations and forwards the attributes to the controller.
- In the *authentication-service architecture* the signature generation in the TA step of the EAC protocol is outsourced to a dedicated authentication server which holds the long-term signing key. The other steps are carried out again by the controller.

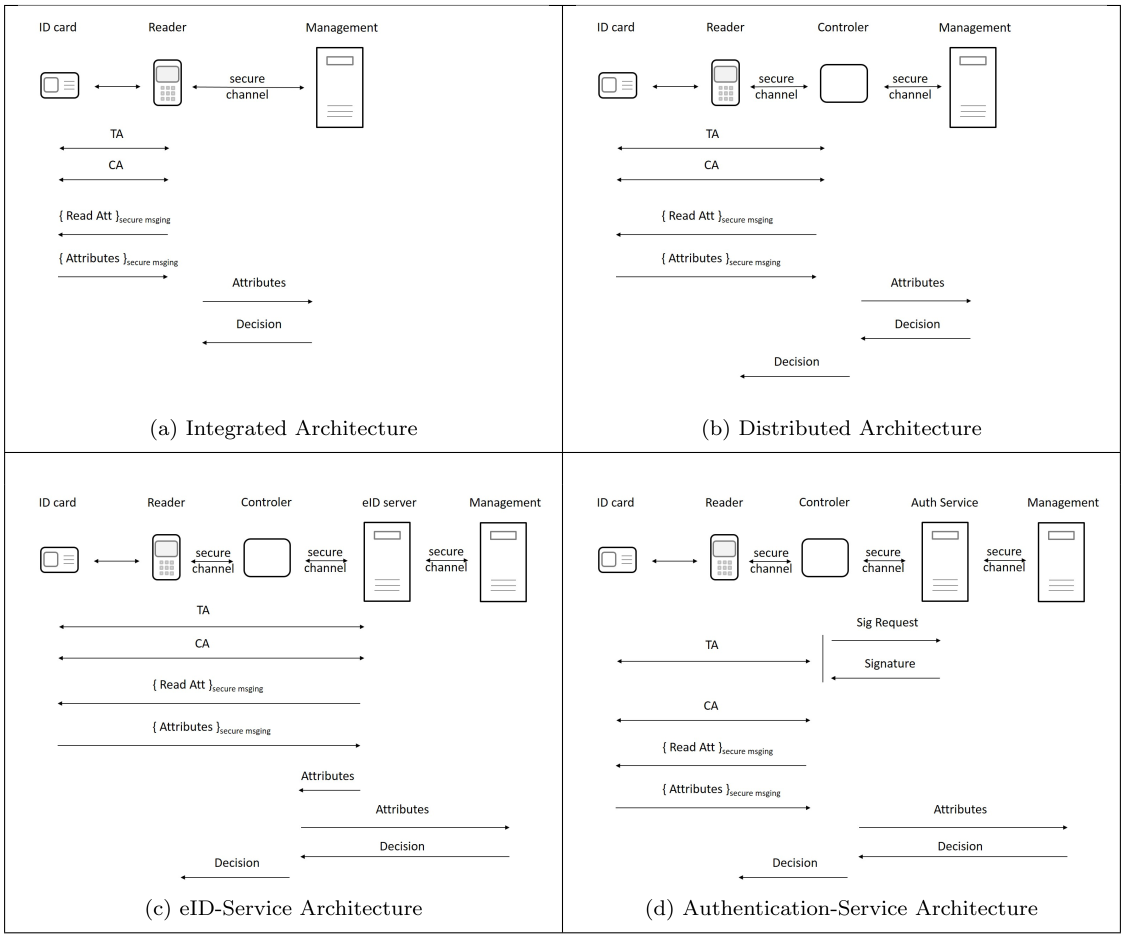
Restoring Sessions. Another extension concerns the possibility to authenticate faster through recognition. Here we can rely on the session contexts provided by the EAC protocol, version 2. Roughly, the EAC protocol offers the possibility to store the derived keys for secure messaging (and the send sequence counter) and to re-establish a connection with these keys. In EAC this allows to switch session contexts when changing communication partners.

By using the session context switching mechanism we can add a recognition step to our authentication procedure (for any setting). That is, the responding party checks if it has already successfully authenticated the card (and stored the session context under some identifier) and tries to re-establish the session with the card. If the card is responsive then both parties re-start the secure channel under the stored keys and the card transmits its attributes. By this the parties do not have to perform the more expensive public-key operations again.

1.2 Security of the Architectures

Our main result is to show that the proposed protocols provide strong security guarantees in a cryptographic sense. This boils down to two important security properties: *impersonation resistance*, preventing the adversary to trick the responding party into falsely accepting a card, and *attribute-privacy*, preventing the adversary from learning the attributes transmitted by genuine cards. The latter may be necessary in cases where the attributes carry confidential data, such as general access information or data facilitating the identification of the person using the card.

We envision very strong adversarial capabilities in attacks against either property, such that showing infeasiblity of attacks gives strong security guarantees. The adversary in our security



(a) Integrated Architecture

(b) Distributed Architecture

(c) eID-Service Architecture

(d) Authentication-Service Architecture

Figure 2: Architectures for attribute-based access control.

model corresponds to similar attackers on key exchange protocols (such as in the Bellare-Rogaway model [8]), giving the adversary full control over the network, and allowing it to modify or inject messages in communications, and to corrupt parties.

For impersonation resistance the adversary wins except in the trivial case that the identified card has been corrupted, or that the adversary has only relayed communication between the reader and the genuine card. Relying on previous results about EAC [17, 23] we show that all architectures achieve this strong notion. Analogously, we argue that all architectures satisfy our strong privacy notion which postulates indistinguishability of used attributes, except for the case of corrupt cards or corrupt responders. We also discuss security peculiarities of the different architectures.

1.3 Related Work

Some of the aforementioned architectures and the idea of using session contexts have already been discussed in the master thesis of one of the authors [6]. Our contribution here is to define appropriate models and argue security according to cryptographic standards.

The different versions of the access architectures should not be confused with the Enhanced

Role Authentication (ERA) protocol for the eIDAS token [2]. There, an attribute provider connects to the card by establishing another channel via the EAC protocol and can then access attribute requests stored by the terminal on the card. For this the card uses the switching operation for session contexts to communicate securely with the corresponding party.

The difference of ERA to our setting here is that we assume that access attributes are stored on the card and not provided by an external service provider. In particular, the card in our setting only communicates with a single responder and executes the EAC protocol only once. This is accomplished in our setting by letting the responding party read out the attributes and having it forward them to the management system. The protocol here also uses the session contexts to re-establish connections, instead of switching channels between the various communication partners. This also means that the session contexts are stored persistently here, whereas eIDAS tokens use them transiently only.

In a related proposal, Bundesdruckerei [5] introduces the possibility to secure transactions data, such as mobile phone numbers, on top of the EAC protocol and its existing eID architecture (requiring only minor modifications to the reader). The proposed transaction system has been analyzed cryptographically in [23]. This idea is orthogonal to our setting here where we discuss different access architectures including attributes. Yet, due to the resemblance with the EAC protocol we can partly use their results in order to show that the various access systems provide the common authentication guaranteed, even if the responding party cannot communicate with the management system for checking the attributes.

The EAC protocol (and its related protocols for the German identity card resp. the eIDAS tokens) has been analyzed in [12, 17, 22, 16, 10, 11, 19, 13, 18]. We merely rely on the EAC analysis in [17] and results related to secure composition of key exchange (like EAC) with secure channels (such as secure messaging).

2 The EAC Protocol and Adaptations

Since the access system strongly relies on the EAC protocol we first recall that protocol and then discuss the modifications.

2.1 The EAC Protocol

The Extended Access Control (EAC) protocol is a two-party key agreement protocol between a chip card and a terminal. It consists of a terminal authentication (TA) step and a chip authentication (CA) step. We omit explicit mentioning of the passive authentication step in between, in which the chip forwards passively authenticated data to the terminal, since the details of this step is irrelevant to our security concerns here.

At the outset both parties each hold a certified long-term key pair, on the card's side for generating a Diffie-Hellman key, and on the terminal's side for signing. Both parties also hold a card identifier id_C which in the execution of the German identity card equals the compressed version of the public-key of the preceding PACE protocol with which the card connects securely to the local reader (see Section 2.3). The PACE step is omitted in our setting. One may for now simply assume that id_C is empty.

In the TA step the terminal authenticates to the chip card. For this it chooses a session-specific ephemeral key pair (esk_T, epk_T), sends over its certificate for the long-term key pk_T (which is also included in its certificate $cert_T$) and a compressed version $\mathrm{Compr}(epk_T)$ of the ephemeral public key. The compression function can be for example the projection onto the x-coordinate of the elliptic curve point. The card returns a random nonce r_C which the terminal signs, together with $\mathrm{Compr}(epk_T)$ and id_C (if present). The terminal sends the signature to the card and the card accepts only if verification succeeds.

Upon successful completion of the TA phase, the card then executes the Chip Authentication (CA) step. For this the chip sends its certificate $cert_C$ and public key pk_C, and the terminal replies with its (uncompressed) ephemeral public key epk_T. The card checks that this value matches the previously sent compressed version. Both parties then compute the Diffie-Hellman key of pk_C and epk_T with the corresponding secret key they hold, and derive an encryption key K_{enc} and the MAC key K_{mac}. This step requires the card to pick another random nonce r'_C and include it in the key derivation process. The chip computes the MAC over epk_T and sends it together with the nonce r'_C to the terminal.

In the protocol description in Figure 3 we also include the so-called session identifiers sid for compatibility with previous analyses [17, 23]. These session identifiers can be roughly seen as unique session-specific labels. These cryptographic identifiers, determined by the protocol communication and used in the proof, should not be confused with the integer-valued session *context* identifiers used by the parties of the EAC protocol. Similarly, we include the partner identifiers pid which refer to the designated partner and are taken from the unique identifier in the certificate.

2.2 Restoring Session Context

A session context in the domain of the German identity card consists mainly of some (session context) identifier, the cryptographic keys for secure messaging, the send sequence counter (and possibly additional entries like the auxiliary data which can be used in the EAC protocol) [2]. A new session context is usually stored after successful reception of the first secure messaging transmission. The card is usually restricted to store at most 127 session contexts (or even less). To restore a session the terminal is supposed to send the context identifier to the card, encapsulated into a corresponding protocol message.

When storing the sequence counter it must be ensured that this value does not interfere with the actual counter value used for secure messaging. The suggested method is to round the current value up to the next multiple of 16 and store this value. Only if the current value reaches this bound then one again needs to update the stored value to the next multiple of 16. For us here the details are irrelevant as long it is guaranteed that sequence counters are used only once in the context of a session.

We note that the card is supposed to immediately delete a session context if an erroneous secure messaging transmission arrives. Similarly, if a terminal tries to re-initialize a session context but receives an error (say, if some other terminal has overwritten the context under the identifier meanwhile) then the terminal should start from scratch running the EAC protocol.

2.3 Modifications

In this part we describe the modifications of the EAC protocoland the eID scenario for our setting.

Omitting the PACE step. First, we do not assume that the password-authenticated connection establishment (PACE) protocol between the card and the reader is executed before initializing the EAC protocol. The PACE protocol requires the card holder to type in the PIN at the reader and then establishes a secure channel between card and reader for the wireless data exchange.

As pointed out in [17] the EAC protocol itself already provides a secure key exchange protocol between card and terminal whose security does not rely on the strength of the PACE protocol. We therefore start with the assumption that card and reader have not executed the PACE protocol. This, however, also means that the card identifier value id_C has not been set yet, because it usually corresponds to data derived during the run of PACE. We simply assume that id_C is empty.

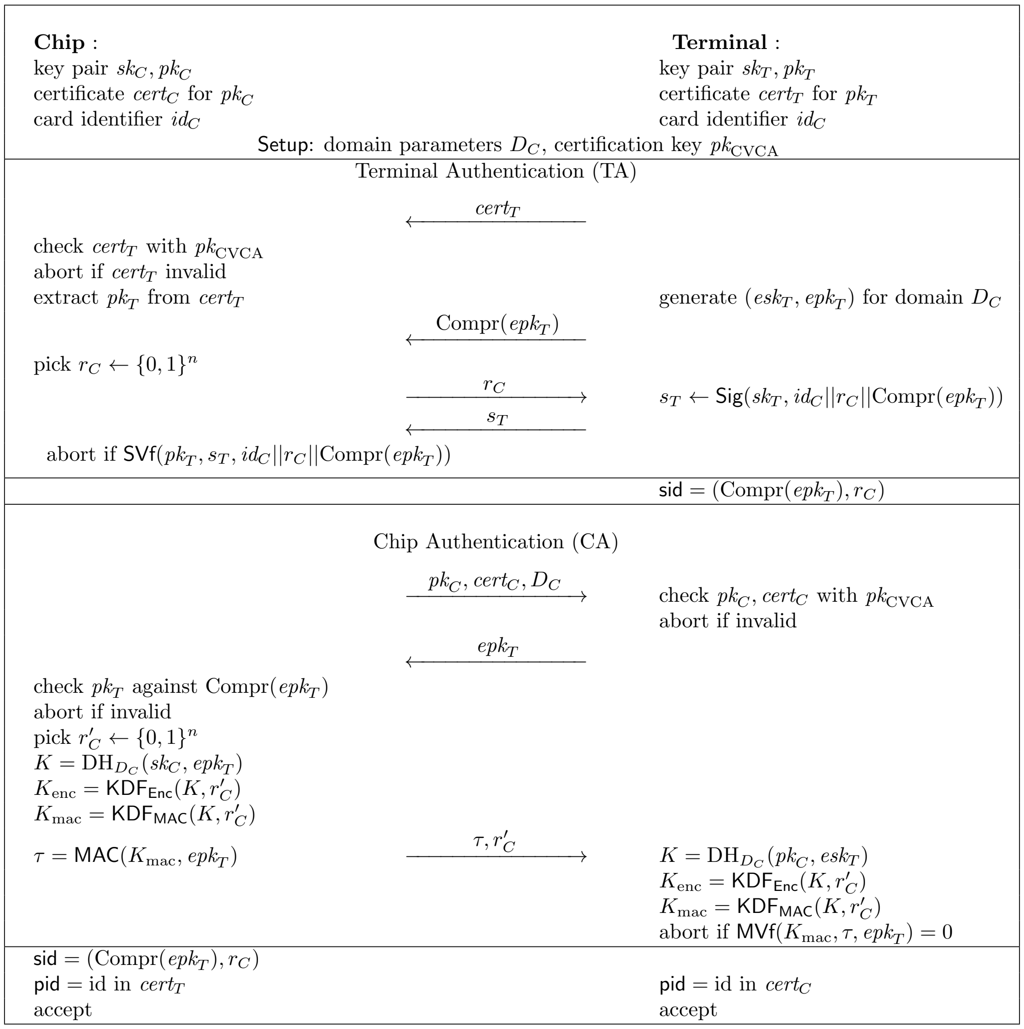

Chip :
key pair sk_C, pk_C
certificate $cert_C$ for pk_C
card identifier id_C
Terminal :
key pair sk_T, pk_T
certificate $cert_T$ for pk_T
card identifier id_C
Setup: domain parameters D_C, certification key pk_{CVCA}
Terminal Authentication (TA)
$\xleftarrow{cert_T}$
check $cert_T$ with pk_{CVCA}
abort if $cert_T$ invalid
extract pk_T from $cert_T$
generate (esk_T, epk_T) for domain D_C
$\xleftarrow{\mathrm{Compr}(epk_T)}$
pick $r_C \leftarrow \{0,1\}^n$
$\xrightarrow{r_C}$
$s_T \leftarrow \mathsf{Sig}(sk_T, id_C \| r_C \| \mathrm{Compr}(epk_T))$
$\xleftarrow{s_T}$
abort if $\mathsf{SVf}(pk_T, s_T, id_C \| r_C \| \mathrm{Compr}(epk_T))$
$\mathsf{sid} = (\mathrm{Compr}(epk_T), r_C)$
Chip Authentication (CA)
$\xrightarrow{pk_C, cert_C, D_C}$
check $pk_C, cert_C$ with pk_{CVCA}
abort if invalid
$\xleftarrow{epk_T}$
check pk_T against $\mathrm{Compr}(epk_T)$
abort if invalid
pick $r'_C \leftarrow \{0,1\}^n$
$K = \mathrm{DH}_{D_C}(sk_C, epk_T)$
$K_{\mathrm{enc}} = \mathsf{KDF}_{\mathsf{Enc}}(K, r'_C)$
$K_{\mathrm{mac}} = \mathsf{KDF}_{\mathsf{MAC}}(K, r'_C)$
$\tau = \mathsf{MAC}(K_{\mathrm{mac}}, epk_T)$
$\xrightarrow{\tau, r'_C}$
$K = \mathrm{DH}_{D_C}(pk_C, esk_T)$
$K_{\mathrm{enc}} = \mathsf{KDF}_{\mathsf{Enc}}(K, r'_C)$
$K_{\mathrm{mac}} = \mathsf{KDF}_{\mathsf{MAC}}(K, r'_C)$
abort if $\mathsf{MVf}(K_{\mathrm{mac}}, \tau, epk_T) = 0$
$\mathsf{sid} = (\mathrm{Compr}(epk_T), r_C)$
pid = id in $cert_T$
accept
pid = id in $cert_C$
accept

Figure 3: Terminal Authentication (TA) and Chip Authentication (CA). All number-theoretic operations are modulo q resp. over the elliptic curve.

Adding attributes. Besides completing the (modified) EAC protocol access permission may depend on the attributes a card can provide. We assume that the responder may request to see the attributes and then the card sends the stored data. Note that these steps are carried out over the secure-messaging channel. If the terminal requests to see the attributes then we set on the card's side, upon successful completion of this step, the cryptographic session identifier to be (sid, C) for the transmitted ciphertext C of the attributes.

We stress again that, if the management system is unreachable, the ordinary authentication process of the EAC protocol is still in effect such that reading the attributes after executing the EAC protocol may be optional. For restored sessions, however, reading out attributes over the secure channel is the only mechanism to ensure that the card actually holds the secret keys.

Persistent session contexts. Session contexts for the German identity card are supposed to be deleted when the card becomes unpowered or is being reset. In contrast we may assume that session contexts are stored over longer periods of time. One may even continuously use a stored context for "cascaded" executions. Furthermore, we do not make any restrictions on the number of stored contexts; the number may depend on the card's architecture.

In addition, we do not pose any stipulations on the choice of the identifiers of session contexts but advise some "collision-free" choice. For example, important responders may be assigned a fixed identifier whereas other terminals may select identifiers at random. Since the choice only affects efficiency but not security we do not discuss possible strategies here further.

We let the cryptographic session identifier sid monotonously grow with the number of restored contexts, because we append the card's latest authenticated ciphertext of the attributes, sent via secure messaging, to the current identifier value $\mathsf{sid}_{\text{old}}^{\mathsf{Cont}}$ of the context upon acceptance. Partner identifiers and attributes remain unaltered. An execution example of a restored session of the distributed architecture is given in Figure 4.

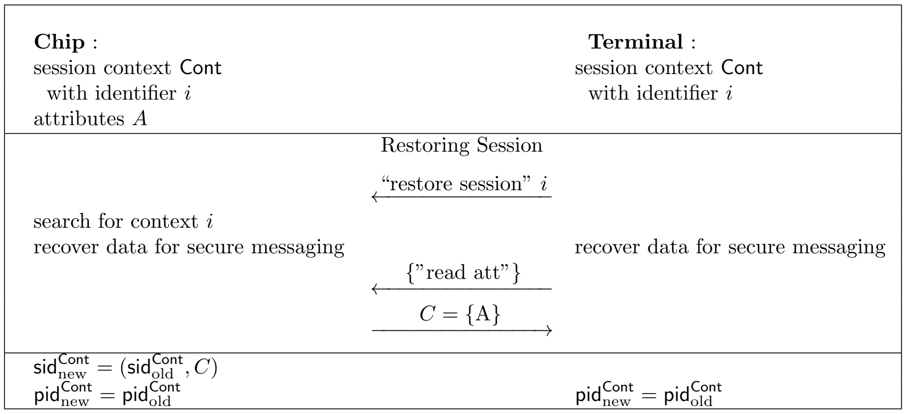

Chip :
session context Cont
with identifier i
attributes A
Terminal :
session context Cont
with identifier i
Restoring Session
"restore session" i
search for context i
recover data for secure messaging
recover data for secure messaging
{"read att"}
$C = \{\text{A}\}$
$\mathsf{sid}_{\text{new}}^{\mathsf{Cont}} = (\mathsf{sid}_{\text{old}}^{\mathsf{Cont}}, C)$
$\mathsf{pid}_{\text{new}}^{\mathsf{Cont}} = \mathsf{pid}_{\text{old}}^{\mathsf{Cont}}$
$\mathsf{pid}_{\text{new}}^{\mathsf{Cont}} = \mathsf{pid}_{\text{old}}^{\mathsf{Cont}}$

Figure 4: Restoring sessions and reading out attributes in the case of the distributed architecture (with the controller acting as the terminal). Here, $\{\dots\}$ denotes protocol messages sent via secure messaging. Note that the updated cryptographic session identifier is augmented by the ciphertexts of the attributes.

3 Access Systems and their Security

Before discussing the security of our (modified) EAC protocol we first abstractly introduce access systems and their desired security features.

3.1 Access Systems

An access system $\mathcal{AS}$ consists of efficient algorithm $(\mathsf{KG}_C, \mathsf{KG}_R, \Pi_S, \Pi_R)$ for generating keys (and attributes) for the card, $(pk_C, sk_C, cert_C) \leftarrow \mathsf{KG}_C(1^n)$. We note that we will later add attributes to cards but in a session-specific way. More formally, we assume that the attributes A are provided "from the outside" and all attributes are stored externally in some list $\mathcal{ATT}$ and given to the managing party. Similarly, the system comprises a key generator for the responding party, $(pk_R, sk_R, cert_R) \leftarrow \mathsf{KG}_R(1^n)$, and stateful algorithms Π_C, Π_R for the party's protocol messages. We sometimes omit mentioning certificates for the public keys explicitly, and that there must be a certification authority; all the details are relevant when taking an in-depth look at EAC, but we are mainly concerned with the fact that EAC provides a secure key exchange protocol.

We assume the usual completeness property, saying that for genuine keys, the card holding attribute $A \in \mathcal{ATT}$, and faithful execution of the algorithms Π_S, Π_R the responder eventually accepts the card. Here, to cover restored sessions we assume that the responder may accept multiple times within a session. Formally, this is captured by running Π_S and Π_R in modes init and $\mathsf{restore}$, and we assume that at the end of the first execution of init on genuine data (involving $A \in \mathcal{ATT}$) the responder accepts, as well at the end of each execution in mode $\mathsf{restore}$. Note that every time the session continues, triggered via a $\mathsf{restore}$ command, the party goes from an accepting state to an unaccepting one.

3.2 Security Model

In all versions of the access system we assume a powerful adversary controlling the network.

Attack Model. We assume that all parties, divided exclusively into cards from set $\mathcal{C}$ and responders from a set $\mathcal{R}$, receive their (certified) key pairs as initial input at the outset of the attack. Since we do not want to make any assumptions about the structure of card attributes we leave it up to the adversary to assign attributes to cards upon initialization of a new session.

The adversary has full control over the network and can, in particular, initiate new sessions of parties and decide when to deliver protocol messages (and potentially to modify such messages or even inject new ones). Formally, this is modeled by giving the adversary the following oracle access:

- $\textsc{Init}$: The adversary can initiate a new card or responder session by calling $\textsc{Init}(\mathrm{id})$ for some identity $\mathrm{id} \in \mathcal{C} \cup \mathcal{R}$. We assume that the identifier id uniquely determines a certificate and vice versa. In case of a card the adversary also has to provide some attribute A. The adversary may thus choose to hand out the same attribute to each card or change attributes depending on the concrete session. This attribute is immediately stored in a list $\mathcal{ATT}$. Upon such a call we spawn a new session of the party for attribute A and assign it a unique label ℓ for administrative purposes. The label ℓ is returned to the adversary and we write $\ell \leftarrow \textsc{Init}(\mathrm{id}, [A])$.

- $\textsc{Send}$: The adversary can send any protocol message m to a session with label ℓ via the $\textsc{Send}(\ell, m)$ command. If the session has not been initialized before, then the oracle immediately returns $\perp$. Else, it makes the corresponding party compute the next protocol message and this message is returned to the adversary (potentially also returning $\perp$ to express rejection).

  In particular, we assume that the adversary may make the party switch to modes, from init to $\mathsf{restore}$ or starting a new $\mathsf{restore}$ session, if receiving $\mathsf{Send}(\ell, \mathsf{restore})$. If the party has not finished successfully the previous mode yet, it may reject. In case the execution is successfully completed, the adversary is informed that the party has accepted.

- $\textsc{Corrupt}$: The adversary can corrupt a party with identity id by using the $\mathsf{Corrupt}(\mathrm{id})$ command. It receives the party's long-term keys and internal state in return, and we put id in the (initially empty) set $\mathsf{Corrupt}$ of corrupt parties. From now on, we assume that the adversary does not send further commands to that session.

To facilitate the notation we use the following mappings. We write $\mathsf{ACC}(\ell)$ for the (current) acceptance status of the (responder) session ($\mathtt{true}$ or $\mathtt{false}$), and $\mathsf{ID}(\ell)$ for the identity id of the session owner, and $\mathsf{PID}(\ell)$ for the intended partner pid, possibly $\mathsf{pid} = \perp$ at this point. Similarly, $\mathsf{SID}(\ell)$ denotes the current value of the session identifier. We also denote by $\mathsf{ATT}(\ell)$ for a session the attribute A the card has been initialized with resp. the attribute the responder it has received (if at all).

```
Experiment $\mathsf{ImpRes}_{\mathcal{A}}^{\mathcal{AS}}(n)$
1 : foreach $i \in \mathcal{C} \cup \mathcal{S}$ do
2 :    if $i \in \mathcal{C}$ then $(sk_i, pk_i, cert_i) \leftarrow \mathsf{KG}_{\mathcal{C}}(1^n)$ fi
3 :    if $i \in \mathcal{R}$ then $(sk_i, pk_i, cert_i) \leftarrow \mathsf{KG}_{\mathcal{R}}(1^n)$ fi
4 : endforeach
5 : $pks \leftarrow \{(pk_i, cert_i) \mid i \in \mathcal{C} \cup \mathcal{R}\}$
6 : $\mathcal{A}^{\textsc{Init}(\cdots),\textsc{Send}(\cdots),\textsc{Corrupt}(\cdot)}(1^n, pks)$
7 : $b \leftarrow \mathsf{ImpResPred}$   // evaluate predicate ImpResPred on execution state
8 : return $\bar{b}$
```

Figure 5: Security of an access system.

Impersonation Resistance. Impersonation resistance of the access system now says that the adversary cannot make the responder accept, unless in the trivial case that a card with attribute A is accepted and the adversary has corrupted a card with these attributes (in which it could easily access the system by using that card), or if the adversary has merely relayed the communication between an honest card and the reader. This is formalized in Figure 5.

While Figure 5 describes the flow of the attack, the predicate $\mathsf{ImpResPred}$ in Figure 6, which is evaluated at the end of the attack, determines when the adversary wins. There are two cases when we declare the adversary to win. The first case is when the adversary has managed to make an honest responder accept an honest card which has not participated in the execution (or with different attributes). This corresponds to the **foreach** loop in Line 3 in Figure 6. The second case, covering replay resistance, is that partnered sessions are unique between cards and responders, or else the adversary wins, too. In particular, there cannot be two honest cards with the same session identifiers. This is checked in the **foreach** loop in Line 9 in Figure 6.

Note that, in the predicate $\mathsf{ImpResPred}$, if the responding party accepts and outputs some session identifier, then the card must have already accepted before, i.e., we assume that the responder receives the last message. This ensures that there is always a card with the same identifier sid at this point, unless the aversary has managed to break security. This holds in our setting here as the card sends the attributes or acknowledges the storage of the session context.

Definition 3.1 (Impersonation Resistance) *An access system $\mathcal{AS}$ is impersonation resistant if for any efficient adversary $\mathcal{A}$ we have that*

$$\text{Prob}\left[\mathit{ImpRes}_{\mathcal{A}}^{\mathcal{AS}}(n) = 1\right] \approx 0$$

is negligible.

Note that we let the adversary $\mathcal{A}$ decide when to stop the execution and to start evaluating the predicate. Hence, if it is advantageous and the adversary already detects a winning situation, it may end the execution immediately (instead of messing up the winning state by, say, corrupting another party). In our case this is easy to detect since all the data required to evaluate the predicate are known to the adversary. In general, if the predicate relies on some information unavailable to the adversary, then the adversary may just guess the point in time for such a state.

3.3 Privacy of Attributes

Privacy of attributes ensures that no adversary can learn the card's attributes (unless it is the responder and controls that party's long-term key). We use an indistinguishability-based approach

Predicate $\mathsf{ImpResPred}$ on execution state

```
$p \leftarrow \mathtt{true}$
// accepting responder session must have honest partner with same sid (or corrupt partner)
foreach $\ell \in \{\ell \mid \mathsf{ID}(\ell) \in \mathcal{R} \setminus \mathsf{Corrupt} \wedge \mathsf{ACC}(\ell) = \mathtt{true}\}$ do
    $p \leftarrow p \wedge [\mathsf{PID}(\ell) \in \mathcal{C} \cap \mathsf{Corrupt}$
                  $\vee\ \exists \ell' \neq \ell : (\mathsf{SID}(\ell') = \mathsf{SID}(\ell) \neq \bot \wedge \mathsf{PID}(\ell) = \mathsf{ID}(\ell')$
                                                  $\wedge\ \mathsf{ATT}(\ell') = \mathsf{ATT}(\ell))]$
endforeach
// Collisions among identifiers only between opposite partners
foreach
    $(\ell, \ell') \in \{(\ell, \ell') \mid \ell \neq \ell' \wedge \mathsf{ID}(\ell), \mathsf{ID}(\ell') \notin \mathsf{Corrupt} \wedge \mathsf{SID}(\ell) = \mathsf{SID}(\ell') \neq \bot\}$
do
    $p \leftarrow p \wedge [(\mathsf{ID}(\ell), \mathsf{ID}(\ell')) \in \mathcal{C} \times \mathcal{S} \cup \mathcal{S} \times \mathcal{C}]$
endforeach
return $p$
```

Figure 6: Security predicate $\mathsf{ImpResPred}$ for impersonation resistance.

here in which a privacy-adversary can, besides regular sessions, also initiate (multiple) executions on a random choice of one of two adversarially chosen attributes A_0, A_1.

The attack model is the same as for impersonation resistance. The only difference is that the adversary now also gets a challenge oracle CHALL, which is initialized with a secret bit $b \leftarrow \{0, 1\}$. When called about identity id $\in \mathcal{C}$ and two attributes A_0, A_1, the challenge oracle executes $\ell \leftarrow$ INIT(id, A_b) to initialize an execution with the card. It returns the session label ℓ to the adversary. From then on the adversary can communicate with the card's sessions via the SEND oracle for the corresponding label. The adversary eventually should predict the bit b.

To rule out trivial attacks, say, in which the adversary controls the corrupt responder, we require that the adversary has only asked the challenge oracle for identities of honest cards which refer to an honest partner. For this we check that for each query (id, A_0, A_1) to CHALL we neither have id $\in \mathsf{Corrupt}$ nor $\mathsf{PID}(\ell) \in \mathsf{Corrupt}$, where a yet unset partner identifier $\mathsf{PID}(\ell) = \bot$ does not belong to $\mathsf{Corrupt}$ by definition.

We can now define privacy with the experiment in Figure 7.

Definition 3.2 (Attribute-Privacy) *An attribute-based access system $\mathcal{AS}$ provides attribute-privacy private if for any efficient adversary $\mathcal{A}$ we have that*

$$\text{Prob}\left[\mathsf{APriv}_{\mathcal{A}}^{\mathcal{AS}}(n) = 1\right] \leq \tfrac{1}{2} + \mathit{negl}(n)$$

is negligibly close to $\frac{1}{2}$.

4 On the Security of EAC with Secure Messaging

Before discussing our analyses let us motivate the setting by the general idea behind the security argument.

Outline. Our general proof strategy is roughly as follows. Dagdelen and Fischlin [17] have basically shown that the EAC protocol is a Bellare-Rogaway secure key exchange protocol. Assuming

Experiment $\mathsf{APriv}_{\mathcal{A}}^{\mathcal{AS}}(n)$

```
1:  $b \leftarrow \{0,1\}$
2:  **foreach** $i \in \mathcal{C} \cup \mathcal{S}$ **do**
3:      **if** $i \in \mathcal{C}$ **then** $(sk_i, pk_i) \leftarrow \mathsf{KG}_{\mathcal{C}}(1^n)$ **fi**
4:      **if** $i \in \mathcal{S}$ **then** $(sk_i, pk_i) \leftarrow \mathsf{KG}_{\mathcal{S}}(1^n)$ **fi**
5:  **endforeach**
6:  $pks \leftarrow \{(i, pk_i) \mid i \in \mathcal{C} \cup \mathcal{S}\}$
7:  $a \leftarrow \mathcal{A}^{\textsc{Init}(\cdot,\cdot),\textsc{Send}(\cdot,\cdot),\textsc{Corrupt}(\cdot),\textsc{Chall}(b,\cdots)}(1^n, pks)$
8:  // check for trivial attacks where card or responder corrupt
9:  $p \leftarrow \mathtt{true}$
10: **foreach** $\ell$ returned by $\textsc{Chall}$ **do**
11:     $p \leftarrow p \wedge [\mathsf{ID}(\ell) \notin \mathsf{Corrupt} \wedge \mathsf{PID}(\ell) \notin \mathsf{Corrupt}]$
12: **endforeach**
13: **return** $p \wedge (a = b)$
```

Figure 7: Attribute privacy experiment

that secure messaging of the eID system, which follows ISO/IEC 9791-1 resp. ISO/IEC 10116, provides a secure channel for fresh keys, we can then apply the composition theorem of Brzuska et al. [14] to conclude that the combined protocol (where the channel keys are now determined by the EAC key exchange protocol) also provides a secure channel. In particular, it follows that the transmissions of the card's attributes via secure messaging can only be carried out by the corresponding party, and that any attack will lead the partner to reject. Let us elaborate on these steps in more detail.

Security of EAC. The result by Dagdelen and Fischlin [17] shows that EAC is a secure key exchange protocol in the Bellare-Rogaway sense. This means that EAC provides keys which are indistinguishable from random, even in presence of active adversaries.[1] In particular, and omitting some negligible terms for collisions among group elements and nonces, they show that the advantage of distinguishing actual keys from random is bounded by the terms to break the used MAC, signature and certification algorithms, to find second pre-images in the compression function, and to solve the Diffie-Hellman problem when given a decisional Diffie-Hellman oracle as help. All formal security notions of these primitives are given in Appendix A:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A},\mathrm{EAC}}^{\mathrm{AKE}}(n) \quad &\leq \quad q_e \cdot \Big(\mathbf{Adv}_{\mathcal{B}_1,\mathrm{MAC}}^{\mathrm{forge}}(n) + \mathbf{Adv}_{\mathcal{B}_2,\mathrm{Sig}}^{\mathrm{forge}}(n) + \mathbf{Adv}_{\mathcal{B}_3,\mathrm{Compr}}^{\mathrm{SecPre}}(n)\Big) \\ &\qquad + \mathbf{Adv}_{\mathcal{B}_4,\mathrm{Cert}}^{\mathrm{forge}}(n) + 2q_e^2 \cdot \mathbf{Adv}_{\mathcal{B}_5,\mathrm{DH}}^{\mathrm{GapDH}}(n)\end{aligned}$$

Here, q_e is the number of executions in the attack, and $\mathcal{B}_1, \ldots, \mathcal{B}_5$ are adversaries with a comparable run time as the attacker $\mathcal{A}$ on the EAC protocol. Since all advantages for the underlying primitives are assumed to be negligible, this shows security of the EAC protocol as an authenticated key exchange. The authors of [17] also discuss that security holds in case of a projection of the curve point onto the x-coordinate, making the compression function two-to-one.

[1]Dagdelen and Fischlin actually show a slight modification of EAC (with an independent authentication key) to be a BR-secure protocol; without this modification such a proof cannot go though. We also adopt this approach here, but as pointed out in [17] one can in principle use the strategy in [15, 14] to lift this to a security for the original protocol, at the cost of a more complicated proof.

```
Secure Channel
$(k, \mathsf{st}_S, \mathsf{st}_R) \leftarrow \mathsf{KG}(n)$
$b \leftarrow \{0,1\}$
$\mathcal{Q} \leftarrow ()$
$a \leftarrow \mathcal{A}^{\textsc{Send}(\cdots),\textsc{Rcv}(\cdots)}$
return $a = b$
```

```
Oracle $\textsc{Send}(m_0, m_1)$
$(C_0, \mathsf{st}'_{S,0}) \leftarrow \mathsf{Send}(k, m_0, \mathsf{st}_S)$
$(C_1, \mathsf{st}'_{S,1}) \leftarrow \mathsf{Send}(k, m_1, \mathsf{st}_S)$
if $C_0, C_1 \neq \perp$ then
    $\mathsf{st}_S \leftarrow \mathsf{st}'_{S,b}$
    $\mathcal{Q}.\mathsf{enqueue}(C_b)$
    return $C_b$
else
    return $\perp$
fi
```

```
Oracle $\textsc{Rcv}(C)$
$(m, \mathsf{st}_R) \leftarrow \mathsf{Rcv}(k, C, \mathsf{st}_R)$
if $b = 1$ and
    $\mathcal{Q}.\mathsf{dequeue}() \neq C$ then
    return $m$
else
    return $\perp$
fi
```

Figure 8: Security Experiment of (single instance of) channel protocol $(\mathsf{KG}, \mathsf{Send}, \mathsf{Rcv})$.

Security of Secure Messaging. The proposed channel protocol is secure messaging [3], which either uses 3DES in CBC mode with IV = 0 according to ISO/IEC 10116 for encryption, and in retail mode (MAC algorithm 3 with DES as block cipher) with IV = 0 according to ISO/IEC 9797-1 for authentication, with the data prepended by send sequence counter SSC which is incremented for each operation. The other option is to use AES in CBC mode according to ISO/IEC 10116 with IV = AES(K_{enc}, SSC) and to use AES in CMAC for authentication with 8 bytes of output according to SP 800-38B, where, again, the data is prepended by SSC before authentication.

In [25] Rogaway analyzes the encryption modes proposed in SP 800-38A resp. ISO/IEC 10116, including the CBC mode used in secure messaging with the IV = AES(K_{enc}, SSC) computed by applying the block cipher to the current send sequence counter SSC. Rogaway proposes an attack if the adversary has full control over the value SSC. The attack does not carry over to the setting used in secure messaging, where the encrypting party increments the value for each operation. This version can be actually shown to be secure [7]. Rogaway [25] confirms the authentication properties of the proposed MAC algorithms in ISO/IEC 9791-1 which are proposed here for secure messaging.

We thus assume that secure messaging provides a secure channel (in the sense of [14, Section 6.3] which in turn is based on the the notion of stateful authenticated encryption [24, 21]). The experiment lets an adversary call a challenge oracle to enqueue one of two message blocks m_0, m_1 into the channel, the choice made according to a secret but then fixed bit b, and to dequeue arbitrary ciphertexts on the receiver's side. The adversary wins if it manages to predict b or to make the receiver accept a decryption of an out-of-order sent ciphertext. See Figure 8. The advantage $\mathbf{Adv}^{\mathsf{SecCh}}_{\mathcal{A},\mathsf{Ch}}(n)$ of the adversary $\mathcal{A}$ is then defined to be the probability of predicting b beyond the pure guessing probability of $\frac{1}{2}$ (in the multi-instance setting).

Note that a secure channel in our scenario here comprises both confidentiality of the attributes, as well as authenticity. The latter suffices if the goal is to ensure that only the designated card can send the attributes, whereas the former also guarantees privacy of the attributes. If authentication suffices may be application dependent.

Compositional Security. Next we apply the compositional result in [14, Section 4] The theorem says that the combined protocol $\mathsf{EAC};\mathsf{SM}$ (where the channel keys for secure messaging in the multiple instances are determined by executing the EAC key exchange protocol first) also provides a secure channel, as if the keys have been chosen freshly. In particular we apply the composition theorem for so-called single-restricted games. This is a property which basically says that multiple concurrently running instances of a game correspond to several independent sessions, as in case of secure channels. For such games, it is shown that:

$$\mathbf{Adv}^{\text{SecCh}}_{\mathcal{A},\mathsf{EAC};\mathsf{SM}}(n) \leq q_e \cdot \mathbf{Adv}^{\text{AKE}}_{\mathcal{B}_1,\mathsf{EAC}}(n) + \mathbf{Adv}^{\text{SecCh}}_{\mathcal{B}_2,\mathsf{SM}}(n).$$

Note that this also requires for EAC to provide *match-security*, a property about collision-freeness of session identifiers, and to have *public session matching*, the ability to determine partnered sessions from the public transcript. Neither property has been discussed in [17] but it is easy to show them to hold for EAC.

EAC and Impersonation Resistance. The EAC protocol not only provides a secure key exchange protocol but it also ensures impersonation resistance. This roughly means that, at the end of the EAC protocol, no adversary can make the responder accept a card, unless the card has been corrupted before or if the adversary merely relayed the communication between the card and the reader. Based on the results in [17] this has been proven formally in [23] for EAC with auxiliary data, when the system is viewed as a transaction system. For "empty" transaction data their protocol is identical to the EAC protocol here, including also the choice for session identifiers, and their security guarantee of session-definite unforgeability is stronger than the requirement of impersonation resistance here. (It has also been shown there that session identifiers collide with negligible probability only.)

More precisely, Morgner et al. [23] show that for any efficient adversary $\mathcal{A}$ (against their unforgeability notion and thus our impersonation resistance notion of the EAC protocol without attributes) there exists efficient adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ against the underlying cryptographic primitives for forging signatures on behalf of terminals, forging MACs on behalf of cards, and solving the computational Diffie-Hellman problem in the presence of a decisional DH oracle. Concretely, the probability of forging transaction resp. impersonating here is bounded from above by:

$$\begin{aligned}\text{Prob}\left[\mathsf{ImpRes}^{\text{EAC}-\mathcal{AS}}_{\mathcal{A}}(n) = 1\right] \quad &\leq \quad \binom{s}{2} \cdot \left(2^{-n} + \tfrac{R}{q}\right) + S \cdot \mathbf{Adv}^{\text{unf}}_{\mathcal{B}_1,\mathcal{SIG}}(n) \\ &\qquad + S \cdot \mathbf{Adv}^{\text{unf}}_{\mathcal{B}_2,\mathcal{MAC}}(n) + C \cdot S \cdot \mathbf{Adv}^{\text{GapDH}}_{\mathcal{B}_3,D_C}(n)\end{aligned}$$

where it is assumed that Compr is a R-regular function, i.e., every image has exactly R pre-images, q is the group size specified by D_C, the adversary initiates at most s sessions, and there are at most C cards and S terminals.

5 Security of the Architectures

Here we discuss the cryptographic strength of the various settings of the access system. For the analysis we assume that the other channels between parties, e.g., connecting the controller with the management system, are strongly secure. This is modeled by disallowing the adversary to tamper with, or even read the data, sent over these secured channels. We first treat the cases of the integrated, distributed, and eID-service architecture. By the assumption about secure connection between the various parties, we can view the reader, controller, server and management as a single entity in these settings. Only the authentication-service architecture with the split cryptographic operations requires a special treatment.

5.1 The Integrated, Distributed and eID-Service Architectures

We give the security statements for the integrated architecture only. Recall that the integrated terminal architecture, for example, assumes all eID operations are carried out by the reader itself. After completion of the TA and CA phase, the reader gets the attributes of the card (where the communication is secured via the secure messaging), and forwards the attributes of the card to the management system for approval. The communication with the management system is secured via a TLS channel. Upon approval, the reader grants access. Analogously, if restoring a session, then the reader only accepts if the securely sent attributes are approved.

Theorem 5.1 (Impersonation Resistance) *The integrated terminal architecture provides an impersonation-resistant access system, such that for any efficient adversary $\mathcal{A}$ there exists efficient adversaries $\mathcal{B}_1, \mathcal{B}_2$ such that*

$$\text{Prob}\left[\mathsf{ImpRes}_{\mathcal{A}}^{\mathcal{AS}}(n) = 1\right] \leq \text{Prob}\left[\mathsf{ImpRes}_{\mathcal{B}_1}^{EAC-\mathcal{AS}}(n) = 1\right] + 2 \cdot \boldsymbol{Adv}_{\mathcal{B}_2,\mathsf{EAC;SM}}^{SecCh}(n)$$

Moreover, adversaries $\mathcal{B}_1, \mathcal{B}_2$ have roughly the same running time $\mathcal{A}$ plus the time to carry out the other steps in the experiment. Note that since the terms on the right hand side are assumed to be negligible, as discussed in Section 4, it follows that the system is impersonation resistance

Proof. We consider three cases: (a) either the adversary manages to create collisions in two honest card sessions or in two honest reader sessions; or (b) an honest reader accepts at the end of the EAC protocol but such that the identified card is neither corrupt and there is no genuine session of the card with the same session identifier; or (c) an honest reader accepts some encrypted and authenticated attributes in a secure-messaging protocol where this ciphertext has not been sent by an honest card.

The first two cases are covered by the impersonation resistance of the EAC protocol. It is straightforward to build an adversary $\mathcal{B}_1$ simulating the environment for $\mathcal{A}$ through its own attack and by adding the extra steps for the secure messaging. If $\mathcal{A}$ breaks the impersonation resistance of the combined protocol then $\mathcal{B}_1$ breaks the security of the EAC protocol. This in particular also implies that the continuously growing session identifiers in restored sessions stay distinct among card sessions as well as among responder sessions, i.e., a subsequent collisions cannot happen anymore.

It remains to argue that case (c) cannot occur. In order to violate the predicate in the impersonation experiment, an honest reader accepts a ciphertext C_i^* in some session with identifier $\mathsf{sid} = (\text{Compr}(\mathit{epk}_T), r_C, C_1, C_2, \ldots, C_{i-1})$ for previously sent ciphertexts $C_1, C_2, \ldots$ in the restored sessions before. Since we quantify over all adversaries we can assume that C_i^* is the first ciphertext which deviates from a session of an honest card, i.e., there exists a session of a card with $\mathsf{sid} = (\text{Compr}(\mathit{epk}_T), r_C, C_1, C_2, \ldots, C_{i-1})$ or already with $\mathsf{sid} = (\text{Compr}(\mathit{epk}_T), r_C, C_1, C_2, \ldots, C_i)$. But then we must have that C_i^* is new or that $C_i^* \neq C_i$, and yet C_i^* decrypts to some attribute which the reader accepts. This can be straightforwardly used in a reduction against the authenticity of the composed channel protocol such that the probability of this event is bounded by the security of the secure messaging channel.

More formally, construct algorithm $\mathcal{B}_2$ against the combined protocol $\mathsf{EAC;SM}$ from $\mathcal{A}$ as follows. Recall that in this combined protocol the keys for the channel are generated by the EAC protocol, such that this perfectly simulates $\mathcal{A}$'s environment up to this step. We also assume, by the above discussion, that session identifiers are unique between cards and responders, and that each honest responder has a unique partnered honest card.

For simulating the channel transmission of an honest card, algorithm $\mathcal{B}_2$ calls the Send oracle for the pair (A, A) for the card's attribute A to get a (valid) ciphertext. For simulating the receipt of a ciphertext at the responder's side, for an honest responder, there are two cases. If the intended partner is a corrupt card, then we can assume that $\mathcal{B}_2$ already knows the shared key (via a reveal query at the end of the key exchange protocol) and can simply act as the original responder. Otherwise, algorithm $\mathcal{B}_2$ forwards the ciphertext to its Rcv oracle. If this oracle returns a message $m \neq \perp$, then $\mathcal{B}_2$ immediately outputs 1. Else, $\mathcal{B}_2$ lets the responder in the simulation for $\mathcal{A}$ accept if and only if the ciphertext has been created by a partnered card before, complying with the "queue property" of the channel protocol.

If, at the end, $\mathcal{B}_2$ has not returned 1 yet, then it outputs a random guess. This completes the description of the perfect simulation (up to the point where an honest responder can correctly decrypt a new ciphertext sent through the channel protocol). If the above now happens in an actual attack, that C_i^* is new or that $C_i^* \neq C_i$ but the original responder would accept but we reject, then we would break the authenticity of the channel protoocl with the same probability. That is, we then have

$$\text{Prob}[\,\text{case (c)}] \leq 2 \cdot \mathbf{Adv}^{\text{SecCh}}_{\mathcal{B}_2,\mathsf{EAC;SM}}(n)$$

where the factor of 2 is due to the fact that we have $b = 1$ and can thus see an output by the Rcv oracle with probability $\frac{1}{2}$ only.

Note that this argument about the channel is independent of whether restored sessions have been overwritten or not. □

Attribute privacy follows analogously, using again the fact that secure messaging provides a secure channel:

Theorem 5.2 (Privacy) *The integrated terminal architecture provides an attribute-private access system, such that for any efficient adversary $\mathcal{A}$ there exists an efficient adversary $\mathcal{B}$ such that*

$$\text{Prob}\left[\mathsf{APriv}^{\mathcal{AS}}_{\mathcal{A}}(n) = 1\right] \leq \tfrac{1}{2} + \boldsymbol{Adv}^{SecCh}_{\mathcal{B},\mathsf{EAC;SM}}(n)$$

Proof. Note that the adversary $\mathcal{A}$ against privacy can only win in the experiment if it does not query the challenge oracle about a card identity such that the card or its partner is corrupt. This in particular means that the derived key for secure messaging must still be secure, and the adversary here can only distinguish the attributes if it breaks confidentiality of the channel protocol. This can again be formalized easily via a reduction to the corresponding game.

More formally, construct adversary $\mathcal{B}$ as in the previous theorem, running the combined protocol $\mathsf{EAC;SM}$ to simulate $\mathcal{A}$'s attack. For every call of $\mathcal{A}$ about attributes A_0, A_1 to the challenge oracle Chall to start a new session in this attack, adversary $\mathcal{B}$ simply initiates a new session and stores A_0, A_1 for later use. If the card is later supposed to send its attributes in this session, then $\mathcal{B}$ calls its Send oracle about A_0, A_1 to get a ciphertext. The (honest) responder of that ciphertext in the simulation simply accepts.

Eventually, if $\mathcal{A}$ outputs a bit b, then $\mathcal{B}$ copies this bit to its output and stops. Since the simulation is perfect, it follows that $\mathcal{A}$'s advantage is at most the one of $\mathcal{B}$. □

5.2 The Authentication-Service Architecture

In principle one can show the same results as for the other architectures to the case of the authentication-service scenario. Recall that there an authentication server signs the TA data forwarded by the controller, and the controller continues the execution with that signature. The other steps are as in the other cases.

Note that the signature and the (ephemeral) DH key in the EAC protocol serve different cryptographic purposes. The signature only binds the ephemeral key to the terminal's identity and prevents the adversary to inject its own key. The DH key is used to establish the session key and, as long as the adversary does not get to learn the ephemeral secret key, the adversary is not able to compute the joint DH key with the card. This has been discussed in [17] in the context of key-compromise impersonation (KCI) resistance.

For us this means that even corruption of the authentication service's signing key does not allow to complete the EAC protocol and to learn the channel keys, for sessions in which an honest reader picks the ephemeral key. In particular, knowledge of the signing key does not allow to break security of previously completed sessions (forward security). Formally, we can augment both attack model by granting the adversary another SigKey oracle which, when queried about a responder's identity, returns the party's secret signing key. The party may still act as an honest party in sessions, with the internal choices hidden from the adversary.

The conditions for impersonation resistance remain unchanged, except for giving the adversary access to the SigKey oracle. Since a responder party only completes a session (and accepts with a session identifier) if it executes the protocol steps itself, any such session in question uses an honestly chosen ephemeral key epk_T on the responder's side. Hence, the security of session keys for such honestly ephemeral keys argued in [23] still holds.[2]

For attribute privacy we need to change the non-triviality check in Line 11 of the experiment in Figure 7. There, we checked for each card session ℓ returned by the challenge oracle that the card nor its intended partner is corrupt:

$$[\mathsf{ID}(\ell) \notin \mathsf{Corrupt} \wedge \mathsf{PID}(\ell) \notin \mathsf{Corrupt}]$$

Here, we need to check that the responder's party has contributed the ephemeral key honestly:

$$[\mathsf{ID}(\ell) \notin \mathsf{Corrupt} \wedge \exists \ell' \neq \ell : (\mathsf{SID}(\ell) = \mathsf{SID}(\ell) \neq \bot \wedge \mathsf{PID}(\ell) = \mathsf{ID}(\ell'))]$$

Given this, attribute privacy follows as before, because session keys are still fresh for such sessions.

6 On Achieving Forward Security

While the EAC protocol has been shown to provide forward security (against leakage of the terminal's long-term secrets) [17], learning a card's long-term secret key or the secure-messaging keys for restored sessions allows both to impersonate towards readers and to learn the attributes in past executions. Since any protocol with fixed symmetric keys is amenable to a-posteriori leakage we briefly discuss here a version which allows to protect attributes sent before. This, however, comes at the price of keeping state and synchronization issues.

We only describe here one way to update the symmetric keys securely. Assume that a party has sent (or received) the attributes over secure messaging and goes into an accepting state again. Then, instead of storing the same keys $K_{\mathrm{enc}}, K_{\mathrm{mac}}$ and (incremented) counter SSC, the party updates its keys as $(K'_{\mathrm{enc}}, K'_{\mathrm{mac}}) \leftarrow \mathsf{KDF}(\mathrm{AES}(K_{\mathrm{enc}}, \mathrm{SSC}))$ for some pseudorandom generator KDF, and re-sets the counter to $\mathrm{SSC}' \leftarrow 1$. It stores $(K'_{\mathrm{enc}}, K'_{\mathrm{mac}}, \mathrm{SSC}')$ in the session context.

The reason that this method of updating the keys is secure is based on the observation that $\mathrm{AES}(K_{\mathrm{enc}}, \mathrm{SSC})$ yields a secure random initialization vector for the channel encryption. It can hence also been used as a seed for the pseudorandom generator KDF to generate quasi fresh keys $K'_{\mathrm{enc}}, K'_{\mathrm{mac}}$. In other words, if the adversary corrupts the party holding the session context we may as well hand over fresh random keys $K'_{\mathrm{enc}}, K'_{\mathrm{mac}}$. This follows as, without knowledge of K_{enc}, the value $\mathrm{AES}(K_{\mathrm{enc}}, \mathrm{SSC})$ is indistinguishable from random, and so is then the output of KDF.

We note that any other forward-secure pseudorandom generators [9] may be used. In general, if we cannot rely on the properties of encryption with counters and AES, it is then safe to augment the state by another key K_{prg} which is used to update keys via $(K'_{\mathrm{enc}}, K'_{\mathrm{mac}}, K'_{\mathrm{prg}}) \leftarrow \mathsf{KDF}(K_{\mathrm{prg}})$.

7 Conclusion

The access system based on EAC with session restoring provides an impersonation resistant and attribute-hiding solution. Here, both security properties hold in a very strong sense, thwarting active adversaries with strong control over the network, and leaving the adversary essentially only trivial attacks from a cryptographic viewpoint. On top, the system is very similar to the existing

[2]The proof relies on the unforgeability of signatures only to ensure that the adversary cannot inject its own ephemeral key, which is guaranteed by construction here.

EAC system and may thus be easy to implement on existing infrastructures for the German identity card (or the future eIDAS system).

Acknowledgments

We thank the anonymous reviewers of SSR 2016 for valuable comments.

References

[1] Bundesamt für Sicherheit in der Informationstechnik (BSI): *Advanced Security Mechanism for Machine Readable Travel Documents – Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI)*, BSI-TR-03110, Version 2.0, 2008.

[2] Bundesamt für Sicherheit in der Informationstechnik (BSI): *Technical Guideline TR-03110-2: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token*, Part 2, Protocols for electronic IDentification, Authentication and trust Services (eIDAS). BSI-TR-03110, Version 2.20, 2015.

[3] Bundesamt für Sicherheit in der Informationstechnik (BSI): *Technical Guideline TR-03110-3: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token*, Part 3, Common Specifications. BSI-TR-03110, Version 2.20, 2015.

[4] Dan Boneh, Ben Lynn, and Hovav Shacham: *Short Signatures from the Weil Pairing*, Asiacrypt, Lecture Notes in Computer Science, Volume 2248, pp. 514-532, Springer-Verlag, 2001.

[5] Frank Morgner: Transaktionsabsicherung mit der Online-Ausweisfunktion. Kryptographische Bindung von Transaktionsdaten an den Personalausweis. Presentation, CeBit 2014, March 2014.

[6] P. Bastian: Physical Access Control Systems Using Asymmetric Cryptography, Master-Arbeit, Humboldt-Universität zu Berlin, 2015.

[7] Mihir Bellare, Anand Desai, E. Jokipii, Phillip Rogaway: A Concrete Security Treatment of Symmetric Encryption. FOCS, pp. 394-403, IEEE, 1997.

[8] M. Bellare and P. Rogaway: Entity Authentication and key distribution. Crypto 93, Lecture Notes in Computer Science, Volume 773, Springer-Verlag, 1994.

[9] M. Bellare and B.S. Yee: Forward-security in private-key cryptography. CT-RSA 2003, Lecture Notes in Computer Science, Volume 2612, pp. 1-18, Springer-Verlag, 2003.

[10] Jens Bender, Özgür Dagdelen, Marc Fischlin, Dennis Kügler: The PACE|AA Protocol for Machine Readable Travel Documents, and Its Security. Financial Cryptography, Lecture Notes in Computer Science, Volume 7397, pp. 344-358, Springer-Verlag, 2012.

[11] Jens Bender, Özgür Dagdelen, Marc Fischlin, Dennis Kügler: Domain-Specific Pseudonymous Signatures for the German Identity Card. Information Security Conference (ISC) 2012, Lecture Notes in Computer Science, Volume 7483, pp. 104-119, Springer-Verlag, 2012.

[12] Jens Bender, Marc Fischlin, Dennis Kügler: Security Analysis of the PACE Key-Agreement Protocol Information Security Conference (ISC) 2009, Lecture Notes in Computer Science, Volume 5735, pp. 33-48, Springer-Verlag, 2009.

[13] Jens Bender, Marc Fischlin, Dennis Kügler: The PACE|CA Protocol for Machine Readable Travel Documents INTRUST 2013, Lecture Notes in Computer Science, Volume 8292, pp. 17-35, Springer-Verlag, 2013

[14] Christina Brzuska: On the Foundations of Key Exchange. Dissertation, Technische Universität Darmstadt, `http://tuprints.ulb.tu-darmstadt.de/id/eprint/3414`, 2013.

[15] Christina Brzuska, Marc Fischlin, Nigel P. Smart, Bogdan Warinschi, Stephen C. Williams: Less is more: relaxed yet composable security notions for key exchange. Int. J. Inf. Sec. Volume 12(4), pp. 267-297, 2013.

[16] Jean-Sebastien Coron, Aline Gouget, Thomas Icart, Pascal Paillier: Supplemental Access Control (PACE v2): Security Analysis of PACE Integrated Mapping. In: Cryptography and Security: From Theory to Applications, Lecture Notes in Computer Science, Volume 6805, pp. 207-232, Springer-Verlag, 2012.

[17] Özgür Dagdelen, Marc Fischlin: Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents. ISC 2010, Lecture Notes in Computer Science, pp. 54-68, Springer-Verlag, 2010.

[18] Lucjan Hanzlik, Miroslaw Kutylowski: Restricted Identification Secure in the Extended Canetti-Krawczyk Model. J. UCS 21(3): 419-439 (2015)

[19] Lucjan Hanzlik, Lukasz Krzywiecki, Miroslaw Kutylowski: Simplified PACE—AA Protocol. ISPEC 2013: 218-232

[20] International Civil Aviation Organization: *Doc 9303, Machine Readable Travel Documents*, Part 11, Security Mechanisms for MRTDs, 7th Edition, 2015.

[21] Tibor Jager, Florian Kohlar, Sven Schäge, Jörg Schwenk: On the Security of TLS-DHE in the Standard Model. CRYPTO, Lecture Notes in Computer Science, pp. 273-293, Springer-Verlag, 2012.

[22] Miroslaw Kutylowski, Lukasz Krzywiecki, Przemyslaw Kubiak, Michal Koza: Restricted Identification Scheme and Diffie-Hellman Linking Problem. INTRUST 2011: 221-238

[23] F. Morgner, P. Bastian, M. Fischlin: Securing Transactions with the eIDAS Protocols. The WISTP International Conference on Information Security Theory and Practice Series, Lecture Notes in Computer Science, Volume 9895, Springer-Verlag, 2016.

[24] Kenneth G. Paterson, Thomas Ristenpart, Thomas Shrimpton: Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol. ASIACRYPT, Lecture Notes in Computer Science, pp. 372-389, Springer-Verlag, 2011.

[25] P. Rogaway: Evaluation of Some Blockcipher Modes of Operation, Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan, February 2011.

A Security Notions of Cryptographic Primitives

This part of the paper here is almost verbatim from the full version of [10].

Message Authentication Codes. A message authentication code $\mathcal{M}$ consists of three efficient algorithms $(\mathsf{MKGen}, \mathsf{MAC}, \mathsf{MVf})$ where $\mathsf{MAC}(k, m)$ maps any key k generated by key generation algorithm MKGen and any message m to a MAC (resp. tag) T which is verifiable with the help of $\mathsf{MVf}(k, m, T)$ with binary output. Completeness demands again that for any valid key k and any message m the value $T \leftarrow \mathsf{MAC}(k, m)$ makes $\mathsf{MVf}(k, m, T)$ return 1.

We require that the message authentication code $\mathcal{M}$ is unforgeable under adaptively chosen-message attacks. That is, the adversary is granted oracle access to $\mathsf{MAC}(k, \cdot)$ and $\mathsf{MVf}(k, \cdot, \cdot)$ for random key k generated by MKGen and wins if it, at some point, makes a verification query (m, T) about a message m which has not been sent previously to MAC, and such that MVf returns 1 for this message. We denote by $\mathbf{Adv}^{\text{forge}}_{\mathcal{M}}(t, q_m, q_v)$ a (bound on the) value ϵ for which no attacker in time t can win (making at most q_m MACs queries and q_v verification queries) with probability more than ϵ. For a concrete attacker $\mathcal{A}$ we write $\mathbf{Adv}^{\text{forge}}_{\mathcal{A},\mathcal{M}}(n)$ to denote the fact that $\mathcal{A}$ attacks the scheme in the above sense (for security parameter n).

Signatures and Certificates. A signature scheme $\mathcal{S} = (\mathsf{SKGen}, \mathsf{Sig}, \mathsf{SVf})$ consists of efficient algorithms for creating key pairs (sk, pk), signing messages $s \leftarrow \mathsf{Sig}(sk, m)$, and verifying signatures, $d \leftarrow \mathsf{SVf}(pk, m, s)$ with $d \in \{0, 1\}$. It must be that for signatures created under valid key pairs SVf always returns 1 (correctness). Unforgeability says that no algorithm should be able to forge the signer's signature. That is, a signature scheme $\mathcal{S} = (\mathsf{SKGen}, \mathsf{Sig}, \mathsf{SVf})$ is (t, q_s, ϵ)-unforgeable if for any algorithm $\mathcal{A}$ running in time t the probability that $\mathcal{A}$ outputs a signature to a fresh message under a public key is $\mathbf{Adv}^{\text{forge}}_{\mathcal{S}}(t, q_s)$ (which should be negligible small) while $\mathcal{A}$ has access (at most q_s times) to a singing oracle. As before, for a concrete attacker $\mathcal{A}$ we write $\mathbf{Adv}^{\text{forge}}_{\mathcal{A},\mathcal{S}}(n)$ to denote the fact that $\mathcal{A}$ attacks the scheme in the above sense (for security parameter n).

We also assume a certification authority CA, modeled like the signature scheme through algorithms $\mathcal{CA} = (\mathsf{CKGen}, \mathsf{Certify}, \mathsf{CVf})$, but where we call the "signing" algorithm $\mathsf{Certify}$. This is in order to indicate that certification may be done by other means than signatures. We assume that the keys $(sk_{\text{CA}}, pk_{\text{CA}})$ of the CA are generated at the outset and that pk_{CA} is distributed securely to all parties (including the adversary). We also often assume that the certified data is part of the certificate. We define unforgeability for a certification scheme $\mathcal{CA}$ analogously to signatures, and denote the advantage bound of outputting a certificate of a new value in time t after seeing q_c certificates by $\mathbf{Adv}^{\text{forge}}_{\mathcal{CA}}(t, q_c)$. We assume that the certification authority only issues unique certificates in the sense that for distinct parties the certificates are also distinct; we besides assume that the authority checks whether the keys are well-formed group elements. For a concrete attacker $\mathcal{A}$ we again write $\mathbf{Adv}^{\text{forge}}_{\mathcal{A},\mathcal{CA}}(n)$ to denote the fact that $\mathcal{A}$ attacks the scheme in the above sense (for security parameter n).

Second Preimage Resistance. We say that the compression function Compr is (t, ϵ)-second preimage resistant if the probability $\mathbf{Adv}^{\text{SecPre}}_{\text{Compr}}(t)$ of finding to a random ephemeral public key epk_T another key epk^*_T with the same compressed value is bounded by ϵ. For a concrete attacker $\mathcal{A}$ we again write $\mathbf{Adv}^{\text{SecPre}}_{\text{Compr}}(t)$ to denote the fact that $\mathcal{A}$ finds a second preimage in the above sense (for security parameter n).

Gap Diffie-Hellman Problem. We need the following gap Diffie-Hellman problem [4]. For a group $\mathcal{G}$ generated by g let $\text{DH}(X, Y)$ be the Diffie-Hellman value X^y for $y = \log_g Y$ (with g being an implicit parameter for the function). Then the gap Diffie-Hellman assumption says that solving the computational DH problem for (g^a, g^b), i.e., computing $\text{DH}(g^a, g^b)$ given only the random elements (g^a, g^b) and $\mathcal{G}, g$, is still hard, even when one has access to a decisional oracle $\text{DDH}(X, Y, Z)$ which returns 1 iff $\text{DH}(X, Y) = Z$, and 0 otherwise. We say that the GDH problem is $(t, q_{\text{DDH}}, \epsilon)$-hard if no algorithm can in time t compute the DH value with probability larger than ϵ, if making at most q_{DDH} queries. We let $\mathbf{Adv}^{\text{GDH}}_{\mathcal{G}}(t, q_{\text{DDH}})$ denote (a bound on) the value ϵ

for which the GDH problem is (t, q_{DDH}, ϵ)-hard. For a concrete attacker $\mathcal{A}$ we write $\mathbf{Adv}^{\mathrm{GDH}}_{\mathcal{A},\mathcal{G}}(n)$ to denote the fact that $\mathcal{A}$ attacks the problem in the above sense (for security parameter n).