Chapter 5
ePassport and eID Technologies

Lucjan Hanzlik and Mirosław Kutyłowski

Abstract This chapter is devoted to the design and implementation of electronic ID (eID) such as ePassports and electronic personal identity documents. We present an overview of existing and emerging concepts, both concerning threats and possible countermeasures. Thereby we aim to shed light on the development of ubiquitous systems, where many artifacts will require strong electronic identification with similar properties to those in the case of eIDs issued for humans.

5.1 Application Scenarios

The initial reason to develop an electronic identity document (eID)—an identity document with an electronic layer—was to prevent its forgery. The problem is that purely visual security measures have their limitations. The introduction of the electronic layer changed the situation substantially.

After introducing the electronic layer into identity documents it became evident that there are multiple opportunities to design and/or improve systems requiring strong authentication. The first obvious example is the eGate automatic border control.

5.1.1 Remote vs. Local Use

The motivation for designing eID systems comes from situations such as automated border control and enabling eGovernment applications. The first option is presenta-

Lucjan Hanzlik
Stanford University and CISPA, Stanford, CA, USA

M. Kutyłowski (✉)
University of Science and Technology, Wrocław, Poland
e-mail: miroslaw.kutylowski@pwr.edu.pl

G. Avoine, J. Hernandez-Castro (eds.), *Security of Ubiquitous Computing Systems*,
https://doi.org/10.1007/978-3-030-10591-4_5

tion of an eID to a reader and processing all information locally. The second option is using an eID as a secure token that enables us to create authenticated connections with a remote terminal. An alternative is to use no eID and realize its functions using an ID infrastructure. Each option has its advantages and disadvantages:

Option 0: Virtual ID An ID document contains data that may be fetched from an appropriate registry. So one can use a "virtual ID document" containing only a key to a database (e.g., an ID number printed on a sheet of paper). There are certain advantages of this approach: a negligible technical effort for issuing and delivery of an ID document, and ease of ID document updating and revocation. However, this solution has also substantial disadvantages:

- The service operator (as well as a cyber criminal that breaks into the system) is aware of all activities concerning identity verification. This violates the data minimization principle of security engineering.
- In the case of a system failure (e.g., due to a cyber attack, telecommunication infrastructure failure, etc.), all activities requiring identity verification are suspended.
- There is a non-negligible communication latency and overhead.
- Responding to a query should be preceded by checking the rights to get an answer. This is hard if the verifier has no eID.

Option 1: Local Use An eID token holding crucial ID data and verifiable for its originality has substantial advantages in certain situations:

- The presence of an eID is indirect proof of the physical presence of its holder.
- Identity and data verification does not require online access despite strong guarantees originating from the eID issuer.
- Interaction with a chip may enable biometric verification without the involvement of any external database. The biometric data obtained by the reader can be directly compared with data stored in the eID. Consequently, any security breach in the system would not expose the biometric data of the whole population.

Option 2: Remote Use In this case an eID serves as a secure cryptographic token for remote authentication. The advantages of this approach are as follows:

- An eID is involved in the authentication process as a "what you have" and a "what you know" component, since typically an eID requires us to provide the user's activation password.
- In remote authentication it is hard to check that the user is really on the other side of the line. An eID serves as indirect proof of this presence: its participation in a protocol execution is checked in a cryptographic way, while one can reasonably assume that the owner of the eID would not borrow/give it to a third person.

5.1.2 Actors and Scenarios

Determining the actors of the process involving an eID enables us to see the variety of solutions associated with this term. Below we list a number of cases:

- **Owner–eID–Reader:** Example: border control booth.
  A traveler presents their ePassport to an eGate. No border control officer is involved in this process (unless additional processing is needed or the traveler needs assistance). The holder of the ePassport is involved in the protocol as biometric data are scanned and compared with the data stored in the ePassport. In some data protection scenarios, providing the password of the ePassport holder is required.
- **eID–Reader:** Example: vending machine with age verification.
  A vending machine selling stuff for adults only (alcohol, etc.) has to verify the age of the buyer. The process must work smoothly without annoying the buyer. The protocol should guarantee that a genuine eID is involved, and that this eID has been issued for a person that has reached the legal age. No other data about the eID holder should be revealed.
- **Owner–eID–Reader–Terminal:** Example: submitting claims to an e-government authority.
  In this case the reader serves as a man-in-the-middle transmission device located between the eID and the remote terminal. It provides technical means to establish a channel between them, while the essential part of the protocol is run between the eID and the terminal. The second role of the reader is to enable authorization of the eID owner to perform some actions with the eID—such as signing a digital document submitted to the terminal.
- **eID–Reader(s)–eID:** Example: vehicle to vehicle communication.
  In the near future a new application area may emerge where autonomous devices, such as vehicles, communicate directly and make decisions about their behavior. In many cases this will require strong authentication. For instance:

  – One has to recognize non-authorized devices that may work in a malicious way or merely misuse the protocol for their own profit.
  – It might be necessary to identify traffic rules violators and vehicles responsible for traffic accidents.

  At the same time the privacy of the authenticating parties should be protected.
- **eID–PC:** Example: user presence verification.
  In certain cases a user operates a terminal and apart from the initial authentication we need continuous verification of their presence. This concerns cases such as operating a terminal for performing certain financial operations or safety-relevant operations (e.g., in railway or air traffic). We have to make sure that an unauthorized person will not be able to perform any action when the authorized person leaves the terminal without closing a session.
  Another major application area is medical services and, in particular, authenticating medical records presented to an insurance company. Evidence of presence

created together with a patient's eID is required in order to witness a medical transaction.

- **Owner–eID–PC:** Example: signature creation.
  An eID may enable us to create digital signatures that are legally equivalent to handwritten signatures. In this case the eID holds signing keys and creates signatures provided that the eID owner explicitly gives their consent.

5.1.3 Goals of Protocol Execution

The execution of a protocol with an eID may serve different purposes. Below we list those that are either currently in operation or claimed as a near target:

- **identity data confirmation**: confirming data regarding the owner that are either printed on the eID or otherwise available for the verifier (e.g., a face image of the eID holder),
- **attribute confirmation**: an eID shows no identification information (including indirect ones such as the eID's serial number), instead it presents (authenticated) attributes of the eID owner,
- **key exchange**: the eID and the terminal establish a mutually authenticated secret key for establishing secure communication,
- **authentication and proof of eID's presence**: a volatile or non-volatile proof of the eID presence, in the second case the proof can be used against third parties,
- **authentication and proof of user's presence**: we check that the owner holds an eID participating in a protocol (the difference with the previous case is that the eID holder is involved in the process, for instance through biometric or password verification),
- **terminal authentication**: authentication of the (remote) terminal concerning its identity and access rights,
- **confirming owner's consent**: providing an (implicit) proof that the owner of the eID has agreed to something.

5.2 Threats and Security Requirements

5.2.1 Assets

The first type of asset concerns secret and private information:

- **password:** a password or a PIN number used to activate the electronic layer of the eID or used in a password authentication protocol,
- **secret key:** a secret key used for eID or terminal authentication,

- **personal data:** data about the document owner including name and other identification attributes that are printed on the physical layer of the identity document,
- **sensitive data:** data stored in the electronic layer of the eID, which are not present on the physical layer (typically, biometric data about the document owner: iris scan, fingerprint, etc.).

On the other hand, there are assets related to a property or state of a protocol execution. The following assets fall into this category:

- **authenticity of data:** we consider the integrity and originality of the data stored in the memory of the eID,
- **authenticity of eID:** the integrity and originality of the eID as a device,
- **confidentiality of communication:** preventing access of unauthorized parties to data exchanged over communication channels established with an eID,
- **access limited to authorized terminals:** limiting access to sensitive data to authorized terminals,
- **privacy of eID usage and location:** confidentiality of data regarding eID usage, including for instance the identity of terminals involved in an interaction, the interaction time, and the data exchanged.

  Note that in some scenarios a proof that the document interacted with a reader is required. However, in general user privacy should be protected and access to the data should be confined to authorized parties.
- **robustness:** an eID must work properly regardless of previous, possibly faulty, executions.

5.2.2 Threats

A major threat against identification documents is **forgeries**. An attacker may attempt to create a fake eID that behaves like a genuine one and presents data that will be accepted just like in the case of interaction with a genuine eID.

An adversary may attempt to **clone** an eID.

A clone can be used by a person with a similar appearance as well as in remote applications, unless protection via biometric authentication or password verification continues to work effectively. In particular, eID cloning may enable identity theft with profound consequences. Note that breaking the secret key of an eID may be regarded as a partial forgery or cloning.

Another threat is using an eID without the **owner's consent**. This is particularly likely in the case of wireless communication, where interaction with an eID can be initiated even without the owner's knowledge. Typically, an eID is secured via a password either entered manually by the owner or read optically from the document's surface.

Since the password has usually low entropy (e.g., 4–6 digits in the case of a PIN) it might be guessed by an adversary. A common protection mechanism is to block an eID after a limited number of failed attempts, but this would allow an adversary to mount a denial of service attack—especially in the case of wireless communication. In the case of non-blocking passwords, an adversary can try to guess the correct password using a brute-force dictionary attack. If the protocol execution is artificially slowed down so that a single interaction takes, say, 2 s, the threat is only reduced.

On the other hand there is a threat of offline attacks, where the attacker analyzes transcripts of communications between the eID and honest as well as dishonest readers. Another scenario is simply leaking passwords from a malicious reader.

A different kind of threat comes from malicious terminals that interact with the electronic layer of an eID and attempt to receive more data than allowed. This may concern eID identity (e.g., if no password has been provided by the eID holder) or sensitive data such as biometric data (if the terminal has not been properly authenticated). In any case we are talking about **escalating access rights** via bypassing the access control mechanism employed by the eID. Note that breaking the secret key used for terminal authentication is essentially a step of such an attack.

Malicious terminals as well as parties observing communication may use an interaction with an eID to convince a third party of their location and activities. In the weaker form of **location and activity tracing** an attacker derives these data for its own purposes, e.g., by observing interaction or initiating a session with the purpose of learning the identity of eIDs within its range.

An adversary can also try to extract personal data by **eavesdropping** on secure and authenticated communication between an honest eID and an honest reader/terminal. To perform this kind of attack, the adversary has to break the confidentiality of the communication channel or hijack a session already established.

In Table 5.1 we summarize the dependencies between the above assets and threats.

5.3 Cryptographic Protocols for eIDs

In this section we present some cryptographic protocols that are implemented and used in various existing eID solutions and which tackle the problems described above.

5.3.1 Preventing eID Forgeries

A simple approach adopted, among others, by the ICAO (International Civil Aviation Organization) (see [291]), is to store all relevant data $D_1, \dots, D_n$ in the electronic layer, compute $h := \mathrm{Hash}(\mathrm{Hash}(D_1), \dots, \mathrm{Hash}(D_n))$ and an electronic signature of the eID issuer on h. Then, it is possible to check the signature and verify that the data $D_1, \dots, D_n$ are authentic. (Note that it is also possible to verify

Table 5.1 Threat-asset dependency table

Threat/asset	Password	Secret key	Personal data	Sensitive data	Authenticity of data	Authenticity of eID	Communication confidentiality	Limited access	Usage/location privacy
eID forgery					✓	✓			
eID cloning	✓	✓	✓	✓		✓			?
Owner's consent	✓		✓	✓			?	✓	✓
Escalating access rights			✓	✓					✓
Location and activity tracing									✓
Eavesdropping	✓		✓	✓			✓	✓	✓

the signature when only some D_i are given, while for the remaining values D_j only $\mathrm{Hash}(D_j)$ is presented.) What is more, this data can be compared with the data stored in the physical layer of the document.

The solution presented is called *Passive Authentication* (PA).

Unfortunately, it provides no protection against eID cloning as all digital data stored in the eID can be presented to a reader in the course of a normal interaction.

5.3.2 Enforcing Owner's Consent

An access control mechanism based on an activation password can be deployed in order to protect against using an eID without the owner's consent, Unfortunately, it can turn out to be ineffective. There are at least two attack scenarios:

1. an adversary is communicating directly with an eID,
2. an adversary is eavesdropping on an interaction between honest parties.

In the former case, the adversary may try all possible passwords by starting an interaction with the eID. However, the eID may increase its response time to say 1–2 s in order to slow down the attack.

Moreover, the password may include a code printed on the eID in a *machine readable zone* (MRZ) and optically read by the reader. This code may have much higher entropy than human memorizable passwords.

Because of this latter scenario, the password cannot be transmitted in a form that would enable the adversary to learn it and reuse it later to communicate with the eID. This has been taken into account in a solution adopted by the ICAO called Basic Access Control (BAC). In this protocol a fixed password corresponding to an eID is used (1) to encrypt random nonces used to derive a session key, and (2) to create MACs authenticating the nonces. If on any side an incorrect password is used, then the protocol will fail. The password is derived from data scanned optically from the MRZ area—therefore an ePassport must be *shown* by its owner. Unfortunately, BAC allows an offline attack, i.e., given a transcript of a communication an adversary can apply a brute-force dictionary attack to learn the password [375].

The successor to BAC—Password Authenticated Connection Establishment (PACE) introduced by the German Federal Office for Information Security (BSI) in 2008—changes the situation. It becomes impossible to verify a password guess given only a transcript of a communication between an eID and a reader. The PACE protocol consists of four main phases (see Fig. 5.1 for details):

1. sending a ciphertext $\mathrm{Enc}(K_\pi, s)$ to the reader, where s is random and the key K_π is derived from the password,
2. using a mapping function based on the secret s to derive new parameters $\hat{\mathbb{G}}$ with a new generator $\hat{g}$ for the Diffie-Hellman protocol,
3. applying Diffie-Hellman key exchange to derive a master secret key K,
4. exchanging message authentication tags T_R, T_C based on a key derived from K.

eID:		reader:
password π, parameters $\mathbb{G}$		password π, parameters $\mathbb{G}$
	Protocol Execution	
$K_\pi = \mathsf{KDF}(\pi)$		$K_\pi = \mathsf{KDF}(\pi)$
choose $s \leftarrow \mathbb{Z}_q$		
$z = \mathrm{Enc}(K_\pi, s)$	$\xrightarrow{z}$	
		$s = \mathrm{Dec}(K_\pi, z)$
	Mapping Function	
$\hat{\mathbb{G}} = \mathbf{Map}(\mathbb{G}, s)$		$\hat{\mathbb{G}} = \mathbf{Map}(\mathbb{G}, s)$
choose $y'_C \leftarrow \mathbb{Z}_q^*$		choose $y'_R \leftarrow \mathbb{Z}_q^*$
$Y'_C = \hat{g}^{y'_C}$	$\xleftarrow{Y'_R}$	$Y'_R = \hat{g}^{y'_R}$
abort if $Y'_R = Y_R$	$\xrightarrow{Y'_C}$	abort if $Y'_C = Y_C$
$K = {Y'_R}^{y'_C}$		$K = {Y'_C}^{y'_R}$
$K_{\mathrm{Enc}} = \mathsf{KDF}_{\mathrm{Enc}}(K)$		$K_{\mathrm{Enc}} = \mathsf{KDF}_{\mathrm{Enc}}(K)$
$K_{\mathsf{Mac}} = \mathsf{KDF}_{\mathsf{Mac}}(K)$		$K_{\mathsf{Mac}} = \mathsf{KDF}_{\mathsf{Mac}}(K)$
$K'_{\mathsf{Mac}} = \mathsf{KDF}(K, 4)$		$K'_{\mathsf{Mac}} = \mathsf{KDF}(K, 4)$
if $\mathsf{Verify}(K'_{\mathsf{Mac}}, (Y'_A, \mathbb{G}), T_R) = 0$, then abort	$\xleftarrow{T_R}$	$T_R = \mathsf{Mac}(K'_{\mathsf{Mac}}, (Y'_C, \mathbb{G}_1))$
$T_C = \mathsf{Mac}(K'_{\mathsf{Mac}}, (Y'_R, \mathbb{G}_1))$	$\xrightarrow{T_C}$	if $\mathsf{Verify}(K'_{\mathsf{Mac}}, (Y'_B, \mathbb{G}), T_C) = 0$, then abort
$\mathsf{key} = (K_{\mathrm{Enc}}, K_{\mathsf{Mac}})$		$\mathsf{key} = (K_{\mathrm{Enc}}, K_{\mathsf{Mac}})$

Fig. 5.1 PACE protocol

There are two variants of the mapping function specified in the ICAO standard: Generic and Integrated Mapping. In the former case, $\hat{g} = h \cdot g^s$, where h is the secret key generated using a standard Diffie-Hellman protocol for parameters $\mathbb{G}$. In the latter case, $\hat{g} := \mathrm{Hash}(s, r)$ where r is a random number chosen by the reader.

The security of the Generic Mapping version of the protocol is based on the following argument. We can create a virtual protocol, where h is not derived by the Diffie-Hellman protocol but is a random element. Such a change cannot be detected by an adversary performing an offline attack due to the hardness of the Diffie-Hellman Problem. However, in the virtual protocol all data exchanged are stochastically independent of s. Therefore, it is impossible to derive any information about s. A similar argument applies to Integrated Mapping. Note that $\hat{g}$ never occurs in the communication and the values Y'_C, Y'_R are uniformly distributed, as the group used has a prime order. The only relation to $\hat{g}$ is hidden in the way the protocol partners derive K. However, again we can consider a virtual protocol where K is replaced by a random element. The change is not observable to the attacker even if they learn the key K. Some partial security proofs for the PACE protocol were presented by Bender et al. [72] and for the PACE Integrated Mapping by Coron et al. [153].

An important feature of password-based schemes is how to respond to authentication attempts with a wrong password. In any case, a failed attempt is information indicating incorrectness of the password. The important point is that no other information should be revealed to the attacker. Note that PACE has this property: first, the ciphertext of a random number s does not reveal any information about the password dependent encryption key. Second, the random challenges exchanged within the final Diffie-Hellman key exchange are stochastically independent of the password. The password dependent message T_C is sent *after* positive verification of T_R—so T_C is not transmitted, if the password is incorrect!

5.3.3 EID Authentication and Preventing Cloning

The threat of eID cloning is also called an *impersonation attack*, since a user of a cloned eID may impersonate its legitimate owner. This kind of attack can be easily performed in the case of passive authentication, as all data in an eID are also available outside it. An obvious remedy to this problem is to use an eID public key as part of the data authenticated by the document issuer, store the corresponding secret key in the memory of the eID, and run an authentication protocol based on these keys. Obviously, this solution makes sense only if it is infeasible to export the secret key from the eID.

The ICAO standard for Machine Readable Travel Documents specifies two cryptographic protocols that can be used to authenticate a chip. The first is called Active Authentication (AA) and relies on the challenge-response paradigm. The terminal sends a challenge to the ePassport, which responds with a signature under this challenge. Finally the terminal verifies the signature with respect to the public key stored in the authenticated data read from the ePassport's memory.

The second solution called Chip Authentication v.2 (ChA v.2) was introduced by the BSI as part of the so-called Extended Access Control (EAC) protocol stack, where within the same protocol the rights of the terminal are checked. ChA v.2 is a static Diffie-Hellman protocol: it is simply the Diffie-Hellman key exchange protocol where the challenge from the eID is replaced by its public key, say $y = g^x$. As in the Diffie-Hellman key exchange protocol deriving the shared key is possible provided that one knows the discrete logarithm of the challenge, the ability of the eID to derive the secret key serves as evidence that it knows the secret key x.

The main advantage of ChA v.2 is that it generates a new session key that can be used to secure communication between the eID and the terminal. Note that the protocols discussed in the previous subsection establish a secure channel between the eID and a reader, which is not necessarily part of the terminal. For example, if the eID is used for online activities the card reader is at best a part of the user's system.

The main disadvantage of both solutions is that they require additional computations that are expensive, i.e., exponentiations. For this reason the ICAO adopted a protocol called PACE with Chip Authentication mapping (PACE-CAM) that

eID:		reader:
choose $y_C \leftarrow_R \mathbb{Z}_q^*$		choose $y_R \leftarrow_R \mathbb{Z}_q^*$
$Y_C = g^{y_C}$	$\xleftarrow{Y_R}$	$Y_R = g^{y_R}$
abort if $Y_R \notin \langle g \rangle \backslash \{1\}$	$\xrightarrow{Y_C}$	abort if $Y_C \notin \langle g \rangle \backslash \{1\}$
$h = Y_R^{y_C}$		$h = Y_C^{y_R}$
$\hat{g} = h \cdot g^s$		$\hat{g} = h \cdot g^s$

Fig. 5.2 Generic mapping of PACE

combines PACE with authentication of the eID [298]. The initial part of the protocol is exactly the same as in the case of PACE Generic Mapping (see Fig. 5.2)—which is important for reasons of backwards compatibility and the costs of upgrading the protocols. However, in the final part of the protocol the eID must show the discrete logarithm of its challenge Y_C with respect to its public key pk_C as the generator. Note that the eID cannot pass the protocol without being able to derive h and this requires knowledge of y_C such that $Y_C = g^{y_C}$. As the eID has to present a w such that $Y_C = pk_C^w$, it follows that $pk_C = g^{y_C/w}$ where y_C and w are both known to the eID. Consequently, after checking that $Y_C = pk_C^w$ the terminal can safely conclude that the eID knows the discrete logarithm of pk_C.

Leakage Resistance The protocol PACE-CAM may serve as an example of a possible defense against malicious or poor implementations. One of the critical threats for this protocol is the danger of leaking the discrete logarithm of pk_C (thereby enabling cloning of the eID concerned).

Assume that during protocol execution the eID first chooses y_C and later computes $w = y_C/sk_C$ (as in [73]). Then obviously the key sk_C is exposed in case a weak random number generator has created y_C. In [257] the computations are performed in a slightly different way. Namely, $Y_C = pk_C^w$, for w chosen at random, and there is no computation of w during the final stage. The most important feature is that the value y_C does not appear at all. However, the element h has to be computed by the eID in a slightly different way: instead of computing $h = Y_R^{y_C}$, the eID computes $h = (Y_R^w)^{sk_C}$. The exponentiation with sk_C can be executed in a separate hardware zone which performs no other operation. In this way the secret key sk_C is essentially secure even if the adversary has access to all values outside this hardware zone.

Particular care is needed in the case of protocols using signatures such as DSA and Schnorr. In this case leakage of ephemeral random values leads to leakage of the long time secret keys and it seems that there is no simple remedy for this problem.

5.3.4 Authenticating the Terminal and Its Rights

A terminal that has guessed or knows the activation password of an eID may attempt to read sensitive data from the eID. Since it knows the activation password it can interact with the eID, but should not necessarily be able to access all data. This problem is solved by Terminal Authentication v.2 (TA v.2), which in fact is a standard challenge-response protocol: the eID generates a random nonce and the terminal responds with a signature of this nonce and a fingerprint of an ephemeral public key—this public key is the challenge for the static Diffie-Hellman key agreement executed as part of ChA v.2. For the sake of signature verification the terminal presents a certificate of its public key. The certificate declares in particular which data from the eID the terminal is allowed to read. To verify the authenticity of the certificate, the document verifies a chain of certificates supplied by the terminal. The first supplied certificate is verified using the root of trust, i.e., a public key of the issuer stored inside the eID (for details see Sect. 5.4).

5.3.5 Proof of Interaction

In this scenario we consider a malicious reader/terminal that tries to sell a transcript of communications or interactions to a third party. Such a transcript may contain valuable information about the document owner (e.g., location, services used). The best protection one can achieve is that a reader/terminal can create a protocol transcript (including values normally available only to the reader/terminal) that is indistinguishable from transcripts originating from real executions. This should also concern executions where a reader/terminal deviates from the protocol description. Then a transcript presented by a reader/terminal has no value to a third party.

In general, any protocol implemented in an eID should support deniability: neither the eID nor the terminal/reader should be able to prove that an interaction with the eID has taken place and had a given form. Note that AA provides a strong proof of interaction for third parties, while for ChA v.2, PACE, PACE-CAM the deniability property is fulfilled. For TA v.2 a signature collected from the terminal is a proof of interaction (with an unspecified eID).

5.3.6 Passive Tracing

Protocols such as PA enable tracing—an eID responds with explicit identification data. The situation is slightly better for BAC. However, once the adversary learns the secret key used by the eID of the traced person to set up connections, the adversary can make trial decryptions of all recorded communications and identify those of the traced eID. The situation is different for protocols such as PACE and PACE-

CAM—where identity information is revealed after establishing a secure channel. Even password related data are transmitted in a very careful way—via a ciphertext of a random value s. For ChA v.2 the situation is slightly more complicated: if PACE is executed first, then ChA v.2 is secured by the secret channel established by PACE. Otherwise, transmission of the public key used for static Diffie-Hellman reveals the identity of the eID. This was one of the reasons for designing ChA v.3, where the protocol starts with the regular Diffie-Hellman key exchange.

5.3.7 Eavesdropping

While most of the discussed protocols run key exchange protocols and automatically support confidentiality of the established session, there are some subtle issues concerning for instance the threat of hijacking of a session. Especially if the protocols are executed one after another, the protocol partners need to be sure that in the meantime an adversary has not taken over the communication on one side. For instance, TA v.2 and ChA v.2 are coupled by a signature created during TA v.2 execution for a fingerprint of the terminal's ephemeral key used during execution of ChA v.2. Similarly, while PACE is not resilient to MiTM attacks (by an adversary knowing the password), it seems to be infeasible to run an active attack enabling the eID and the reader to establish the shared key K, so that it would be known also to the adversary.

Summary

Table 5.2 depicts threats addressed successfully by the protocols discussed above.

Table 5.2 Threats addressed by the discussed protocols

Threat/protocol	PA	PACE	ChA v.2	TA	PACE-CAM
eID forgery	✓		✓		✓
eID cloning			✓		✓
Lack of owner's consent		✓			✓
Unauthorized data access				✓	
Location and activity tracing		✓			✓
Eavesdropping		✓	✓	Indirectly	✓

5.4 PKI

In standard situations the Public Key Infrastructure (PKI) is built on top of trust relations. A user chooses *roots of trust*—entities that are trusted by him. A root of trust can delegate trust to other entities, creating a *chain of trust*. In the end, the user will accept/trust any entity that can prove itself to be a member of a chain of trust. To confirm trust relations, the entities issue certificates signed with their secret keys.

A similar approach could be implemented in the eID scenario, but we face substantial technical problems. The hardware chip of an eID might be severely limited in computational power and memory size. Storing many roots of trust and verifying long chains of trust would immediately make an eID solution quite inefficient. For this reason a simplified approach has been adopted:

- the root of trust is set to one entity, namely *the country verifying certificate authority* (CVCA),
- the CVCA delegates its rights to domestic and foreign document verifiers that delegate rights to terminals—so the trust chains have a length of at most 2,
- certificates have a form enabling easy verification on hardware with limited resources (*card verifiable certificates*, CVC).

This PKI allows for interoperability between passports issued by different countries. However, in order to inspect a passport, a document verifier must be trusted by the document owner's CVCA, which requires cooperation between two countries. Practically it turns out to be a problem.

Another serious problem is revoking certificates. In a standard situation it is implemented using certificate revocation lists (CRL) and an online certificate status protocol (OCSP). In the case of eID systems this may lead to technical problems:

- an eID has no memory available for storing CRLs,
- an eID has no direct access to the OCSP server,
- verification may turn out to be too slow from a practical point of view,
- an eID has no internal battery and therefore no internal clock. It can only store the time last seen, so a terminal may authenticate itself with an outdated CRL.

The solution implemented in ePassports is to use short-term certificates for terminals. Thereby, a terminal that has to be revoked simply does not receive a new certificate.

5.5 Challenges for eID Systems

In this section we focus on some fundamental issues that do not concern eIDs directly, but nevertheless are crucial for their success in practice.

Deployment of eIDs Due to financial costs, the deployment of eIDs has to be a continuous process, where the production capacities are used evenly over time.

Hence invalidating eIDs and replacing them with new ones on a large scale might be infeasible. Unfortunately, one cannot exclude that a security flaw or an exploit will be found after deployment. Therefore it would be reasonable to design in advance pragmatic strategies for such situations which are not based on replacement.

As an eID is issued for a long period of time (typically 10 years in the case of personal ID documents and passports), inevitably many generations of eIDs will have to coexist and interact with the existing systems. On the other hand, if a user gets a new eID replacing the expired one, they should be able to continue all activities as the same person. This might be a problem, since transferring the signing keys from the old eID to a new eID might be technically impossible or forbidden due to, e.g., requirements for "secure signature creation devices".

Malicious Provider/Document Issuer Before an eID is given to its owner it has to go through procedures implemented by the hardware and software provider and the document issuer responsible for personalization of the eID. These procedures may be not strict enough and/or depend on the internal policies of these parties. This can open doors for implementing trapdoors or intentional creation of security weaknesses.

A malicious hardware provider can implement secret backdoors that reveal the secret keys of the eID (created during personalization) or provide a means to trace the eID. Similar backdoors can be implemented in software by the document issuer. A malicious issuer can still infer information about users, even if the software is somehow protected (e.g., installed by a different authority). In many cases the secret keys used by the cryptographic protocols can be generated on-card (i.e., by the eID without revealing the secret key to anyone). However, the owner of the document has no *guarantee* that this is how the keys have been generated. What is more, in many cases the secret keys cannot be generated on-card since secret system keys have to be involved.

A different attack technique is to use a flawed random number generator that would allow an attacker to predict the secret key generated by the document and break any protocol based on randomness generated by an eID. The problem can be also caused by a flawed software library. This was the case for the Estonian eID, where many RSA moduli were created in a way that made their factorization easy.

Interoperability While in the area of biometric passports we have to deal with a de facto worldwide standard, in the case of personal ID documents there are a variety of different solutions and concepts. At the same time, the expectation is that an eID may interact with any system and communicate with any reader. This is costly and hardly possible from the technical point of view. The problem has been recognized by the European Union as an obstacle to the internal market. The eIDAS regulation [546] attempts to find a solution to this problem. The approach chosen by eIDAS is as follows. Assume that an eID J attempts to connect to a terminal T in a third country. Then:

- the terminal T redirects the connection request of J to an online authentication service A of the country of origin of J,

- A runs the verification process,
- after a positive answer from A the terminal T establishes a connection with J.

Unfortunately, this approach has limited usability:

- it does not work for the offline eID–reader scenario,
- a reader must run all protocols notified in the EU,
- when the authentication service learns that the eID is active at some place. for some protocols the online authentication service can learn the session key and be able to eavesdrop on the communication.

Sometimes cooperation between eID providers is limited, e.g., due to different approaches to personal data protection (such as for the EU versus the US).

User–eID interaction While declaratively users pay attention to their own security, the standard behavior is to trade security for convenience. A good eID system should take this into account. On the other hand, a user has to be given a minimal level of control over their eID. This should include in particular the possibility to:

- temporarily block their own eID,
- get a record of their former activities.

In existing systems, there are limited possibilities to suspend or invalidate an eID based on inserting entries into a central database. This is not enough to prevent lunchtime attacks, where an eID is seized by an adversary for a short time. To keep track of eID activity one could deploy for instance a transaction counter or some more sophisticated solution (see, e.g., [349].

5.6 Future Directions

Composability The main goal of eID protocol designers should be composability. That is, given a limited number of cryptographic procedures it should be possible to implement all required functionalities and even leave room for new applications. This approach not only ensures that an eID can be created with cheaper hardware (i.e., less code means less memory used), but also makes it easier to create and reuse a formal security analysis.

Extensions The protocol stack implemented in an eID should allow extensions. In particular, it should be possible to build new security features on top of the existing stack.

Simple Protocol Changes If a security flaw is found, the protocol designers should focus on simple fixes that make only small changes to the existing protocols. This would not only simplify the security analysis of the new protocol but also speed up the certification process. What is more, the same approach should be taken in the case of an extension to existing protocols. A good example is the PACE-CAM protocol, which is only slightly different from PACE, but significantly improves it.

Acknowledgments Both authors "Lucjan Hanzlik and Mirosław Kutyłowski" have been supported by Polish National Science Centre, project OPUS no 2014/15/B/ST6/02837.