Journal of Universal Computer Science, vol. 21, no. 3 (2015), 419-439
submitted: 30/6/14, accepted: 28/2/15, appeared: 1/3/15

Restricted Identification Secure in the Extended Canetti-Krawczyk Model

Lucjan Hanzlik
(Wrocław University of Technology, Poland
lucjan.hanzlik@pwr.edu.pl)

Mirosław Kutyłowski
(Wrocław University of Technology, Poland
miroslaw.kutylowski@pwr.edu.pl)

Abstract: In this paper we consider restricted identification (RI) protocols which enable strong authentication and privacy protection for access control in an unlimited number of domains. A single secret key per user is used to authenticate and derive his identity within any domain, while the number of domains is unlimited and the scheme guarantees unlinkability between identities of the same user in different domains. RI can be understood as an universal solution that may replace unreliable login and password mechanisms. It has to secure against adversaries that gather personal data by working on a global scale, e.g. by breaking into one service for getting passwords that a user frequently re-uses at different places.

We consider security of an extended version of the Chip Authentication Restricted Identification (ChARI) protocol presented at the 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2012). We preserve the features of ChARI (avoiding the critical security problems of group keys in the RI solution deployed in the German personal identity cards), but provide security proof in the well-studied Canetti-Krawczyk model (such a proof has not been provided for ChARI). Our extension has similar computational complexity as the original ChARI protocol in terms of the number of modular exponentiations.
Key Words: Restricted Identification, Chip Authentication, ChARI, Diffie-Hellman key agreement, sector identity, unlinkability, eCK model, personal identity card
Category: D.4.6, E.3, K.6.5

1 Introduction

1.1 Practical challenges of authentication

User authentication is one of the key problems of emerging large scale omnipresent IT systems. The solutions deployed so far are borrowed directly from isolated island systems. Consequently, the mechanisms tailored for the early age of multiuser systems, such as authentication based on the user's login and password, are applied in a way that creates severe data security problems. One of the most spectacular problems is re-using the same passwords in different systems by a user that is forced to remember more and more passwords. However, in this situation stealing passwords from one system automatically opens access to the accounts in other (may be better protected) systems. The attacks like this lead to massive scale security breaches and illegal wholesale trade of authentication data.

The problem has at least two different dimensions. First, the authentication procedures should be user friendly. Today they are not, for the sheer reason that a normal user cannot remember several passwords (even worse, sometimes the user is forced to change the passwords frequently). An immediate consequence is that the passwords are stored unprotected or the same password is used in different systems. There are attempts to deal with this problem by federated identity management, but the price for improved usability is that there is a trusted party that gathers knowledge about user accounts and can impersonate the user.

The second problem is linkability: with the growing number of services, it is becoming more and more sensitive which services are used by a user. Analyzing these data is one of the major privacy threats today. In particular, it can be used for creating quite accurate user profiles; the resulting knowledge can be misused in multiple ways.

1.2 Authentication with restricted identification

The restricted identification (RI) [BSI 2013] is an idea to provide strong authentication for multiple systems with just a single secret key that may be implemented on a secure cryptographic device. Moreover, it has to hide users' real identities in the following sense:

- For each domain a user has a separate anonymous identity (which is a public key). This domain identity can be derived on-the-fly from the domain parameters and the user's private key.
- For a given domain, the user cannot acquire more than one identity.
- For two domains and two identities in these domains, it is infeasible to determine whether they correspond to the same user. This property holds even if the adversary performs authentication protocols with the user or users holding these identities.

Note that the main difference between the RI and anonymous credentials protocols is that for RI there is a unique permanent identity for a user in a given domain. In almost all application scenarios we need such a limitation. For instance, if the anonymous identity is used for activities in a social network it is useful to prevent Sybil attacks, where a malicious user appears under different IDs. Even more acute is the situation on auction servers: a dishonest participant may gain trust, then make frauds and disappear with money. Later he may use a new ID and start the game from the beginning. However, with the RI authentication this is impossible: an anonymous participant will be linked with the previous misconduct for the lifetime. This is perhaps the most efficient mechanism to force the users to behave honestly.

Applying RI as the standard mean of authentication would solve all security and usability problems related to the current login+password practice. It would be particularly convenient for the users as they would be obliged to protect just one key instead of potentially unlimited number of passwords and keys for diverse systems. As the users have already learned to protect such documents as credit cards or personal identity documents, we may expect the same proper behavior in case of restricted identification.

1.3 Choice of devices implementing RI

Even the best RI scheme cannot provide security unless we can guarantee that the user holds exactly one private key used for RI authentication. A simple way to ensure that only one key is obtained by a user is to implement the key on the official personal identity document. As generally a person obtains exactly one identity document and issuing the official personal identity documents is under a strict control, we could be fairly sure that one person cannot use diverse private keys for authentication with RI. However, we have to guarantee the following properties:

- Preferably, the architecture of the microcontroller should guarantee that there is no room for installing more than one RI private key. Presumably, changing the value of the private key should be technically infeasible.
- The user should have the full control over the private key, just as in the case of signing keys for electronic signatures. In particular, it is preferable to create the keys within the microcontroller.
- During the authentication procedure, the server should check that it is interacting with a valid identity document but without learning the real identity of the document.

The first two conditions can be fulfilled in a standard way by applying appropriate hardware. The third condition is harder and requires a very careful design of cryptographic protocols.

1.4 Electronic personal identity documents

Nowadays many countries have decided to replace traditional personal documents with identity cards equipped with an electronic chip (for more details refer to the handbook [Fumy and Paeschke 2010]). The primary goal is to protect against forgeries, however there is an opportunity to use the identity document as a secure personal device for electronic communication. It may provide new functionalities regarding e-Government services. To name only a few use cases we may apply e-ID's to submit tax forms (e.g. in Netherlands), access online services offered by public authorities, create digital signatures (Lithuania) or to provide simple credentials (e.g. age verification for cigarette vending machines in Germany). The use of e-ID's in remote voting has also been considered (and even implemented in a controversial way in Estonia). Electronic chips are also deployed in so called Biometric Passports deployed worldwide according to ICAO

(International Civil Aviation Organization) standard for machine readable travel documents (MRTD) [ICAO Doc 2008].

A solution related to RI on personal identity documents was first implemented in Austria as so-called *Bürgerkarte*. However, the separation of identity information from authentication has appeared long time ago in [Flinn and Maurer 1995]. *Bürgerkarte* is a system of passwords computed with a symmetric algorithm from the citizen's personal number. However, only public administration domains were used and replay attacks are possible due to the use of static passwords generated by symmetric algorithms.

The restricted identification in the present form has been introduced together with the new German identity card (*neuer Personalausweis* or *nPA*). Here, the concept of separating authentication in different domains evolved into a protocol called Restricted Identification (RI) [BSI 2013] and introduced by the German Bundesamt für Sicherheit in der Informationstechnik. The RI protocol has been included as one of the key components of the e-ID architecture implemented on a native smart card. Apart from the general properties mentioned so far the following features had to be fulfilled during the design of nPA:

mutual authentication: the e-ID document should be convinced that it is communicating with a certified terminal. On the other hand, the terminal must be sure that the keys used for authentication are from a valid e-ID document.

feasibility on smart cards: the RI protocol should be executed in a reasonable time on smart cards (including both computation and communication time). Moreover, memory usage for private keys, certificates and the program code should be low and take into account smart card limitations.

cross-domain anonymity: two (and more) cooperating service providers from two different domains communicating with some e-ID's by means of RI cannot determine if they are interacting with the same e-ID or with different e-ID's.

untraceability: an adversary should not learn the domain identity of a user by eavesdropping on the communication.

deniability: a protocol transcript cannot be used as a proof of communication against third parties.

seclusiveness: no group of malicious and colluding users can create a valid key material for a new user.

1.5 Related work and group key problem

The German RI solution [BSI 2013] consists of three protocols: Terminal Authentication (TA), Chip Authentication (ChA) and Restricted Identification (RI).

The TA protocol authenticates the terminal of a domain, i.e., a domain specific group generator is extracted from the certificate issued by a Certificate Authority and the terminal sends a signed ephemeral public key. Then the ChA protocol is executed in order to authenticate the card (chip). ChA is basically a static Diffie-Hellman authentication where the public key of the card is computed using a secret key from the card and the terminal's ephemeral key. Both sides establish a secret shared key in order to encrypt the forthcoming communication and to verify whether the ephemeral key came from the genuine terminal. Finally, the RI protocol is executed where the card computes his domain specific pseudonym using the private key x_{RI} devoted exclusively for RI. So, if g_{dom} is the domain specific group generator, then $H((g_{dom})^{x_{RI}})$ is the smart card's domain specific pseudonym (H is a cryptographic hash function). The pseudonym delivered during execution of the RI protocol is only cross-checked against a blacklist.

1.5.1 Group keys and their security problems

During the ChA authentication the terminal verifies that the card knows one of secret keys issued by a certification authority. These keys are called *group keys*. Note that the key from ChA must be shared between multiple cards, since otherwise the group key could be used to link user's domain specific identities. This leads to a problem discussed in [Hanzlik et al. 2012] (see also [Dagdelen 2013] and [Poller et al. 2012], as well as paper [Feld and Pohlmann 2011] ignoring this issue). Namely, it is risky to assume that a group key will never be leaked. For instance, we may expect that a very powerful adversary eventually breaks into a single smart card and reads its internal memory. As in this construction there is no connection between the group key and the key used to compute the domain specific identity, an adversary holding a group key can forge new identities, which cannot be blacklisted. The only solution is to revoke all cards using the leaked group key, also the honest ones. The weakness seems to be very serious, as an attack may lead to a large scale exchange of personal identity documents with enormous social costs.

Moreover, as indicated in [Hanzlik 2015], for the scheme described in [BSI 2013] the issuer of the identity documents may easily use the group key mechanism to derive session keys used by a smart card and a terminal, learn the card owner's domain specific identity and later impersonate him against this terminal. It means in particular that the authorities issuing the identity documents may gain full access to citizen's accounts where RI authentication is used. Note that this is possible without learning the private key used for RI authentication. So it does not help that the ID cards are tamperproof.

1.5.2 ChARI protocol

A solution to the group key problem was given in [Hanzlik et al. 2012]. The idea is to merge the Chip Authentication and Restricted Identification protocols. Therefore, instead of using a group key, the construction uses only one secret key on the card to establish a session key for securing the channel and to compute a domain specific pseudonym. Verification that the e-ID was issued by a legitimate authority is possible thanks to a list of anonymous identities computed separately for each domain as a whitelist. Thus, if a secret key of a card gets exposed, it is still possible to block this card by removing it from the whitelists. Moreover, if an adversary attempting to impersonate as a valid user has to hit an identity which is already on the whitelist. The probability of such an event is negligible.

Note that some ideas of [Hanzlik et al. 2012] and of this paper have been used in the scenario of two mobile devices communicating without involving a terminal [Hanzlik et al. 2013].

1.6 Paper contribution

We present an extended version of the ChARI protocol called ChARI eCK-Secure. Our protocol is secure in the well-studied extended Canetti-Krawczyk model (eCK) [LaMacchia et al. 2007] widely used for inspecting authenticated key exchange protocols. This is a major improvement, since the ChARI protocol was not proven secure in a widely used model. Nevertheless, the proposed scheme has a similar computational complexity, in terms of modular exponentiations, as the original ChARI protocol.

Let us note that the RI algorithm deployed on the German personal identity documents solution was proven secure [Dagdelen and Fischlin 2010] in a weaker model, i.e., in an extended version of the Bellare-Rogaway model [Bellare and Rogaway 1994].

A complexity comparison of the discussed algorithms is given in Table 2.

2 ChARI eCK-Secure

In this section we present our eCK secure version of the ChARI protocol, called ChARI eCK-Secure. This algorithm is executed interactively by two parties: a smart card C and a terminal T. Both parties share domain specific parameters $\mathcal{G}_i = (G, q, g_i, \lambda)$, where G is a multiplicative group of a prime order q, g_i is the group generator specific for the ith domain, and λ is a security parameter. We assume that the Decisional Diffie-Hellman Problem (see Sect. 3.1) is hard for G. As it is clear from the context, we do not use a special notation for the computations performed in G.

In addition, the card C contains a secret key sk_C, which is used to derive domain-specific public key $pk_C^i = g_i^{sk_C}$ of C in the ith domain. The terminal holds a secret key sk_T, the public key $pk_T = g_i^{sk_T}$ and a certificate $cert_T$ for pk_T.

In the following description, CVer denotes the certificate verification procedure; SVer denotes the signature verification procedure, MAC(K, M) denotes a cryptographic message authentication code created for the message M with the key K; ENC(K', M) denotes the ciphertext obtained with the symmetric key K' from the message M;

Card C:		Terminal T:
	PRIVATE AND PUBLIC PARAMETERS	
secret key sk_C		private/public key pair sk_T, $pk_T{=}g_i^{sk_T}$
		domain-specific generator g_i
		whitelist W_i of all users admitted to the ith domain serviced by T
public key pk_{CA} of a certificate authority		certificate $cert_T$ for pk_T and g_i
parameters $\mathcal{G}_i = (G, q, g_i, \lambda)$		parameters $\mathcal{G}_i = (G, q, g_i, \lambda)$
	PROTOCOL EXECUTION	
	.... Terminal Authentication Phase (similar to Terminal Authentication from the ICAO documents)	
	$\xleftarrow{cert_T}$	
extract pk_T and g_i from $cert_T$		$esk_T \leftarrow_R \mathbb{Z}_q^*$
abort if CVer$(pk_{CA}, cert_T)$='false'		$epk_T = g_i^{H_0(esk_T, sk_T) \cdot sk_T}$
	$\xleftarrow{epk_T}$	
$r_1 \leftarrow_R \{0,1\}^\lambda$	$\xrightarrow{r_1}$	
		$s = \text{Sign}(sk_T, (r_1, epk_T))$
	$\xleftarrow{s}$	
abort if SVer$(pk_T, (r_1, epk_T))$='false'		
	.... Domain-specific Chip Authentication Phase	
$esk_C \leftarrow_R \mathbb{Z}_q^*$		
$hesk_C = H_0(esk_C, sk_C)$		
$epk_C = g_i^{H_0(hesk_C) \cdot sk_C}$	$\xrightarrow{epk_C}$	
$K = epk_T^{H_0(hesk_C) \cdot sk_C}$		$K = epk_C^{H_0(esk_T, sk_T) \cdot sk_T}$
		$r_2 \leftarrow_R \{0,1\}^\lambda$
		$K_{\text{MAC}}{=}H_1(K, r_2)$, $K_{\text{ENC}}{=}H_2(K, r_2)$
		$T_R = \text{MAC}(K_{\text{MAC}}, epk_C)$
$K_{\text{MAC}}{=}H_1(K, r_2)$, $K_{\text{ENC}}{=}H_2(K, r_2)$	$\xleftarrow{r_2, T_R}$	
check $T_R \stackrel{?}{=} \text{MAC}(K_{\text{MAC}}, epk_C)$		
$\sigma_C = \text{ENC}(K_{\text{ENC}}, H_0(esk_C, sk_C))$		
	$\xrightarrow{\sigma_C}$	$hesk_C = \text{DEC}(K_{\text{ENC}}, \sigma_C)$
		$pk_C^i = epk_C^{H_0(hesk_C)^{-1}}$
		check if pk_C^i is on the whitelist W_i
$K_{\text{session}}{=}H_0(K, epk_C, epk_T, pk_C^i, pk_T)$		$K_{\text{session}}{=}H_0(K, epk_C, epk_T, pk_C^i, pk_T)$

Figure 1: Description of the ChARI eCK-Secure protocol

DEC(K', Z) stands for the plaintext obtained with the symmetric key K' from the ciphertext Z. Moreover, H_0, H_1, H_2 denote hash functions such that $H_0 : \{0,1\}^* \to \mathbb{Z}_q$, $H_1 : \{0,1\}^* \to \mathcal{KS}_{\text{MAC}}$ and $H_2 : \{0,1\}^* \to \mathcal{KS}_{\text{ENC}}$, where $\mathcal{KS}_{\text{MAC}}$ and $\mathcal{KS}_{\text{ENC}}$ are respectively the keyspaces of MAC and ENC.

The protocol ChARI eCK-Secure executes the steps described in Table 1. They can be summarized as follows:

Terminal Authentication: – The terminal presents its certificate $cert_T$ confirming that it is a terminal of the domain where the card should authenticate itself. The card verifies the certificate $cert_T$ and extracts the public key pk_T and domain specific generator g_i from $cert_T$.

- The terminal chooses an ephemeral secret key $esk_T \leftarrow_R \mathbb{Z}_q^*$ at random, computes the ephemeral public key $epk_T = pk_T^{H_0(esk_T, sk_T)}$ and sends this public key to the card. The key epk_T plays the role of a nonce for terminal authentication, however it will be reused for Diffie-Hellman key agreement executed during chip authentication. The role of the hash function in the exponent $H_0(esk_T, sk_T)$ is to prevent derivation of any property of the exponent when only one of the keys esk_T, sk_T is known to the adversary. The above idea of using both keys was inspired by the NAXOS protocol [LaMacchia et al. 2007].
- The card creates a random nonce $r_1 \leftarrow_R \{0,1\}^\lambda$ and sends it to the terminal.
- The terminal signs r_1, epk_T with its private key sk_T. The resulting signature $s = \text{Sign}(sk_T, (r_1, epk_T))$ is sent to the card and verified there with the public verification key pk_T obtained from the terminal's certificate.

Domain Specific Chip Authentication:

- The card C chooses an ephemeral secret key $esk_C \leftarrow_R \mathbb{Z}_q^*$ at random, computes the ephemeral public key $epk_C = g_i^{H_0(H_0(esk_C, sk_C)) \cdot sk_C}$ and sends it to the terminal. The ephemeral key esk_C is retained for the future use, while the intermediate values like $H_0(esk_C, sk_C)$ can be erased. Again, the hash functions in the exponent have to prevent derivation of information on the exponent from only part of the secret keys.
- Both parties compute a shared Diffie-Hellman key. Namely, the card C computes $K = epk_T^{H_0(H_0(esk_C, sk_C)) \cdot sk_C}$ while the terminal T computes K as $epk_C^{H_0(esk_T, sk_T) \cdot sk_T}$. Note that the exponents used contain both the random nonces and the private keys of the communicating parties. The hash functions are applied for the sake of a formal security proof. At this moment the terminal still does not know the public key $g_i^{sk_C}$ of the card C for the domain of the terminal T. It is used only implicitly.

- At the next stage, the terminal has to prove that it knows the shared key K. Namely, T chooses at random a nonce $r_2 \leftarrow_R \{0,1\}^\lambda$, derives the keys $K_{\text{MAC}} = H_1(K, r_2)$, $K_{\text{ENC}} = H_2(K, r_2)$ and computes the tag $T_R = \text{MAC}(K_{\text{MAC}}, epk_C)$. T sends the nonce r_2 and the tag T_R to the card C.
- The card C recomputes K_{MAC}, K_{ENC} and verifies T_R with K_{MAC} and epk_C.
- The card C creates a ciphertext $\sigma_C = \text{ENC}(K_{\text{ENC}}, H_0(esk_C, sk_C))$ and sends it to the terminal.
- Using the shared key K_{ENC} the terminal decrypts σ_C and gets the key $hesk_C = H_0(esk_C, sk_C)$. At this moment T can compute $pk_C^i = epk_C^{H_0(hesk_C)^{-1}}$ which should be equal to $g_i^{sk_C}$, the public key of C in the domain with the parameter g_i. Finally, the terminal verifies that the identifier pk_C^i is on its whitelist, i.e. the list of accepted users.

Establishing a session: both T and C compute locally the session key

$$K_{\text{session}} = H_0(K, epk_C, epk_T, pk_C^i, pk_T).$$

3 Security model

In this section we present details about the formal security model for ChARI eCK-Secure protocol. Let us note that the differences between the ChARI protocol from [Hanzlik et al. 2012] and ChARI eCK-Secure are due to technicalities of the security proof and of the security model. On the other hand, we do not claim that the modifications are necessary to patch any security flaw of the ChARI protocol.

Note that the German RI protocol [BSI 2013] is a simple pseudonym computation scheme. However, the authentication of this pseudonym is assured by the Extended Access Control (EAC) protocol, which is an authenticated key exchange protocol (AKE) run before the pseudonym computation. Thus, security of the German RI scheme must be considered in terms of the combination of these protocols.

ChARI eCK-Secure uses a similar construction. The identification scheme is incorporated into an authenticated key exchange protocol. For this reason, we recall the popular extended Canetti-Krawczyk model [LaMacchia et al. 2007] for AKE protocols, which will be used to prove AKE security of ChARI eCK-Secure. Furthermore, we will use these results to show that a successful impersonation attack would lead to an attack against the AKE security of ChARI eCK-Secure.

3.1 Number-theoretical assumptions

Now we recall computationally hard problems used to prove security properties of our protocol. We start by recalling the discrete logarithm problem for cyclic group $\mathbb{G}$ of a prime order q.

Definition 1 (Discrete Logarithm Problem (DLP)). Given two elements $g, g^x \in \mathbb{G}$, solving DLP means outputting x. We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving DLP in $\mathbb{G}$, if

$$\Pr[x \leftarrow \mathcal{A}(g, g^x)] \geq \epsilon,$$

where the probability is taken over the random choice of the generator $g \in \mathbb{G}$, the random choice of $x \in \mathbb{Z}_q$, and the random bits of $\mathcal{A}$. In addition, by $\mathbf{Adv}^{\text{DLog}}$ we mean the maximal advantage for any PPT adversary in solving DLP, and by $\text{DLog}(g, h)$ we mean x such that $g^x = h$.

Definition 2 (Computational Diffie-Hellman Problem (CDH)). Given three elements $g, g^a, g^b \in \mathbb{G}$, the goal is to output the element g^{ab}. We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the CDH problem in $\mathbb{G}$ if:

$$\Pr[g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b)] \geq \epsilon,$$

where the probability is taken over the random choice of the generator $g \in \mathbb{G}$, the random choice of $a, b \in \mathbb{Z}_q$, and the random bits of $\mathcal{A}$. In addition, by $\mathbf{Adv}^{\text{CDH}}$ we denote the maximal advantage for any PPT adversary in solving the computational Diffie-Hellman problem. For $X = g^a$, $Y = g^b$, by $\text{CDH}(X, Y)$ we denote g^{ab}.

Let us also recall the decisional version of the Diffie-Hellman problem:

Definition 3 (Decisional Diffie-Hellman Problem (DDH)). Given four elements $g, g^a, g^b, g^z \in \mathbb{G}$, the output should be 1, if $z = a \cdot b \mod q$, and 0 otherwise. We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the DDH in $\mathbb{G}$ if:

$$|\Pr[1 \leftarrow \mathcal{A}(g, g^a, g^b, g^{ab})] - \Pr[1 \leftarrow \mathcal{A}(g, g^a, g^b, g^z)]| \geq \epsilon,$$

where the probability is taken over the random choice of the generator $g \in \mathbb{G}$, the random choice of $a, b, z \in \mathbb{Z}_q$, and the random bits of $\mathcal{A}$. In addition, by $\mathbf{Adv}^{\text{DDH}}$ we denote the maximal advantage for any PPT adversary in solving the DDH problem.

3.2 Extended Canetti-Krawczyk model

In this subsection we recall the popular eCK model for two party authenticated key exchange protocols (AKE). In contrary to other models, such as the Bellare-Rogaway [Bellare and Rogaway 1994] and the Canetti-Krawczyk [Canetti and Krawczyk 2001], eCK considers forward secrecy and key-compromise impersonation resilience (KCI). These attacks are substantial from the point of view of secure use of identification documents. Forward secrecy protects the session key, and thereby the exchanged data, even if the adversary later learns the long-term secrets of the participants. Security against KCI ensures that leaking the long-term secret of a party, say Alice, does not enable a party talking with Alice to impersonate other parties.

3.2.1 AKE session

Each user has a view on the protocol execution consisting of the messages exchanged with his alleged partner. This sequence of messages is called a *session*. If the protocol is executed properly, then the sessions of both communicating parties consist of the same messages. In this case we talk about *matching sessions*. The initiator of the session is called the *owner* of the session and the other party is called the *peer*. Note that a *matching session* may not exist, if the communication between the owner and the peer has been corrupted, e.g. by a man-in-the-middle. The *session identifier* consists of the parties' identities and the messages exchanged during the session.

3.2.2 AKE experiment informally

To prove security in the eCK model, we have to show that there is no adversary $\mathcal{A}$ that wins the AKE experiment with a non-negligible probability. In the AKE experiment the adversary $\mathcal{A}$ is given access to different oracles. These oracles allow, among others, to: initiate AKE sessions, control communication between protocol participants, reveal long-term, ephemeral and session keys. At some time of the AKE experiment the adversary $\mathcal{A}$ must query a Test oracle. This oracle returns a random key or the real session key of the session tested, according to a randomly chosen bit b. The task of $\mathcal{A}$ is to guess the value of the bit b. As a random answer yields the correct response with the probability $\frac{1}{2}$, we look for adversaries $\mathcal{A}$ that answer correctly with a probability non-negligibly higher than $\frac{1}{2}$.

3.2.3 AKE experiment formally

We consider a system consisting of a number of communicating honest parties, the certification authority $\mathcal{CA}$ and the adversary $\mathcal{A}$. We assume that all communication links between all parties are fully controlled by the adversary $\mathcal{A}$. Furthermore, we define the session identifier to be:

$$sid = (\mathit{role}, \mathit{ID}, \mathit{ID}^*, \mathit{comm}_1, \ldots, \mathit{comm}_n),$$

where *ID* is the identity of the initiator, ID^* is the identity of the peer, $\mathit{role} \in \{I, R\}$ is the role in the protocol (initiator/responder) and $\mathit{comm}_j \in \{0,1\}^*$ is the jth message sent during this session.

In our case we assume without loss of generality that there are two sets of users, i.e., users $U_1, U_2, \ldots$ that play either the role of terminals (denoted by $T_1, T_2, \ldots$ or simply T if only one terminal is considered) or the role of e-ID cards (denoted by $C_1, C_2, \ldots$ or simply C if only one card is considered).

Each party computes communications $comm_j$ as a function of all previous messages and data corresponding to the party and the partner (i.e. the long-term key, the public keys etc.). After receiving all communications, the party finishes the execution by computing the session key.

3.2.3.1 AKE experiment

The AKE experiment proceeds as follows. At first the adversary selects the identities of all honest parties (which can be arbitrary distinct binary strings) and these parties generate and register their public keys with the $\mathcal{CA}$. The adversary may also register arbitrary public keys, even the same as those of honest parties. Then the adversary makes any sequence of queries to the following oracles:

Send$(U_i, U_j, comm)$**:** the oracle sends a message *comm* to U_i on behalf of U_j. This oracle returns U_i's response to the message *comm*. This query allows the adversary $\mathcal{A}$ to start an AKE session with U_j and to provide communications between U_i and U_j.

Long-Term Key Reveal(U_i)**:** this oracle reveals the long-term key of the party U_i.

Ephemeral Key Reveal(sid)**:** this oracle reveals an ephemeral key of a, possibly incomplete, session sid.

Reveal(sid)**:** this oracle reveals the session key of a completed session sid.

Furthermore, at any time of the experiment, the adversary $\mathcal{A}$ may select a completed session sid and, only once during the experiment, send the following query to the Test oracle:

Test(sid)**:** The oracle picks $b \leftarrow_R \{0,1\}$. If $b = 1$, then $K \leftarrow$ Reveal(sid), otherwise $K \leftarrow_R \{0,1\}^\lambda$. The oracle returns K.

The adversary can then continue the AKE experiment. However, $\mathcal{A}$ terminates immediately after making a query to the following oracle:

Guess(b') - If $b = b'$, return 1, otherwise return 0.

The adversary *wins* the AKE experiment, if the selected test session remains *clean* (see below for the meaning of a "clean session") until the end of the experiment and the **Guess** oracle returns 1.

3.2.4 Clean session

Let sid be an AKE session completed by parties U_i and U_j. In addition, by sid^* we denote the matching session to sid, supposedly executed by U_j (note that sid^* may not exist in the experiment). Furthermore, let sk_{U_i} and sk_{U_j} denote the long-term secret keys of U_i and U_j, respectively. By esk_{U_i} and esk_{U_j} we denote the ephemeral secret keys generated by U_i and U_j in sid and sid^* (the latter exists only if sid^* exists). We now say that an AKE session is *clean* if none of the following events occurs:

– U_i or U_j is controlled by the adversary,

- $\mathcal{A}$ reveals the session key of sid or sid^* (if it exists) with an oracle call to **Reveal**,
- The session sid^* exists and $\mathcal{A}$ reveals either both sk_{U_i} and esk_{U_i}, or both sk_{U_j} and esk_{U_j} with appropriate calls to oracles **Long-Term Key Reveal** and **Ephemeral Key Reveal**.
- The session sid^* does not exist and $\mathcal{A}$ reveals either sk_{U_j} or both sk_{U_i} and esk_{U_i} via appropriate oracle calls.

Definition 4 (Extended Canetti-Krawczyk (eCK) security). We define the advantage of an adversary $\mathcal{A}$ in the AKE experiment with AKE protocol Π as:

$$\mathbf{Adv}_{\Pi}^{\mathrm{AKE}} = |\Pr[\mathcal{A} \text{ wins the AKE experiment}] - \tfrac{1}{2}| \ .$$

We say that an AKE protocol is secure in the eCK model if no efficient adversary $\mathcal{A}$ has a non-negligible advantage $\mathbf{Adv}_{\Pi}^{\mathrm{AKE}}$ in winning the above experiment.

3.2.5 Security achieved by the eCK model

Confidentiality: the messages exchanged while establishing the session key may provide only a negligible advantage for an adversary aiming to break confidentiality of the data exchange.

Forward secrecy: the session key of a previous session remains confidential even if:

- the adversary knows three out of four secret keys (two ephemeral keys and two long-term keys), if this session has a matching session,
- the adversary knows two out of three secret keys (two long-term keys and the ephemeral key of the owner), if the session has no matching session (the peer's ephemeral key does not exist).

Resistance against key-compromise impersonation: this covers attacks in which an adversary learns the long-term key of a user and then tries to use this key to impersonate other users to this party.

Confidentiality in case of leakage of ephemeral values: a session remains confidential even if the adversary reveals the internal states of the session (i.e. ephemeral values of the communicating parties), unless he also knows one of long-term keys.

3.3 Cross-domain anonymity

Informally, cross-domain anonymity (also called unlinkability) means that the adversary cannot distinguish user identifiers in different domains, i.e., given two identifiers from two domains the adversary cannot tell, if these identifiers correspond to one user.

3.3.1 System parameters

In order to formally capture cross-domain anonymity we define a game played by an adversary $\mathcal{A}$ and a challenger. First, the adversary sends n, the number of users, and d, the number of domains, to the challenger. Then, the challenger creates a system corresponding to the identification protocol Π. That is, he defines system parameters such as the group used for cryptographic computations, creates a Certificate Authority CA, creates valid cards (or users) $C_1, \ldots, C_n$, possibly using the secret keys of the CA, and defines the domain parameters for each terminal $T_1, \ldots, T_d$. Moreover, for each domain j (i.e., for each terminal T_j) the challenger creates $C_1^j, \ldots, C_n^j$ as genuine copies of the cards $C_1, \ldots, C_n$. Namely, C_i^j is a copy of C_i.

The adversary will see these cards shuffled at random. That is, for each $j \leq d$ the challenger chooses at random a permutation $\pi_j : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$. Then we may think that the adversary can address the cards in the way that from his point of view the ith card for domain j is in fact the card $C^j_{\pi(i)}$.

We assume that all terminals are controlled by the adversary. So if protocol Π requires authenticated terminals, then the adversary is given the secret keys of all of them.

3.3.2 Capabilities of the adversary

An ULK adversary $\mathcal{A}$ is given a full control of the communication in the system, i.e., the challenger gives $\mathcal{A}$ access to the following oracles:

Execute(i, j): this oracle returns a transcript of execution of the protocol Π between card instance $C^j_{\pi_j(i)}$ and terminal T_j.

Send(i, j, m): this oracle sends message m to the card instance $C^j_{\pi_j(i)}$ and returns its response (specified by the protocol Π).

$\mathcal{A}$ is also given access to several other oracles:

Corrupt(i, j): this oracle returns the secret keys of card $C_{\pi_j(i)}$.

Reveal(i, j, k): this oracle returns the number $\pi_k^{-1}(\pi_j(i))$. Note that according to the adversary's point of view the ith card in domain j and the card $\pi_k^{-1}(\pi_j(i))$ in domain k correspond to the same card in the batch $C_1, \ldots, C_n$.

Furthermore, if protocol Π allows revocation, the adversary can model this using the following oracles:

Revoke(i, j): this oracle revokes card $C_{\pi_j(i)}$ in all domains, i.e., the cards $C^1_{\pi_j(i)}, \ldots, C^d_{\pi_j(i)}$ will not pass the verification.

DomainRevoke(i, j): this oracle revokes only the card instance $C^j_{\pi_j(i)}$.

Finally, at the end of the game the adversary queries the following test oracle:

Test(i, j, k, l): the oracle returns 1, if $\pi_j(i) = \pi_l(k)$, otherwise it returns 0.

3.3.3 Adversary's goal and winning conditions

We denote a Test query (i, j, k, l)-*fresh* if:

- two cards $C^j_{i_1}, C^j_{i_2}$ in domain j and two cards $C^l_{k_1}, C^l_{k_2}$ in domain l have been neither revoked nor corrupted,
- no query of the form:

$$\text{Reveal}(i_1, j, l),\ \text{Reveal}(i_2, j, l),\ \text{Reveal}(k_1, l, j),\ \text{Reveal}(k_2, l, j)$$

  has been stated,
- $i \in \{i_1, i_2\}$ and $k \in \{k_1, k_2\}$.

$\mathcal{A}$ wins the unlinkability game, if the Test oracle returns 1 and the issued Test query was fresh. Ideally, the probability of winning the above game should be not greater than for a random guess. Note that the highest probability of success is for 2 cards and 2 domains. Then if the first user and domain is set, with probability $\frac{1}{2}$ we may guess the card of this user in the second domain. Let $\mathbf{Adv}^{\text{ULK}}_{\Pi}$ denote the maximal advantage of the adversary over the random answer for the unlinkability game for Π.

4 Security analysis

In this section we present security proofs for the protocol ChARI eCK-Secure. First, in the random oracle model, we show that ChARI eCK-Secure protocol is secure in the eCK model, provided that the CDH problem is hard and that the signature scheme used guarantees unforgeability. Then we show that ChARI eCK-Secure satisfies cross-domain anonymity assuming hardness of the DDH problem.

4.1 AKE security

Theorem 5. *ChARI eCK-Secure is secure in the Extended Canetti-Krawczyk model (eCK) if H_0, H_1 and H_2 are modeled by independent random oracles. Namely, for any AKE adversary $\mathcal{A}$ against ChARI eCK-Secure that runs in polynomial time in the security parameter λ, involves at most n honest parties, activates at most k sessions and makes at most q_{H_0}, q_{H_1}, q_{H_2} oracle queries to H_0, H_1, H_2 respectively, there is a CDH solver S such that:*

$$\begin{aligned}\mathbf{Adv}^{\text{CDH}} \geq \tfrac{1}{2} \cdot \min\left(\tfrac{1}{2n \cdot k \cdot q_H}, \tfrac{2}{k^2 \cdot q_{H_0}}\right) \cdot \mathbf{Adv}^{\text{AKE}}_{ChARIeCK} \\ - \tfrac{1}{4} \cdot \mathbf{Adv}^{\textit{forge}}_{sign} - n \cdot \mathbf{Adv}^{\text{DLog}} - O\left(\tfrac{k^2}{2^\lambda}\right),\end{aligned}$$

where $q_H = q_{H_0} + q_{H_1} + q_{H_2}$.

Proof. Let $\mathcal{A}$ be an AKE adversary in the AKE security model for ChARI eCK. The adversary $\mathcal{A}$ has only two ways to distinguish the session key $K_{session} = H(\sigma)$, for some 5-tuple σ, from a random string:

1. Forging attack: at some moment $\mathcal{A}$ queries H_0 for the same 5-tuple σ.
2. Key-replication attack: $\mathcal{A}$ succeeds to enforce establishing the same session key in another session.

If we assume that the random oracles produce no collisions, the key-replication attack is infeasible as equality of the session keys requires equality of the corresponding 5-tuples (which are hashed to produce the session keys). However, the ephemeral values are chosen at random, and therefore the probability of a repetition is negligible. If the 5-tuples are different, then according to the random oracle model, a collision occurs with probability $O(k^2/2^\lambda)$. It follows that $\mathcal{A}$ must perform a forging attack in order to get a non-negligible success probability.

We show that if $\mathcal{A}$ can mount a forging attack, then we can construct a solver S breaking the CDH problem. Informally, S works as follows. It takes as input a CDH challenge (X_0, Y_0) and executes the extended Canetti-Krawczyk experiment with $\mathcal{A}$. Then S modifies the data returned by the honest users in such a way that if $\mathcal{A}$ breaks the security of ChARI eCK-Secure, then S can output a solution to the CDH problem.

We consider two cases of $\mathcal{A}$'s behavior. Namely, we consider the case that $\mathcal{A}$ selects a test session for which a matching session exists, and separately the case that the test session has no matching session. The solver S tries to anticipate the behavior of $\mathcal{A}$ by choosing one of those cases at random with the probability $\frac{1}{2}$.

Case 1: a matching session exists

Assume that $\mathcal{A}$ selects a test session for which a matching session exists. S modifies the experiment as follows. It selects at random two sessions. If they are the matching sessions and one of them is the test session, then the experiment does not fail at this point. Note that this happens with probability higher than $\frac{2}{k^2}$ as there are $\frac{k(k-1)}{2}$ ways to choose a pair of sessions. S generates all values according to the protocol, but sets $epk_C = X_0$ and $epk_T = Y_0$ for the chosen pair of sessions. If $\mathcal{A}$ wins the forging attack it must have queried the oracle $H_0(\sigma)$ for the value σ which is the 5-tuple containing the key $K = \text{CDH}(X_0, Y_0)$. Note that S can choose the right query of $\mathcal{A}$ with the probability $1/q_{H_0}$.

If the session selected by S is indeed the test session, then $\mathcal{A}$ is allowed to query for a subset of the values $\{esk_C, esk_T, sk_C, sk_T\}$, but not for (esk_C, sk_C) and (esk_T, sk_T) (containing both secrets of a party.)

$\mathcal{A}$ can distinguish the simulated experiment from a real AKE experiment, if $\mathcal{A}$ queries (esk_C, sk_C) or (esk_T, sk_T) to oracle H_0. However, in this case the test session would not be clean. In other cases $\mathcal{A}$ cannot distinguish the simulated experiment. Indeed, if $\mathcal{A}$ reveals both ephemeral keys, then it must solve the discrete logarithm problem for pk_T or pk_C^i, to distinguish the simulated experiment. Indeed, only in this case $\mathcal{A}$ may compute the hash values and verify whether the values epk_C or epk_T were computed according to the protocol, i.e., $epk_C = (pk_C^i)^{H_0(H_0(esk_C, sk_C))}$, $epk_T = (pk_T)^{H_0(esk_T, sk_T)}$. On the other hand, if $\mathcal{A}$ reveals both long-term keys, then it cannot distinguish the simulated experiment, since there is no "reference data" for esk_C, esk_T and $\mathcal{A}$ is unable to distinguish $H_0(esk_C, sk_C)$ or $H_0(esk_T, sk_T)$ from a random value. Therefore, the probability that $\mathcal{A}$ detects a difference and the simulation fails is at most $2n \cdot \mathbf{Adv}^{\text{DLog}}$.

We conclude that if $\mathcal{A}$ selects a test session which has a matching session, then:

$$\mathbf{Adv}^{\text{CDH}} \geq \frac{2}{k^2 \cdot q_{H_0}} \cdot \mathbf{Adv}^{\text{AKE}}_{ChARIeCK} - 2n \cdot \mathbf{Adv}^{\text{DLog}} - O\left(\frac{k^2}{2^\lambda}\right) .$$

Case 2: no matching session exists

Now we assume that $\mathcal{A}$ selects a test session for which there is no matching session. We consider two subcases: the owner of the session is the responder (i.e. the card) or the initiator (i.e. the terminal). Note that we have assumed that a user can only play either the role of an initiator or the role of an responder. Thus, the solver S has again to anticipate the behavior of $\mathcal{A}$ by choosing one of those subcases at random with probability $\frac{1}{2}$.

Case 2.1: Session owned by a terminal

We start with the case when the owner of the test session is a terminal. In this case S modifies the experiment as follows. S selects a card C at random and sets $pk_C^i = X_0$. Note that since the secret key corresponding to this public key is unknown, S cannot properly simulate the eCK sessions executed by C. However, S can handle such sessions as follows. S randomly picks esk_C and h at random from $\mathbb{Z}_q^*$ and sets $epk_C = (pk_C^i)^h$. Obviously, S cannot compute the key K from epk_C, epk_T, as this would require solving the CDH problem. However, the other party, i.e, the terminal T must send the tag T_R to the card before S has to use the key K. In order to compute T_R, terminal T must have used $K_{\text{MAC}} = H_1(K, r_2)$. Thus, S may search for a response K_{MAC} outputted by the oracle H_1 which verifies T_R. If such a key exists, then S can take key K from the query to H_1. Otherwise S terminates. Such a simulation works in both cases, i.e., if terminal T is adversary-controlled or not. Note that $\mathcal{A}$ cannot detect that it is a simulated eCK experiment unless it either queries $(esk_C, \text{DLog}(X_0))$ to oracle H_1 or reveals the long-term secret key of C. The first event reveals $\text{DLog}(X_0)$ and allows S to solve the CDH problem, this happens with probability at most $n \cdot \mathbf{Adv}^{\text{DLog}}$. The second event is impossible as otherwise the test session will no longer be clean.

Now S randomly selects an eCK session in which C is the card. Let T be the terminal for this session. S generates esk_T in a regular way, but sets $epk_T = Y_0$. Additionally,

T chooses the shared Diffie-Hellman key K at random and uses it to derive the keys K_{MAC} and K_{ENC}. With probability at least $\frac{1}{nk}$ ($\frac{1}{n}$ to pick the correct party C and $\frac{1}{k}$ to pick the correct session) S picks the right combination (the user and the session) for the test oracle. $\mathcal{A}$ can distinguish this simulation from a real eCK experiment either if it queries (esk_T, sk_T) to H_0 or if $\mathcal{A}$ noticed that T_R and σ_C (if generated by C) were computed using false keys K_{MAC} and K_{ENC}. The first case happens with probability at most $n \cdot \mathbf{Adv}^{\text{DLog}}$. In the second case $\mathcal{A}$ must have queried $H_1(\text{CDH}(X_0, Y_0)^{H_0(hesk_C)}, r_2)$ to get the key K_{MAC} or $H_2(\text{CDH}(X_0, Y_0)^{H_0(hesk_C)}, r_2)$ to get the key K_{ENC}. In either case, S can use the query made by $\mathcal{A}$ to capture the value $\text{CDH}(X_0, Y_0)^{H_0(hesk_C)}$. S can choose this value from all queries to H_1 (respectively, H_2) with probability at least $1/q_{H_1}$ (respectively, $1/q_{H_2}$). Moreover, $\mathcal{A}$ (or the party C) must have queried $hesk_C$ to oracle H_0. Note that this query can be found by searching for a query $hesk_C$ such that $epk_C = pk_C^{H_0(hesk_C)}$. It follows that S can compute $\text{CDH}(X_0, Y_0)$ and solve the CDH problem.

Now if $\mathcal{A}$ wins the AKE experiment, at some point it must query for $H_0(\sigma)$, where σ is a 5-tuple containing the value $\text{CDH}(X_0, Y_0)^{H_0(hesk_C)}$. The solver S can find the right query with probability at least $1/q_{H_0}$. As before, we can see that either $\mathcal{A}$ or the party C must have queried for the value $hesk_C$. Thus, S is able to compute $\text{CDH}(X_0, Y_0)$ and solve the CDH problem.

We conclude that if $\mathcal{A}$ selects a test session which has no matching session and the owner of this session is a terminal, then:

$$\mathbf{Adv}^{\text{CDH}} \geq \frac{1}{n \cdot k \cdot q_H} \cdot \mathbf{Adv}^{\text{AKE}}_{ChARIeCK} - 2n \cdot \mathbf{Adv}^{\text{DLog}} - O\left(\frac{k^2}{2^\lambda}\right) .$$

Case 2.2: Session owned by a card

We will now consider the case when the owner of the test session is a card C. Let us assume that the other party for this session is a terminal T. Note that T cannot be adversary-controlled, since this would mean that the test session is not clean. Thus, the adversary may only query for the ephemeral key esk_T (corresponding to this session) and not for the secret key sk_T. We may observe that the value epk_T cannot be changed by the adversary, since this would mean that $\mathcal{A}$ can forge the signature s. Note that $\mathcal{A}$ cannot manipulate the nonce r_1 sent by the card because C would terminate the session after verifying signature s.

The solver S randomly selects an eCK session in which C is the card. With probability $\frac{1}{k}$ the selected session is the test session. Then, S sets $epk_T = Y_0$ and $epk_C = X_0$. Note that at this point $\mathcal{A}$ can distinguish the simulated experiment from a real eCK experiment only if it queries H_0 for either (sk_C, esk_C) or (sk_T, esk_T). However, as previously shown, this happens with the probability at most $2n \cdot \mathbf{Adv}^{\text{DLog}}$.

We now distinguish two subcases. Either the value T_R is computed by $\mathcal{A}$ or by S (when simulating C). In the first case $\mathcal{A}$ must have queried the oracle H_1 for the pair

$(\text{CDH}(epk_T, epk_C), r_2)$ in order to derive K_{MAC}. Thus, S may search for queries to H_1 for $\text{CDH}(epk_T, epk_C)$ and thereby get an answer to the CDH problem. In the second case S may compute T_R using a random key and omit the verification while simulating C. Note that $\mathcal{A}$ can detect this manipulation only if it queries the oracle H_1 for $(\text{CDH}(epk_T, epk_C), r_2)$. However, then S would be able to extract the result for $\text{CDH}(X_0, Y_0)$ with probability $1/q_{H_1}$.

If $\mathcal{A}$ does not verify T_R, then S computes σ_T at random. As before, we see that if $\mathcal{A}$ decrypts σ_T, then it must have queried the oracle H_2 for $(\text{CDH}(epk_T, epk_C), r_2)$ in order to derive the key K_{ENC}. Thus, in such a case, with probability $1/q_{H_2}$, S can extract the solution $\text{CDH}(X_0, Y_0)$. Otherwise, if $\mathcal{A}$ wins the AKE experiment, it must query for $H_0(\sigma)$, where σ is the 5-tuple containing the result $\text{CDH}(X_0, Y_0)$. Thus, with probability $1/q_{H_0}$, S can determine $\text{CDH}(X_0, Y_0)$.

Therefore, if $\mathcal{A}$ selects a test session which has no matching session and the owner of this session is a card, then:

$$\mathbf{Adv}^{\text{CDH}} \geq \tfrac{1}{k \cdot q_H} \cdot \mathbf{Adv}^{\text{AKE}}_{ChARIeCK} - \mathbf{Adv}^{\textit{forge}}_{sign} - 2n \cdot \mathbf{Adv}^{\text{DLog}} - O\left(\tfrac{k^2}{2^\lambda}\right)$$

We conclude that in Cases 2.1 and 2.2 we always have:

$$\mathbf{Adv}^{\text{CDH}} \geq \tfrac{1}{2n \cdot k \cdot q_H} \cdot \mathbf{Adv}^{\text{AKE}}_{ChARIeCK} - \tfrac{1}{2} \cdot \mathbf{Adv}^{\textit{forge}}_{sign} - n \cdot \mathbf{Adv}^{\text{DLog}} - O\left(\tfrac{k^2}{2^\lambda}\right)$$

□

4.2 Cross-domain anonymity

Theorem 6. *ChARI eCK-Secure satisfies cross-domain anonymity if H_0, H_1 and H_2 are modeled by independent random oracles and the DDH problem is hard. Namely, for any ULK adversary $\mathcal{A}$ against ChARI eCK-Secure that runs in polynomial time in the security parameter λ, involves at most n honest parties and d domains, we show that there exists a DDH solver S such that:*

$$\frac{1}{n^2} \cdot \mathbf{Adv}^{\text{ULK}}_{ChARIeCK} \leq \mathbf{Adv}^{\text{DDH}}$$

Proof. Let $\mathcal{A}$ be an ULK adversary against the cross-domain anonymity of ChARI eCK-Secure. We show that we can construct a DDH solver S which uses the advearsary $\mathcal{A}$ as a subprocedure. Let $(X_0 = g^x, Y_0 = g^y, Z_0 = g^z)$ be an instance of the DDH problem.

Note that S can simulate a card with the public key $pk^i_C = g_i^{sk_C}$, even if the secret key sk_C is not known to S. The solver only differs from the protocol by choosing $hesk_C \in \mathbb{Z}^*_q$ at random instead of deriving it as $H_0(esk_C, sk_C)$. The adversary $\mathcal{A}$ cannot see any difference, since in the unlinkability game $\mathcal{A}$ cannot query for the ephemeral secret key esk_C. Obviously, S cannot compute the key K. However, $\mathcal{A}$ must compute T_R using the key $K_{\text{MAC}} = H_1(K, r_2)$. Thus, S may search for the response of the oracle H_1, which verifies T_R, and for the corresponding query (K, r_2).

The solver works as follows. First, S creates $n-1$ cards with known (to S) but random secret keys and one special card with $sk_C = y \cdot b$, for a random b (the randomizer b is known to S, but neither sk_C nor y is known to S – simply the public key is set to Y_0^b). Then, S creates two types of domains. For the first type, $g_i = g^{r_i}$ (for a random r_i). In such a domain the public key of the special card is $pk_C^i = (Y_0)^{r_i \cdot b}$. For the second domain type $g_j = (X_0)^{r_j}$ (for a random r_j). In this case the public key of the special card is $pk_C^j = (Z_0)^{r_j \cdot b}$. The remaining public keys can be easily derived by S as it knows the private keys of the remaining users.

At the beginning of the experiment the probability that $\mathcal{A}$ makes a Test query for the user i in the domain j and the user k in the domain l is $\frac{2}{d^2 \cdot n^2}$ (Test queries (i,j,k,l) and (k,l,i,j) are symmetric cases). Hence, with probability $\frac{1}{2}$, the domains j and l are of different types and one is built using X_0. In addition, with probability $\frac{1}{n \cdot n}$ the cards i and k chosen by $\mathcal{A}$ are the special card. Thus, with probability $2 \cdot \frac{1}{2 \cdot n \cdot n} = \frac{1}{n^2}$ the unlinkability game corresponds to the problem, whether $(g^{r_j}, g^{y \cdot r_j \cdot b}, g^{x \cdot r_l}, g^{z \cdot r_l \cdot b})$ corresponds to a correct DDH tuple (note that g_j is known to S). $\square$

5 Conclusions and future work

The protocol ChARI eCK-Secure presented here avoids the group key problem and is secure in the extended Canetti-Krawczyk model. Moreover, it has a similar computational complexity as the previous versions (see Table 2).

However, ChARI eCK-Secure requires whitelists of the users. This approach may require handling big data on side of the public key infrastructure. Thus, it would be helpful to design an RI scheme, which would neither require the white lists nor the group keys. Note that this issues are particularly important not only for authentication of humans holding electronic identity documents, but also for machine-to-machine authentication in heterogenous ad hoc networks run by multiple providers, mobile autonomous devices and for other scenarios of Internet of Things.

This problem has been solved in [Hanzlik 2015] at the price of using bilinear mappings on the side of the terminals. We are not aware of any solution that would work in the general case (not only for pairing friendly groups).

Acknowledgments

This research has been supported by the Polish National Science Centre, project HARMONIA, DEC-2013/08/M/ST6/00928.

References

[Bellare and Rogaway 1994] Bellare M., Rogaway P.: “Entity authentication and key distribution”; Lect. Notes Comp. Sci. 773, Springer, Berlin (1994), 232-249.

	German RI [BSI 2013]	ChARI [Hanzlik et al. 2012]	ChARI eCK-Secure This papers contribution
E-ID	2E + 2 SignVer	2E + 2M + 2 SignVer	2E + 2M + 2 SignVer
Terminal	2E + 1 SignGen	3E + 1I + 1 SignGen	3E + 2M + 1I + 1 SignGen

Table 2: Efficiency of discussed algorithms: E stands for the number of exponentiations, M for the number of multiplications, I for the number of inversions, SignGen and SignVer denote the cost of signature generation and verification, the cost of all symmetric algorithms (i.e. encryption, MAC and hash) is neglected.

[BSI 2013] BSI: "Advanced Security Mechanisms for Machine Readable Travel Documents 2.11," Technische Richtlinie TR-03110, (2013), https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/TR-03110_v2_11_P3pdf.pdf?__blob=publicationFile

[Canetti and Krawczyk 2001] Canetti K., Krawczyk H.: "Analysis of key-exchange protocols and their use for building secure channels,"; Lect. Notes Comp. Sci. 2045, Springer, Berlin (2001), 453-474.

[Dagdelen 2013] Dagdelen Ö.: "The Cryptographic Security of the German Electronic Identity Card". PhD Dissertation, Technische Universität Darmstadt (2013), http://tuprints.ulb.tu-darmstadt.de/3538/7/%C3%96zg%C3%BCrDagdelen-Thesis.pdf

[Dagdelen and Fischlin 2010] Ö. Dagdelen and M. Fischlin: "Sicherheitsanalyse des EAC-Protokols," Technical report, BSI, 2010. [Online]. Available at: http://www.personalausweisportal.de/SharedDocs/Downloads/DE/Studie_Kryptographie_Volltext.pdf

[Feld and Pohlmann 2011] Feld S., Pohlmann N.: "Security analysis of OpenID followed by a reference implementation of an nPA-based OpenID provider"; ISSE 2010 Securing Electronic Business Processes, Springer Science & Business Media, Berlin (2011), 13-25.

[Fischlin et al. 2010] Fischlin M., Lehmann A., Ristenpart T., Shrimpton T., Stam M., Tessaro S.: "Random oracles with(out) programmability"; Lect. Notes Comp. Sci. 6477, Springer, Berlin (2010), 303-320.

[Flinn and Maurer 1995] Flinn B., Maurer H. A.: "Levels of anonymity", J. UCS, 1, 1 (1995), 35-47.

[Fumy and Paeschke 2010] Fumy W., Paeschke M.: "Handbook of eID Security: Concepts, Practical Experiences, Technologies"; John Wiley & Sons (2010), ISBN 978-3-89578-379-1.

[Hanzlik 2015] Hanzlik L.: "Cryptographic Protocols for Modern Identification Documents". PhD Dissertation submitted at Inst. of Computer Science, Polish Academy of Sciences (2015).

[Hanzlik et al. 2012] Hanzlik L., Kluczniak K., Kubiak P., Kutyłowski M.: "Restricted identification without group keys"; Proc. TrustCom 2012, IEEE Computer Society, Los Alamitos (2012), 1194-1199.

[Hanzlik et al. 2013] Hanzlik L., Kluczniak K., Kutyłowski M., Krzywiecki Ł.: "Mutual restricted identification"; Lect. Notes Comp. Sci. 8341, Springer, Berlin (2013), 119–133.

[ICAO Doc 2008] , ICAO: "Machine Readable Travel Documents - Part 3: Machine Readable Passport, Specifications for electronically enabled official travel documents with biometric identification capabilities" (2008), http://www.icao.int/publications/Documents/9303_p3_v2_cons_en.pdf

[LaMacchia et al. 2007] LaMacchia B. A., Lauter K., Mityagin A.: "Stronger security of authenticated key exchange"; Lect. Notes Comp. Sci. 4784, Springer, Berlin (2007), 1-16.

[Poller et al. 2012] Poller A., Waldmann U., Vowé S., Türpe S.: "Electronic identity cards for user authentication - promise and practice"; IEEE Security & Privacy, 10, 1 (2012), 46–54.