BioID: a Privacy-Friendly Identity Document*

Fatih Balli[1], F. Betül Durak[1,2**], and Serge Vaudenay[1]

[1] Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{fatih.balli, serge.vaudenay}@epfl.ch
[2] Robert Bosch LLC – Research and Technology Center, Pittsburgh, USA
betul.durak@us.bosch.com

Abstract. We design a suite of protocols so that a small tamper-resistant device can be used as a biometric identity document which can be scanned by authorized terminals. We target both strongly secure identification and strong privacy. Unlike biometric passports, our protocols leak no digital evidence and are essentially deniable. Besides, getting the identity information from the device requires going through access control. Access control can follow either a strong PKI-based path or a weak password-based path which offer different functionalities. We implemented our protocols on JavaCard using finger-vein recognition as a proof of concept.

Keywords: privacy, deniability, ID document, smart card

1 Security vs Privacy in Identification

Informally, *secure identification* means that (1) acceptance of invalid identities do not happen, (2) making multiple copies of the same identity document is hard, and (3) the identity of the holder matches the identity indicated by the document. Public-key cryptography, tamper-resistance, and biometry solve these problems. However, the same cannot be said for privacy, which used to come as a secondary goal in the design process. Fortunately, EU has recently passed GDPR to enforce companies and institutions to put forth "appropriate technical and organizational measures" to protect users' privacy. In identification context, privacy means that (1) identity documents leak no tracking information to unauthorized terminals, (2) the information transferred to authorized terminals is always deniable, and (3) biometric templates leave the card only if the terminal is strongly authenticated, i.e. the certificate of the interacting terminal is still valid.

Previous Work. International Civil Aviation Organization standardized globally recognized passports as Machine Readable Travel Documents (MRTD)

* This is the full version of the work with the same title that appears in the proceedings of the 15th International Workshop on Security and Trust Management (STM) 2019. The work was supported by Innosuisse project 19339.1 PFES-ES and done in collaboration with Global ID SA, Switzerland.

** This work was done when the author was at EPFL.

[1]. The MRTD standard realizes the goals of secure identification and some minimalistic level of privacy. In a more detail, false or invalid recognition of identities are prevented with classical digital signatures and Public-Key Infrastructures (PKI) in the MRTD standard. This is referred to as the Passive Authentication Protocol. Although this solution is simple and effective in terms of secure identification, it leaks a transferable evidence to the communicating terminal. An adversarial terminal can collect these identities with signatures and sell them on the black market (because their validity is provable), or use them in a global surveillance system. Therefore, it ignores the privacy of the bearer.

Even worse, the MRTD standard allows the chip to communicate with the terminal only through a password-authenticated channel (formerly BAC, currently PACE protocol). Although this prevents arbitrary devices from communicating with the chip and skim the contained data, a state-level adversary collecting password page copies can easily thwart this protection. In practice, passwords are printed in the photo pages of passports and they are frequently shared via instant messaging, emails or even uploaded to the cloud. These passwords can also be guessed as they are sampled from a small domain.

The MRTD standard also proposes an Active Authentication Protocol (version 1). In order to prove its genuineness, the chip signs a challenge message chosen by the terminal. Although it allows the terminal to ensure that it is talking to the genuine chip but not a copy, it leaks a signature to the terminal and harms the privacy of the owner.

To address some of the problems above, German Federal Office for Information Security (BSI) has proposed eIDAS tokens with TR-03110 standard which is an improvement over the MRTD [2]. The first main contribution is the introduction of the Extended Access Control (EAC). One of the motivations behind the EAC proposal was to provide a better alternative for the terminal authentication based on a PKI instead of a shared password. The EAC consists of the Terminal Authentication and the Chip Authentication protocols. The Terminal Authentication (version 2) is a simple challenge-response protocol based on digital signatures. The Chip Authentication (version 2) is an authenticated key-exchange protocol based on a PKI and a fixed public-key of the chip. The security of both protocols is proven with respect to the Bellare-Rogaway model [3], which is weaker than eCK [4]. However, the privacy of the Chip Authentication is not addressed and neither formalized as a security game. It actually leaks undeniable and transferable information from the chip. Finally, the hash values of privacy-sensitive data groups are stored in a publicly readable security object called $\mathsf{EF.SO_D}$. Hence, if these data groups leak, $\mathsf{EF.SO_D}$ can be used as evidence of their validity [5].

Bichsel et al. [6] proposes an identification scheme based on smart cards. Their idea (henceforth referred to as eID) is implementation of an RSA-based anonymous credential scheme on smart cards. Each eID contains a credential and it can be used to sign arbitrary messages, e.g. a proof of having a certain age. In that sense, eID is treated as a trusted platform module (TPM), where tamper-resistance of smart cards acts as a core security assumption. Hence, if

tamper-resistance of a (single) smart card is circumvented, then the security of the whole identification scheme is threatened, i.e. the terminals will accept inaccurate statements or even fake identity information[3]. Even though eID scheme supports revocation of leaked credentials, it is unclear how to detect leakages in a timely manner and prevent malicious identification sessions during the transient periods, i.e. the interval from leakage of a credential to its revocation. Moreover, signatures released by an eID can be linked later if its credential is released (happens when an eID is revoked), and thus anonymous credential scheme destroys deniability in order to support a revocation scheme.

Overall, ICAO's design solves secure identification, but fails with respect to the aforementioned privacy goals of identification systems. On the other hand, eID of Bichsel et al. [6] provides a more generic solution through anonymous credentials, yet treating smart cards as TPM provides weaker security in identification, and at the same time prevents deniability.

Our Design Paradigm. Our design from scratch allows us to keep the privacy goals in mind at every step, in order to place *appropriate measures* encouraged by the GDPR. We refer to the identity document as *chip*, and any device which supports the same contactless communication technology as *terminal*. We adhere to the following list of rules:

1. Privacy-sensitive biometric data should be stored on chips but not on a central database. The data inside the chip must also be clearly separated with clear boundaries based on the sensitivity level.
2. The communication with the chip should be allowed only if the terminal can prove its authorization, (interchangeably if the chip can authenticate the terminal). The interactions from two different chips in the same domain[4] should look indistinguishable to unauthorized terminals.
3. The self-authenticating data, e.g. signatures verifiable with a public key, of the identity should never leave the chip. This authentication should be done via deniable zero-knowledge interactive protocols. On the contrary, both MRTD and EAC disclose signatures on all data groups with the document security object $\mathsf{EF.SO_D}$[5] as discussed by Monnerat et al. [5].
4. If a transferable proof is leaked from the chip, then it should not be publicly-verifiable and undeniable, e.g. a trusted third-party is required for verification. On the contrary, both MRTD and EAC releases publicly-verifiable undeniable proofs with $\mathsf{EF.SO_D}$.
5. Any biometric comparison must be done on the chip if computational resources of the chip permits. Otherwise it must be strictly restricted to au-

[3] In comparison, in our scheme if tamper-resistance of a single device is violated, only the data contained in the card is affected. It does not lead to a system-wide collapse of identification.

[4] We cannot guarantee that two chips with different hardware or software configuration remain indistinguishable.

[5] In practice, this means that anyone who can get hold of passport can extract undeniable, transferable, universally-verifiable proofs of its identity with the help of a NFC reader e.g. an Android phone.

thorized terminals. The EAC strengthens the MRTD with the Terminal Authentication Protocol.

6. Interactions should be deniable from the chip's perspective. No terminal should be able to store information which can be later used to prove a previous interaction of the chip, e.g. the owner was at given location at some specific time[6]. On the contrary, interactions with the MRTD (even when enhanced by the EAC) cannot be denied because of $\mathsf{EF.SO_D}$.

Deniability remains vulnerable to adversaries in the trusted agent model [7]: if a tamper-proof device implements the terminal and outputs a signed result and if it can be proven that the signing key never left this device, then any protocol leaks undeniable evidence. This is an inherent limitation to deniability. We exclude this attack model in our contribution.

Our suite does not necessarily address the scenarios involving passports and border controls, but instead provides a more generic treatment to the identification problem. Namely, our design can be used by governments to identify their citizens, hospitals to identify patients, and companies to issue cards to its employees. For this reason, our main goal is to improve privacy, ensure that the identification session is deniable and no evidence leaks. The use cases where a user wants to obtain an evidence for a successful session, e.g. a tourist proving a valid entry/exit to/from a country, falls outside the scope of our work.

Section 2 defines our notation and common cryptographic primitives. In Section 3, we give the high level picture of our design. The detailed descriptions of our protocols are in Section 4 (strong path) and 5 (weak path). The results of a prototype implementation can be found in Section 6. Security models and the proofs of our protocols are presented in Appendix A.

2 Preliminaries

2.1 Notation

We use $\mathcal{G} = (p, a, b, G, G_2, G_3, q, h)$ to represent elliptic curve (EC) defined over $\mathbb{F}_p$ with $y^2 = x^3 + ax + b$ such that G generates a group with a prime order q, and the cofactor is h. We use additive notation for groups. $\mathcal{O}$ denotes the neutral element and $\langle G \rangle$ denotes the subgroup generated by G. G_2, G_3 are additional generators uniformly sampled from $\langle G \rangle \setminus \mathcal{O}$. We further assume that membership to this subgroup is easily verifiable, i.e. given a point R on the elliptic curve, we can decide whether $R \in \langle G \rangle \setminus \mathcal{O}$ by checking $q \cdot R = \mathcal{O}$ and $R \neq \mathcal{O}$.

We denote bitstring concatenation operator with '|'. For encoding a field element x, we use fixed-length encoding, i.e. each $x \in \mathbb{F}_p$ is mapped to $\lceil \log_2 p \rceil$ bits exactly by left-padding with zeros. In our notation, we omit this conversion and leave it implicit, e.g. $\mathsf{H}(x)$ means binary-encoding of the field element x is fed

[6] More formally, any information obtained by the terminal after interacting with the chip should remain indistinguishable from some simulated information obtained without interaction.

to hash function H. Similarly, conversions from bitstrings to integers are implicit. We assume that a point R on the elliptic curve is represented by (x, y). We consider the binary encoding of R as the concatenation of x and y as bitstrings. We denote with $(R)_{\mathbb{X}}$ the x-coordinate of the point R on the elliptic curve. With $x \leftarrow\!\!\$\, \mathcal{D}$, we mean that x is uniformly sampled from the domain $\mathcal{D}$. With $x \leftarrow\!\!\$\, A(y)$, we mean that the algorithm A takes y as input along with some implicit random coins and returns x.

2.2 PKIs and Entities

We rely on Public-Key Infrastructures (PKI) to bootstrap trust among entities. The public/private keys of an entity A are denoted by $\mathsf{A}_{\mathsf{pub}}/\mathsf{A}_{\mathsf{prv}}$. When a higher level authority A issues a certificate for a lower level entity B, we denote this certificate by $\langle \mathsf{A}, \mathsf{B} \rangle$. We extend the same notation to chains of certificates, e.g. $\langle \mathsf{A}, \mathsf{C} \rangle$ is a shorthand for the combination of $\langle \mathsf{A}, \mathsf{B} \rangle$ and $\langle \mathsf{B}, \mathsf{C} \rangle$.

We assume the following three operations below can be performed on a given certificate $\langle \mathsf{A}, \mathsf{B} \rangle$: (1) verify the certificate using $\mathsf{A}_{\mathsf{pub}}$, by "verify $\langle \mathsf{A}, \mathsf{B} \rangle$", (2) extract the public key, by "extract $\mathsf{B}_{\mathsf{pub}} \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$", (3) extract the expiration date, by "extract $t_e \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$"[7]. They can also be applied to chains of certificates.

Our design consists of the following PKIs:

- *Identity Signer PKI* is managed by an autonomous authority whose task is to produce signatures for identity cards, e.g. the government. $\mathsf{CA\text{-}ID}$ is the root certificate authority, identity signers IS_i are sub-authorities, and identity documents are leaf-level entities.
- *Terminal PKI* issues/authorizes devices (called terminals) who support the same communication technology with chips. Authorized terminals can communicate with chips and eventually decide whether the presented identity is valid or not. $\mathsf{CA\text{-}T}$ is the root certificate authority, $\mathsf{Term\text{-}S}$ is a sub-level authority, and terminals T are leaf-level entities.

Furthermore we employ the following entities:

- *Time server* TS is the entity whose task is to provide secure time information to chips, so that chips can check whether the valid certificate presented by the terminal has not expired.
- *Confirmer* Cnf is a third-party authority whose task is to check whether given message authentication code (MAC) is valid or not. It is trusted from the privacy perspective.
- *Chips* are tamper-resistant devices with small secure memory (henceforth denoted with C).
- *Terminals* are devices which communicate with chips (denoted with T).

[7] We consider the earliest expiration date from all certificates in the chain.

2.3 Primitives

Hash Function. We use $\mathsf{SHA256}$ [8]. In our security proofs, each hash function is treated as an independent random oracle. Hence, we need to separate each of them with a fixed padding: $\mathsf{H}_i(x) := \mathsf{SHA256}(\mathsf{encode}(i)|x)$ where encode maps integer i to a single byte value.

Pseudorandom Function, Message Authentication Code. For MAC and PRF, we use HMAC [9], where the underlying hash function is $\mathsf{SHA256}$ [8].

Authenticated Encryption [10]. A (variable-length) authenticated encryption is a symmetric encryption scheme comprising two algorithms $(\mathsf{Enc}, \mathsf{Dec})$. Enc takes (K, IV, H, pt) as input, that is a symmetric key K, initialization vector IV, authenticated header H and message pt respectively, and returns a ciphertext ct. Dec takes (K, IV, H, ct) as input and returns either a special failure symbol $\perp$ or a message pt. The security of the AE primitive is formally defined by Rogaway [10]. Our practical choice for AE is AES-GCM [11].

Schnorr Signatures [12]. The Schnorr signature scheme over $\mathcal{G}$ is a tuple of algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$ with arbitrary-length bitstrings as the message domain. Below, sk, vk denotes signing, verifying key pair respectively; σ denotes the signature. All algorithms have access to $\mathcal{G}$ which includes G and q.

$\mathsf{Kg}(\mathcal{G})$:	$\mathsf{Sign}(\mathsf{sk}, m)$:	$\mathsf{Ver}(\mathsf{vk}, m, \sigma)$:
$\mathsf{sk} \leftarrow_\$ \mathbb{Z}_q^*; \mathsf{vk} \leftarrow \mathsf{sk} \cdot G$	$k \leftarrow_\$ \mathbb{Z}_q; R \leftarrow k \cdot G$	$\sigma \rightarrow (s, R)$
return $(\mathsf{sk}, \mathsf{vk})$	$h \leftarrow \mathsf{H}_1(m\|R)$	$h \leftarrow \mathsf{H}_1(m\|R)$
	$s \leftarrow (k - \mathsf{sk} \cdot h) \bmod q$	$R' = s \cdot G + h \cdot \mathsf{vk}$
	$\sigma \leftarrow (s, R)$	**return** $[R' = R]$ //true or false
	return σ	

A Schnorr tuple (vk, m, σ) is valid if $\mathsf{Ver}(\mathsf{vk}, m, \sigma) = \mathsf{true}$. Assuming that the discrete logarithm problem is hard, this signature scheme is existentially unforgeable with chosen message attack in the random oracle model, as proven by Pointcheval and Stern [13].

For our Data & Chip Authentication Protocol (Section 4.2), we require that the signature to be of Schnorr type. However, it is possible to use other elliptic-curve-based signature schemes for implementing PKIs. Our efficiency calculations are based on the assumption that signing takes one scalar multiplication on $\mathcal{G}$, where verification takes two. Hence, verifying a certificate chain takes $2d$ scalar multiplications, where d is the depth of the PKI.

3 Suite Overview

In this section, we explain the high-level organization of the protocols. The data contained in the chip is stored at enrollment and is never updated. We categorize the information inside the chip into few data groups depending on their privacy sensitivity:

- DG_1: It contains all necessary public parameters for our protocols. It contains no privacy-sensitive data that can be exploited for tracking.

– DG_2: The data contained in this group is related to the basic identity information of the holder. It is revealed only after the successful completion of either the Strong Access Control Protocol (SAC) (Section 4.1) or the Weak Access Control Protocol (WAC) (Section 5.1).
– DG_3: The data contained in this group is privacy-sensitive, e.g. biometric templates. It can only be given to the terminal if the session is established through the SAC (Section 4.1).
– DG_4: The data contained in this group is highly privacy-sensitive such as signatures of the identity, and never leaves the chip.

In Table 1, we give the actual contents of the data groups. $\mathsf{DG}_{2\text{-}3}$ is a shorthand for the combination of DG_2 and DG_3, and it is authenticated by the identity signer IS such that $\sigma_{\mathsf{DG}_{2\text{-}3}}$ is a Schnorr signature over $\mathsf{DG}_{2\text{-}3}$.

Table 1. Content of data groups. The password pwd lays printed on the front page.

Printed	public	pwd (password for WAC)
DG_1	public	$\mathcal{G}$ (curve domain parameters), $\mathsf{CA\text{-}T}_{\mathsf{pub}}$ (public key for the terminal authority), $\mathsf{TS}_{\mathsf{pub}}$ (public key for the time server), ℓ (nonce length)
DG_2	personal	ID (identity info e.g. full name), $\mathsf{u}_{\mathsf{chip}}$ (identifier value tied to $\mathsf{K}_{\mathsf{chip}}$ for DCA)
DG_3	sensitive	$\mathsf{BIO}_{\mathsf{ID}}$ (biometric reference template), $\langle \mathsf{CA\text{-}ID}, \mathsf{IS} \rangle$ (certificate chain of IS)
DG_3	chip-only	$\sigma_{\mathsf{DG}_{2\text{-}3}}$ (signature over $\mathsf{DG}_{2\text{-}3}$), $\mathsf{K}_{\mathsf{chip}}$ (MAC key for CDA), G_{pwd2} (precomputed $\mathsf{pwd}\cdot G_2$), G_{pwd3} (precomputed $\mathsf{pwd}\cdot G_3$)

The protocol works in two different paths as shown in Figure 1: strong and weak paths. The control flow can follow either one based on the choice of the terminal. The holder has no control on this choice but he can accept or refuse to have his biometry scanned, which is necessary in the strong path[8]. If either T or C encounters an error, then the session is aborted immediately.

The strong path continues in the following order:

SAC: C proceeds to verify the authorization of T. On success, both C and T end up deriving a shared secret key K.
TCH: C requests a reliable time information from TS and the communication between the two is relayed through T.
SSE: C and T use the shared key K to establish secure communication. After this protocol, any following interaction between the two is encrypted.
$\mathsf{DG}_{2\text{-}3}$: C transfers DG_2 and DG_3 to T.
DCA: C proves the authenticity of $\mathsf{DG}_{2\text{-}3}$ and that the chip is genuine.

[8] We cannot guarantee that the biometric scanner is not malicious and will not store or share the real-time template obtained from the user during the session. Each time a user accepts (or coerced) to give a biometric sample, she inherently faces the risk that the scanner will store the real-time sample.

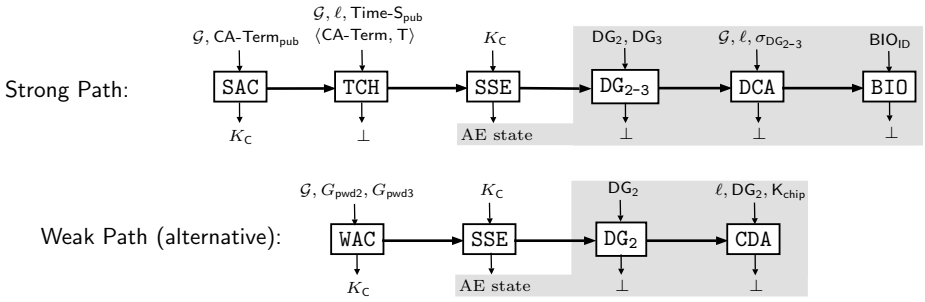

Fig. 1. The high-level picture of the suite. The protocols with gray background are encapsulated by the secure session. Input/output of the chip for protocols are shown.

`BIO`: T checks if the real-time biometric template matches the reference template $\mathsf{BIO_{ID}}$ included in $\mathsf{DG_3}$ by running the biometric matching algorithm. On success, T successfully accepts the presented identity and terminates.

In the strong path, C ensures that T is authorized for access by the authority. This allows C to trust T and release its biometric template for biometric matching. On the other side, T ensures that (1) the identity information on C is valid, (2) C is genuine (not a skimmed copy) and (3) the holder's biometric identity matches the one carried by C. The strong path requires the online connectivity between T and TS. All security and privacy goals of identification are fulfilled in the strong path.

The weak path continues in the following order:

`WAC`: C proceeds to verify that T knows the password. On success, both C and T end up deriving a shared secret key K.
`SSE`: Same as in the strong path.
$\mathsf{DG_2}$: C transfers only $\mathsf{DG_2}$ to T (but not $\mathsf{DG_3}$).
`CDA`: C proves the authenticity of $\mathsf{DG_2}$ with a confirmer-based proof. T terminates after receiving the proof. Later, T needs to interact with the remote confirmer Cnf to verify the authenticity of the proof. On success, T concludes that the presented identity is valid and that C is genuine. T never receives access to $\mathsf{DG_3}$ in this path and no biometric matching is made.

The weak path provides a more lightweight solution and does not require T to have online connection to TS (or the revocation server) during the interaction. C ensures that T has a visual access to the front page of the identity document and thereby knows the password. This does not necessarily mean that T is authorized by the identification system. It could be a rogue device who had access to the front copy of the identity document. Therefore, C shares its basic identity information with T along with the confirmer-verifiable proof of the identity. Also, C does not release its biometric template. On the other side, T cannot decide on the validity of the identity information by itself. T keeps the offline proof and later requests from the trusted confirmer to verify the proof. Only when the

confirmer validates the proof, T is fully convinced that the presented identity is valid, and the interacting chip is genuine. There is no guarantee that the holder's biometric identity matches the one carried by C. Therefore, the weak path cannot be used to securely identify the holder. It can only be used to authenticate the document. This sort of verification could be useful for mobile terminals that does not have a reliable connection to TS (or the revocation server), e.g. in a refugee camp.

4 Strong Path

4.1 Strong Access Control (SAC)

Given the current wide-spread use of NFC-enabled smart phones, attackers have a large number of devices that can be exploited to act as terminals. To prevent arbitrary devices from communicating with chips, we rely on a PKI. When T prompts C to establish a communication session, T is asked to provide a chain of *valid* certificates as an attestation. The chip comes embedded with $\mathsf{CA}\text{-}\mathsf{T}_{\mathsf{pub}}$. On success, both C and T end up with a shared secret, that is the key material for the following symmetric session. Hence, the access control protocol serves two purposes: (1) C is convinced that it is talking to an authorized terminal, (2) C and T derive the key material.

The full description of our *strong access control* (SAC) is given in Figure 2. The security model and proofs are given in Appendix A.1. Briefly, the security argument states that (1) the passive adversary cannot distinguish the derived session key from random, and (2) the active adversary cannot make C accept by modifying the exchanged messages. The final derived key material remains secure even if $\mathsf{T}_{\mathsf{prv}}$ is given to the adversary after the protocol is completed, i.e. forward-secrecy.

Efficiency: The chip has to perform 1 scalar multiplication for checking the subgroup membership of R[9]; 4 multiplications for computing X_1, X_2, K_{C}; and $2d_{\mathsf{T}}$ multiplications for verifying the certificate chain. d_{T} is the depth of the Terminal PKI ($d_{\mathsf{T}} = 2$ is a reasonable choice). The chip does $5 + 2d_{\mathsf{T}}$ multiplications in total[10]. The terminal does 5 multiplications.

Deniability: Because the protocol does not require a private input from C side, any execution transcript can be later denied by C. I.e. C can claim that T produced the transcript by simulation. For terminals, we do not need to consider deniability; as they do not carry any personal information.

Forward-secrecy: Gathered executions will not be useful, even if the key of the terminal is compromised. This assumes that the values x_2 and r are destroyed

[9] Note that asserting $R \in (\langle G \rangle \setminus \mathcal{O})$ is done by $(R \neq \mathcal{O}) \wedge (q \cdot R = \mathcal{O})$; so each group membership check takes 1 multiplication.

[10] As a speed up technique, the chip can store the hash of the previously seen certificate chains in its non-volatile memory. If the chip interacts with a small set of terminals, simple hash-then-compare saves $2d_{\mathsf{T}}$ multiplications in each interaction.

SAC: Strong Access Control Protocol

Chip C		**Terminal** T
IN: $\mathcal{G}, \mathsf{CA\text{-}T}_{\mathsf{pub}}$		**IN:** $\mathcal{G}, \mathsf{T}_{\mathsf{prv}}, \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$
verify $\langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$	$\xleftarrow{\langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle}$	
extract $\mathsf{T}_{\mathsf{pub}} \leftarrow \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$		
assert $R \in \langle G \rangle \setminus \mathcal{O}$	$\xleftarrow{R}$	$r \leftarrow\!\!\$\, \mathbb{Z}_q^*;\ R \leftarrow r \cdot G$
$x_1 \leftarrow\!\!\$\, \mathbb{Z}_q^*;\ X_1 \leftarrow x_1 \cdot G$		
$x_2 \leftarrow\!\!\$\, \mathbb{Z}_q^*;\ X_2 \leftarrow x_2 \cdot G$	$\xrightarrow{X_1, X_2}$	**assert** $X_1, X_2 \in \langle G \rangle \setminus \mathcal{O}$
$K_{\mathsf{C}} \leftarrow (x_1 \cdot \mathsf{T}_{\mathsf{pub}} + x_2 \cdot R)_{\mathbb{X}}$		$K_{\mathsf{T}} \leftarrow (\mathsf{T}_{\mathsf{prv}} \cdot X_1 + r \cdot X_2)_{\mathbb{X}}$
erase x_2		**erase** r
$K'_v \leftarrow \mathsf{H}_2(K_{\mathsf{C}} \vert \mathsf{trans}_{\mathsf{C}})$	$\xleftarrow{K_v}$	$K_v \leftarrow \mathsf{H}_2(K_{\mathsf{T}} \vert \mathsf{trans}_{\mathsf{T}})$
assert $K'_v = K_v$		
OUT: K_{C}		**OUT:** K_{T}

Fig. 2. T's key pair is such that $(\mathsf{T}_{\mathsf{prv}}, \mathsf{T}_{\mathsf{pub}}) \in \mathbb{Z}_q^* \times (\langle G \rangle \setminus \mathcal{O})$. On "assert" and "verify", parties abort with failure if given statement is false or given certificate/signature is invalid. "erase" means given parameters must be destroyed. $\mathsf{trans}_{\mathsf{T}}$, $\mathsf{trans}_{\mathsf{C}}$ are the local transcripts, i.e. the concatenation of R, X_1, X_2 seen by each party.

by the chip and the terminal following the "erase" instruction in SAC. If the Gap Diffie-Hellman Problem remains intractable in $\mathcal{G}$, then all past sessions will be protected against key leakages of the future. More lightweight solution without forward-secrecy can be obtained by letting the chip pick $x_2 = 0$ and removing **assert** $X_2 \in \langle G \rangle \setminus \mathcal{O}$ on the terminal side.

4.2 Data & Chip Authentication (DCA)

Data Authentication. C is obliged to prove to T that $\mathsf{DG_{2\text{-}3}} = \mathsf{DG_2} | \mathsf{DG_3}$ is valid, i.e. has a signature verifiable w.r.t. the identity server PKI. We propose an interactive zero-knowledge (ZK) protocol for proving the authenticity of the presented identity that is similar to offline non-transferable authentication protocol [14]. Briefly, C carries a Schnorr tuple $(\mathsf{IS}_{\mathsf{pub}}, \mathsf{DG_{2\text{-}3}}, \sigma_{\mathsf{DG_{2\text{-}3}}})$ along with the certificate chain $\langle \mathsf{CA\text{-}ID}, \mathsf{IS} \rangle$. T knows $\mathsf{CA\text{-}ID}_{\mathsf{pub}}$ in advance, and also receives $\mathsf{DG_{2\text{-}3}}$ as described in Figure 1. Then, C and T follow the interaction described in Figure 3; in which C proves the knowledge of a valid signature, that is $\sigma_{\mathsf{DG_{2\text{-}3}}}$. During this interaction, R point of the signature $\sigma_{\mathsf{DG_{2\text{-}3}}} = (s, R)$ is revealed to the verifier in clear. This allows us to develop a very efficient Σ-protocol, which yields a zero-knowledge proof of knowledge interactive protocol with a commitment scheme under the random oracle assumption. The security of this generic commitment based transformation is proven by Monnerat et al. [15], and our proofs for Σ-protocol can be found in Appendix A.2. Preferring interactive

DCA: Data & Chip Authentication Protocol

Chip C		**Terminal** T
IN: $\mathcal{G}, \ell, \sigma_{\mathsf{DG}_{2\text{-}3}}$		**IN:** $\mathcal{G}, \ell, \mathsf{DG}_{2\text{-}3}, \mathsf{CA\text{-}ID}_{\mathsf{pub}}$
$(s, R) \leftarrow \sigma_{\mathsf{DG}_{2\text{-}3}}$		**extract** $\langle \mathsf{CA\text{-}ID}, \mathsf{IS} \rangle \leftarrow \mathsf{DG}_{2\text{-}3}$
		extract $\mathsf{IS}_{\mathsf{pub}} \leftarrow \langle \mathsf{CA\text{-}ID}, \mathsf{IS} \rangle$
		verify $\langle \mathsf{CA\text{-}ID}, \mathsf{IS} \rangle$
		$r \leftarrow\$ \{0,1\}^{\ell}$; $v \leftarrow\$ \mathbb{Z}_q^*$
	$\xleftarrow{h}$	$h = \mathsf{H}_5(r \mid v)$
$u \leftarrow\$ \mathbb{Z}_q$; $U \leftarrow u \cdot G$	$\xrightarrow{U, R}$	**assert** $R \in (\langle G \rangle \setminus \mathcal{O})$
assert $(v \in \mathbb{Z}_q^*) \wedge (r \in \{0,1\}^{\ell})$	$\xleftarrow{r, v}$	**assert** $U \in \langle G \rangle$
assert $h = \mathsf{H}_1(r \mid v)$		
$s' \leftarrow (s + v \cdot u) \bmod q$	$\xrightarrow{s'}$	**assert** $s' \in \mathbb{Z}_q$
		$e \leftarrow \mathsf{H}_1(\mathsf{DG}_{2\text{-}3} \mid R)$
		assert $(s' \cdot G + e \cdot \mathsf{IS}_{\mathsf{pub}} = R + v \cdot U)$

Fig. 3. The identity signature $\sigma_{\mathsf{DG}_{2\text{-}3}}$ over $\mathsf{DG}_{2\text{-}3}$ is actually a tuple $(s, R) \in \mathbb{Z}_q^* \times \langle G \rangle$.

proofs over signature schemes has two benefits in our scenario. First, the transcript remains deniable from the chip's perspective, hence he can later deny the interaction, even if the transcript is released by the terminal[11]. Secondly, the actual signature is never released to the terminal, which prevents the terminal from obtaining publicly-provable and transferable proof over the identity.

Efficiency: The chip only performs one scalar multiplication. The terminal performs $4 + 2d_{\mathsf{IS}}$ multiplications, where d_{IS} is the depth of the Identity Signer PKI ($d_{\mathsf{IS}} = 2$ is a reasonable choice).

Chip Authentication. With the assumption that the memory of C is secure and the ZK property of DCA, $\sigma_{\mathsf{DG}_{2\text{-}3}}$ does not leak. Hence, the proof of knowledge of $\sigma_{\mathsf{DG}_{2\text{-}3}}$ is also a proof that C is genuine.

4.3 Time Check Protocol (TCH) & Revocation Check Protocol (REV)

To protect against compromise of long-term secret keys, we need a secure revocation verification in the Terminal PKI.

First remedy is the revocation-through-time. Briefly, we enforce that the terminal server issues certificates only for a limited amount of time. The compromised certificates remain useful only for a short period of time. All required by the chip is a trusted clock to gradually expire certificates. As C usually has no such clock, it must query a time server. Concretely, C prompts T with a

[11] We exclude attacks from the trusted agent model [7].

TCH: Time Check Protocol

Chip C		**Time Server** TS
IN: $\mathcal{G}, \ell, \mathsf{TS}_{\mathsf{pub}}, \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$		**IN:** $\mathcal{G}, \ell, \mathsf{TS}_{\mathsf{prv}}$
extract $t_e \leftarrow \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$		
$n \leftarrow\!\!\$\ \{0,1\}^\ell$	$\xrightarrow{\quad n \quad}$	**assert** $n \in \{0,1\}^\ell$
		$t \leftarrow$ current time
		$h \leftarrow \mathsf{H}_3(t \mid n)$
$h' \leftarrow \mathsf{H}_3(t \mid n)$	$\xleftarrow{\quad t, \sigma \quad}$	$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{TS}_{\mathsf{prv}}, h)$
verify $(\mathsf{TS}_{\mathsf{pub}}, h', \sigma)$		
assert $t_e > t$		

REV: Revocation Check Protocol

Chip C		**Rev. Server** Rev
IN: $\mathcal{G}, \ell, \mathsf{Rev}_{\mathsf{pub}}, \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$		**IN:** $\mathcal{G}, \ell, \mathsf{Rev}_{\mathsf{prv}}, \mathsf{CA\text{-}T}_{\mathsf{pub}}$
$n \leftarrow\!\!\$\ \{0,1\}^\ell$	$\xrightarrow{\quad n, \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle \quad}$	**assert** $n \in \{0,1\}^\ell$
		verify $\langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$
		revoked? $\langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$
		$h \leftarrow \mathsf{H}_3(n \mid \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle)$
$h' \leftarrow \mathsf{H}_3(n \mid \langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle)$	$\xleftarrow{\quad \sigma \quad}$	$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{Rev}_{\mathsf{prv}}, h)$
verify $(\mathsf{Rev}_{\mathsf{pub}}, h', \sigma)$		

Fig. 4. The instruction "revoked?" asserts that the input certificate is not revoked. $t_e > t$ means the certificate has not expired yet.

challenge value n to produce a signed timestamp value t (see Figure 4). T acts as a proxy, relaying the communication between C and TS. Then, TS provides a signature over the timestamp t and nonce n to prevent replay attacks.

Alternatively, we can use OCSP-based approach to check the revocation status of certificates with the REV described in Figure 4. In this case, the public key of the revocation server Rev is stored in C at enrollment time. Using REV instead of TCH could also allow us to outsource the verification of the certificate chain $\langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$ in SAC and save $2d_{\mathsf{T}}$ multiplications by trusting the revocation server. *Efficiency:* The chip has to verify the validity of one signature, which takes 2 scalar multiplications over $\mathcal{G}$. The time server does $1 + 2d_{\mathsf{T}}$ multiplications.

4.4 Secure Communication (SSE)

We denote by $K = K_{\mathsf{C}} = K_{\mathsf{T}}$ the common secret which is derived by either SAC or WAC. We create an authenticated and confidential symmetric channel based on K. For this, we use a variable-length authenticated encryption scheme AE which consists of two algorithm (Enc, Dec) as explained in Section 2.3. Each peer keeps

WAC: Weak Access Control Protocol

Chip C		**Terminal** T
IN: $\mathcal{G}, G_{\mathsf{pwd2}}, G_{\mathsf{pwd3}}$		**IN:** $\mathcal{G}, \mathsf{pwd}$
$a \leftarrow\$ \mathbb{Z}_q^*;\ A \leftarrow a \cdot G$		
$M \leftarrow (A + G_{\mathsf{pwd2}})$	$\xrightarrow{M}$	$b \leftarrow\$ \mathbb{Z}_q^*;\ B \leftarrow b \cdot G$
		$L \leftarrow B + \mathsf{pwd} \cdot G_3$
		$K_{\mathsf{T}} \leftarrow (b \cdot M - b \cdot \mathsf{pwd} \cdot G_2)_{\mathbb{X}}$
$K_{\mathsf{C}} \leftarrow (a \cdot (L - G_{\mathsf{pwd3}}))_{\mathbb{X}}$	$\xleftarrow{L, K_v}$	$K_v \leftarrow \mathsf{H}_6(K_{\mathsf{T}} \vert \mathsf{trans}_{\mathsf{T}})$
assert $K_v = \mathsf{H}_6(K_{\mathsf{C}} \vert \mathsf{trans}_{\mathsf{C}})$		
OUT: K_{C}		**OUT:** K_{T}

Fig. 5. T obtains pwd from the front page of the identity document. $\mathsf{trans}_{\mathsf{T}}$, $\mathsf{trans}_{\mathsf{C}}$ are the local transcripts, i.e. the concatenation of M, L seen by each party.

an active state $\mathsf{st} = (\mathsf{K}_{\mathsf{AE}}, i_0, i_1)$ for the live session where K_{AE} is the authenticated encryption/decryption key, and i_0, i_1 are nonce counters for each sent and received messages respectively. A peer has access to the pair of algorithms Send and Recv. Send takes (st, pt) as input and returns (st, ct), i.e. each send/receive updates the corresponding internal counter of the state. Similarly, Recv takes (st, ct) and returns (st, pt). Authenticated header in not used (so we put $\perp$ as input below). In case of $pt = \perp$, the receiver halts. The full description is given below:

$\mathsf{Send}(\mathsf{st}, pt)$:	$\mathsf{Recv}(\mathsf{st}, ct)$:	$\mathsf{Init}_b(\mathsf{K})$:
$\mathsf{st} \rightarrow (\mathsf{K}_{\mathsf{AE}}, i_0, i_1)$	$\mathsf{st} \rightarrow (\mathsf{K}_{\mathsf{AE}}, i_0, i_1)$	$\mathsf{K}_{\mathsf{AE}} \leftarrow \mathsf{H}_4(\mathsf{K} \vert \mathsf{encode}(1))$
$ct \leftarrow \mathsf{Enc}(\mathsf{K}_{\mathsf{AE}}, i_0, \perp, pt)$	$pt \leftarrow \mathsf{Dec}(\mathsf{K}_{\mathsf{AE}}, i_1, \perp, ct)$	$\mathsf{IV}_0 \leftarrow \mathsf{H}_4(\mathsf{K} \vert \mathsf{encode}(2))$
$\mathsf{st} \leftarrow (\mathsf{K}_{\mathsf{AE}}, i_0 + 1, i_1)$	**if** $pt = \perp$ **then abort**	$\mathsf{IV}_1 \leftarrow \mathsf{H}_4(\mathsf{K} \vert \mathsf{encode}(3))$
return (st, ct)	$\mathsf{st} \leftarrow (\mathsf{K}_{\mathsf{AE}}, i_0, i_1 + 1)$	$\mathsf{st} \leftarrow (\mathsf{K}_{\mathsf{AE}}, \mathsf{IV}_b, \mathsf{IV}_{\bar{b}})$
	return (st, pt)	**return** st

The initial state st is bootstrapped with algorithm Init_b (above) which takes the key material K as input. We swap the order of counters to synchronize the initial states. Hence, C runs Init_0 and T runs Init_1. encode is a byte-encoding function explained in Section 2.3 and $\bar{b}$ is the complement of b such that $\{b, \bar{b}\} = \{0, 1\}$.

5 Weak Path

5.1 Weak Access Control (WAC)

WAC is an alternative way of ensuring that a connection attempt is from an authorized terminal is based on a knowledge of a password instead of a PKI. This is done with PACE in the MRTD standard [1]. The password usually lays printed on the document and can be either typed manually by the inspector or directly read with visual scanning of the document.

Password-based authentication is already a well-established topic in the academic literature, as surveyed by Boyd and Mathuri [16]. The rigorous formalization of security can be found by Bellare et al. [17]. Nevertheless, password-based authenticated key exchange protocols did not receive its merit in practice, especially troubled by patents and legal ambiguities. Our own version in Figure 5 is a reduced version of SPAKE which is available for royalty-free use, and has been drafted for a standardization [18].

In password-based access control protocols, passwords are chosen from a small domain, and it is feasible for an adversary to enumerate each possible password in this set. Therefore, the adversary always has an inevitable not negligible chance of guessing the password. In terms of security, the usual expectations from such protocols are that an adversary who obtains multiple execution transcripts of the protocol should not be able to eliminate any candidate password from the set and an adversary who is performing an active attack should only eliminate a single candidate password from the domain, and strictly no more.

We specify the dictionary $\mathcal{D}$ as the password domain such that $\log_2 |\mathcal{D}|$ is far smaller than the security parameter λ. Each chip receives a randomly sampled password $\mathsf{pwd} \in \mathcal{D}$ at enrollment.

Efficiency: C and T do 2 and 4 scalar multiplications respectively. C also does 2 group additions.

5.2 Confirmer Data & Chip Authentication (CDA)

After a weakly secure session is established, C releases DG_2 as seen in Figure 1. In order to convince T, without any publicly-verifiable proof of valid identity, C produces a MAC that can only be verified by Cnf. Therefore, without the help of Cnf, T can only learn the basic identity information without any proof.

We require that Cnf is in possession of a master secret $\mathsf{K}_{\mathsf{Cnf}}$ that is sampled uniformly from the key domain of the pseudorandom function PRF. During enrollment, each chip receives a unique identity and key pair $(\mathsf{u}_{\mathsf{chip}}, \mathsf{K}_{\mathsf{chip}})$ from Cnf such that $\mathsf{K}_{\mathsf{chip}} = \mathsf{PRF}(\mathsf{K}_{\mathsf{Cnf}}, \mathsf{u}_{\mathsf{chip}})$. When C and T interact, both of them add nonces to the protocol to guarantee freshness. By using timestamps, T commits to a specific time interval, i.e. $[t, t + \Delta_t]$, in which the authentication code will be useful. On the other side of the MAC verification, the confirmer decides on the value of Δ_t and applies this time interval policy (see Figure 6).

This protocol allows weakly transferable proofs from the terminal side, but they are not publicly verifiable. Moreover, the confirmation protocol makes sure that a certain terminal can be kept accountable for verification requests. The proof σ_t is not deniable to the confirmer Cnf. If Cnf is honest, it provides verification access for a limited time Δ_t, thus σ_t is deniable to anyone else after this time. If Cnf is malicious, σ_t is deniable in the sense that it could have been forged by Cnf itself. We exclude attacks in the trusted agent model [7] which are inherent to deniability.

Similar to DCA presented in Section 4.2, we place $\mathsf{K}_{\mathsf{chip}}$ into the secure memory to obtain chip authentication at the same time. Since $\mathsf{K}_{\mathsf{chip}}$ never leaves the chip,

CDA1: Confirmer Data & Chip Auth. Protocol 1

Chip C		**Terminal** T
IN: $\ell, \mathsf{DG}_2, \mathsf{K}_{\mathsf{chip}}$		**IN:** ℓ, DG_2
$m' \leftarrow \mathsf{H}_7(\mathsf{DG}_2)$		**extract** $\mathsf{u}_{\mathsf{chip}} \leftarrow \mathsf{DG}_2$
		$t \leftarrow$ current time
		$m \leftarrow \mathsf{H}_7(\mathsf{DG}_2)$
assert $n_{\mathsf{T}} \in \{0,1\}^\ell$	$\xleftarrow{t, n_{\mathsf{T}}}$	$n_{\mathsf{T}} \leftarrow\!\!\$\ \{0,1\}^\ell$
$n_{\mathsf{C}} \leftarrow\!\!\$\ \{0,1\}^\ell$		
$\sigma_t \leftarrow \mathsf{MAC}(\mathsf{K}_{\mathsf{chip}}, n_{\mathsf{T}} \vert t \vert n_{\mathsf{C}} \vert m')$	$\xrightarrow{n_{\mathsf{C}}, \sigma_t}$	**assert** $n_{\mathsf{C}} \in \{0,1\}^\ell$
		OUT: $m, n_{\mathsf{T}}, t, n_{\mathsf{C}}, \mathsf{u}_{\mathsf{chip}}, \sigma_t$

CDA2: Confirmer Data & Chip Auth. Protocol 2

Terminal T		**Confirmer** Cnf
IN: $m, n_{\mathsf{T}}, t, n_{\mathsf{C}}, \mathsf{u}_{\mathsf{chip}}, \sigma_t$		**IN:** $\mathsf{K}_{\mathsf{Cnf}}, \Delta_t$
	$\leftarrow$ **TLS** $\rightarrow$	
	$\xrightarrow{m, n_{\mathsf{T}}, t, n_{\mathsf{C}}, \mathsf{u}_{\mathsf{chip}}, \sigma_t}$	
		$\mathsf{K}_{\mathsf{chip}} \leftarrow \mathsf{PRF}(\mathsf{K}_{\mathsf{Cnf}}, \mathsf{u}_{\mathsf{chip}})$
		$\sigma'_t \leftarrow \mathsf{MAC}(\mathsf{K}_{\mathsf{chip}}, n_{\mathsf{T}} \vert t \vert n_{\mathsf{C}} \vert m)$
		assert $\sigma'_t = \sigma_t$
		$t_{\mathsf{now}} \leftarrow$ current time
		assert $t_{\mathsf{now}} \in [t, t + \Delta_t]$
assert $b = 1$	$\xleftarrow{b}$	$b \leftarrow 1$

Fig. 6. We assume that T and Cnf can communicate through a mutually authenticated and confidential channel established through TLS.

a terminal who verifies the validity of this offline proof can also be convinced that it was interacting with the genuine chip.

6 Implementation

We implemented our protocols on smart cards. We used a biometric finger-vein reader with infrared camera attached to Raspberry Pi that communicates through the WebSocket protocol. We developed the card applets with the technology of java cards (JCs) provided by Oracle. The JC we used is NXP JCOP version 2.4.2 with 144KB of EEPROM and 7.5KB of RAM [19].

The general restrictions related to JCs are also summarized by Bichsel et al. [6], especially the fact that APIs related to group operations are quite limited and there are no big integer utilities. Since our design relies on elliptic curves

Table 2. The running time and the number of operations (in $\mathcal{G}$) for each protocol. d_T is the depth of the terminal PKI. SAC is implemented without forward-secrecy (requiring no group addition).

	Strong Path					**Weak Path**			
	SAC	TCH	SSE	$\mathsf{DG}_{2\text{-}3}$	DCA	WAC	SSE	DG_2	CDA
# of multiplications in $\mathcal{G}$	$5 + 2d_\mathsf{T}$	2	0	0	1	2	0	0	0
# of additions in $\mathcal{G}$	1	0	0	0	0	2	0	0	0
running time (in milliseconds)	1819			2815	2734	8765			

(compared to their RSA-based group that requires modulus to be larger in terms of bit size), single scalar multiplication in the group takes much less time in our case (respectively 0.6 seconds versus 4.3 seconds). As a conclusion, our seemingly more detailed and fine-grained protocols take less time to complete than that of eID in total [6].

For scalar multiplications we use elliptic curve Diffie-Hellman key exchange (ECDH) API. However, group additions become more tricky (no closely related API). Therefore, we used an external library called JCMathlib [20]. It performs memory management and contains useful elliptic curve and big integer utilities. Despite the fact that we found some bugs in the library, it is a handy tool to implement elliptic curve cryptography on smart cards.

In JC API, ECDH algorithm returns the x-coordinate of a general purpose scalar multiplication. For scalar multiplications in the protocols, we use the ECDH hardware implementation through API which takes around 600 milliseconds per operation. However, the point addition is done in software through JCMathlib, and each addition operation takes more than 3 seconds (much slower than a scalar multiplication). This results in a considerable delay in WAC-CDA compared to SAC-TCH-SSE (where SAC is without forward-secrecy and contains no addition operation). It should be noted that this is solely due to over-restricted API imposed by smart cards. In general, point additions can be done much faster than multiplication. We used the secp256r1 curve.

For the feature extraction algorithm, we used the maximum curvature method and for the matching algorithm, we used the Mirua Matching [21]. We integrated the biometric feature extraction and matching algorithm developed by IDIAP [22]. The algorithms are coded in python with reliable performance. They are easy to integrate with simple calls from the java code.

This implementation was a proof of concept. It was used as a demonstration during a science fair. We enrolled and matched about 300 people during the event. For this implementation, we simplified the protocols, mostly due to time constraints before the event. We used no forward secrecy in SAC (hence $x_2 = 0$). We did not implement TCH. Our PKI was flat (with depth 0). This way, only WAC required a point addition to be made by the chip. Finally, the biometric extraction and matching were done by the terminal.

7 Conclusion

All of our protocols are designed to be efficient and they only require elliptic curve based operations. On the chip side, the strong path takes $8 + 2d_{\mathsf{T}}$ scalar multiplications, and the weak path takes only 3 multiplications in total. On the terminal side, they take $10 + 2d_{\mathsf{IS}}$ and 4 multiplications respectively. In practice, $d_{\mathsf{IS}} = d_{\mathsf{T}} = 2$ could be a realistic choice.

Our main improvement over the existing MRTD and EAC standards are in the privacy direction. Essentially, any interaction from the identity document can later be denied, and no transferable information is leaked. We achieved this with the help of interactive zero knowledge proofs instead of passively authenticating the data with digital signatures.

We thank Lambert Sonna for his support and valuable insights during the design and implementation of our protocols. We thank Innosuisse for providing financial support and Global ID for providing the necessary equipment for the implementation part.

References

1. ICAO. *Machine Readable Travel Documents Part 11 (Doc. 9303)*. International Civil Aviation Organization, 2015.
2. BSI. *Advanced Security Mechanisms for Machine Readable Travel Documents*. TR-03110 Technical Guideline. in der Informationstechnik, 2016.
3. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993.
4. Brian LaMacchia, Kristin Lauter, and Anton Mityagin. *Stronger Security of Authenticated Key Exchange*, pages 1–16. Springer Berlin Heidelberg, 2007.
5. Jean Monnerat, Serge Vaudenay, and Martin Vuagnoux. About machine-readable travel documents. *RFID Security 2007*, 2007.
6. Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard java card. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pages 600–610, 2009.
7. Paulo Mateus and Serge Vaudenay. On tamper-resistance from a theoretical viewpoint. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 411–428, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
8. National Institute of Standards and Technology. *FIPS PUB 180-2: Secure Hash Standard*. 2004.
9. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, February 1997.
10. Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 98–107. ACM, 2002.
11. Morris J. Dworkin. SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Technical report, Gaithersburg, MD, United States, 2007.

12. C. P. Schnorr. *Efficient Identification and Signatures for Smart Cards*, pages 239–252. Springer New York, New York, NY, 1990.
13. David Pointcheval and Jacques Stern. Security proofs for signature schemes. EUROCRYPT'96, pages 387–398, Berlin, Heidelberg, 1996. Springer-Verlag.
14. Jean Monnerat, Sylvain Pasini, and Serge Vaudenay. Efficient deniable authentication for signatures. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security*, pages 272–291, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
15. Jean Monnerat, Sylvain Pasini, and Serge Vaudenay. Efficient deniable authentication for signatures. In *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ACNS '09. Springer-Verlag, 2009.
16. Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer Publishing Company, Incorporated, 1st edition, 2010.
17. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*. Springer Berlin Heidelberg, 2000.
18. SPAKE2, a PAKE, draft-irtf-cfrg-spake2-04. https://tools.ietf.org/html/draft-irtf-cfrg-spake2-04, 2017. Accessed: 2018-01-18.
19. NXP JCOP J2E145 v2.4.2 R3 Java Card 144K. https://www.cardlogix.com/product/nxp-j2e145-2-4-2-rel-3-java-card-145k-jcop-3-0-1, 2018. Accessed: 2018-10-01.
20. JCMathLib. https://github.com/OpenCryptoProject/JCMathLib, 2017. Accessed: 2018-10-01.
21. Naoto Miura, Akio Nagasaka, and Takafumi Miyatake. Feature Extraction of Finger-Vein Patterns Based on Repeated Line Tracking and Its Application to Personal Identification. *Mach. Vis. Appl.*, 15(4):194–203, 2004.
22. IDIAP finger-vein matching, 2016. Accessed: 2018-10-01.
23. Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *Proceedings of PKC 2001, volume 1992 of LNCS*, pages 104–118. Springer-Verlag, 1992.
24. Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In Alfred Menezes, editor, *CT-RSA*, 2005.

A Security Models and Proofs

In this section, we present models that capture the intuitive security of our primitives and complement the protocols with their proofs.

A.1 Strong Access Control (SAC)

SAC normally uses a certificate. We consider the security of the core protocol where the certificate $\langle \mathsf{CA\text{-}T}, \mathsf{T} \rangle$ is omitted. We also assume that the final session keys are defined as the hashes of the derived keys, i.e. $K_{\mathsf{C}} \leftarrow \mathsf{H}(K_{\mathsf{C}}|R|X_1|X_2)$ and $K_{\mathsf{T}} \leftarrow \mathsf{H}(K_{\mathsf{T}}|R|X_1|X_2)$. This small change makes sense, because these key materials are hashed before use in SSE.

The goal of the adversary is to succeed in either of the following: (1) make the chip accept by actively manipulating the communication between the chip and the terminal (without trivially relaying with the terminal); or, (2) for an honest session between the chip and the terminal, distinguish the session key of $\mathcal{O}_{\mathsf{C}}$ from random. Our model is weaker than extended Canetti-Krawczyk (eCK) model, because we do not allow revelation of an ephemeral state [4]. However, we allow revelation of session keys and the corruption of the long-term key. We also consider the explicit authentication which is omitted in the eCK model. The latter is allowed only after the challenge query is made, in order to capture the forward-secrecy notion. We consider the single-chip single-terminal notion as their multi-user versions directly follows from the former models. $\mathtt{Game}_{\mathsf{SAC\text{-}SEC}}$ proceeds as follows:

1. *Setup.* The challenger picks $\mathcal{G}$, and initializes $\mathcal{O}_{\mathsf{T}}$ by sampling a secret key $y \leftarrow_{\$} \mathbb{Z}_q^*$ and public key $Y \leftarrow y \cdot G$. It returns Y and $\mathcal{G}$ to the adversary $\mathcal{A}$.
2. *Oracle* $\mathcal{O}_{\mathsf{T}}$. $\mathcal{A}$ can interact with the oracle $\mathcal{O}_{\mathsf{T}}$ which simulates the terminal side of SAC, and the number of oracle queries is bounded by Q_{T}.
3. *Oracle* $\mathcal{O}_{\mathsf{C}}$. $\mathcal{A}$ can interact with $\mathcal{O}_{\mathsf{C}}$ which simulates the chip only once.
4. *Reveal.* $\mathcal{A}$ can obtain the session key K_{T} from his choice of session of $\mathcal{O}_{\mathsf{T}}$.
5. *Corrupt.* $\mathcal{A}$ can obtain the long-term secret key y.
6. *Test.* If $b = 0$, the session key K_{C} of the chip is returned. If $b = 1$, a random key $K_{\$}$ whose size is same with K_{C} is returned.
7. *Guess.* Finally, $\mathcal{A}$ outputs a guess bit b'.

$\mathcal{A}$ can keep playing with $\mathcal{O}_{\mathsf{T}}$; make hash, *Reveal* and *Corrupt* queries even after *Test* query. However, until $\mathcal{O}_{\mathsf{C}}$ completes its session successfully and assigns a key to K_{C}, *Test* and *Corrupt* are not allowed. We define partnering to rule out trivial wins for $\mathcal{A}$.

Definition 1 (Partnering). *Suppose that oracles record timestamps whenever they are queried by* $\mathcal{A}$*. In that case, the session of* $\mathcal{O}_{\mathsf{C}}$ *(and similarly each* $\mathcal{O}_{\mathsf{T}}$ *session) reaches to a final list of tuples* $(\mathsf{m_i}, \mathsf{d_i}, \mathsf{t_i})$ *at the end of the game. Let this list be sorted such that* $\mathsf{t}_1 < \mathsf{t}_2 < \cdots < \mathsf{t}_\tau$. $\mathsf{m_i} \in \{0,1\}^*$, $\mathsf{d_j} \in \{0,1\}$*, where each* $\mathsf{t_j}$ *is a point in time. We use* $\mathsf{m_j}$ *to denote the exchanged message,* $\mathsf{d_j} = 0$ *to denote that the message was received (resp.* $\mathsf{d_j} = 1$ *implies sent), and* $\mathsf{t_j}$ *to*

denote the time the message was received (resp. sent). Let $\mathsf{t_k} < \mathsf{t_j}$ *mean* $\mathsf{m_k}$ *was received (resp. sent) before* $\mathsf{m_j}$*, for* $k, j \in \{1, 2, \ldots, \tau\}$*. Then* $\mathcal{O}_\mathsf{C}$ *with timestamped transcript* $(\mathsf{m_i}, \mathsf{d_i}, \mathsf{t_i})$ *is partnered with* $\mathcal{O}_\mathsf{T}$ *with* $(\mathsf{m'_i}, \mathsf{d'_i}, \mathsf{t'_i})$ *if and only if* $\forall i \in \{1, 2, \ldots, \tau\}$ $[(\mathsf{m_i} = \mathsf{m'_i}) \wedge (\mathsf{d_i} \neq \mathsf{d'_i}) \wedge ((\mathsf{d_i} = 1 \wedge \mathsf{t_i} < \mathsf{t'_i}) \vee (\mathsf{d_i} = 0 \wedge \mathsf{t_i} > \mathsf{t'_i}))]$.

We say that event `Relay` happened if there exists a session of $\mathcal{O}_\mathsf{T}$ that is partnered with $\mathcal{O}_\mathsf{C}$. We say that event `Violate` happened if a *Test* or *Corrupt* query is made before $\mathcal{O}_\mathsf{C}$ completes. We further define event `Clean` as:

- Relay happens; and,
- `Violate` does not happen; and,
- no *Reveal* is made on the partner session of $\mathcal{O}_\mathsf{T}$ (Relay implies that it exists).

Definition 2. *Let* b, b' *be bits and events* `Relay`, `Clean` *be defined as above in* $\mathtt{Game}_{\mathsf{SAC\text{-}SEC}}$*. We say* `SAC` *is secure, if for every polynomially bounded adversary* $\mathcal{A}$*, the two advantages to be defined are negligible:* $\mathcal{A}$*'s active advantage* $\mathsf{Adv}^{\mathsf{act}}_{\mathcal{A}} := \Pr[\mathcal{O}_\mathsf{C}$ *accepts*$, \neg\mathtt{Relay}, \neg\mathtt{Violate}]$ *and* $\mathcal{A}$*'s passive advantage* $\mathsf{Adv}^{\mathsf{pas}}_{\mathcal{A}} = \Pr[b' = 1, \mathtt{Clean} | b = 1] - \Pr[b' = 1, \mathtt{Clean} | b = 0]$.

Definition 3 (GDH Assumption [23]). *Let* $\mathcal{G}$ *define a group. Let* $\mathcal{O}_{DH}$ *be an oracle which decides whether* $(G, x{\cdot}G, x{\cdot}Y, K)$ *is a valid DH-tuple, i.e.* $K = x{\cdot}y{\cdot}G$*. Let* $x, y \leftarrow_\$ \mathbb{Z}_q$*. Given* $(G, x{\cdot}G, y{\cdot}G)$ *and polynomially many access to* $\mathcal{O}_{DH}$*, it is hard to recover* $K = x{\cdot}y{\cdot}G$*.*

Theorem 1. `SAC` *is secure under the random oracle assumption if GDH assumption holds over* $\mathcal{G}$*.*

Proof. Active Attacks. We show that $\mathsf{Adv}^{\mathsf{act}}_{\mathcal{A}}$ is negligible. Let $\mathcal{O}_\mathsf{T}$ and $\mathcal{O}_\mathsf{C}$ be oracles simulating T and C of `SAC` in Figure 2. The hash queries receive $(K|R|X_1|X_2)$ as input. We do not need to simulate *Corrupt* oracle, since the simulation will stop as soon as the chip accepts.

$\underline{\mathtt{Game}_0}$: The original game $\mathtt{Game}_{\mathsf{SAC\text{-}SEC}}$.

$\underline{\mathtt{Game}_1}$: Let (R, X_1, X_2) denote the transcript seen by $\mathcal{O}_\mathsf{C}$. We modify the game such that we abort if any $\mathcal{O}_\mathsf{T}$ session makes a hash query containing (R, X_1, X_2). This is upper bounded by $|\mathsf{Adv}^{\mathtt{Game}_0}_{\mathcal{A}} - \mathsf{Adv}^{\mathtt{Game}_1}_{\mathcal{A}}| \leq \frac{Q_\mathsf{T}}{q}$ (i.e. collisions over R).

$\underline{\mathtt{Game}_2}$: We modify the game such that we abort if $\mathcal{A}$ had made a hash query $\mathsf{H}(K|R|X_1|X_2)$ with an arbitrary key K, and later (R, X_1, X_2) happens to be a view obtained by $\mathcal{O}_\mathsf{C}$. This is upper bounded by $|\mathsf{Adv}^{\mathtt{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathtt{Game}_2}_{\mathcal{A}}| \leq \frac{Q_\mathsf{H}}{q^2}$ (i.e. collisions over (X_1, X_2)).

$\underline{\mathtt{Game}_3}$: We change the game simulation. We never calculate the key $K_\mathsf{C} = x_1{\cdot}Y + x_2{\cdot}Y$, but instead use modified hash oracles $\mathsf{H_T}, \mathsf{H_C}, \mathsf{H}_\mathcal{A}$ through transcripts of executions. $\mathcal{O}_\mathsf{T}$ gets K_T by solely querying $\mathsf{H_T}(\perp|r|X_1|X_2)$. Similarly $\mathcal{O}_\mathsf{C}$ queries $\mathsf{H_C}(\perp|R|X_1|x_2)$. And $\mathcal{A}$ can query hash with $(K|R|X_1|X_2)$.

We simulate $\mathtt{Game}_3$ and play the role of adversary in GHD game. In GDH game, we receive (G, X_0, Y_0) and try to output $DH(X_0, Y_0)$. In Figure 7, we give algorithms to correctly simulate the random oracle. We use two lists L_1 and L_2 for bookkeeping, and overwrite responses of L_1 by L_2 whenever there is a match. If $\mathcal{A}$ makes a game-winning query, then $\mathsf{H}_\mathcal{A}$ aborts and wins the GDH game.

$\mathsf{H}_{\mathsf{T}}(\perp|r|X_1|X_2)$:

$k' \leftarrow_{\$} \{0,1\}^{|\mathsf{H}|}$; $R \leftarrow r{\cdot}G$
if $\exists K\ \mathsf{L}_1[K, R, X_1, X_2] \neq \perp$:
  return $\mathsf{L}_1[K, R, X_1, X_2]$
if $\mathsf{L}_2[R, X_1, X_2] \neq \perp$:
  $(r', x_2', k) \leftarrow \mathsf{L}_2[R, X_1, X_2]$
  return k
$\mathsf{L}_2[R, X_1, X_2] \leftarrow (r, \perp, k)$
return k

$\mathsf{H}_{\mathsf{C}}(\perp|R|X_1|x_2)$:

$k' \leftarrow_{\$} \{0,1\}^{|\mathsf{H}|}$; $X_2 \leftarrow x_2{\cdot}G$
if $\exists K\ \mathsf{L}_1[K, R, X_1, X_2] \neq \perp$:
  return $\mathsf{L}_1[K, R, X_1, X_2]$
if $\mathsf{L}_2[R, X_1, X_2] \neq \perp$:
  $(r', x_2', k) \leftarrow \mathsf{L}_2[R, X_1, X_2]$
  return k
$\mathsf{L}_2[R, X_1, X_2] \leftarrow (\perp, x_2, k)$
return k

$\mathsf{H}_{\mathcal{A}}(K|R|X_1|X_2)$:

if $\mathsf{L}_2[R, X_1, X_2] \neq \perp$:
  $(r, x_2, k) \leftarrow \mathsf{L}_2[R, X_1, X_2]$
  if $r = \perp$ **then** $Z' \leftarrow x_2{\cdot}R$
  else $Z' \leftarrow r{\cdot}X_2$
  if $\mathcal{O}_{DH}(G, X_1, Y_0, K - Z') = 1$:
    if $X_0 = X_1$ **then ABORT with** $(K - Z')$
    return k
if $\mathsf{L}_1[K, R, X_1, X_2] = \perp$:
  $\mathsf{L}_1[(K, R, X_1, X_2)] \leftarrow_{\$} \{0,1\}^{|\mathsf{H}|}$
return $\mathsf{L}_1[(K, R, X_1, X_2)]$

Fig. 7. Simulation of hash oracles.

We also simulate $\mathcal{O}_{\mathsf{C}}$ and a randomly chosen session of $\mathcal{O}_{\mathsf{T}}$. For $\mathcal{O}_C$, we only set $X_1 = X_0$ and for $\mathcal{O}_{\mathsf{T}}$, we set the long-term public key to Y_0. When $\mathcal{A}$ makes $\mathcal{O}_{\mathsf{C}}$ accept, we deduce that $\mathcal{A}$ must have made a query with the correct key K. Otherwise the tested key $K_{\mathsf{C}}/K_{\$}$ is independent of $\mathcal{A}$'s view. Hence, our simulated hash oracle will abort the game and output the key for (G, X_0, Y_0) tuple. Hence, we reach to $\frac{1}{Q_{\mathsf{T}}} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathtt{Game}_3} \leq \mathsf{Adv}_{\mathcal{A}}^{\mathtt{GDH}} + negl$.

Passive Attacks. We bound $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{pas}}$, where the adversary $\mathcal{A}$ tries to distinguish the key after relaying the communication between the chip and the terminal.
$\underline{\mathtt{Game}_0}$: This is the original game. Let $\mathcal{O}_{\mathsf{C}}$'s partial transcript be (R, X_1, X_2).
$\underline{\mathtt{Game}_{1,2}}$: We make the same modifications from the previous proof. No $\mathcal{O}_{\mathsf{T}}$ session makes a hash query with a collision on R. $\mathcal{A}$ does not make a lucky hash query with the same (X_1, X_2) values. Hence, $|\mathsf{Adv}_{\mathcal{A}}^{\mathtt{Game}_0} - \mathsf{Adv}_{\mathcal{A}}^{\mathtt{Game}_2}| \leq \frac{Q_{\mathsf{T}}}{q} + \frac{Q_{\mathsf{H}}}{q^2}$.
$\underline{\mathtt{Game}_3}$: We simulate the random oracle for hash queries. Essentially, we pick a random terminal session among Q_{T}, hoping that it will be the one partnered with $\mathcal{O}_{\mathsf{C}}$. Then for inputs (G, X_0, Y_0) from GDH game, we can set $X_2 = X_0$ in $\mathcal{O}_{\mathsf{C}}$ and $R = Y_0$ in this $\mathcal{O}_{\mathsf{T}}$ sessions. x_1 and Y are chosen randomly as in the original game.

We also isolate and overwrite the hash query $(\perp|R|X_1|X_2)$ from the partnered $\mathcal{O}_{\mathsf{T}}$ and $\mathcal{O}_{\mathsf{C}}$. In order to keep hash responses consistent, when $\mathcal{A}$ makes hash queries of the form $(K|R|X_1|X_2)$ for some arbitrary K, we check whether

$\mathcal{O}_{DH}(G, X_0, Y_0, K - x_1 \cdot Y) = 1$. If this holds, we call it *special query* and we output $K - x_1 \cdot Y$ in GDH game. If $\mathcal{A}$ does not make the special query, then the tested key $K_{\mathsf{C}}/K_{\$}$ is independently sampled from the view of $\mathcal{A}$. Therefore, $\frac{1}{Q_{\mathsf{T}}} \cdot \mathsf{Adv}^{\mathsf{pas}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathtt{GDH}}_{\mathcal{A}} + \frac{1}{2^{|\mathsf{H}|}}$. □

A.2 Data & Chip Authentication (DCA)

We give the skeletal core of our DCA in Figure 8, which is a Σ-protocol.

Prover C		**Verifier** T
IN: $s, R, \mathsf{IS}_{\mathsf{pub}}, m$		**IN:** $R, \mathsf{IS}_{\mathsf{pub}}, m$
$u \leftarrow_{\$} \mathbb{Z}_q;\ U \leftarrow u \cdot G$	$\xrightarrow{U}$	
	$\xleftarrow{v}$	$v \leftarrow_{\$} \mathbb{Z}_q^*$
$s' \leftarrow (s + vu) \bmod q$	$\xrightarrow{s'}$	$e \leftarrow \mathsf{H}(m \mid R)$
		assert $s' \cdot G + e \cdot \mathsf{IS}_{\mathsf{pub}} = R + v \cdot U$

Fig. 8. The core Σ protocol.

Suppose that (s, R) is a Schnorr signature over message m and public key $\mathsf{IS}_{\mathsf{pub}}$. That is to say, $R = s \cdot G + e \cdot \mathsf{IS}_{\mathsf{pub}}$ holds (see Section 2.3 for Schnorr's verification algorithm). Then we define the relation predicate $\mathcal{R}(\chi, \omega)$ as $R = s \cdot G + e \cdot \mathsf{IS}_{\mathsf{pub}}$ where χ is $(R, m, \mathsf{IS}_{\mathsf{pub}})$ and ω is s.

We give a sketch proof that the core protocol satisfies the requirements of a Σ-protocol. 3-move and polynomial running time properties are straightforward.

- *Knowledge Extractor.* The extractor E takes two valid transcripts (U, v_1, s'_1) and (U, v_2, s'_2) as input and returns a witness $\hat{s} = (v_2 s'_1 - v_1 s'_2)/(v_2 - v_1)$, given $v_1 \neq v_2$. The validity of $\hat{s}$ follows from the verification equations.
- *Honest Verifier ZK:* On fixed $v \in \mathbb{Z}_q^*$, we show that the same view can be simulated. For $u \leftarrow_{\$} \mathbb{Z}_q$, we have one-to-one mapping $u \mapsto s'$, because $\gcd(v, q) = 1$. Hence the simulator first uniformly picks $s' \leftarrow_{\$} \mathbb{Z}_q$ and then finds the corresponding U with $U \leftarrow v^{-1}(s' \cdot G + e \cdot Y - R)$, where v^{-1} is the inverse of v modulo q. This also gives one-to-one mapping, $s' \mapsto U$, therefore providing the same distribution with the honest view.

In order to take into account that the terminal might not behave honestly, the honest-verifier zero knowledge property is augmented with a hash-based commitment. The enhanced protocol becomes fully zero-knowledge and weak proof of knowledge in the random oracle model. The formal proofs can be found in Monnerat et al. [15]. Additionally, we updated the protocol such that R is being sent by the chip, instead of being an input.

A.3 Weak Access Control (WAC)

Our weak access control protocol is a variant of SPAKE1 [24], where the difference is that G_2, G_3 points are fixed and same for all chips. We define a security notion similar to that of SAC prove the security of this modified protocol. We use the simplified version of the password-based chosen-basis Diffie-Hellman assumption introduced by Abdalla and Pointcheval [24], that is equivalent to the computational Diffie-Hellman assumption.

Similar to SAC-SEC, the goal of the adversary is to either make the chip accept by actively modifying the exchanged messages, or distinguish the key of an honest session. We again assume that the final session keys are defined as the hashes of the derived keys, i.e. $K_{\mathsf{C}} \leftarrow \mathsf{H}(K_{\mathsf{C}}|M|L)$ and $K_{\mathsf{T}} \leftarrow \mathsf{H}(K_{\mathsf{T}}|M|L)$. That is because these key materials are hashed before use in SSE.

The security game consists of Q_{C} different chips (simulated by $\mathcal{O}_{\mathsf{C}}$) whose passwords are uniformly sampled from the password domain $\mathcal{D}$ at the beginning of the game. $\mathcal{O}_{\mathsf{T}}$ simulates the terminal which has access to all passwords of the chips. The security game $\mathtt{Game}_{\mathsf{WAC\text{-}SEC}}$ works as follows:

1. *Setup.* The challenger picks $\mathcal{G}$, and initializes each C_i by sampling a password pwd_i for $i \in \{1, \ldots, Q_{\mathsf{C}}\}$. $\mathcal{A}$ receives $\mathcal{G}$.
2. *Execute.* $\mathcal{A}$ chooses a chip C_i. Then an honest session transcript using pwd_i between the chip and the terminal is returned. It can be queried Q_e times.
3. *Oracle* $\mathcal{O}_{\mathsf{T}}$. $\mathcal{A}$ chooses a chip C_i and the oracle $\mathcal{O}_{\mathsf{T}}$ simulates the terminal side of WAC with pwd_i. It can be queried Q_{T} times.
4. *Oracle* $\mathcal{O}_{\mathsf{C}}$. $\mathcal{A}$ chooses a chip C_i and $\mathcal{O}_{\mathsf{C}}$ simulates the chip side of WAC with pwd_i. It can be queried Q_{C} times.
5. *Reveal.* $\mathcal{A}$ can request the session key K_{T} from any session of $\mathcal{O}_{\mathsf{T}}$, or K_{C} from any session of $\mathcal{O}_{\mathsf{C}}$.
6. *Test.* $\mathcal{A}$ picks a chip session. If $b = 0$, the session key K_{C} is returned. If $b = 1$, a random key $K_{\$}$ whose size is same with K_{C} is returned.
7. *Guess.* Finally, $\mathcal{A}$ outputs a guess bit b'.

Since the active attacks allow $\mathcal{A}$ to brute-force password domain, we assume that $Q_{\mathsf{T}} + Q_{\mathsf{C}} \ll Q_e$. Any adversary with $Q_{\mathsf{C}} + Q_{\mathsf{T}}$ close to $|\mathcal{D}|$ can practically recover the password with good probability simply by trial-and-error. We define the partnering exactly same as in Section A.1.

We say that event Relay happened if the tested $\mathcal{O}_{\mathsf{C}}$ has some partner $\mathcal{O}_{\mathsf{T}}$ session. We further define event Clean as:

- *Test* is made to $\mathcal{O}_{\mathsf{C}}$ session queried with *Execute*; and,
- no *Reveal* is made to the tested $\mathcal{O}_{\mathsf{C}}$ or its partner session from $\mathcal{O}_{\mathsf{T}}$.

Definition 4. *Let b, b' be bits and events* Relay, Clean *be defined as above. We say* WAC *is secure, if for every $\mathcal{A}$, for the two advantages: $\mathcal{A}$'s active advantage defined as* $\mathsf{Adv}^{\mathsf{act}}_{\mathcal{A}} := \Pr[\textit{tested } \mathcal{O}_{\mathsf{C}} \textit{ accepts}, \neg\mathtt{Relay}]$ *is at most negligibly better than* $\frac{Q_{\mathsf{T}}+Q_{\mathsf{C}}}{|\mathcal{D}|}$*, and $\mathcal{A}$'s passive advantage defined as* $\mathsf{Adv}^{\mathsf{pas}}_{\mathcal{A}} = \Pr[b' = 1, \mathtt{Clean}|b = 1] - \Pr[b' = 1, \mathtt{Clean}|b = 0]$ *is negligible.*

Definition 5 (Simplified password-based chosen-basis DH). *Let $\mathcal{D}$ be the dictionary of passwords. Simplified version of the said assumption states that the following advantage is bounded by $\frac{1}{|\mathcal{D}|}$ with some negligible difference for all $\mathcal{A}$:*

$$\Pr\left[\begin{array}{l} pick\ \mathcal{G}; G_2, G_3 \leftarrow_{\$} \langle G\rangle; \mathsf{pwd} \leftarrow_{\$} \mathcal{D} \\ A \leftarrow_{\$} \langle G\rangle; M \leftarrow A + \mathsf{pwd}{\cdot}G_3 \\ L, K \leftarrow \mathcal{A}(\mathcal{G}, G_2, G_3, M) \end{array} : DH(M - \mathsf{pwd}{\cdot}G_2, L - \mathsf{pwd}{\cdot}G_3) = K\right]$$

Theorem 2 ([24]). $CDH \implies \mathsf{PCCDH} \implies \mathsf{S\text{-}PCCDH}$.

Theorem 3. `WAC` *is secure under the random oracle assumption if CDH assumption holds over $\mathcal{G}$.*

Proof. Passive Attacks. We first bound $\mathsf{Adv}^{\mathsf{pas}}_{\mathcal{A}}$, where the adversary $\mathcal{A}$ tries to distinguish the key for a honest session between the chip and the terminal. Essentially, we can pick a random session among Q_e, hoping that it will be the one attacked by $\mathcal{A}$. Then we can simulate CDH game by setting $A = X_0$ and $B = Y_0$ respectively in the game for inputs (G, X_0, Y_0) from the GDH game.

We again simulate the random oracle just as in the previous proof. If $\mathcal{A}$ can distinguish K_{C} with good probability, then we can find the key $DH(X_0, Y_0)$ in one of the hash queries made by $\mathcal{A}$. If $\mathcal{A}$ does not make such query, then the tested key $K_{\mathsf{C}}/K_{\$}$ is independent of $\mathcal{A}$'s view. Other oracle queries can be perfectly simulated by honestly running the protocol. We deduce that: $\frac{1}{Q_e} \cdot \mathsf{Adv}^{\mathsf{pas}}_{\mathcal{A}} \leq \mathsf{negl.} + \frac{1}{2^{|\mathsf{H}|}}$.

Active Attacks. Now we show that $\mathsf{Adv}^{\mathsf{act}}_{\mathcal{A}}$ is bounded by $\frac{Q_{\mathsf{T}}+Q_{\mathsf{C}}}{|\mathcal{D}|}$ with some negligible difference. The idea is to make a guess the active attack session which will be tested by $\mathcal{A}$, and simulate the `S-PCCDH` game. For $\mathcal{A}$'s execute queries, we sample a random tuple (M, N, K_v) from $\langle G\rangle^2 \times \{0,1\}^{|\mathsf{H}|}$. Given that $\mathcal{A}$'s passive advantage above is negligible, this change can reduce $\mathcal{A}$'s active advantage negligibly.

We again simulate the random oracle just as in the previous proof. Finally, when $\mathcal{A}$ makes a correct guess on bit b', we look at his hash queries to find a key K that would win in `S-PCCDH` game. Therefore: $\frac{1}{Q_{\mathsf{T}}+Q_{\mathsf{C}}} \cdot \mathsf{Adv}^{\mathsf{act}}_{\mathcal{A}} \leq \mathsf{negl.} + \frac{1}{|\mathcal{D}|}$. □

A.4 Confirmer Data & Chip Authentication (`CDA`)

The security is defined w.r.t. the terminal: $\mathcal{A}$ playing the role of a chip without possession of a valid $(\mathsf{u}_{\mathsf{chip}}, \mathsf{K}_{\mathsf{chip}})$ pair should not be able to convince the terminal. The game is defined in Figure 9 with the core of our protocol. Although stronger notions that capture the collusion of confirmer and the terminal exist in public-key setting, the actual use scenario renders collusion attacks meaningless.

Definition 6. *We say* `CDA` *is secure, if for every polynomially bounded $\mathcal{A}$, the probability $p(Q_s, Q_v) := \Pr[\mathsf{CONF\text{-}SEC}_{\mathcal{A}}(Q_s, Q_v)]$ is negligible (in terms of ℓ).*

Theorem 4. `CDA` *is secure under the random oracle assumption if* MAC *is existentially-unforgeable and* PRF *is a pseudo-random function.*

$\mathsf{CONF\text{-}SEC}_{\mathcal{A}}(Q_s, Q_v)$

$\mathsf{L}_{\mathsf{MAC}} \leftarrow \emptyset; \mathsf{K}_{\mathsf{Cnf}} \leftarrow \{0,1\}^{\ell}$

$(u', W', \sigma') \leftarrow \mathcal{A}^{\mathsf{Send},\mathsf{Verify}}(\ell)$

return $(u', W', \sigma') \notin \mathsf{L}_{\mathsf{MAC}}$

$\quad \wedge\ \mathsf{MAC}(\mathsf{PRF}(\mathsf{K}_{\mathsf{Cnf}}, u'), W') = \sigma'$

$\mathsf{Send}(u, W)$

$\sigma \leftarrow \mathsf{MAC}(\mathsf{PRF}(\mathsf{K}_{\mathsf{Cnf}}, u), W)$

$\mathsf{L}_{\mathsf{MAC}} \leftarrow \mathsf{L}_{\mathsf{MAC}} \cup \{(u, W, \sigma)\}$

return σ

$\mathsf{Verify}(u, W, \sigma)$

return $\mathsf{MAC}(\mathsf{PRF}(\mathsf{K}_{\mathsf{Cnf}}, u), W) = \sigma$

Fig. 9. The ad hoc security definition for CDA. $\mathcal{A}$ can interact with Send that simulates chips (at most Q_s queries), and with Verify that mimics the confirmer (at most Q_v queries). W and u combined can be interpreted as the full tuple $(m', n_{\mathsf{T}}, t, n_{\mathsf{C}}, \mathsf{u}_{\mathsf{chip}}, \sigma_t)$. We use u in place of $\mathsf{u}_{\mathsf{chip}}$, W for $(m', n_{\mathsf{T}}, t, n_{\mathsf{C}})$ and σ for σ_t.

Proof. In PRF game, let ϵ_{prf} be the max advantage of distinguishing PRF from random with multiple $\mathcal{O}_{\mathsf{PRF}}$ queries. In multi-attempt unforgeability game, let ϵ_{mac} be the max advantage of an adversary forging a MAC with multiple message, verification queries where $\mathcal{A}$ wins if any of its many attempted forgery is valid.
$\underline{\mathtt{Game}_0}$ The original game (p_i denotes the advantage in $\mathtt{Game}_i$ below).
$\underline{\mathtt{Game}_1}$: We replace PRF with a random function, and simulate it by bookkeeping active u values. Hence $|p_0(Q_s, Q_v) - p_1(Q_s, Q_v)| \leq \epsilon_{\mathsf{prf}}$.

We show reduction to the adversary $\mathcal{B}$ playing against MAC forgery game. $\mathcal{B}$ picks a random index $i \in [1, Q_s + Q_v]$ and uses MAC oracles to simulate oracles to $\mathcal{A}$. The other keys are simulated with bookkeeping as before. $\mathcal{B}$ wins if $\mathcal{A}$ can produce a valid forgery for index i. Therefore, $p_1(Q_s, Q_v) \leq (Q_v + Q_s) \cdot \epsilon_{\mathsf{mac}}$. □