Solving Terminal Revocation in EAC by Augmenting Terminal Authentication[*]

(Fullpaper version)

Rafik Chaabouni[1,2]

[1]EPFL
CH-1015 Lausanne, Switzerland

[2]University of Tartu
Ülikooli 18, 50090 Tartu, ESTONIA

Abstract: In this paper we propose a solution to enable an accurate terminal revocation in the Extended Access Control (EAC). Chaabouni and Vaudenay in [CV09] pointed out the need for an accurate revocation procedure, but failed to provide a complete solution description. We aim at filling this gap. Our solution relies on augmenting terminal authentication with a t-out-of-ℓ threshold signature provided by neighboring terminals. These terminals will be in charge of checking the revocation status of the requested terminal. As Terminals have a real clock embedded and more computational power than Machine Readable Travel Documents (MRTDs), they are better suited for checking revocation status.

1 Introduction

In response to the initial weak standard for Machine Readable Travel Documents (MRTDs), produced by the International Civil Aviation Organization (ICAO), the European Union has mandated the Federal Office for Information Security (BSI) to provide and maintain a stronger standard for MRTDs. In that regard, the BSI has issued the Extended Access Control (EAC) which provides a stronger privacy protection for MRTDs. Its first initial release [BfSidI12a] was made in 2006, while the last version [BfSidI12b, BfSidI12c, BfSidI12d] was published in 2012. It was believed that with the introduction of EACv2 in 2009, the majority of threats were solved. Unfortunately, Chaabouni and Vaudenay [CV09] pointed out several remaining flaws and threats. The major flaw pointed out was the absence of a good terminal revocation. The other issues are now considered marginal as they are or will be gradually solved with the evolution of previous standards (notably the one from the ICAO [ICAO08, ICAO13]). However, no progress has been made regarding terminal revocation nor terminal authentication. Chaabouni and Vaudenay

[*] This work has been supported from research theme IUT2-1 and European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

suggested a solution for terminal revocation but they omitted to give a detailed description. We aim at filling this gap by providing an efficient and secure solution.

Our concern in this paper targets two types of threats. We are first concerned by the threat of a stolen integrated terminal device. These are considered to be Portable Computing Devices (PCD) in the Technical Guideline TR-03110 [BfSidI12b, BfSidI12c, BfSidI12d], when terminal key pairs are explained. An integrated terminal as explained in [BfSidI09], consists of a single reader with integrated hardware security module and proximity coupling device. Moreover a stolen integrated terminal could still be used to read MRTD, as long as its certificate is not expired. This threat even applies with an expired certificate if the date approximated in the MRTD is outdated. Hence there is no real revocation system present for terminals. This is a known problem for the BSI and is even mentioned in [BfSidI09], section 1.2.1:

> The disadvantage of this architecture is, that a stolen reader can be used to perform Terminal Authentication at least as long as the current CV certificate is valid.

Secondly, we have to keep in mind that threats come often from an inside attack. This pushes us to study the threat of a compromised terminal that is remained in place, acting maliciously. With the actual standard, a stolen or compromised terminal could be used to target a group of person (e.g. by nationality), or a specific person (e.g. important politicians).

The implications of these threats are threefolds. They introduce an obvious privacy breach in the sense that any compromised integrated terminal will have an illegitimate access to all MRTDs data including biometrics. From there, an attacker can filter and target specific individuals or even groups with specific attributes e.g. a specific nationality. Moreover, the terminal can be used to acquire all information from all MRTDs that come nearby in order to build an illegitimate database of biometrics. With this kind of database, attackers can train themselves and select the closest match for a cloned ID.

Furthermore, we need to take into account efficiency. In [Fri], it is mentioned that more than 56 millions passengers traveled through Frankfurt airport in 2011. As around half of them are only transfer passengers, and thus do not necessarily need a passport control, we can see that big hubs need to process more than 2 millions passport checks per month.

1.1 Prior and Related Work

Terminal revocation has received little amount of interest as the BSI community is convinced that the Password Authenticated Connection Establishment (PACE) protocol mitigate this threat, as explained in [BDFK12]. Indeed, when executing EACv2, PACE is the initial phase before Terminal Authentication. After its successful completion, the MRTD is ensured that the terminal has knowledge of a shared password, and can proceed with Terminal Authentication. Moreover, the ISO/IEC JTC1 SC17 WG3 mentioned in [ICAO13] that:

> At present the fact that BAC MUST always be present on the eMRTD ensures that inspection systems that do not support PACE (yet) will still be able to access the MRTD's chip. To access eMRTDs supporting only PACE, inspection systems MUST support PACE. In its meeting on 19-21 February 2013 the NTWG concluded that as of the date 01 January 2018 eMRTDs supporting only PACE will be considered to be ICAO compliant. The chosen date should provide enough time for inspection system owners and vendors to implement the necessary modifications to their systems.

However no guarantees are provided in the obtention of this password. If the shared password has been obtained by social engineering, or read directly by eavesdropping on the MRTD, then a successful terminal authentication will allow the stolen terminal to access all sensitive data contained in the MRTD. This issue has been raised by Belguechi et al. in [BLR12]. Unfortunately they concentrate on the protection of biometric data and do not provide a solution for terminal revocation. Li et al. in [LZJX10] also mention the threat of terminal revocation. However they concentrate on presenting the Singapore solution that implicates Authorized Smartcard with Identity Based Cryptography. Hence to solve the terminal revocation issue they require heavy hardware modifications.

1.2 Contribution

In our new method, we make use of threshold cryptography in order to verify the revocation status of terminals. We assume that Document Verifiers (DVs) in the EAC standard are trusted participants. In our general case, several terminals are present. If the number of terminals is considered too low, our scheme can easily be modified to provide equivalent properties. Moreover the required modifications to enable this method are solely software upgrades and the existence assumption of a communication channel between terminals. Hence no hardware modification is needed in MRTDs.

1.3 Organization

In section 2 we recall the general structure of the EAC [BfSidI12b, BfSidI12c, BfSidI12d] which includes how terminal authentication and revocation are achieved. In section 3 we recall Shamir's secret sharing [Sha79], Non-Interactive Zero-Knowledge (NIZK) Proofs and see how it can be applied to achieve threshold signatures with the example of threshold RSA [Sho00, DF94, Kin00]. Section 4 will be devoted to our new solution. We will precise our security assumptions, explain how terminal authentication should be augmented to achieve a realistic terminal revocation and finish with some discussions on the security outcomes, potential efficiency tuning and some remarks.

2 EAC

The BSI TR-03110 Technical Guideline [BfSidI12b, BfSidI12c, BfSidI12d] defines the EAC. It is a specification for mutual authentication between terminal readers and MRTD, such as biometric passports. As it is an evolving standard (11 versions since its public release in 2006, with one major enhancement in 2008), we will focus only on the latest 2.10 version. We refer readers to [CV09] for an analysis and survey of EACv2.01 and EACv1. Regarding terminal authentication and terminal revocation, no progress was made between version 2.01 and version 2.10 of the EAC standard.

The aim of EAC, with its mutual authentication, is threefolds. It allows first to verify that a MRTD is genuine. Secondly, it allows authenticated terminals to access sensitive data contained in the MRTD, such as fingerprints. And at last, it provides a secure channel between the MRTD and the terminal. This authentication process relies on an international Public Key Infrastructure (PKI) composed by three entity types : Country Verifying Certificate Authorities (CVCAs), Document Verifiers (DVs) and Terminals. The description of EAC PKI can be found in [BfSidI12d]. Each participating country will possess a national CVCA that will act as a national root authority. The national CVCA will be in charge of issuing national MRTDs and DVs certificates (especially foreign DVs certificates). DVs are organizational units in charge of managing a group of terminals, notably by issuing their certificate. Within a same organizational unit, if terminals are supposed to access sensitive data contained in foreign MRTD, the DV in charge has to request for a DV certificate from all foreign CVCAs corresponding to the MRTD countries that terminals would encounter.

The EACv2 general authentication procedure is composed by four steps in the following order: Password Authenticated Connection Establishment (PACE), Terminal Authentication, Passive Authentication, and Chip Authentication. PACE is also a mutual authentication procedure, but it is solely based on a shared password. This password is either known by the MRTD bearer or is directly printed on the MRTD. Nevertheless, PACE provides a secure authenticated key agreement as proven by Bender, Fischlin and Kügler in [BFK09]. Once PACE has succeeded, the MRTD is ensured that the terminal has knowledge of the shared password and thus provides access to its less-sensitive data. Terminal Authentication is then preformed. We will detail it in section 2.1. After the terminal has been authenticated, Passive Authentication enables terminals to confirm that a MRTD has not been altered. This step does not protect against cloning attacks. In order to achieve cloning protection, Chip Authentication is performed at last. This last step insures that the MRTD is genuine.

It is interesting to note that the Data Group 2 (DG2) of the ePassport Application, which corresponds to the facial image, and the Document Security Object (SO_D) are still considered as less-sensitive data. As such, they are provided to terminals before terminal authentication. The threats implications of their classification under less-sensitive data are described in [MVV07, CV09]. We do not intent to provide a solution in this paper for this issue as we focus solely on terminal revocation.

2.1 Terminal Authentication

We refer to section 3.4 of [BfSidI12c] for the terminal authentication complete description. It is essentially composed by three major phases. First, the terminal sends a certificate chain starting from the CVCA certificate corresponding to the MRTD country. The chain ends with the terminal own certificate. In the second phase, the MRTD checks the certificates contained in the certificate chain with a Certificate Validation process (section 2.5 of [BfSidI12d]). The third phase consists of setting up an authenticated ephemeral Diffie-Hellman key pair for the terminal. The resulting ephemeral public key will then be used to secure messages for the terminal.

If the terminal authentication succeeds, the MRTD will grant access rights to its sensitive data according to the terminal effective authorization. The terminal effective authorization is derived from the certificate chain as the smallest authorization set present in all certificates of the certificate chain.

2.2 Terminal Revocation

The terminal revocation status is checked during the terminal Certificate Validation (section 2.5 of [BfSidI12d]). Surprisingly enough, the revocation process is achieved only with the expiration date contained in the certificate and with a “Current Date” approximation stored in the MRTD. The major problem, as expressed in [CV09], is that MRTDs do not have a reliable clock. This is why they try to approximate the current date. Unfortunately, due to the requirements on this approximation, the “Current Date” could be outdated by an entire month. Indeed, this update is done solely with the date of certificate creation contained in a certificate issued by the same country as the MRTD. Notice that there is no passport control within the Schengen zone. For departure from the Schengen zone, an identity control will be required only at the last Schengen airport before a non-Schengen country. More information can be found in [EP06]. As it is quite rare for a MRTD to encounter a terminal of its own country, the update will be done with the date of certificate creation contained in foreign DV certificates. These are issued by the same country as the MRTD one.

Hence, a stolen terminal can still be used for a long period of time, even if his expiration date has passed. This is an important threat that must not be neglected. Without a proper terminal revocation scheme, a stolen terminal could be set up to use solely EACv1 without PACE and thus be used to detect and target individuals or a specific group of persons, while the attacker is absent of the crime scene. Even in the case where EACv2 with PACE has to be used, if the shared password is compromised then all sensitive data will be accessed after completion of the terminal authentication.

3 Preliminaries

Threshold cryptosystems are an important building block for our new solution. We are going to focus on the case of threshold RSA signatures as it provides a good balance between security and efficiency considering our protocol participants. Indeed, we need to keep in mind that the computational power of MRTDs is much more limited compared to the one of terminals. Nevertheless, to understand threshold signatures and its related security, we first need to explain secret sharing and Non-Interactive Zero-Knowledge proofs.

3.1 Secret Sharing

This notion has been introduce by Shamir in [Sha79]. It aims at dividing the knowledge of a secret among ℓ servers. The motivation behind it was to protect a secret against the corruption of some servers. To achieve secret sharing, Shamir used Lagrange interpolation to divide the secret into multiple shares. The main idea is that any polynomial function f of degree t can be reconstructed from $t + 1$ distinct points. $f(0)$ is considered to be the secret s to be shared. If given t or less points, the function cannot be reconstructed. Hence every participant will be given a point of the function as a secret share. Mathematically speaking, we define f as follows:

$$f(x) = \sum_{i=0}^{t} a_i \cdot x^i. \tag{3.1}$$

As mentioned, the secret is $s = f(0) = a_0$. Every participant $i > 0$ will be provided the secret share $s_i = f(i)$. Given a set of participants Ψ with $|\Psi| = t + 1$, we can reconstruct f as follows:

$$f(x) = \sum_{j \in \Psi} s_j \cdot \lambda^{\Psi}_{x,j}, \tag{3.2}$$

where $\lambda^{\Psi}_{x,j}$ are the Lagrange coefficients defined by $\lambda^{\Psi}_{x,j} = \prod_{i \in \Psi \setminus j} (i - x) \cdot (i - j)^{-1}$, and with $x \notin \Psi$. Moreover, note that due to the inversion present in the Lagrange coefficients, we need to work in a field. This restriction can be relaxed with an efficiency cost, e.g. instead of computing $f(x)$ we could compute $i_{max}! \cdot f(x)$ where i_{max} is larger or equal than the largest index among participants.

3.2 Non-Interactive Zero-Knowledge Proofs

NIZK proofs are also an important and basic cryptographic building block. A NIZK proof allows a prover to convince a verifier on the veracity of a statement, by sending him a single message (non-interactive), without leaking any other information than the veracity of the statement proven (zero-knowledge). NIZK proofs are often constructed directly in the Common Reference String (CRS) model or by converting an interactive equivalent proof using the Fiat-Shamir [FS86] transformation in the random oracle model. The random oracle model makes the assumption that hash functions, and more generally pseudo-random functions, are replaced by truly random oracles. The drawback of this method is that proofs in this model are weaker than in the standard model. However they achieve a much higher efficiency. We will limit ourselves to the random oracle model as the EAC also uses hash functions, and thus falls in the random oracle model too.

Let us see the example of the Chaum-Pedersen [CP92] NIZK proof of discrete logarithm equality. We assume that a prover knows the discrete logarithm $d = \log_G X$, where G is a group generator of order q. The public parameters are (G, q, X). The goal of the prover is to convince a verifier that, given two group elements (R, S), the following statement holds: $\log_G X = \log_R S$. To do so, the prover picks a random $a \in_R \mathbb{Z}_q$ and computes $A = G^a$, $B = R^a$, $c = \mathcal{H}(R, S, A, B)$, and $z = a + cd$. Notice that $\mathcal{H}$ is a hash function that maps four group elements into $\mathbb{Z}_q$. The NIZK proof will thus consist of (c, z). Indeed, any verifier can be convinced of the statement by checking if $c \overset{?}{=} \mathcal{H}(R, S, G^z X^{-c}, R^z S^{-c})$. However, this proof would lose soundness if the prover is free to choose X. More details can be found in section 3 of [BPW12]. In our solution X is fixed by the verifier and then given to the prover.

3.3 Threshold Cryptosystems

When secret sharing is used, the participant or authority in charge of reconstructing the secret will obviously learn the secret. This is of course not a desirable property and to be instantiated in practice, secret sharing needs some modifications. Ideally we would like the secret to be shared (i.e. divided) among ℓ servers, with the constraint that servers could perform computations based on the secret without reconstructing it. One such application is threshold cryptography. Here we have the additional constraint that to be able to use the secret, $t + 1$ servers need to collaborate. Using the secret is achieved without being able to reconstruct it. Furthermore, no t or less servers could use the secret. In this paper we will focus on the example of threshold signatures rather than threshold decryption. Nevertheless, the latter could still be used at a higher efficiency cost.

Threshold Signatures. Participants to a threshold signature scheme consist of ℓ *signers*, a *trusted dealer* and an *adversary*. The scheme itself is composed by a set of five algorithms: $(KG, \Sigma_i, \Sigma_v, \Sigma_c, V_\sigma)$.
$KG(1^k, t, \ell) \longrightarrow (pk, \{sk_1, ..., sk_\ell\}, \{vk_1, ..., vk_\ell\}, vk)$. The *key generation algorithm* takes as input the security parameter k, the threshold parameter t and the number of participants ℓ. It outputs the public key pk of the system, ℓ secret keys sk_i together with their corresponding verification keys vk_i and the general verification key vk of the system.
$\Sigma_i(M, pk, vk, sk_i, vk_i) \longrightarrow (\sigma_i, \pi_i)$. The *partial signature algorithm* takes as input a message M, the general public key pk, the general verification key vk and the secret share sk_i with its verification key vk_i. It outputs a partial signature σ_i with a verification proof π_i on the validity of σ_i.

$\Sigma_v(M, pk, vk, \sigma_i, \pi_i, vk_i) \longrightarrow \{0,1\}$. The *partial signature verification algorithm* takes as input a message M, the general public key pk, the general verification key vk, the partial signature σ_i, its corresponding verification proof π_i and verification key vk_i. It checks the validity of σ_i and outputs the result. The verification of σ_i with π_i is used to achieve robustness.
$\Sigma_c(M, pk, \{\sigma_i\}_\Psi) \longrightarrow \sigma$. The *combining share algorithm* takes as input a message M, the public key pk and a set Ψ of size $t+1$ of valid partial signatures σ_i. It outputs the signature σ of m.
$V_\sigma(M, \sigma, pk) \longrightarrow \{0,1\}$. The *signature verification algorithm* takes as input a message M, its signature σ and the public key pk. It checks the validity of σ and outputs the result.

Threshold Signature Security Requirements. The security requirements for threshold signatures are *robustness*, *threshold security*, *unforgeability*, and optionally *proactive security*.
Robustness states that if all partial signatures used to create a signature σ on message M are valid then the signature σ is a valid signature of m.
The *threshold security* requirement states that no subset of t signers can produce a valid threshold signature on m. Moreover, any subset of $t+1$ or more signers can produce a valid threshold signature.
A threshold signature scheme is said to be *unforgeable* if a computationally bounded adversary is enable to forge a valid signature on a chosen message. In this case, the adversary is allowed to corrupt up to t signers.
Proactive security states that an update mechanism exists for signers to update their secret key share, without modifying the general public key of the system (nor the general verification key).

Threshold RSA Signatures. Let us remind briefly the RSA signature scheme [RSA78]. Let p and q be two large primes such that $n = pq$. Let $ed \bmod \phi(n) = 1$, where ϕ is the Euler's Totient function. Hence $\phi(n) = (p-1)(q-1)$. The public key of this system is (n, e). Let $\bar{\mathcal{H}}$ be a one-way mapping from the message space to $\mathbb{Z}_n^*$. To obtain a signature σ on a message M, the signer computes $\sigma = m^d \bmod n$, where $m = \bar{\mathcal{H}}(M)$. Hence d is part of the signer's private key. To verify the signature, it suffices to check the following: $\bar{\mathcal{H}}(M) \stackrel{?}{=} \sigma^e \bmod n$.

To obtain the threshold version of the RSA signature scheme, the secret d needs to be shared among ℓ servers. We assume the presence of a trusted party in charge of the key generation algorithm. In our case, this trusted party will be the DV. To share d, secret sharing will be used. However this cannot be done directly as revealing $\phi(n)$ to the signers would allow them to factorize n and thus compute d from e. Hence a single signer would be able to sign on behalf of the entire group. Extensive research has been done regarding Threshold RSA signatures and we will present here the solution proposed by Shoup in [Sho00]. However, depending on actual values for t and ℓ, solutions from King [Kin00] and Desmedt-Frankel [DF94] should also be considered.

Shoup [Sho00] suggested to use safe primes for the RSA modulus. Hence $n = pq =$

$(2p' + 1)(2q' + 1)$ such that p, p', q and q' are primes. Let $m = p'q'$. The value of m will not be revealed and should be kept secret from all parties. If no proactive security is needed, then m can be safely erased after the key generation phase. The public exponent e will be chosen as a prime with $e > \ell$. d will be picked such that $ed \equiv 1 \bmod m$ and shared using Shamir's secret sharing. Hence the secret share of signer i will be of the form $d_i = f(i) \bmod m$, where f is defined as in equation 3.1 with $a_i \in_R \mathbb{Z}_m$. Let $\mathcal{Q}_n$ be the subgroup of squares in $\mathbb{Z}_n^*$. The general verification key will be randomly chosen in $\mathcal{Q}_n$. The verification key of signer i will be set as $vk_i = vk^{d_i} \in \mathcal{Q}_n$. To compute the Lagrange coefficients we use the trick explained at the end of section 3.1 with $\Delta = \ell!$. To generate a partial signature on message M, signer i will first compute $x = \bar{\mathcal{H}}(M)$ and then $\sigma_i = x^{2\Delta d_i}$. The validity proof π_i consists of proving the statement $\log_{vk} vk_i = \log_{x^{4\Delta}} \sigma_i^2$. As we are working in a group of unknown order, computation is a bit more difficult. To solve this issue, it is enough to work with sufficiently large numbers. Hence the random a in Chaum-Pedersen [CP92] NIZK proof will be picked in $a \in_R \mathbb{Z}_{2^{\log_2 n + 2L_1}}$, where L_1 is a secondary security parameter (Shoup suggests $L_1 = 128$). The hash function $\mathcal{H}$ will thus output a L_1 bit integer such that $c = \mathcal{H}(vk, x^{4\Delta}, vk_i, \sigma_i^2, vk^a, x^{4\Delta a})$. Thus $\pi_i = (c, z)$ with $z = cd_i + a$. The corresponding verification will be $c \overset{?}{=} \mathcal{H}(vk, x^{4\Delta}, vk_i, \sigma_i^2, vk^z vk_i^{-c}, x^{4\Delta z} \sigma_i^{-2c})$. Suppose we have $t + 1$ valid partial signatures σ_j, combining them means computing the signature $\sigma = (\prod_{j \in \Psi} \sigma_j^{2\Delta\lambda_{0,j}^{\Psi}})^{\alpha} x^{\beta} \bmod n$, where α and β are obtained by solving the Euclidian algorithm $\alpha \cdot 4\Delta^2 + \beta \cdot e = 1$. Notice that α and β can be precomputed by the trusted authority. Finally, the verification of σ is the same as the standard RSA signature scheme. We refer readers to [Sho00] for more details as well as for the security proof.

4 New Solution

Let us now look out how we can go from threshold signatures to terminal revocation. The main idea is that we introduce terminal collaboration in order to achieve terminal authentication. Terminal revocation will thus be achieved with the help of neighboring terminals. Let us first specify our security assumptions. Then we will explain how we extend terminal authentication in order to achieve a better terminal revocation.

4.1 Security Assumptions

We assume the same structure of participants than the EAC model. However we make some precisions. Each DV is responsible for ℓ terminals (ℓ differs from one DV to the other). DVs play the role of trusted authority amongst their terminals. We assume the existence of secure and authenticated channels between all ℓ terminals. This is easily achieved with public key encryption as it is the same DV, i.e. a trusted party, that issued every terminal key pairs. When a terminal is stolen, its certificate will be revoked. This revocation will disable its use. Moreover, the lack of online connectivity should affect

only CVCAs and DVs as they are Public Key Generators. As such they should be turned offline once their keys setup generation has been achieved ([Sha84]). This is not the case for terminals.
Furthermore, we assume attackers to be *computationally bounded*. We will focus on threats targeting terminals, as they are somehow neglected in the current EAC. Nevertheless, we assume CVCAs and DVs to be honest. We consider a threshold security assumption, i.e. cases where the adversary can corrupt up to t terminals among $\ell \geqslant 2t + 1$. We will expect adversaries to be either *passive adversaries*, where attackers corrupt targets by reading their contents and secrets, or *active adversaries*, where attackers will additionally change the behavior of corrupted terminals. Lastly we restrict ourselves to *static adversaries*, meaning that the adversary will select which terminals to corrupt before the start of the protocol. Moreover, the adversary is free to corrupt them when he wants to. When a terminal is corrupted, all his communications will be revealed to the adversary. We set aside cases of dynamic adversaries as the corresponding solutions will induce a high loss in efficiency.

4.2 Augmented Terminal Authentication

Figure 1 gives a sketch of the general structure of our additional part to the current terminal authentication protocol. Our Setup phase is very similar to the original EAC one. DVs have to contact CVCAs from every other country, in order to obtain their DV certificate. The main difference is that now, certificates will contain an additional public key PK_{DV} corresponding to a secret key SK_{DV} only known by the DV and that will be shared among terminals. Moreover, certificates will contain additional information regarding how many terminals are required to collaborate in order to authenticate themselves (parameters t and ℓ). When a DV will set up his terminals, he will additionally give them a share d_i of his secret such that every terminal authentication will require the collaboration of at least $t+1$ of them. Hence our scheme tolerates up to t corrupted terminals. As long as $t + 1$ honest terminals are available, terminal authentication will be able to proceed. Once the Setup phase has been completed, only terminals and MRTDs are present in the interactions. Hence the DV can be used offline as described in the EAC standard.

DVs are in charge of the setup phase. They will run the key generation algorithm and distribute to each terminal its corresponding secret key, the public key pk of the system and the verification keys of all participants. After this step, DVs can be turned offline.

During the terminal authentication and just after the Certificate Chain Validation process, a MRTD will first select a random challenge M in the message space $\mathcal{M}$. He will then challenge the terminal with $(M||\widetilde{date})$ where $||$ denotes concatenation and $\widetilde{date}$ is the approximation of the current date stored in the MRTD. Moreover M must be independent from the MRTD identity, otherwise a tracking privacy threat would rise. Indeed, in this case the signature will prove that a given identity was at a given specific location and time. In order to sign the challenge, the terminal will have to collaborate with at least t other terminals. The revocation process takes place during the terminal collaboration. It will be the role of other terminals to determine whether the requesting terminal T_r is revoked or not. As terminals have real clocks and better computation capabilities than MRTDs, they will be able to check this revocation status much more efficiently. Any standard strong revocation mechanism can then be used here. The basic solution is to apply Certificate Validation as described in section 2.5 of [BfSidI12d], but with a real clock. More complex solution can also be used such as Certificate Revocation Lists (CRL) or with an Online Certificate Status Protocol (OCSP) if an OCSP responder is set up for terminals. If the requesting terminal is revoked, then his request can be simply ignored. If T_r status is not revoked, then a partial signature σ_i can be computed and sent to him, possibly with a verification proof π_i. At this stage, the requesting terminal will collect all valid t partial signatures and combine them with his own to create a global signature on the MRTD challenge. The latter will be sent to the MRTD as a proof of authenticity and non-revocation.

Once the MRTD receives the global threshold signature, he will have to verify it with the global public key of the DV. If the check is successful, he can be ensured that either the terminal knows the DV secret or that he has gone through a threshold signature involving some revocation checks. As we assume the DV to have correctly achieved the initial setup, the MRTD is ensured on the non-revocation status of the terminal.

$$\begin{tikzcd}
MRTD & T_r & T_{i \neq r} \\
M \in_R \mathcal{M} \arrow[r, "(M||\widetilde{date})"] & \sigma_r = \Sigma^{d_r}_{i=r}(M||\widetilde{date}) \arrow[r, "(M||\widetilde{date})"] & \text{Check revocation status} \\
 & \text{Check } \Sigma_v(\sigma_i, \pi_i) & (\sigma_i, \pi_i) = \Sigma^{d_i}_i(M||\widetilde{date}) \arrow[l, "\sigma_i, \pi_i"'] \\
\begin{array}{c}\text{Check}\\ V_\sigma(M||\widetilde{date})\end{array} & \sigma = \Sigma_c(\{\sigma_i\}_\Psi) \arrow[l, "\sigma"'] &
\end{tikzcd}$$

Figure 1: Terminal Authentication with Revocation

At this point, any efficient and secure threshold signature scheme can be used. In that regard, we suggest to use Shoup's threshold RSA signature [Sho00]. In this case, the MRTD computation will be dominated by one single exponentiation. The terminal communicating directly with the MRTD and in charge of combining the partial signatures, will have a computational complexity dominated by $(5t + 4)$ exponentiations. However this computational cost can be reduced to $(t + 5)$ exponentiations as explained in section 4.4. For the collaborating terminals, the computational cost is dominated by 3 exponentiations.

4.3 Security Outcome

We can distinguish two types of terminals: a requesting terminal and collaborating terminals. The requesting terminal communicates directly with the MRTD. Collaborating terminals have the responsibility of verifying the requesting terminal revocation status and they also have to participate in the creation of the threshold signature in case of a non-revoked requesting terminal. Terminals are either corrupted or honnest, and either revoked or non-revoked. The cases of revoked terminals is easy as no honnest terminals will interact with them.

As our augmented terminal authentication is enforced with threshold signatures, the security achieved is highly dependent on the security of the threshold signature scheme used. We assume a threshold signature scheme that is robust, unforgeable and threshold secure, as the one from [Sho00]. Hence any computationally bounded adversary corrupting at most t terminals will not be able to learn the master secret of the threshold signature scheme $(sk = d_0 = f(0))$. Moreover adversaries will not be able to forge valid signatures on chosen messages.

Our augmented terminal authentication is complete in the sense that if all parties are honest, the MRTD will obtain a valid threshold signature on his challenge, with the additional guarantee that the requesting terminal to which it is communicating is non-revoked and authenticated.

In the case where the requesting terminal is non-revoked and compromised, the adversary could gain access to sensitive data from the MRTDs that it encounters. However to do so, the requesting terminal will have to collaborate honestly with the other terminals. This adversarial behavior can be mitigated by monitoring the network and making sure that terminals only communicate with other known terminals.

When the requesting terminal is honnest and some collaborating terminals are corrupted and non-revoked, these terminals will be easily identified if they fail to provide valid verification proofs on their partial signatures. Moreover, the adversary will only learn contents of challenges without being able to link them to the MRTD that generated them.

Proactive security can be achieved by frequently renewing the global secret of the threshold signature scheme. This can be done efficiently by resharing the same secret with the means of sharing the “secret” value ’0’ and adding the obtained partial secrets to the previous ones. We can easily see this property with Lagrange interpolation. Assume the general secret is contained in $f(0)$ and that another function g, with $g(0) = 0$ is shared and added to the previous secret shares. The resulting addition will form another function f' such that $f'(0) = f(0)$. This method reduces the threat of terminal keys being exposed. In order to compromise the general secret key, an adversary will have to obtain $t + 1$ key shares during a same time frame. This allows DV certificates to protect their general secret used for threshold signature throughout their entire time validity. Notice that this step is highly efficient if performed by the DV, i.e. the DV generates the additional secret key shares and distribute them to their corresponding terminal. Verification keys will also have to be redistributed to every participants. However, this can be achieved without the need of the DV with secure multiparty computation.

To sum up, a stolen terminal will not be able to authenticate itself. A corrupted collaborating terminal will learn no information except that a MRTD with some approximation date has requested an authentication process. However, a corrupted requesting terminal interacting with a MRTD will be granted access to the MRTD sensitive data if the terminal behaves honestly with the other collaborating terminals. As long as at most t terminals are corrupted, the secret key used to authenticate terminals remains protected. Furthermore, the leakage of the secret key can be achieved only if at least $t + 1$ key shares are compromised within the same time frame of a resharing phase. These security properties are desirable as they improve the current state of the EAC. By lowering the trust in terminals, we increase the DV level of trust. This is an acceptable change as DVs are less exposed than terminals.

4.4 Efficiency Tuning

Regarding computational costs, several modifications can be brought to reduce them. First, the terminal in charge of combining partial signatures could perform the robustness checks solely if the resulting combined signature is not valid. Hence instead of computing $4t$ exponentiations he would first check the validity of the signature with one exponentiation. Furthermore, a minor enhancement consists of letting the DV precompute Δ, (α, β) and the Lagrange coefficients $\lambda_{0,j}^{\Psi}, \forall\Psi$, and storing them in each terminals during the set up phase. The drawback of this method is that it will require a storage space in terminals. This can be an issue especially with the Lagrange coefficients, as there are $(t+1)C_{\ell}^{t+1} = \frac{\ell!}{t!(\ell-t-1)!}$ elements to compute. In the case of a large ℓ (e.g. $\ell > 100$), we can see that exponentiation by Δ will slow down the system. In this scenario, the threshold signature scheme of Gennaro et al. [GHKR08] will be preferable as it will be more efficient.

Furthermore, a small efficiency gain could be obtained by using the threshold signatures of King [Kin00] which is itself derived from the Desmedt-Frankel [DF94] scheme. However, the gain in efficiency is achieved by an increased difficulty to implement them and a higher storage requirement.

4.5 Remarks

If a CVCA considers that the threshold t used in an organizational unit managed by a DV is too low, he can request the participation of a special terminal that will act as a revocation server (e.g. OCSP). The drawback with this method is that it introduces a single point of failure. Moreover, the DV participation in the revocation process should be avoided as it breaks the principle of closing the Public Key Generator (PKG) after key generation (first mentioned in [Sha84]). Ideally, the set up of the organizational unit under a DV should include enough terminals. As CVCAs provide foreign DV the ability to read their passport, it would be desirable that these DV protect this privilege and avoid its missuse. If the number of terminal needed is low (in a hotel, etc...), then new terminals should join the infrastructure of another existing DV organizational unit.

Let us also mention the existence of *multisignatures*. These are a type of threshold signature where the identity of signers is provided in the general signature. However, even the latest result in multisignatures that we could use, namely the scheme from Boldyreva [Bol03], would imply an important efficiency decrease.

The overhead in time of our suggested solution should be less than 0.1 seconds, assuming 30 MHz CPU for MRTDs, 520 MHz CPU for terminals, 802.11g wireless communication between terminals (net average of 22 Mbit/s) and 200 Kbit/s communication speed between MRTDs and terminals. We consider an upperbound of 50 exponentiations for requesting terminals, 3 exponentiations for collaborating terminals and 1 exponentiation for the MRTD. Each message sent is around 1 Kbit except the messages from collaborating terminals that are around 3 Kbits.

References

[BDFK12] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. The PACE—AA Protocol for Machine Readable Travel Documents, and Its Security. In Angelos D. Keromytis, editor, *Financial Cryptography*, volume 7397 of *Lecture Notes in Computer Science*, pages 344–358. Springer, 2012.

[BFK09] Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio Agostino Ardagna, editors, *ISC*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2009.

[BfSidI09] Bundesamt für Sicherheit in der Informationstechnik. PKIs for Machine Readable Travel Documents – Protocols for the Management of Certificates and CRLs. Technical report, Federal Office for Information Security, 53133 Bonn, Germany, 2009. Technical Guideline TR-03129, Version 1.10.

[BfSidI12a] Bundesamt für Sicherheit in der Informationstechnik. Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control (EAC). Technical report, Federal Office for Information Security, 53133 Bonn, Germany, 2012. Technical Guideline TR-03110, Version 1.00.

[BfSidI12b] Bundesamt für Sicherheit in der Informationstechnik. Advanced Security Mechanisms for Machine Readable Travel Documents – Part 1. Technical report, Federal Office for Information Security, 53133 Bonn, Germany, 2012. Technical Guideline TR-03110-1, Version 2.10.

[BfSidI12c] Bundesamt für Sicherheit in der Informationstechnik. Advanced Security Mechanisms for Machine Readable Travel Documents – Part 2. Technical report, Federal Office for Information Security, 53133 Bonn, Germany, 2012. Technical Guideline TR-03110-2, Version 2.10.

[BfSidI12d] Bundesamt für Sicherheit in der Informationstechnik. Advanced Security Mechanisms for Machine Readable Travel Documents – Part 3. Technical report, Federal Office for Information Security, 53133 Bonn, Germany, 2012. Technical Guideline TR-03110-3, Version 2.10.

[BLR12] Rima Belguechi, Patrick Lacharme, and Christophe Rosenberger. Enhancing the privacy of electronic passports. *IJITM*, 11(1/2):122–137, 2012.

[Bol03] Alexandra Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643. Springer, 2012.

[CP92] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.

[CV09] Rafik Chaabouni and Serge Vaudenay. The Extended Access Control for Machine Readable Travel Documents. In Arslan Brömme, Christoph Busch, and Detlef Hühnlein, editors, *BIOSIG*, volume 155 of *LNI*, pages 93–103. GI, 2009.

[DF94] Yvo Desmedt and Yair Frankel. Perfect Homomorphic Zero-Knowledge Threshold Schemes over any Finite Abelian Group. *SIAM J. Discrete Math.*, 7(4):667–679, 1994.

[EP06] Council European Parliament. Regulation (EC) No 562/2006 of the European Parliament and of the Council of 15 March 2006 establishing a Community Code on the rules governing the movement of persons across borders (Schengen Borders Code). `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32006R0562:EN:NOT`, March 2006.

[Fri] Friedhelm. 2012 Facts and Figures on Frankfurt Airport. `http://www.frankfurt-airport.com/content/frankfurt_airport/en/misc/container/facts-and-figures-2011/jcr:content.file/zadafa-2012_e_lowres.pdf`.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GHKR08] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Threshold RSA for Dynamic and Ad-Hoc Groups. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 88–107. Springer, 2008.

[ICAO08] International Civil Aviation Organization. Machine Readable Travel Documents – Document 9303. Technical report, ICAO, 2005-2008. `http://www.icao.int/Security/mrtd/Pages/Document9303.aspx`.

[ICAO13] International Civil Aviation Organization. Machine Readable Travel Documents – SUPPLEMENT to Document 9303. Technical report, ICAO, 2013. `http://www.icao.int/Security/mrtd/Pages/Document9303.aspx`.

[Kin00] Brian King. Improved Methods to Perform Threshold RSA. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 359–372. Springer, 2000.

[LZJX10] C. H. Li, X. F. Zhang, H. Jin, and W. Xiang. E-passport EAC scheme based on Identity-Based Cryptography. *Inf. Process. Lett.*, 111(1):26–30, 2010.

[MVV07] Jean Monnerat, Serge Vaudenay, and Martin Vuagnoux. About Machine-Readable Travel Documents. In *In Proceedings of the International Conference on RFID Security 2007*, pages 15–28, 2007.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha84] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[Sho00] Victor Shoup. Practical Threshold Signatures. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2000.