The PACE|AA Protocol for Machine Readable Travel Documents, and Its Security

Jens Bender[1], Özgür Dagdelen[2], Marc Fischlin[2], and Dennis Kügler[1]

[1] Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany
[2] Darmstadt University of Technology, Germany

Abstract. We discuss an efficient combination of the cryptographic protocols adopted by the International Civil Aviation Organization (ICAO) for securing the communication of machine readable travel documents and readers. Roughly, in the original protocol the parties first run the Password-Authenticated Connection Establishment (PACE) protocol to establish a shared key and then the reader (optionally) invokes the Active Authentication (AA) protocol to verify the passport's validity. Here we show that by carefully re-using some of the secret data of the PACE protocol for the AA protocol one can save one exponentiation on the passports's side. We call this the PACE|AA protocol. We then formally prove that this more efficient combination not only preserves the desirable security properties of the two individual protocols but also increases privacy by preventing misuse of the challenge in the Active Authentication protocol. We finally discuss a solution which allows deniable authentication in the sense that the interaction cannot be used as a proof towards third parties.

1 Introduction

Through ISO/IEC JTC1 SC17 WG3/TF5 [ICA10] the International Civil Aviation Organization (ICAO) has adopted the Password Authenticated Connection Establishment (PACE) protocol [BSI10] to secure the contactless communication between machine-readable travel documents (including identity cards), and a reader. Roughly, the protocol generates a secure Diffie-Hellman key out of a low-entropy password which the owner of the passport has to enter at the reader, or which is transmitted through a read-out of the machine-readable zone. The Diffie-Hellman key is subsequently used to secure the communication. In [BFK09] it has been shown that the PACE protocol achieves the widely accepted security notion of password-based authenticated key agreement of Bellare-Pointcheval-Rogaway [BPR00], in its strong form of Abdalla et al. [AFP05]. This holds under a variant of the Diffie-Hellman assumption, assuming secure cryptographic building blocks, and idealizing the underlying block cipher and the hash function.

According to European endeavors, the PACE protocol should be followed by the extended access control (EAC) authentication steps, called Chip Authentication (CA) and Terminal Authentication (TA), with high-entropic certified keys. This should ensure that access for either party is granted based on strong

A.D. Keromytis (Ed.): FC 2012, LNCS 7397, pp. 344–358, 2012.

cryptographic keys (i.e., not relying on low-entropy passwords only). The security of the EAC protocols and of the composition with PACE has been discussed in [DF10].

In the specifications of the ICAO 9303 standard [ICA06] for the border control scenario, the normative document about machine-readable travel documents, however, only passive authentication of the passport is mandatory, where the passport essentially merely sends its (authenticated) data. Active Authentication (AA) of the passport, implemented through a signature-based challenge-response protocol, is only optional. If AA is not enforced this potentially allows to bypass authentication through cloning of passports. Even if AA is used, then the (plain) challenge-response protocol introduces a potential threat to privacy, as discussed in [BSI10] (see also [BPSV08b, BPSV08a, MVV07]). Namely, if the terminal can encode a time stamp or the location into the challenge, then the signature on that challenge can be used as a proof towards third parties about the location or time of the border check. In this sense, the passport cannot deny this interaction. This problem has been explicitly addressed in the European Chip Authentication protocol (where a message authentication code for a shared key is used for the challenge-response step instead).

Combining PACE and AA. We discuss that, on the chip's side, we can re-use some of the (secret) data in the PACE step for the AA step to save the exponentiation for the signature in AA on the chip's side, giving Active Authentication (almost) for free.

To understand our technique, we need to take a closer look at the PACE protocol. The PACE protocol first maps the short password to a random group element through an interactive sub protocol Map2Point, followed by a Diffie-Hellman key exchange step for this group element, and concludes with an authentication step. While the latter steps are somewhat canonical, the Map2Point step can be instantiated by different means and allows a modular design. The most common instantiations are based on another Diffie-Hellman step (used within the German identity card), or on hashing into elliptic curves as proposed by Icart [Ica09] and Brier et al. [BCI+10]. The security proof for PACE [BFK09] holds for general Map2Point protocols satisfying some basic security properties.

Our improvement works for the Diffie-Hellman based Map2Point protocol as implemented on the German identity cards, for example, since the chip can re-use its secret exponent from the Diffie-Hellman step of the Map2Point protocol. We discuss two alternatives how to carry out the AA step with this exponent more efficiently, one based on DSA signatures and the other one using Schnorr signatures. We note that the idea applies more generally to other discrete-log based signature schemes. The challenge in the new AA step is now the authentication data sent by the terminal in the PACE step.

Security of the Combined Protocol. Whenever secret data is used throughout several sub protocols great care must be taken in cryptography not to spoil the security of the overall protocol. We thus show that sharing the data between the PACE protocol and the new AA sub protocol preserves the desirable security properties. More precisely, we show that:

- In the combined PACE|AA protocol we still achieve the security of a password-based authenticated key exchange protocol (thus showing that the deployment of the randomness in the extra AA step does not violate the security of the PACE protocol), and
- the overall protocol still authenticates the chip securely (in a high-entropy sense), even when many executions of PACE|AA take place. To this end, we define a strong security model for authentication, essentially only excluding trivial attacks, e.g., if the adversary gets possession of the secret key, or simply relays information in executions.

It follows that the PACE|AA protocol achieves the previous security standards of the individual protocols but comes with a clear efficiency improvement. We note that the underlying assumptions are essentially the same as for PACE and AA, i.e., besides the common assumptions about secure encryption, signature, and MAC algorithms, we reduce the security of the combined protocol to the security of PACE (as an authenticated key-exchange protocol) and to a variant of the security of Schnorr signatures resp. DSA signatures (where the adversary now also gets access to a decisional Diffie-Hellman oracle and can decide upon the message to be signed after seeing the first half of the signature).

A Deniable Schnorr Version. As explained before, for privacy reasons it may be important that the terminal cannot derive a proof for others from the interaction with the passport or identity card *that* an interaction took place. Put differently, the protocol should provide *deniable authentication* [DDN00]. This roughly means that the terminal could have generated its view in the protocol itself from the public data, without communicating with the passport. This implies that the passport holder can deny any actual interaction and claim the terminal to have made up this conversation.

We note that the previously discussed signature based protocols do not support deniability. The reason is that the terminal could not have created the signature under the passport's key without the signing key —or without communicating with the actual chip. For the (ordinary) AA variant the terminal is even allowed to encode *any* information in the challenge, in our improved combinations the challenge is "only" a MAC computed over data provided by the passport and the shared Diffie-Hellman key. If this allows to encode information depends on the MAC.

In contrast, our proposed deniable variant does not rely on Schnorr signatures, but in some sense rather on the interactive Schnorr identification scheme for honestly chosen challenges. This identification scheme is deniable because one can simulate the interaction via the well-known zero-knowledge simulator.[1]

[1] It is this property which is not known to work for the DSA case and why we restrict ourself to the Schnorr scheme. Note also that Schnorr signatures are also somewhat simulatable but only if one programs the random oracle hash function; this, however, is not admissible for the notion of deniability. We nonetheless still use a hash function in the solution but use programmability only to show the unforgeablity/impersonation resistance property, not the deniability proof.

Interestingly, our variant is essentially as efficient as the signature based one, but comes with the advantage of deniability.

2 Security Model

We use the real-or-random security model of Abdalla et al. [AFP05] which extends the model of Bellare et al. [BPR00] for password-based key exchange protocols. Due to space limitations, we refer the reader to [BFK09] for the description of the attack model and security notion. Some changes are necessary, though, because we now incorporate a long-term signing key of the chip. These minor modifications follow next.

Attack Model. We consider security against active attacks where the adversary has full control over the network, and the adversary's goal is to distinguish genuine keys from random keys in executions, which are picked independently of the actual protocol run. This corresponds to the so-called real-or-random setting [AFP05], a stronger model than the original find-then-guess model of [BPR00], where the adversary can see several test keys (instead of a single one only).

In the attack, each user instance is given as an oracle to which an adversary has access, basically providing the interface of the protocol instance (via the usual Send, Execute, Reveal, and Test commands to send messages to parties, to observe executions between honest parties, to reveal the session key, and to be challenged for a test key). In addition, there exists a Corrupt oracle in the model from [AFP05]. The adversary can gain control over a user during the execution by issuing a Corrupt query with which the adversary obtains the secrets of an honest party. For sake of convenience, here we split these queries into Corrupt.pw and Corrupt.key queries, where the former reveals the password only and the latter discloses the long-term key only (in case of a chip); in both cases, the other secret remains private. Note that we now can model Corrupt queries by both queries (since we work in the weak corruption model where the parties' internal states are not revealed upon corruption). An honest party gets adversarially controlled if it does not have any secrets left (i.e., if the adversary issues both Corrupt query types for a chip, or the Corrupt.pw query for the terminal).

The adversary can make the following queries to the interface oracles other than these from [AFP05]:

Corrupt.pw(U) The adversary obtains the party's password π.

Corrupt.key(U) The adversary obtains the party's cryptographic key sk (if it exists).

In addition, since the original PACE protocol was cast in the random oracle and ideal cipher model where oracles providing a random hash function oracle and an encryption/decryption oracle are available, the attacker may also query these oracles here. (We note that we only use the ideal cipher implicitly through the reduction to the security to PACE.)

Partners, Correctness and Freshness. Upon successful termination, we assume that an instance U_i outputs a session key k, the session ID *sid*, and a user ID *pid* identifying the intended partner (assumed to be empty in PACE for anonymity reasons but containing the chip's certificate in the combined PACE|AA protocol). We note that the session ID usually contains the entire transcript of the communication but, for efficiency reasons, in PACE it only contains a part thereof. This is inherited here. We say that instances A_i and B_j are *partnered* if both instances have terminated in accepting state with the same output. In this case, the instance A_i is called a partner to B_j and vice versa. Any untampered execution between honest users should be partnered and, in particular, the users should end up with the same key (this correctness requirement ensures the minimal functional requirement of a key agreement protocol).

Neglecting forward security for a moment, an instance (U, i) is called *fresh* at the end of the execution if there has been no $\mathsf{Reveal}(U, i)$ query at any point, neither has there been a $\mathsf{Reveal}(B, j)$ query where B_j is a partner to U_i, nor has somebody been corrupted (i.e., neither kind of Corrupt query has been issued). Else, the instance is called *unfresh*. In other words, fresh executions require that the session key has not been leaked (by neither partner) and that no Corrupt-query took place.

To capture forward security we refine the notion of freshness and further demand from a fresh instance (U, i) as before that the session key has not been leaked through a Reveal-query, and that for each $\mathsf{Corrupt.pw}(U)$- or $\mathsf{Corrupt.key}(U)$-query there has been no subsequent $\mathsf{Test}(U, i)$-query involving U, or, if so, then there has been no $\mathsf{Send}(U, i, m)$-query for this instance at any point.[2] In this case we call the instance *fs-fresh*, else *fs-unfresh*. This notion means that it should not help if the adversary corrupts some party after the test query, and that even if corruptions take place before test queries, then executions between honest users are still protected (before or after a Test-query).

AKE Security. The adversary eventually outputs a bit b', trying to predict the bit b of the Test oracle. We say that the adversary wins if $b = b'$ and instances (U, i) in the test queries are fresh (resp. fs-fresh). Ideally, this probability should be close to 1/2, implying that the adversary cannot significantly distinguish random keys from session keys.

To measure the resources of the adversary we denote by t the number of steps of the adversary, i.e., its running time, (counting also all the steps required by honest parties); q_e the maximal number of initiated executions (bounded by the number of Send- and Execute-queries); q_h the number of queries to the hash oracle, and q_c the number of queries to the cipher oracle. We often write $Q = (q_e, q_h, q_c)$ and say that $\mathcal{A}$ is (t, Q)-bounded.

Define now the AKE advantage of an adversary $\mathcal{A}$ for a key agreement protocol P by

[2] In a stronger notion the adversary may even issue a Corrupt.key command for the user before the testing; Due to the entanglement of the PACE and the AA protocol here our protocol does not achieve this, though.

$$\mathbf{Adv}_P^{ake}(\mathcal{A}) := 2 \cdot \mathrm{Prob}[\mathcal{A} \text{ wins}] - 1$$
$$\mathbf{Adv}_P^{ake}(t, Q) := \max\left\{ \mathbf{Adv}_P^{ake}(\mathcal{A}) \;\middle|\; \mathcal{A} \text{ is } (t, Q)\text{-bounded} \right\}$$

The forward secure version is defined analogously and denoted by $\mathbf{Adv}_P^{ake-fs}(t, Q)$.

Impersonation Resistance. This security property says that the adversary, in the above attack, *successfully impersonates* if an honest reader in some session accepts with partner identity *pid* and session id *sid*, but such that (a) the intended partner U in *pid* is not adversarially controlled or the public key in *pid* has not been registered, and (b) no Corrupt.key command to U has been issued before the reader has accepted, and (c) the session id *sid* has not appeared in another accepting session. This roughly means that the adversary managed to impersonate an honest chip or to make the reader accept a fake certificate, without knowing the long-term secret or relaying the data in a trivial man-in-the-middle kind of attack.

Define now the IKE advantage (I for impersonation) of an adversary $\mathcal{A}$ for a key agreement protocol P by

$$\mathbf{Adv}_P^{ike}(\mathcal{A}) := \mathrm{Prob}[\mathcal{A} \text{ successfully impersonates}]$$
$$\mathbf{Adv}_P^{ike}(t, Q) := \max\left\{ \mathbf{Adv}_P^{ike}(\mathcal{A}) \;\middle|\; \mathcal{A} \text{ is } (t, Q)\text{-bounded} \right\}$$

Note that we do not need to define a forward secure version here.

3 The PACE|AA Protocol

In this section, we describe the PACE|AA protocol and both options for authentication in the last message, i.e., active authentication (AA) via Schnorr and via DSA. The deniable Schnorr variant and its security is addressed in Section 6.

3.1 Protocol Description

Figure 1 illustrates the PACE|AA protocol with both options of authentication at the end. The scheme itself uses a block cipher $\mathcal{C}(K_\pi, \cdot) : \{0,1\}^\ell \to \{0,1\}^\ell$ and a hash function $\mathcal{H}$, with values $1, 2, \ldots$ in fixed-length encoding prepended to make evaluations somewhat independent.

The chip already holds a certificate $cert_C$ for its public key X_A under the authorities' public key $pk_{\mathcal{CA}}$, and (authenticated) group parameters $\mathcal{G} = (a, b, p, q, g, k)$ describing a subgroup of order q, generated by g, of an elliptic curve for parameters a, b, p for security parameter k. We also note that, throughout the paper, we use the multiplicative notation for group operations. It is understood that, if working with elliptic curves, multiplications correspond to additions and exponentiations to multiplications. Then the parties run the PACE protocol,

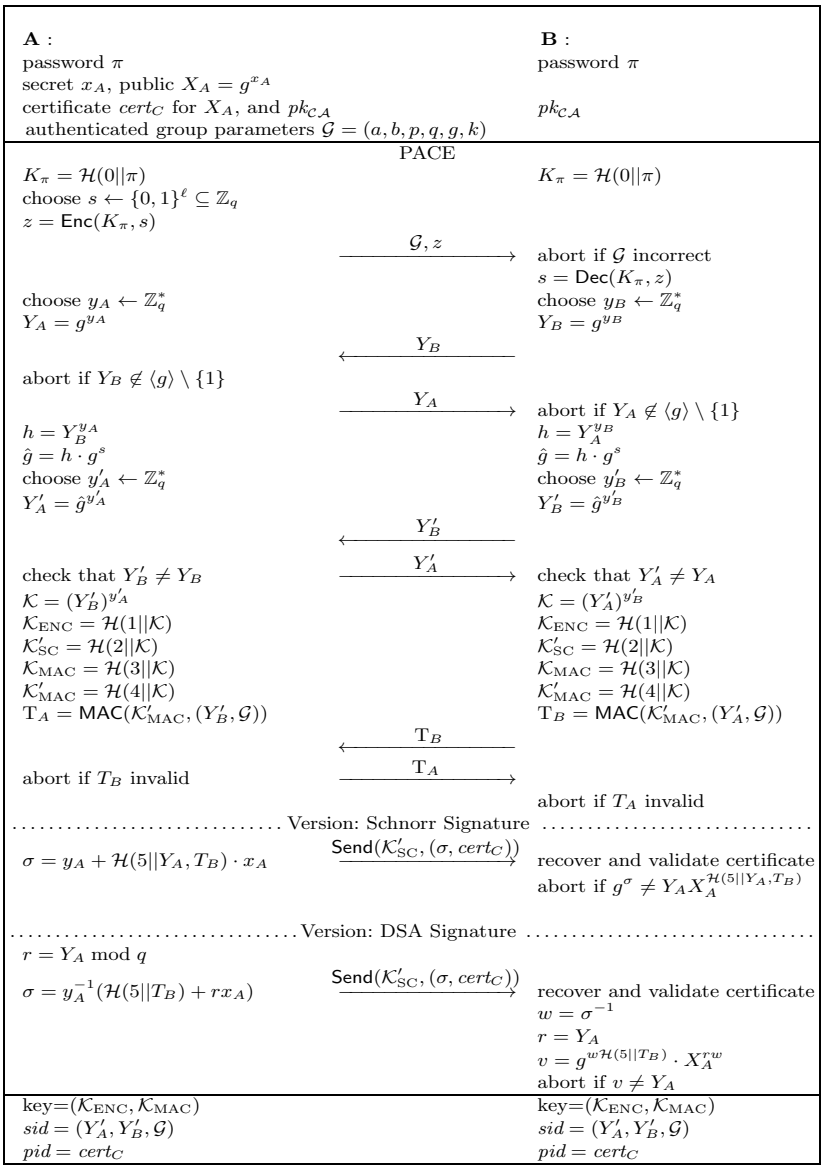

Fig. 1. The PACE|AA protocol (all operations are modulo q)

with the chip sending a nonce encrypted under the password, running the Diffie-Hellman based Map2Point protocol to derive another generator $\hat{g}$ on which another Diffie-Hellman key exchange is then performed. In this Map2Point step the chip uses some secret exponent y_A to send $Y_A = g^{y_A}$. The parties in the PACE protocol finally exchange message authentication codes $\mathrm{T}_A, \mathrm{T}_B$.

The idea is now roughly to re-use the secret exponent y_A in the Map2Point sub protocol on the chip's side for the signature generation, and use the authentication value T_B of the terminal as the challenge on which the signature is computed. The chip then sends its certificate (along with the missing signature part) over the secure channel, via a Send command for the key $\mathcal{K}'_{\mathrm{SC}}$ derived from the Diffie-Hellman exchange. The reader may think for now of the secure channel as an authenticated encryption, but other channel instantiations work as well.

3.2 Instantiations

There are essentially two possible instantiations. One is based on the Schnorr signature scheme [Sch90] where the chip uses the values y_A and Y_A as the (private resp. public) randomness and T_B as the challenge for creating the signature under its long-term signature key X_A. We call this option *Active Authentication via Schnorr signatures*. Alternatively, the chip card might prove its authenticity by providing a DSA signature where again y_A and Y_A are used as the randomness for the signature generation [Kra95]. This version is called *Active Authentication via DSA signatures*. We note that the computation of the final signatures requires only modular multiplications (and, in case of DSA, an inversion) instead of exponentiations.

4 Security Assumptions

As remarked above we carry out our security analysis assuming an ideal hash function (random oracle model). Basically, this assumption says that $\mathcal{H}$ acts like a random function to which all parties have access. We do not make any explicit assumption about the cipher $\mathcal{C}$ here, but note that the security proof for PACE in [BFK09] (to which we reduce AKE security to) relies on an ideal cipher.

4.1 Cryptographic Primitives

For space reasons, we omit the standard definitions of the cryptographic primitives for message authentication, signatures, certificates, and for secure channels. In the theorems' statements, we denote by $\mathbf{Adv}^{attack}_{\mathcal{S}}(t, Q)$ an upper bound on an adversary running in time t (and making Q queries of the corresponding type) and breaking the scheme $\mathcal{S}$ in an attack of type *attack*. For secure channels we consider a simultaneous attack in which the adversary either tries to distinguish messages sent through the channel or to successfully inject or modify transmissions. We denote the adversary's advantage in this case by $\mathbf{Adv}^{lor}_{\mathcal{SC}}(t, Q)$.

4.2 Number-Theoretic Assumptions

Our proof for the AKE security of the PACE|AA protocol follows by reduction to the security of the original PACE protocol (and from the security of cryptographic primitives for the channel). For the IKE security against impersonators,

we nonetheless need two number-theoretic assumptions related to the Diffie-Hellman resp. discrete-log problems. The first one is the gap Diffie-Hellman problem [BLS01]. For a group $\mathcal{G}$ generated by g let $\mathrm{DH}(X, Y)$ be the Diffie-Hellman value X^y for $y = \log_g Y$ (with g being an implicit parameter for the function). Then the gap Diffie-Hellman assumption says that solving the computational DH problem for (g^a, g^b), i.e., computing $\mathrm{DH}(g^a, g^b)$ given only the random elements (g^a, g^b) and $\mathcal{G}, g$, is still hard, even when one has access to a decisional oracle $\mathrm{DDH}(X, Y, Z)$ which returns 1 iff $\mathrm{DH}(X, Y) = Z$, and 0 otherwise. We let $\mathbf{Adv}^{\mathrm{GDH}}(t, q_{DDH})$ denote (a bound on) the value ϵ for which the GDH problem is (t, q_{DDH}, ϵ)-hard.

Furthermore, for the Schnorr signature based solution we rely on the following version which (a) allows access to a decisional DH oracle for the forger, and (b) considers access to a signer in an online/offline fashion in the sense that the adversary may ask to see the public randomness part first before deciding on a message to be signed. Still, the goal is to create a signature on a new message for which the signing has not been completed. We note that the proof in [PS00] for Schnorr signatures still holds, assuming that computing discrete-logarithms relative to a DDH-oracle is hard. In particular, the hardness of this "gap discrete-log problem" is implied by the GDH hardness. We call this security notion *robust* unforgeability as it should still hold in presence of the DDH oracle and the delayed message choice.

Definition 1 (Robust Unforgeability of Schnorr Signatures). *The Schnorr signature scheme is (t, Q, ϵ)-robustly-unforgeable with $Q = (q_R, q_{DDH})$ if for any adversary $\mathcal{A}$ running in total time t, making at most q_{DDH} DDH oracle queries and at most q_R init-queries to oracle $\mathcal{O}$ the probability that the following experiment returns 1 is most ϵ:*

pick $\mathcal{G}$ (including a generator g of prime order q)
pick $sk \leftarrow \mathbb{Z}_q$ and let $pk = g^{sk}$
let $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(sk,\cdot),DDH(\cdot,\cdot,\cdot)}(\mathcal{G}, g, pk)$ for $\sigma^* = (c^*, s^*)$
where stateful oracle $\mathcal{O}$ upon input init picks $r \leftarrow \mathbb{Z}_q$ and returns $R = g^r$; and upon input $(\mathsf{complete}, R, m)$ checks if it has returned $R = g^r$ to a request init before, and if so, returns $r + \mathcal{H}(R, m)sk \bmod q$;
output 1 iff $c^* = \mathcal{H}(g^{s^*} pk^{c^*}, m^*)$ and m^* was no input to a $\mathsf{complete}$-query

We let $\boldsymbol{Adv}^{r-forge}_{Schnorr}(t, Q)$ be the maximal advantage for any adversary running in time t, making in total $Q = (q_R, q_{DDH})$ queries.

As it turns out to be useful for our deniable version, we remark that the proof of Pointcheval and Stern [PS00] holds as long as the input to the hash oracle in the forgery is new, i.e., one can extract the discrete-logarithm of the public key even if the hash function in signature requests is evaluated on quasi unique inputs, and the forgery, too, uses a previously unqueried hash function input. For the notion of signature unforgeability this holds because each signature request uses a high-entropic random group element and the message m^* in the forgery cannot have been signed before. We take advantage of this fact for our deniable

version where we insert $(Y'_A, \mathcal{G})$ instead of (R, m) into the hash function for the random group element Y'_A chosen by the chip respectively, signer. We also show that for the proof of impersonation resistance the adversary cannot re-use one of these values $(Y'_A, \mathcal{G})$ but needs to pick a new value Y'_A, thus showing the second property.

For the DSA based solution, we require an analogous assumption which is omitted here for space reasons and refer to the full version of this paper.

5 Security Analysis of PACE|AA

In this section, we discuss the security of the PACE|AA protocol when active authentication is done via Schnorr signatures; the case of DSA signatures follows, too, because we do not use any specific properties of the underlying signature scheme (except for the robust unforgeability). That is, we assume that the chip, holding public key $X_A = g^{x_A}$ with certificate $cert_C$, signs the message Y_B with key x_A and randomness Y_A. The signature is given by $\sigma = y_A + cx_A \bmod q$ for $c = \mathcal{H}(5||Y_A, T_B)$. After the final authentication step of PACE, the chip sends (using already a secure channel) the values σ and $cert_C$ to the reader who verifies the signatures and the certificate (and aborts in case one of the verification fails).

As noted in [BFK09] using the derived keys already in the key agreement step does not allow for a proof in the Bellare-Pointcheval-Rogaway model. We hence also use the variant that the keys $\mathcal{K}'_{\mathrm{SC}}$ and $\mathcal{K}'_{\mathrm{MAC}}$ are independent from the keys output as the result of the key agreement.

5.1 Security as a Key Exchange Protocol

Theorem 1. *The protocol PACE|AA (with Schnorr or DSA signatures) satisfies:*

$$\boldsymbol{Adv}^{ake}_{PACE|AA}(t, Q) \leq \frac{q_e^2}{2q} + \boldsymbol{Adv}^{lor}_{\mathcal{SC}}(t^*, q_e, q_e) + \boldsymbol{Adv}^{ake}_{PACE}(t^*, Q)$$

where $t^* = t + O(kq_e^2 + kq_h^2 + kq_c^2 + k^2)$ *and* $Q = (q_e, q_c, q_h)$.

We remark that the time t^* covers the additional time to maintain lists and perform look-ups. Since PACE is secure (under cryptographic assumptions) it follows together with the security of the underlying encryption scheme that the PACE|AA scheme is secure as well.

The idea of the proof is roughly that the additional Schnorr signature does not violate the security of the underlying PACE protocol as it is encrypted. This is shown through a reduction to the security of the original PACE protocol, mildly exploiting the structure of the original proof in [BFK09] and the properties of the Schnorr signature scheme. We roughly show that, in the PACE|AA protocol, we can simulate the final transmission of the signature token by sending dummy values through the channel, because the keys used to secure this transmission are "as secure as" the PACE keys. That is, even though the strength of the keys is only password-protected (i.e., one can try to guess the low-entropy password), this is sufficient for our purpose, as we do not plan to be more secure than that.

Proof. The proof uses the common game-hopping technique, gradually taking away adversarial success strategies and discussing that each modification cannot contribute significantly to the overall success probability. Note that the original proof of PACE in [BFK09] actually shows something stronger than indistinguishability of keys (from random). The proof rather shows that computing the Diffie-Hellman key $\mathcal{K}$ in an execution is hard (unless one knows or has guessed the password); key indistinguishability then follows from this. We will use this more fine-grained view on the proof below and also consider the adversary on the PACE|AA protocol in this regard, i.e., we measure its success probability according to the probability of making a hash query about $\mathcal{K}$ in a Test session (called target hash query).

Game 0: Corresponds to an AKE attack on the PACE|AA protocol (with the more fine-grained success notion).

Game 1: Abort Game 0 if an honest chip would compute the same Diffie-Hellman key in two executions.

Note that, since the honest chip always goes second for the Diffie-Hellman key exchange step, sending Y'_A, the keys in such executions are random elements and the probability that such a collision occurs is thus at most $\frac{1}{2}q_e^2/q$.

Game 2: Change the previous game slightly such that, an honest chip when sending the encrypted signature, instead picks and uses random and independent (independent of the hash function output) keys $\mathcal{K}'_{\mathrm{SC}}$.

Note that the only difference between the two cases can occur if the adversary makes a target hash query since Reveal and Test sessions never output these keys and Diffie-Hellman keys are distinct by the previous game. It follows that the adversarial success can only decrease by the probability of making a target hash query in this new game.

Game 3: Change the game once more and replace channeled transmissions of the signatures sent by an honest chip by encryptions of 0-bits of the same length and, at the same time, let any honest terminal reject any final message unless it has really been sent by the honest chip in the same session.

Note that the length (of the signature part and the certificate) is known in advance. Note also that the probability of making a target hash query in Game 3 cannot be significantly larger, by the distinguishing advantage of genuine transmissions from all-zero transmissions. To make this claim more formally, assume that we mount an attack on the left-or-right security of the (multi-user) encryption scheme by simulating the entire Game 2 with two exceptions: (1) If an honest chip is supposed to send the signature and certificate, then we simply call the next transmission challenge oracle about the signature part and the certificate and about an all-zero message of the same length. Then the challenge bit of the left-or-right oracle corresponds exactly to the difference between the two games. (2) If the adversary successfully modifies the final transmission of an honest chip and the honest terminal would accept the message, then this would also constitute a security breach of the channel protocol. Hence, if the success probabilities of the adversary dropped significantly, we would get a successful attacker against the secure channel scheme.

The final game can now be easily cast as an attack on the original PACE protocol. That is, if there was a successful attacker in Game 3 (making a target hash query), then there was a straightforward attacker with the same probability in the original PACE protocol: this attacker would run the Game 3-adversary and simulate the additional signature steps itself (i.e., creating keys and certificates), inject the values from the PACE protocol (i.e., relay the communication), but send dummy values $0 \ldots 0$ through the channel on behalf of honest chips under independent random keys. It follows that the probability of making a target hash query in Game 3 is also bounded by the PACE security.

Given that no target hash query is made, the advantage in the final game is now bounded from above by the advantage against PACE. Note that the advantage of breaking PACE simultaneously covers both the case of target hash queries and of breaks otherwise (such that we do not need to account for the advantage of target hash queries and then of other breaks, resulting in a factor 2). □

On Forward Security. Note that the PACE|AA protocol inherits the forward security of PACE (when used as authenticated key exchange protocol). That is, even if the adversary knows the password, then executions between honest parties remain protected. Since the security of PACE|AA essentially reduces to the security of PACE any successful attack against the forward security of PACE|AA would yield a successful attack against PACE; the other protocol steps do not violate this property.

5.2 Security against Impersonation

It remains to show that the protocol is IKE-secure. Here, we only rely on the unforgeability of certificates and MACs and the robust unforgeability of the Schnorr/DSA signature scheme.

Theorem 2. *For the PACE|AA protocol (with Schnorr or DSA signatures) it holds:*

$$\begin{aligned}
&\boldsymbol{Adv}^{ike}_{PACE|AA}(t, Q) \\
&\quad \leq \frac{q_e^2 + q_e q_h}{q} + \boldsymbol{Adv}^{forge}_{\mathcal{CA}}(t^*, q_e) + 2q_e \cdot \boldsymbol{Adv}^{forge}_{\mathcal{M}}(t^*, 2q_e, 2q_e) \\
&\qquad + \boldsymbol{Adv}^{r-forge}_{\{Schnorr|DSA\}}(t^*, q_e)
\end{aligned}$$

where $t^* = t + O(kq_e^2 + kq_h^2 + k^2)$ *and* $Q = (q_e, q_h)$.

The idea is to show first that the adversary cannot inject its own unregistered key (unless it breaks the unforgeability of the certification authority). Since any successful attack must be then for an uncorrupt party whose secret signing key was not revealed, it follows that the adversary must produce a signature under the (registered) public key of an honest user. Because the session id must be new and is somewhat signed via T_B, it follows that the adversary must forge Schnorr respectively DSA signatures in order to break the IKE property.

The formal proof appears in the full version of the paper.

6 A Deniable Schnorr Variant

Deniability basically demands that for any (possibly malicious) party on either side, there exists a simulator $\mathcal{S}$ which produces the same output distribution as the malicious party but without communicating with the honest party (but only receiving the honest party's public input and the malicious party's secrets). This implies that the malicious party could have generated these data itself, without the help of the other party, and cannot use it as a proof towards a third party.

Since we work in the random oracle model, there is a peculiarity due to the (non-)programmability of the hash function [Pas03]. Roughly, it is important that the distinguisher (receiving either the view of the malicious party or the simulated view) cannot distinguish these two random variables, even if it gets access to the same random oracle as the parties and the simulator. The distinguisher's access to the same hash function prevents the simulator from programming the hash values (as in the case for a real-world hash function).

We omit a formal definition of deniability (in the random oracle model) and refer to [Pas03]. We note that there are even stronger versions, called *online* deniability [DKSW09] where the distinguisher can communicate with the malicious party resp. the simulator while the protocol is executed. This notion, however, is much harder to achieve and not known to work here.

Deniability of Our Protocol. Our deniable version of the Schnorr schemes works as before, only that this time we hash $(Y'_A, \mathcal{G})$ instead of T_B. We call this protocol the *deniable Schnorr-based PACE|AA protocol*. Roughly, the idea is now that the chip itself determines the challenge! Hence, given that the challenge can be determined beforehand, and that it is created independently of the first signature step one can simulate the final signature part as in the interactive Schnorr identification protocol [Sch91]. We only need to take care that the other security properties are not violated through this.

Note that security as an AKE protocol follows as in the Schnorr signature based version (with the very same bounds), even for such challenges, as discussed after Definition 1. It suffices to show impersonation resistance —which follows similar to the case of signatures, using the fact that the chip in the PACE protocol already provides some form of authentication through the token T_A— and to show deniability. We note that our deniability simulator will actually need some assistance in form of a decisional Diffie-Hellman oracle (which, for sake of fairness, we then also give the adversary and the distinguisher). We comment that this does not trivialize the task as such a decision oracle is not known to help compute discrete logarithms, such that the simulator cannot simply derive the chip's secret key from the public key and use this key to show deniability.

We omit a formal treatment of these two properties but merely sketch how the deniability simulator $\mathcal{S}^{\mathcal{H}}$ works for this case. More insights can be found in the full version of this paper. The simulator only has access to the chip's public key X_A, the group data, and the password since it is considered a secret input to the terminal (but not the chip's secret key). The simulator now proceeds as follows, running a black-box simulation of the adversarial terminal (playing the honest

chip). In each execution, the simulator initially picks values $y_A, y'_A \leftarrow \mathbb{Z}_q$ and computes $Y'_A = g^{y'_A}$ as well as $c = \mathcal{H}(Y'_A, \mathcal{G})$ and $Y_A = X_A^{-c} g^{y_A}$. Note that both values are not computed according to the protocol description but still have the same distribution. In particular, even though the simulator may not be able to compute the shared Diffie-Hellman key $\mathcal{K}$ in the execution, it can later complete the signature generation by setting $s = y_A$ (such that $g^s = Y_A X^{\mathcal{H}(Y'_A, \mathcal{G})}$). For the other steps the simulator proceeds as the chip would, using knowledge of the password. Only when the simulator receives T_B from the malicious token, it searches (with the decisional Diffie-Hellman oracle) in the list of hash queries of the malicious terminal for queries about a key $\mathrm{DH}(Y'_A, Y'_B)$. If no key is found, then abort this execution (this means that the adversary must have forged a MAC for an unknown key); else use the found key $\mathcal{K}$ to finish the execution (using the signature tokens as computed above). If the adversary stops, then let the simulator output the same value.

Acknowledgments. We thank the anonymous reviewers of FC'12 for helpful comments. This work was supported by CASED (`http://www.cased.de`).

References

[AFP05] Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-Based Authenticated Key Exchange in the Three-Party Setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)

[BCI+10] Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient Indifferentiable Hashing into Ordinary Elliptic Curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010)

[BFK09] Bender, J., Fischlin, M., Kügler, D.: Security Analysis of the PACE Key-Agreement Protocol. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 33–48. Springer, Heidelberg (2009)

[BLS01] Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)

[BPR00] Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)

[BPSV08a] Blundo, C., Persiano, G., Sadeghi, A.-R., Visconti, I.: Improved Security Notions and Protocols for Non-transferable Identification. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 364–378. Springer, Heidelberg (2008)

[BPSV08b] Blundo, C., Persiano, G., Sadeghi, A.-R., Visconti, I.: Resettable and non-transferable chip authentication for e-passports. In: RFIDSec 2008 (2008)

[BSI10] Advanced security mechanism for machine readable travel documents extended access control (eac). Technical Report (BSI-TR-03110) Version 2.05 Release Candidate, Bundesamt fuer Sicherheit in der Informationstechnik, BSI (2010)

[DDN00] Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)

[DF10] Dagdelen, Ö., Fischlin, M.: Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 54–68. Springer, Heidelberg (2011)

[DKSW09] Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and On-Line Deniability of Authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (2009)

[ICA06] Machine readable travel documents. Technical Report Doc 9303, Part 1 Machine Readable Passports, 6th edn., International Civil Aviation Organization, ICAO (2006)

[Ica09] Icart, T.: How to Hash into Elliptic Curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009)

[ICA10] ICAO. Supplemental access control for machine readable travel documents (2010), `http://www2.icao.int/en/MRTD/Pages/Downloads.aspx`

[Kra95] Kravitz, D.W.: Digital signature algorithm. Computer Engineering 44(5), 6–17 (1995)

[MVV07] Monnerat, J., Vaudenay, S., Vuagnoux, M.: About machine-readable travel documents – privacy enhancement using (weakly) non-transferable data authentication. In: RFIDSEC 2007 (2007)

[Pas03] Pass, R.: On Deniability in the Common Reference String and Random Oracle Model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)

[PS00] Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (2000)

[Sch90] Schnorr, C.-P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)

[Sch91] Schnorr, C.-P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)