Derandomized PACE with Mutual Authentication

Adam Bobowski(✉) and Mirosław Kutyłowski

Department of Computer Science, Wrocław University of Science and Technology,
Wrocław, Poland
adam.bobowski@pwr.edu.pl, miroslaw.kutylowski@pwr.wroc.pl

Abstract. We present a derandomized version of the ICAO protocol PACE – a PAKE protocol (password authenticated key exchange) used for identity documents including biometric passports and future European personal ID documents. The modification aims to remove necessity of implementing random number generator and thereby reduce the cost of the chip and its certification, while maintaining the level of security. As a side effect we achieve better verifiability properties as well as chip and terminal authentication.

Keywords: PAKE · PACE · Chip and terminal authentication · Derandomization

1 Introduction

Electronic identification is one of fundamental tasks when developing ubiquitous systems. For the sake of personal identity documents a number of cryptographic protocols has been developed by the ICAO organization [2]. They have been deployed on biometric passports around the world and on some national ID cards. A recent decision European Union authorities is to follow the ICAO specification on all national ID cards issued by the member states.

One of the core schemes in the ICAO specification is PACE developed for German identity documents by the German IT security authority BSI [1]. PACE enables to establish a session (and its session keys) provided that both the chip and the reader are using the same password. The goal of PACE is to prevent activation of the chip without the consent of it holder. – in case of wireless communication it might be particularly easy (skimming ID documents). PACE is resistant to offline and online attacks aiming to derive the password used – the only attack vector is to guess the password at random and check if the protocol execution terminates successfully.

Current widespread adoption of PACE for the classical identity documents suggests to adapt and reuse it in other areas, such as the IoT.

This research has been initiated in project OPUS 2014/15/B/ST6/02837 funded by Polish National Science Centre, later supported by grant 049U/0044/19 at Wrocław University of Science and Technology.

J. K. Liu and X. Huang (Eds.): NSS 2019, LNCS 11928, pp. 697–705, 2019.
https://doi.org/10.1007/978-3-030-36938-5_44

Moreover, due to its specific and proven features regarding (un)traceability, PACE is particularly attractive when high standards of personal data protection apply.

The PACE protocol (and the proposed extension) are presented on Fig. 1.

Implementation Issues. Selection of random elements, like in most cryptographic schemes, plays a crucial role in security of standard PACE protocol. Convenient assumption of having access to source of truly random elements solves many issues. On the other hand, in real world this theoretical assumption is almost practically unobtainable.

The problems might be diverse. First, a physical source of randomness might be affected by aging problems and start to provide biased and predictable output – in this case the whole security argumentation collapses. Second, if we apply solutions based on PRNG, then it is hard to guarantee that nobody knows the seed. Finally, verifying that a device is following the protocol and there is neither trapdoor utilizing randomness nor implementation errors is difficult. This increases substantially the costs of certification process and well as reduces the trust level.

Our Goals. Let us summarize our design approach:

derandomization: While there are cryptographic methods for constructing verifiable randomness, any solution to be deployed in practice must be extremely simple and effective. We follow a different strategy – we eliminate random elements and replace them by unpredictable elements.
backwards compatibility: The devices running our protocol should interact smoothly with the devices running the regular PACE (or PACE CAM).
indistinguishability: It should be infeasible for an observer to distinguish which version of the protocol is run. Therefore, many security and privacy properties of the regular PACE should be inherited.
deniability: Moreover, even a party executing the protocol should be unable to convince a third party that a presented protocol transcript has not been forged.
authentication: The protocol should enable strong authentication of communicating parties.
reuse: We wish to reuse the old designs and therefore the number of changes in the original protocol should be small. This reduces the effort in adjusting the old software to the new protocols.
password as a context: We concern the password not only as a guard against an illegitimate device activation. It should provide a context – only the devices with the same context should establish a session.

2 Deterministic PACE with Mutual Authentication

The protocol consists of 4 phases depicted on Fig. 1.

device A (eID chip)		device B (reader)
holds the keys:		holds the keys:
$\boxed{sk_A \text{ (secret)}, pk_A = g^{sk_A} \text{ (public)}}$		$\boxed{sk_B \text{ (secret)}, pk_B = g^{sk_B} \text{ (public)}}$
password π		password π (input)
	initialization	
$\boxed{\text{adjust the password } \pi \text{ to the context}}$		$\boxed{\text{adjust the password } \pi \text{ to the context}}$
$\boxed{\text{determine the seed } \omega_A}$		$\boxed{\text{determine the seed } \omega_B}$
	PACE initial phase	
$K_\pi := H(\pi\|\|0)$		$K_\pi := H(\pi\|\|0)$
~~choose s at random~~		
$\boxed{s := H_q(\omega_A\|\|4)}$		
$z := \mathrm{Enc}(K_\pi, s)$	$\xrightarrow{\mathcal{G}, z}$	abort if $\mathcal{G}$ incorrect
		$s := \mathrm{Dec}(K_\pi, z)$
	DH2Point Start	
~~choose x_A at random~~		~~choose x_B at random~~
$\boxed{x_A := H_q(\omega_A\|\|5)}$		$\boxed{x_B := H_q(\omega_B\|\|6)}$
$X_A := g^{x_A}$	$\xleftarrow{X_B}$	$X_B := g^{x_B}$
	$\xrightarrow{X_A}$	
$h := X_B^{x_A}$		$h := X_A^{x_B}$
abort if $h = 1$		abort if $h = 1$
$\hat{g} := h \cdot g^s$		$\hat{g} := h \cdot g^s$
	DH2Point End	
~~choose y_A at random~~		~~choose y_B at random~~
$\boxed{y_A := H_q(\omega_A\|\|7)}$		$\boxed{y_B := H_q(\omega_B\|\|8)}$
$Y_A := \hat{g}^{y_A}$	$\xleftarrow{Y_B}$	$Y_B := \hat{g}^{y_B}$
	$\xrightarrow{Y_A}$	
$K := {Y_B}^{y_A}$		$K := {Y_A}^{y_B}$
$K_{\mathrm{Enc}} := H(K\|\|1)$		$K_{\mathrm{Enc}} := H(K\|\|1)$
$K_{\mathrm{MAC}} := H(K\|\|2)$		$K_{\mathrm{MAC}} := H(K\|\|2)$
$K'_{\mathrm{MAC}} := H(K\|\|3)$		$K'_{\mathrm{MAC}} := H(K\|\|3)$
$\boxed{K'_{\mathrm{Enc}} := H(K\|\|4)}$		$\boxed{K'_{\mathrm{Enc}} := H(K\|\|4)}$
$T_A := \mathrm{MAC}(K'_{\mathrm{MAC}}, (Y_B, \mathcal{G}, \hat{g}))$		$T_B := \mathrm{MAC}(K'_{\mathrm{MAC}}, (Y_A, \mathcal{G}, \hat{g}))$
	$\xleftarrow{T_B}$	
abort if T_B incorrect	$\xrightarrow{T_A}$	abort if T_A incorrect
	$\boxed{\text{Authentication procedure}}$	
$\boxed{\text{authenticating the parties based on the the public keys } pk_A, pk_B \text{ and the protocol transcript}}$		

Fig. 1. A deterministic version of the PACE protocol with mutual authentication. Changes to the original PACE are marked in gray boxes (with the original version striked out.)

Initialization Phase. In this phase the participants adjust the shared password and derive the seeds. For this purpose they use the *context* Δ (see Sect. 2.1) and the secret keys of the participants. We propose two versions of the initialization procedure: an non-anonymous version where the public keys pk_A and pk_B are mutually known, and an anonymous version where each participant uses only own keys. The details of the proposal are given on Figs. 2 and 3.

We intend to simplify as much as possible the first case. The price for this is that the scheme is not resilient against an adversary that holds the private key of, say, device B.

device A	device B
holds the keys: sk_A (secret) $pk_A = g^{sk_A}$ (public), π' (password)	holds the keys: sk_B (secret) $pk_B = g^{sk_B}$ (public), π' (password)
A and B exchange their public keys pk_A, pk_B	
$\theta := pk_B^{sk_A}$, acquire context Δ $\pi := H(\theta \| \Delta \| 1 \| \pi')$ $\omega_A := H_G(\theta \| \Delta \| 2)$	$\theta := pk_A^{sk_B}$, acquire context Δ $\pi := H(\theta \| \Delta \| 1 \| \pi')$ $\omega_B := H_G(\theta \| \Delta \| 2)$

Fig. 2. Non-anonymous initialization - the devices must exchange the public keys

device A	device B
holds the keys: sk_A (secret) $pk_A = g^{sk_A}$ (public), π' (password)	holds the keys: sk_B (secret) $pk_B = g^{sk_B}$ (public), π' (password)
acquire context Δ $\pi := H(\Delta \| 1 \| \pi')$ $\zeta := H_G(\Delta \| 2)$, $\omega_A := \zeta^{sk_A}$	acquire context Δ $\pi := H(\Delta \| 1 \| \pi')$ $\zeta := H_G(\Delta \| 2)$, $\omega_B := \zeta^{sk_B}$

Fig. 3. Anonymous initialization: the devices do not show their public keys

Authentication Phase. This phase is executed immediately after PACE (see Fig. 4). It consists of a common part that does not depend on the initialization and the initialization dependent parts. In the case of the non-anonymous case no extra action is required (as the equality $\omega_A = \omega_B$ is checked implicitly). Figure 5 contains details corresponding to the anonymous initialization. Note that the common part may contain also verification of the public keys pk_B and pk_A, e.g. based on certificates.

In case of anonymous initialization it must be checked that $\omega_A = \zeta^{sk_A}$ and $\omega_B = \zeta^{sk_B}$. In fact, the authentication proof is based on the KEA1 assumption: as the discrete logarithm of ζ is unknown, nobody but the holder of sk_A can compute ζ^{sk_A} (resp., only the holder of sk_B can compute ζ^{sk_B}). So we deal with

device A	device B
retrieval and verification of seeds	
procedure dependent on initialization	
verification of the execution transcript	
abort if $X_B \neq g^{H_q(\omega_B\|\|6)}$	abort if $X_A \neq g^{H_q(\omega_A\|\|5)}$
abort if $Y_B \neq \hat{g}^{H_q(\omega_B\|\|8)}$	abort if $Y_A \neq \hat{g}^{H_q(\omega_A\|\|7)}$
	abort if $s \neq H_q(\omega_A\|\|4)$
verify pk_B	verify pk_A
authentication of seeds	
procedure dependant on initialization	

Fig. 4. Common part of authentication.

the problem of equality of discrete logarithms for the tuples $(g, pk_A, \zeta, \omega_A)$ and $(g, pk_B, \zeta, \omega_B)$. A solution based on Schnorr signatures may be the simplest, but it requires random numbers and leaves an undeniable proof of interaction.

In Fig. 5 we instantiate the authentication process with a textbook procedure.

Note that it starts with a choice of random elements. Hence it may appear that we need a source of strong randomness - just violating our assumptions. However, here we only need that these numbers are to some degree unpredictable for the other side of the protocol. Unlike in the case of Schnorr signatures, it does not endanger the secret keys – it only makes the proof of equality of discrete logarithms weaker.

2.1 Comments on Design Approach

Complexity. In practice, for devices like smart cards or sensors a small storage capacity is one of the key limitations. For this reason reusing the same code or adding just a few lines of code might be crucial for implementability of a scheme. For similar reasons, the communication volume and the number of messages exchanged should be minimized.

Modularity. Depending on the application case, there might be different requirements for the strength of authentication. Sometimes the pure password authentication is enough and any kind of strong authentication of a protocol participant would violate the *data minimality* principle. So we follow the approach where authentication comes as a plug-and-play component that can be injected into the original code. Our scheme can be modified easily and enable to authenticate either both communicating parties or only one of them or none of them. Note that in the ICAO specification contains the PACE CAM protocol with strong authentication of the ID document. However, it needs a secure PRNG - otherwise the secret key may leak.

Context. The parameter $\varDelta$ from the protocol description may have different origins and play different roles. For example, in case of VANET, sometimes only

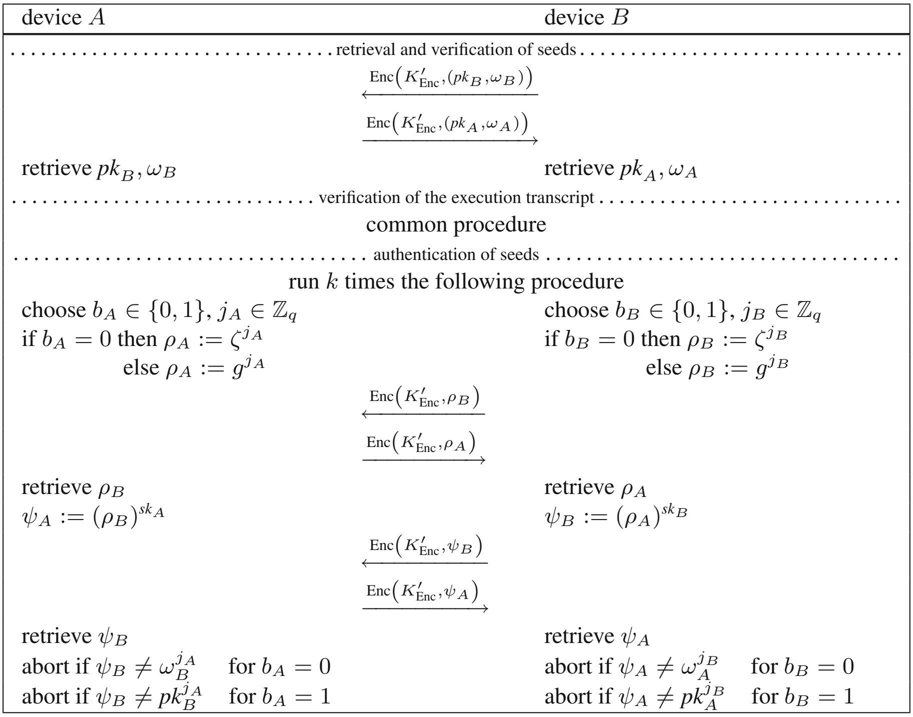
device A
device B
retrieval and verification of seeds
$\mathrm{Enc}\big(K'_{\mathrm{Enc}},(pk_B,\omega_B)\big)$
$\mathrm{Enc}\big(K'_{\mathrm{Enc}},(pk_A,\omega_A)\big)$
retrieve pk_B, ω_B
retrieve pk_A, ω_A
verification of the execution transcript
common procedure
authentication of seeds
run k times the following procedure
choose $b_A \in \{0,1\}$, $j_A \in \mathbb{Z}_q$
if $b_A = 0$ then $\rho_A := \zeta^{j_A}$
else $\rho_A := g^{j_A}$
choose $b_B \in \{0,1\}$, $j_B \in \mathbb{Z}_q$
if $b_B = 0$ then $\rho_B := \zeta^{j_B}$
else $\rho_B := g^{j_B}$
$\mathrm{Enc}\big(K'_{\mathrm{Enc}},\rho_B\big)$
$\mathrm{Enc}\big(K'_{\mathrm{Enc}},\rho_A\big)$
retrieve ρ_B
$\psi_A := (\rho_B)^{sk_A}$
retrieve ρ_A
$\psi_B := (\rho_A)^{sk_B}$
$\mathrm{Enc}\big(K'_{\mathrm{Enc}},\psi_B\big)$
$\mathrm{Enc}\big(K'_{\mathrm{Enc}},\psi_A\big)$
retrieve ψ_B
abort if $\psi_B \neq \omega_B^{j_A}$ for $b_A = 0$
abort if $\psi_B \neq pk_B^{j_A}$ for $b_A = 1$
retrieve ψ_A
abort if $\psi_A \neq \omega_A^{j_B}$ for $b_B = 0$
abort if $\psi_A \neq pk_A^{j_B}$ for $b_B = 1$

Fig. 5. Details of authentication in case of anonymous initialization.

the vehicles in a close proximity should establish communication. In order to enforce this, Δ may contain an (approximate) GPS location or a string broadcasted locally by a Road Side Unit. Another option is an optical channel. Δ may be encoded in a QR code exposed to the communicating devices (like in case of the WeChat and Alipay systems). Finally, Δ might follow from the past events – e.g. contain a nonce from the previous interaction.

3 Security Discussion

A detailed security proof even for the standard PACE turns out to be quite long and tedious, if we take into account active adversaries and privacy issues (see [4] for a draft!). Therefore in this paper we provide only some comments on the crucial issues.

Privacy Features of PACE. Privacy protection was apparently one of the main goals for the designers of PACE. However, it is hard to express exactly what privacy protection means and to present the relevant models. Fortunately, one can allude the problem by determining situations where there are no concerns about privacy protection.

Similarly to the Abdalla model and the left-or-right games we can attempt to show that given access to a protocol execution an adversary cannot distinguish whether it has been executed with the keys attributed to a given participant or situation (when the password is derived from e.g. location), or with ad hoc keys selected at random. In this situation any kind of conclusion regarding the device's identity is impossible.

For PACE one can show that even if an adversary knows a password, then he cannot distinguish between a protocol execution with this password and a protocol execution with a random password - the only exception is when the adversary executes the protocol himself [4]. This property follows from the KEA1 assumption: deriving the shared key in the Diffie-Hellmann protocol by the adversary indicates that they knows at least one of the exponents used. However, this might be an Achilles heel of PACE for weak devices: if a source of randomness is corrupted on one side, then the argument fails.

Full Key Compromise. Now let us consider our schemes. First note that if an adversary knows the key sk_A or sk_B and Δ, then he can easily check whether a given communication transcript corresponds to these keys and derive the session keys. Simply, the adversary mimics the deterministic computation of A or B. If Δ is unknown but has low entropy, then the same attack applies – it suffices to recompute X_A (resp., X_B) and compare with the value transmitted.

The attack described above cannot be prevented as long as the computation is deterministic. Unfortunately, for the regular PACE the situation need not to be better as the protocol might be based on a PRNG with a secret seed. The seed might be compromised or replaced after subverting the device. Detecting or preventing this attack requires nontrivial countermeasures (such as watchdogs).

Reduction Arguments. Security of our schemes is based on the fact that an adversary cannot distinguish between executions of our protocols and the cases where the key steps are randomized again to meet the original specification. So we can argue that certain properties are the same as for the original PACE.

From now on we assume that the adversary knows neither sk_A nor sk_B. However, we admit that the adversary may guess Δ.

Non-anonymous Initialization Case. The first observation is that due to difficulty of the DDH Problem, the adversary should not see any difference in the protocol execution when we replace θ by a random element. Then to proceed we need the Correlated Input Hash Assumption (CIHA) [4]:

One cannot distinguish between the tuples $H(C_1(r)), \ldots, H(C_n(r))$ and the random tuples of the same length and from the same domain provided that the circuits $C_1, \ldots, C_n$ are in some sense independent and min-entropy of their output is sufficiently large.

It is impossible to use directly CIHA for π and ω_A, since different hash functions H and H_G are used. However, one can define H_G as a composition: $H_G(x) = F(H(x))$. Thus, we can assume that the adversary would not see any difference if we replace π and ω_A by random values. Then in the same way we may replace the derivation for s, x_A, x_B, y_A, y_B by the random choice.

As the initialization and authentication phases are executed without any interaction, this finalizes the argument in this case.

Anonymous Initialization Case. In this case the adversary can derive π and ζ, as he knows Δ. However, due to the DDH Assumption, the adversary cannot distinguish the original computation from the computation where ω_A and ω_B are chosen independently at random. The next step is to replace the derivation for s, x_A, x_B, y_A, y_B by the random choice and argue that due to CIHA the adversary cannot see any difference.

After the changes, the main part of the protocol is exactly the original PACE (apart from computing additional key K'_{Enc}). However, there is still the authentication part presented on Fig. 5. Now we can replace the computation of ψ_A and ψ_B by random values and skip the tests from the last two lines in Fig. 5. (Effectively, we can simply replace the last four ciphertexts with random values.) Again, the adversary cannot detect the manipulation, even when given an oracle for decryption of the last 4 messages. The only remaining part dependent on the participants' keys are the ciphertexts exchanged in the first part of the authentication phase. These key are unrelated now to the rest of the computation, so we are now in the situation of the original PACE, where the session key is used for encrypting some data. One can provide an argument based on the security of PACE that the plaintexts can be replaced by random elements. (Note that the key K'_{Enc} has been introduced only for the sake of a formal proof at this point.)

Active Adversary. In general, an active adversary acting as man-in-the-middle is the most interesting and complicated case for any protocol. The difficulty comes from the fact that the number of possible attack scenarios is enormous. However, in our case we can follow [4] showing protocol's fragility: it means that if an adversary manipulates any message, then the protocol execution will change the behavior of the protocol parties always in the same way. We postpone the details to the full version of the paper.

Deniability. A dishonest protocol participant, say A, might create a proof of interaction with B, and offer it to third parties. In case of PACE this is doable due to the extensive use of randomness enabling malicious cryptography. Derandomization applied to our protocols helps here a lot: in case of the non-anonymous initialization the responses of B can be created by A himself. In case of anonymous initialization the issue is slightly more complicated: A cannot create ω_B, but on the other hand if the proof is created by A before presenting the data for sale, then A can present any value as ω_B together with a fake proof from the authentication phase.

Authentication. In the non-anonymous case party A can compute the values x_B and y_B to be used by B. On the other hand, x_A and x_B are computed as hash values, so A may assume that B knows the arguments used to compute these hash values (this feature is captured by the Extractable Hash Function property – cf. [3]). Hence B must know ω_B and then in turn must know θ. However, by the KEA1 Assumption, if B can compute θ given the keys pk_A and pk_B, then B knows either sk_A or sk_B. If A is sure about secrecy of its key sk_A, then the other party is either B (holding sk_B) or a party that has got either sk_B or θ from B. The same argument applies for authentication of B against A.

The price to be paid for protocol simplicity is twofold. First, A can be cheated if sk_A is compromised. Second, B can delegate its ability to talk with A by presenting θ to a third party. If this is unacceptable for an application scenario, then anonymous initialization should be used.

In the case of anonymous initialization the situation is less complicated. A accepts B, if B presents ω_B which, due to the interactive proof in the last phase, equals to ζ^{sk_B} with a high probability. As the discrete logarithm of ζ is unknown (recall that ζ has been computed as a hash value), by KEA1 we can conclude that B must have known sk_B. The same argument concerns authenticating A against B.

References

1. BSI: Technical guideline TR-03110 v2.21 - advanced security mechanisms for machine readable travel documents and eIDAS token (2016)
2. ICAO: machine readable travel documents - part 11: security mechanism for MRTDs. Doc 9303 (2015)
3. Kiayias, A., Liu, F., Tselekounis, Y.: Practical non-malleable codes from l-more extractable hash functions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of ACM CCS 2016, pp. 1317–1328. ACM (2016)
4. Kubiak, P., Kutyłowski, M.: Privacy and security analysis of PACE GM protocol. In: TrustCom/BigData 2019 Proceedings, pp. 763–768 (2019)