Security of Okamoto Identification Scheme - a Defense against Ephemeral Key Leakage and Setup

[Extended Abstract]

Łukasz Krzywiecki
Faculty of Fundamental Problems of Technology
Wrocław University of Science and Technology
Wybrzeże Wyspiańskiego 27
50-370 Wrocław, Poland
lukasz.krzywiecki@pwr.edu.pl

Mirosław Kutyłowski
Faculty of Fundamental Problems of Technology
Wrocław University of Science and Technology
Wybrzeże Wyspiańskiego 27
50-370 Wrocław, Poland
miroslaw.kutylowski@pwr.edu.pl

ABSTRACT

We consider the situation, where an adversary may learn the ephemeral values used by the prover within an identification protocol, aiming to get the secret keys of the user, or just to impersonate the prover subsequently. Unfortunately, most classical cryptographic identification protocols are exposed to such attacks, which might be quite realistic in case of software implementations. According to a recent proposal from SECIT'2017, we regard a scheme to be secure, if a malicious verifier, allowed to set the prover's ephemerals in the *query stage*, cannot impersonate the prover later on.

We focus on the Okamoto *Identification Scheme* (IS), and show how to make it immune to the threats described above. Via reduction to the GDH Problem, we provide security guarantees in case of insufficient control over the unit executing Okamoto identification protocol (the standard Okamoto protocol is insecure in this situation).

Keywords

identification; Okamoto scheme; ephemeral random values; leakage; impersonation; provable security; simulatability; security reduction; GDH

1. INTRODUCTION

Proving one's identity is the very first functionality triggered when entering many cyber systems. It is also an indispensable part of more complex protocols, e.g. secure Authenticated Key Exchange (AKE). Due to increasing proliferation and globalization of cyber systems, the number of interacting parties rapidly increases. At the same time the average prior trust level between the partners seem to decline. Among others, this is due to the fact that more and more frequently the interactions are not limited to local closed computing environments, but take place in systems such as remote cloud platforms.

From the practical point of view the situation gets even more complicated due to heterogeneity of the systems. On one hand, especially in the world of IOT, we might have to deal with cheap micro-controllers, delivered as black box devices, with limited tamper resistance, low level security certificates (if any), no security test features, etc. On the other hand, the protocols may be run in a virtual environment created by powerful servers, within a complex internal system with multiple users running their processes concurrently. Despite powerful security mechanisms, we cannot exclude leakages of sensitive data between processes due to, e.g., cache attacks. Moreover, providing relevant security guarantees via certification process, e.g. within the Common Criteria framework, would imply huge costs due for instance to the documentation size.

SCC'17, April 02 2017, Abu Dhabi, United Arab Emirates
© 2017 ACM. ISBN 978-1-4503-4970-3/17/04. . . $15.00
DOI: http://dx.doi.org/10.1145/3055259.3055267

Identification Protocols - General Outline.

In general, an *Identification Scheme* (IS), allows one party - called a *prover* (or *Alice*) - to prove its identity against the other party - called a *verifier* (or *Bob*). Typically, this functionality takes advantage from an existing Public Key Infrastructure (PKI), and the party holding a secret key proves its knowledge in front of the verifier holding the corresponding public key and its certificate. There are several general approaches to construct an IS. Protocols executed between the prover and the verifier can have two, three or more rounds. The prover and the verifier can use, apart from long term secret and public keys, some temporary values, called *ephemeral keys*, whose secrecy might have critical impact on security of the whole construction.

A typical protocol consists of three phases: In the first phase the prover chooses an ephemeral secret at random and sends its *commitment* to the verifier. In the second phase the verifier replies with an unpredictable random *challenge*. In the last phase the prover sends a *response* to the verifier. Security of such schemes is usually based on *zero-knowledge* arguments: on one hand it should be infeasible to respond to challenges in a correct way without knowledge of the secret key, while on the other hand, a verifier should not gain any advantage in learning the secret key, compared to the situation when he does not interact with the prover and holds only the public key of the prover.

In some cases the IS schemes are required to possess some additional features, like *deniability*. It means that the protocol can be simulated without the knowledge of the secret key - thus making a protocol transcript useless as a proof of interaction with the secret key owner, when presented to a third party.

Protocol security in practice.

Arguments about security of identification protocols are usually based on abstract and somewhat idealistic models. This is quite evident in case when we confine ourselves to the standard "Alice and Bob" framework. Unfortunately, for practical IS implementations, the situation is far more complicated. The user does not perform the identification protocol himself, but instead uses a computing device that interacts with the verifier on behalf of its owner. In particular, the device holds the private keys attributed to the owner. The same situation may occur on the side of the verifier: while he might be honest (and for instance wishes to generate the challenges at random), his device might be in fact behaving maliciously, aiming to attack the prover.

In principle, the prover should fully control the device executing the protocol on behalf of himself. In reality, the situation is more complicated and most of the power may remain in hands of the manufacturers providing software and hardware used, as well as in hands of the system administrator. Therefore, implementation security matters a lot and it becomes a critical issue to what extent one can examine whether the protocol runs exactly as declared. Unfortunately, in a standard case the possibilities to inspect a unit executing cryptographic protocols (even by the owner) is very limited. This follows from the sheer reason that the inspection itself might be malicious aiming to retrieve the private key stored by the inspected unit.

Unfortunately, there are many opportunities for a practical attack, even for the best and *provably secure* cryptographic protocols. According to the famous Third Law of Computer Security by Adi Shamir *Cryptography is typically bypassed, not penetrated.* Unfortunately, not only the attacks can be mounted, but at the same time they may be well hidden so that the users remain totally unaware about the situation. For instance, one of the key features of many schemes, based on the discrete logarithm problem, is use of ephemeral values that are generated independently at random for each protocol execution. The protocol descriptions, as well as the formal security proofs, typically assume that the source of randomness is ideal and in particular unpredictable for the verifier. Does it match reality? And what are the consequences, if the randomness turns out to be poor? The answer for the second question is pessimistic: the security of the scheme may collapse entirely. In particular, this is the case for ElGamal, Schnorr and DSA signatures, where knowing the random value used to generate the signature and the signature itself enables the attacker to derive immediately the secret key of the user.

Of course, majority of implementations involve some kind of self-control over the random number generator used. For this purpose randomness tests (such as NIST tests [16]) are frequently used. However, we need to be aware that such tests detect only rough errors due to, for instance, substantial implementation errors or aging effect of cryptographic hardware. They do not detect the subversion cases, where the (pseudo)random number generator is designed in a clever way to remain undetected. In fact, apart from sophisticated attacks such as hardware Trojans [3], there are trivial attacks based on pseudorandom number generators (PRNG), where it suffices to know the seed in order to learn all (pseudo)random values used. Sometimes even a subtle subliminal adversarial interference, such as the reset of the internal state and/or randomization source of the prover's device, can have influence on the produced values.

So in practice, it seems that we are already defenseless against certain kinds of attacks and we need not to wait until *post-quantum era* to experience lack of firm cryptographic protection. Should we therefore give up?

There are two general strategies to respond to this situation: the first one is to create a framework where there is a *watchdog* checking whether the device is behaving as declared – in particular whether the software and hardware has not been subverted by the adversary. However, one of the problems with a watchdog approach is the old question *who will guard the guards?* Moreover, constructing a watchdog that inspects a complex system, and not an isolated unit, seems to be very hard. The second strategy is to rebuild the protocol in order to eliminate or significantly reduce the effects of implementation weaknesses.

For many IS protocols, the whole execution of the prover is deterministic apart from choosing ephemeral secret values. Therefore our goal might be to rebuild the protocol so that the advantage of the adversary is still negligible even if the ephemeral values on both sides are compromised: the adversary should not be able to impersonate the prover in subsequent protocol executions, even if he managed to learn, modify or even fully determine the ephemeral values used by the prover's device.

Goal of the Paper.

The goal of this paper is to show that protection mechanisms may be built on the cryptographic level into the Okamoto identification scheme [17]. The Okamoto IS, based on the hardness of DLP, is regarded as an important building block for more complex designs, e.g. [8]. Notably, a signature scheme based directly on the Okamoto identification scheme [6] has been included into a suite of protocols for personal identification documents (the eIDAS token) by the Federal Office for Information Security (BSI) in Germany. Unfortunately, the Okamoto IS does not withstand the ephemeral leakage attack, just like the previous Schnorr IS and signature schemes. Therefore, for our Okamoto modification, we aim to make - from the design - the secrets to be more protected than in the traditional case, even if the device used for identification contains subverted random number generator.

In this paper we continue the work done in [14], where the similar solution was proposed for the Schnorr IS. We use a security model for the ephemeral secrets from [14], as well as we use a similar notation.

Paper contribution.

The contribution of the paper is the following:

- We refine the security model from [14]. Particularly we analyze the architecture of the computational platform of the prover, which consist of two parts: a security module of minimal functionality (storing the secret key, exponentiation with the secret key), and the less protected part that provides all the rest computational functionality required by the proving algorithm. Such a separation enables outsourcing a part of the computation to the third party services.

- We propose a modification of the Okamoto authentication protocol [17], which makes it immune to malicious setting of the ephemeral values for the adversary's advantage.

- We prove the security of the modified Okamoto IS according to the model of [14] with our refinements, by reduction to the GDH problem.

The paper is organized in the following way. In Sect. 2 we recall the Okamoto identification protocol. In Sect. 3 we recall the security model from [14]. In Sect. 4 we propose a modified version of Okamoto IS, and prove its security.

Previous Work.

The general problem of security threats regarding cryptographic products has recently drawn more and more attention. On the side of the industrial design, there have been a lot of efforts to design devices that are tamper resistant and resistant to side channel analysis. Most of this development is regarded as an industrial secret and protected by non-disclosure agreements. The similar situation concerns access to information about the attack methods.

On the side of academic research there have been efforts to model the leakage from cryptographic devices. In the *bounded retrieval model* we assume that the number of information bits leaked by a cryptographic device is limited (see e.g. [1]). Among others, this has to model the capabilities of side channel attacks that, presumably, can betray only a partial information of the internal state of the device. Unfortunately, this does not apply to the situation, where we are forced to implement cryptographic protocol in software, with no strict separation offered by dedicated devices.

Even in case of hardware implementation, the bounded retrieval does not cover the case of complete compromise of a random number generator used to create ephemeral random elements. Unfortunately, such a situation cannot be excluded for twofold reasons: first, as already mentioned, a trapdoor can be hidden in the hardware in an undetectable way ([3]). Second, there are sophisticated ways to build a random number generator that yields an output indistinguishable from a random one, and yet containing a trapdoor for a malicious party holding a certain secret key (see e.g. [20]).

Recently, the concept of *subversion resilience* for digital signatures has been expressed explicitly [2]. It has been assumed that an adversary can replace the original algorithm by a new one that behaves in the same way from the point of view of a user, but creates a trapdoor for an attacker. The paper [18] extends the scope of *subversion resilience* to the process of generating signing keys. A kind of general solution for this problem for certain protocols using PRNG has been announced in [15].

The main difference, when comparing our approach to the above *subversion* model, is that we assume that the prover's computation platform is divided into two subcomponents: a secure unit that stores and operates on the secret key, and the remaining insecure part of the system (hardware, memory, software) exposed to setup and subliminal attacks. The adversary can replace the code/algorithms of the insecure part, but cannot access the secure unit otherwise but through predefined interfaces. In this context the leakage of bits of the secret key from *subversion* model, cannot be simply implemented, as the replaced code cannot just refer to the secret key variable located in the secure unit. From our point of view, the attacks presented in [2] can be regarded as mounting a subliminal channel for the secret key leakage by the malicious program code which has access to the secret key storage. The adversary just hides the subsequent bits of the retrieved secret in the randomized part of the result produced by the attacked scheme (e.g. randomized parameters of the signature). In our model we just assume that undetectable subliminal channels could exist for everything but the plain secret key from the secure unit. The particular example of such a subliminal channel we analyze, is the possibility to set/get values of ephemeral keys from the insecure part. In this model one can see major difference between the regular Okamoto (which allows compromising the long term secrets from protocol messages once the ephemerals are leaked) and our modified version (where long term keys are secure regardless of ephemeral key leakage).

Issues regarding particular cryptographic protocols and attacks against them have been considered in a number of papers. The problem of security of identification schemes under reset attacks on ephemeral secrets was raised in [7] in the context of zero-knowledge proofs. Later, the paper [4] presented constructions for making the identification protocols immune against *reset attacks*: the reset-secure identification protocols based on a deterministic, stateless digital signature scheme, the reset-secure identification protocols based on a CCA secure asymmetric encryption scheme, the reset-secure identification schemes based on pseudorandom functions and trapdoor commitments. Each of them has a certain drawback compared to the Okamoto identification scheme: the first one leaves an undeniable proof of interaction, the second one is not directly compatible with the Diffie-Hellman key exchange protocols initiated with the prover's ephemeral public key, the third one requires more than 3 rounds.

Historically, in the PKI setup we have the following fundamental IS, which suffer from the same problem - compromised ephemeral values lead to derivation of the verifier's secret key: 1) RSA based [10, 9, 11]; 2) *three-round* DLP based schemes of Schnorr [19] and Okamoto [17]. Recently in [14] the solution for the above stated problem for the Schnorr IS was proposed. Our goal is to do the same for the Okamoto IS.

Note that the problem with ephemerals in identification schemes, inflicts by analogy the signatures, constructed when Fiat-Shamir transformation is used. Unfortunately, the same applies to the so-called Pseudonymous Signature designed for electronic personal identity documents by the German Federal Office for Information Security [6].

Finally, one has to draw attention to the fact that a lot depends on the implementation details, even if the protocol itself has been fixed. An example of this situation is CAM protocol from ICAO standard [13]. It turns out that the protocol can be either implemented in the way that the ephemeral values reveal a secret key [5], or they bring no advantage to the adversary [12].

2. OKAMOTO IDENTIFICATION SCHEME

2.1 Preliminaries and Notation

We loosely follow the notation from [1]: $x_1, \ldots, x_n \leftarrow_R X$ means that each x_i is sampled independently and uniformly at random from the set X. We shall use a group generation algorithm $\mathcal{G}$ such that if $\mathcal{G}(1^\lambda) = (q, g, G)$, then G is a group of a prime order q and a generator g. We assume the following:

Existence of a *Bilinear Map*: We assume that one can find a bilinear map $\hat{e} : G \times G \to G_T$ into a group G_T of order q. That is, the following conditions hold:
1) *bilinearity*: $\forall a, b \in \mathbb{Z}_q : \hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$,
2) *non-degeneracy*: $\hat{e}(g, g) \neq 1$,
3) *computability*: $\hat{e}$ is efficiently computable.

The *discrete logarithm* (DL) Assumption: For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}_{\mathsf{DL}}$ it holds that:
$\Pr[\mathcal{A}_{\mathsf{DL}}(\mathbb{G}, g^x) = x \mid \mathbb{G} \leftarrow_R \mathcal{G}(1^\lambda), x \leftarrow_R \mathbb{Z}_q] \leq \epsilon_{\mathsf{DL}}(\lambda)$, where $\epsilon_{\mathsf{DL}}(\lambda)$ is negligible.

The *computational Diffie-Hellman* (CDH) Assumption: For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}_{\mathsf{CDH}}$ it holds that: $\Pr[\mathcal{A}_{\mathsf{CDH}}(\mathbb{G}, g^x, g^y) = g^{xy} \mid \mathbb{G} \leftarrow_R \mathcal{G}(1^\lambda), x \leftarrow_R \mathbb{Z}_q, y \leftarrow_R \mathbb{Z}_q] \leq \epsilon_{\mathsf{CDH}}(\lambda)$, where $\epsilon_{\mathsf{CDH}}(\lambda)$ is negligible.

The *decisional Diffie-Hellman* oracle ($\mathcal{O}_{\mathsf{DDH}}$) denotes an oracle which for $\mathbb{G} \leftarrow_R \mathcal{G}(1^\lambda), x \in \mathbb{Z}_q, y \in \mathbb{Z}_q, z \in \mathbb{Z}_q$ and $\mathcal{O}_{\mathsf{DDH}}(\mathbb{G}, g^x, g^y, g^z) = 1$ iff $z = xy \mod q$.

The *gap computational Diffie-Hellman* (GDH) Assumption: For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}_{\mathsf{GDH}}^{\mathcal{O}_{\mathsf{DDH}}}$ that has access to decisional Diffie-Hellman oracle $\mathcal{O}_{\mathsf{DDH}}$ it holds that: $\Pr[\mathcal{A}_{\mathsf{GDH}}^{\mathcal{O}_{\mathsf{DDH}}}(\mathbb{G}, g^x, g^y) = g^{xy} \mid \mathbb{G} \leftarrow_R \mathcal{G}(1^\lambda), x \leftarrow_R \mathbb{Z}_q, y \leftarrow_R \mathbb{Z}_q] \leq \epsilon_{\mathsf{GDH}}(\lambda)$, where $\epsilon_{\mathsf{GDH}}(\lambda)$ is negligible.

2.2 Model of Identification Schemes

Below we recall a formal model for an identification scheme.

DEFINITION 1 (IDENTIFICATION SCHEME). *An identification scheme* IS *is a system which consists of four algorithms* (ParGen, KeyGen, $\mathcal{P}$, $\mathcal{V}$) *and a protocol* π*:*

$\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda)$**:** *inputs the security parameter* λ*, and outputs public parameters available to all users of the system (we omit them from the rest of the description).*

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}()$**:** *outputs the secret key* sk *and the corresponding public key* pk *of a prover.*

$\pi(\mathcal{P}, \mathcal{V})$**:** *denotes the protocol executed between the prover* $\mathcal{P}$ *and the verifier* $\mathcal{V}$*.*

$\mathcal{P}(\mathsf{pk}, \mathsf{sk})$**:** *denotes the prover who interacts with the verifier* $\mathcal{V}$ *in the protocol* π *and aims to prove his (or its) identity,*

$\mathcal{V}(\mathsf{pk})$**:** *denotes the verifier who interacts with the prover* $\mathcal{P}$ *in the protocol* π *in order to check whether the prover's identity is the same as claimed.*

We distinguish two stages of the scheme:

- *Initialization: In this stage all parameters are generated:* $\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda)$*, and users are registered. E.g. on behalf of the user of identity* $\hat{A}$ *the procedure* $(a, A) \leftarrow \mathsf{KeyGen}()$ *yields the secret key a and the corresponding public key A.*
- *Operation: In this stage any user, e.g.* $\hat{A}$*, can demonstrate its identity to a verifier by performing the protocol* $\pi(\hat{A}(a, A), \mathcal{V}(A))$ *related to the keys* a, A*. Finally the verifier outputs* 1 *for "accept" or* 0 *for "reject". For simplicity,* $\pi(\mathcal{P}, \mathcal{V}) \to 1$ *means that* $\mathcal{P}$ *has been accepted by* $\mathcal{V}$ *via execution of* π*.*

We require the scheme to be complete: for any pair of keys $(\mathsf{sk}, \mathsf{pk})$ *generated by* $\mathsf{KeyGen}()$ *we have that* $\pi(\mathcal{P}(\mathsf{sk}, \mathsf{pk}), \mathcal{V}(\mathsf{pk})) \to 1$*.*

Intuitively, an identification scheme is secure, if it is infeasible for any adversary prover algorithm $\mathcal{A}$, to be accepted by the verifier unless $\mathcal{A}$ holds the secret key corresponding to A. That is, the probability $\Pr[\pi(\mathcal{A}(\mathsf{pk}), \mathcal{V}(\mathsf{pk})) \to 1]$ must be negligible.

2.3 Okamoto Identification Scheme

In Fig. 1 we recall the Okamoto identification scheme from [17].

$\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda)$**:** Let $\mathbb{G} = (p, q, g, G) \leftarrow \mathcal{G}(1^\lambda)$, s.t. DL Assumption holds. Set $\mathsf{params} = (p, q, g, G)$. Choose $g_1, g_2 \in G$ such that $\log_{g_1} g_2$ is unknown.

$\mathsf{KeyGen}()$**:** $\mathsf{sk} = (a_1, a_2) \leftarrow \mathbb{Z}_q$, $\mathsf{pk} = (g_1, g_2, A)$ where $A = {g_1}^{a_1} \cdot {g_2}^{a_2}$. Output $(\mathsf{sk}, \mathsf{pk})$.

$\pi(\mathcal{P}(\mathsf{sk}, \mathsf{pk}), \mathcal{V}(\mathsf{pk})$**:** The prover $\mathcal{P}(\mathsf{sk}, \mathsf{pk})$ with identity $\hat{A}$ runs with the verifier $\mathcal{V}(\mathsf{pk})$ the following protocol:

1. $\mathcal{P}$: chooses $x_1, x_2 \in \mathbb{Z}_q$ at random, computes $X = {g_1}^{x_1} \cdot {g_2}^{x_2}$ and sends X to the verifier $\mathcal{V}$.
2. $\mathcal{V}$: chooses $c \leftarrow_R \mathbb{Z}_q$, and sends c to $\mathcal{P}$.
3. $\mathcal{P}$: computes $s_1 = x_1 + a_1 \cdot c \bmod q$, $s_2 = x_2 + a_2 \cdot c \bmod q$ and sends s_1, s_2 to the verifier $\mathcal{V}$.
4. $\mathcal{V}$: accepts $\mathcal{P}$ iff ${g_1}^{s_1} \cdot {g_2}^{s_2} == X \cdot A^c$.

Figure 1: The Okamoto identification scheme.

One of the key features of Okamoto protocol is *simulatability* property recalled below. A passive adversary may observe protocol executions. For each interaction the adversary learns its transcript, that is, a tuple $T = (X, c, s_1, s_2)$. However, the adversary may be easily deceived in the following way. First, the simulator algorithm chooses $\tilde{c} \leftarrow_R \mathbb{Z}_q$, $(\tilde{s}_1, \tilde{s}_2) \leftarrow_R \mathbb{Z}_q$ at random. Then it sets $\tilde{X} = {g_1}^{\tilde{s}_1} \cdot {g_2}^{\tilde{s}_2} / A^{\tilde{c}}$. Then the simulator algorithm can replay the precomputed transcript $\tilde{T} = (\tilde{X}, \tilde{c}, \tilde{s}_1, \tilde{s}_2)$ in the correct order, thus simulating an interaction between the prover and the verifier. The deception is perfect, since the tuples $T = (X, c, s_1, s_2)$ and $\tilde{T} = (\tilde{X}, \tilde{c}, \tilde{s}_1, \tilde{s}_2)$ are identically distributed. Consequently, a protocol transcript has no proof value for a third party.

One can also see that if the prover can predict the challenge c before it sends the commitment X, then the security of the protocol collapses – the prover can authenticate himself without the private keys a_1, a_2.

3. OUR MODEL

3.1 Protocol

In this section we recall the security model for IS from [14], where the active adversary possibly set the prover's random values in the query stage of the security experiment. Furthermore we assume the active mode of the query stage: subsequent choices of the adversary can be adjusted according to the responses from the prover. We follow the notation:

- $\mathcal{A} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ is the adversary standing for a coalition of the malicious prover and the malicious verifier,
- $\bar{x}$ denotes the ephemeral secrets chosen by $\mathcal{A}$,
- $\mathcal{P}^{\bar{x}}$ denotes the honest prover $\mathcal{P}$ with injected $\bar{x}$,
- $\pi(\mathcal{P}^{\bar{x}_i}(\mathsf{sk}, \mathsf{pk}), \tilde{\mathcal{V}}(\mathsf{pk}, \bar{x}_i))$ denotes the i-th protocol execution in the query stage,
- ℓ denotes the maximum number of executions of the protocol π in the query stage,
- $\vec{\bar{x}}(\ell) = \{\bar{x}_1, \ldots, \bar{x}_\ell\}$ denotes all the adaptive choices of $\tilde{\mathcal{V}}$ in the query stage,
- $\mathsf{v}^{\mathcal{P}, \tilde{\mathcal{V}}, \vec{\bar{x}}(\ell)}$ denotes the view of the adversary – the knowledge $\tilde{\mathcal{V}}$ can gain in the query stage.

DEFINITION 2 (CHOSEN PROVER EPHEMERAL – (CPE)). *Let* $\mathsf{IS} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathcal{P}, \mathcal{V}, \pi)$ *be an identification scheme. We define security experiment* $\mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE}, \lambda, \ell}$*:*

`Init stage` : *Let* $\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}()$. *The adversary* $\mathcal{A}$ *is initialized as a party that runs an algorithm* $\tilde{\mathcal{P}}$ *(when acting as a verifier) and runs an algorithm* $\tilde{\mathcal{V}}$ *(when impersonating the owner of* sk *and acting as a prover).*

`Query stage` : $\mathcal{A}$ *runs a polynomial number* ℓ *of executions of the protocol* $\pi(\mathcal{P}^{\bar{x}_i}(\mathsf{sk}, \mathsf{pk}), \tilde{\mathcal{V}}(\mathsf{pk}, \bar{x}_i))$ *with the honest prover* $\mathcal{P}^{\bar{x}_i}$*, collecting* $\mathsf{v}^{\mathcal{P}, \tilde{\mathcal{V}}, \vec{\bar{x}}(\ell)}$*, where* $\bar{x}_i \in \{\bar{x}_1, \ldots, \bar{x}_\ell\}$ *denotes the adaptive choices of* $\tilde{\mathcal{V}}$ *injected to the prover* $\mathcal{P}^{\bar{x}_i}$ *in the ith execution.*

`Impersonation stage` : $\mathcal{A}$ *runs the protocol* $\pi(\tilde{\mathcal{P}}(\mathsf{pk}, \mathsf{v}^{\mathcal{P}, \tilde{\mathcal{V}}, \vec{\bar{x}}(\ell)}), \mathcal{V}(\mathsf{pk}))$ *with an honest verifier.*

The advantage of $\mathcal{A}$ *in the experiment* $\mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE}, \lambda, \ell}$ *is defined as the probability of acceptance in the last stage:*

$$\mathbf{Adv}(\mathcal{A}, \mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE}, \lambda, \ell}) = \Pr[\pi(\tilde{\mathcal{P}}(\mathsf{pk}, \mathsf{v}^{\mathcal{P}, \tilde{\mathcal{V}}, \vec{\bar{x}}(\ell)}), \mathcal{V}(\mathsf{pk})) \to 1].$$

The identification scheme is CPE-secure, if the advantage $\mathbf{Adv}(\mathcal{A}, \mathsf{Exp}_{\mathsf{IS}}^{\mathsf{CPE},\lambda,\ell})$ is negligible function in λ.

The Okamoto protocol, as well as many other cryptographic protocols, depends on the random choices. The step of generating random numbers is one of the most risky ones concerning malicious implementation as it is extremely difficult to check honest behavior of a generator. In the worst case, the adversary may have full knowledge of the random numbers generated during a protocol execution. For the Okamoto protocol this would have catastrophic consequences – given s_1, s_2 as well as x_1, x_2 one can immediately derive a_1, a_2 from the equations $a_1 = (s_1 - \bar{x}_1)/c$, $a_2 = (s_2 - \bar{x}_2)/c$ and afterwards impersonate the owner of the keys.

3.2 Architecture Model

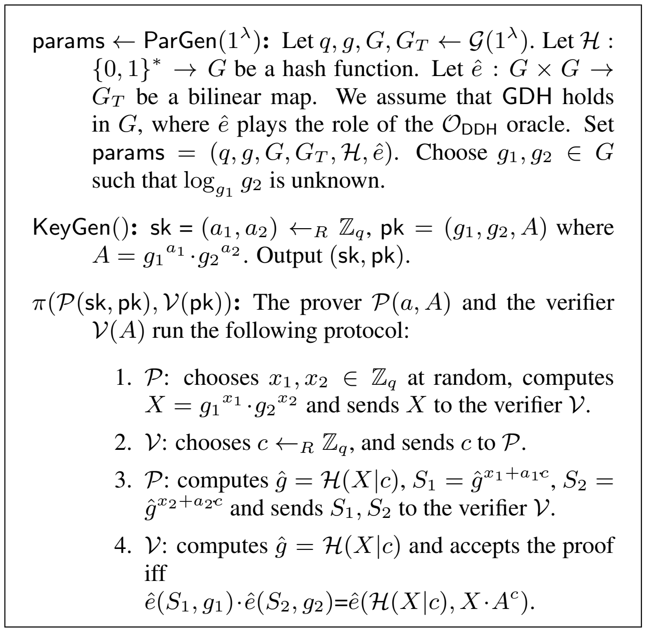
We assume that the prover's part of computation in the protocol $\pi(\mathcal{P}^{\bar{x}_i}(\mathsf{sk}, \mathsf{pk}), \tilde{\mathcal{V}}(\mathsf{pk}, \bar{x}_i))$ is run on a system which is not fully under the prover's control. The prover cannot trust the hardware and the OS software that run the code corresponding to prover's algorithm. However, the only part of that device the prover trust, is the snippet of the platform, called here *secure unit* that stores the long term secret key sk and performs with sk the minimal required functionality. One may ask what such a functionality is, and how it is defined. Here we assume that this is a secure function that *takes some mathematical parameters and produces the public key from the secret key*. Indeed, this is functionality which is required during initial phase of the user/prover registration. Therefore from that moment we will denote that secure functionality of the secure unit as $\mathsf{f}()$, and by $\mathsf{f}(a)$ we denote that this part of the device takes the value a via its input interface, and returns the value a^{sk} through its output interface. From now on we assume the secret key sk is secure, i.e. it is not accessible in other form but as a^{sk} to other parts of the system, nor is available to any subliminal hardware channel. W.l.o.g we can regard $\mathsf{f}()$ as a separate computing unit of *minimal functionality* which is under total control of the user. On the other hand, the ephemeral values $\bar{x}_i$ are under control of the adversary, which has access to the memory part of the OS allocated to the values $\bar{x}_i$, and controls the unit producing randomness for $\bar{x}_i$.

4. MODIFIED OKAMOTO SCHEME

The proposed modified Okamoto IS is shown in Fig. 2. The general idea behind the modification is the following: We play the game against the adversary which knows: the challenge c, the ephemerals x_1, x_2, and which would compute the secret keys (a_1, a_2) given $s_1 = x_1 + a_1 \cdot c$, $s_2 = x_2 + a_2 \cdot c$ from the regular Okamoto IS case. Thus, in the third message of the modified protocol, the prover sends the values $S_1 = \hat{g}^{s_1}$, $S_2 = \hat{g}^{s_2}$ for the generator $\hat{g} = \mathcal{H}(X|c)$ computed with the commitment X and the challenge c. Subsequently, the verifier checks the linear equations $s_1 = x_1 + a_1 \cdot c \bmod q$, $s_2 = x_2 + a_2 \cdot c \bmod q$ in the exponent by using the bilinear map: $\hat{e}(S_1, g_1) \cdot \hat{e}(S_2, g_2) = \hat{e}(\hat{g}, X \cdot A^c)$.

Referring to our device architecture model, the secure unit for the proposed modified Okamoto IS runs the functionality $\mathsf{f}_1(x) := x^{a_1}$ and $\mathsf{f}_2(x) := x^{a_2}$, i.e. the secure unit stores secret keys a_1, a_2 and returns only the results of exponentiations with that keys. Observe that the values $S_1 = \hat{g}^{s_1}$, $S_2 = \hat{g}^{s_2}$ are computable in our device model as $S_1 = \hat{g}^{x_1}\mathsf{f}_1(\hat{g}^c)$, $S_2 = \hat{g}^{x_2}\mathsf{f}_2(\hat{g}^c)$. Hereafter we assume that the computation on the prover's side are done in this particular way.

In Fig. 3 we compare the original Okamoto scheme and our proposition. As for the computational complexity our modification has two exponentiations and one hashing more on the prover's side. The verifier has no extra exponentiation. In fact, it has even got rid



$\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda)$: Let $q, g, G, G_T \leftarrow \mathcal{G}(1^\lambda)$. Let $\mathcal{H} : \{0,1\}^* \to G$ be a hash function. Let $\hat{e} : G \times G \to G_T$ be a bilinear map. We assume that GDH holds in G, where $\hat{e}$ plays the role of the $\mathcal{O}_{\mathsf{DDH}}$ oracle. Set $\mathsf{params} = (q, g, G, G_T, \mathcal{H}, \hat{e})$. Choose $g_1, g_2 \in G$ such that $\log_{g_1} g_2$ is unknown.

$\mathsf{KeyGen}()$: $\mathsf{sk} = (a_1, a_2) \leftarrow_R \mathbb{Z}_q$, $\mathsf{pk} = (g_1, g_2, A)$ where $A = {g_1}^{a_1} \cdot {g_2}^{a_2}$. Output $(\mathsf{sk}, \mathsf{pk})$.

$\pi(\mathcal{P}(\mathsf{sk}, \mathsf{pk}), \mathcal{V}(\mathsf{pk}))$: The prover $\mathcal{P}(a, A)$ and the verifier $\mathcal{V}(A)$ run the following protocol:

1. $\mathcal{P}$: chooses $x_1, x_2 \in \mathbb{Z}_q$ at random, computes $X = {g_1}^{x_1} \cdot {g_2}^{x_2}$ and sends X to the verifier $\mathcal{V}$.
2. $\mathcal{V}$: chooses $c \leftarrow_R \mathbb{Z}_q$, and sends c to $\mathcal{P}$.
3. $\mathcal{P}$: computes $\hat{g} = \mathcal{H}(X|c)$, $S_1 = \hat{g}^{x_1 + a_1 c}$, $S_2 = \hat{g}^{x_2 + a_2 c}$ and sends S_1, S_2 to the verifier $\mathcal{V}$.
4. $\mathcal{V}$: computes $\hat{g} = \mathcal{H}(X|c)$ and accepts the proof iff $\hat{e}(S_1, g_1) \cdot \hat{e}(S_2, g_2) = \hat{e}(\mathcal{H}(X|c), X \cdot A^c)$.

Figure 2: Modified Okamoto identification scheme

of two explicit exponentiations – there is no need to compute ${g_1}^{s_1}$ and ${g_2}^{s_2}$. However, as a price for that, the verifier computes a hash value, three pairings, and a product of two elements in G_T.

4.1 Correctness

THEOREM 4.1. *The modified Okamoto protocol (Fig. 2) is complete according to Definition 1, that is*

$$\Pr[\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda), (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}() : \pi(\mathcal{P}(\mathsf{sk}, \mathsf{pk}), \mathcal{V}(\mathsf{pk})) \to 1] = 1 .$$

PROOF. For well generated keys used by the prover and the verifier the verification will yield the positive answer, as the following equalities hold:

$$\begin{aligned}\hat{e}(S_1, g_1) \cdot \hat{e}(S_2, g_2) &= \hat{e}(\hat{g}^{x_1 + a_1 c}, g_1) \cdot \hat{e}(\hat{g}^{x_2 + a_2 c}, g_2) \\ &= \hat{e}(\hat{g}, {g_1}^{x_1 + a_1 c}) \cdot \hat{e}(\hat{g}, {g_2}^{x_2 + a_2 c}) \\ &= \hat{e}(\hat{g}, {g_1}^{x_1 + a_1 c} \cdot {g_2}^{x_2 + a_2 c}) = \hat{e}(\hat{g}, X \cdot A^c).\end{aligned}$$

□

4.2 Simulatability

The modified Okamoto IS preserves the simulatability property of its original version. A fake protocol transcript can be created as follows. First one chooses $\tilde{s}_1, \tilde{s}_2, \tilde{c}$ at random, then computes

$\tilde{X} = ({g_1}^{\tilde{s}_1} {g_2}^{\tilde{s}_2}) / A^{\tilde{c}}$,
$\hat{g} = \mathcal{H}(\tilde{X}|\tilde{c})$,
$\tilde{S}_1 = \hat{g}^{\tilde{s}_1}$, $\quad \tilde{S}_2 = \hat{g}^{\tilde{s}_2}$.

Obviously, the tuples $(\tilde{X}, \tilde{c}, \tilde{S}_1, \tilde{S}_2)$ created in this way and the tuples (X, c, S_1, S_2) resulting from the genuine protocol executions are identically distributed.

4.3 Simulation in the CPE Model

In this section we provide arguments for security of our version of the Okamoto scheme. We show that it is simulatable in the proposed *Chosen Prover Ephemeral* (CPE) model. Assuming programmable ROM (Random Oracle Model) we can simulate the

the original Okamoto scheme			**the modified variant**		
$\mathcal{P}(a, A = {g_1}^{a_1}{g_2}^{a_2})$		$\mathcal{V}(A)$	$\mathcal{P}(a, A = {g_1}^{a_1}{g_2}^{a_2}))$		$\mathcal{V}(A)$
$x_1 \leftarrow_R \mathbb{Z}_q$			$x_1 \leftarrow_R \mathbb{Z}_q$		
$x_2 \leftarrow_R \mathbb{Z}_q$			$x_2 \leftarrow_R \mathbb{Z}_q$		
$X = {g_1}^{x_1} \cdot {g_2}^{x_2}$			$X = {g_1}^{x_1} \cdot {g_2}^{x_2}$		
	$\xrightarrow{X}$			$\xrightarrow{X}$	
		$c \leftarrow_R \mathbb{Z}_q$			$c \leftarrow_R \mathbb{Z}_q$
	$\xleftarrow{c}$			$\xleftarrow{c}$	
			$\hat{g} = \mathcal{H}(X \mid c)$		
$s_1 = x_1 + a_1 \cdot c \bmod q$			$S_1 = \hat{g}^{x_1 + a_1 \cdot c}$		
$s_2 = x_2 + a_2 \cdot c \bmod q$			$S_2 = \hat{g}^{x_2 + a_2 \cdot c}$		
	$\xrightarrow{s_1, s_2}$			$\xrightarrow{S_1, S_2}$	
					$\hat{g} = \mathcal{H}(X \mid c)$
		accept iff ${g_1}^{s_1} \cdot {g_2}^{s_2} = X \cdot A^c$			accept iff $\hat{e}(S_1, g_1) \cdot \hat{e}(S_2, g_2) = \hat{e}(\hat{g}, X \cdot A^c)$

Figure 3: Side-by-side comparison of the original Okamoto scheme and its modified version

protocol $\pi(\mathcal{P}^{\bar{x}}(\mathsf{pk}), \tilde{\mathcal{V}}^{\mathcal{O}_\mathcal{H}}(\mathsf{pk}, \bar{x})) \rightarrow 1$ on behalf of the prover $\mathcal{P}^{\bar{x}}(\mathsf{pk})$ without the secret key sk, using the injected ephemerals $\bar{x}$, and interacting with the active adversary $\tilde{\mathcal{V}}^{\mathcal{O}_\mathcal{H}}(\mathsf{pk}, \bar{x})$, which injects the ephemerals $\bar{x}$ to the prover and performs adaptive choices of challenges. Note that the adversary calls the oracle $\mathcal{O}_\mathcal{H}$ to compute the hash value for the queried input.

THEOREM 4.2. *The modified Okamoto protocol (Fig. 2) is simulatable in the CPE model (Definition 2).*

PROOF. The simulator $\mathcal{S}_{\mathsf{IS}}^{\mathsf{CPE},\pi}()$ is defined in the following way: For given (g_1, g_2, A) we choose $a_2 \leftarrow_R \mathbb{Z}_q$ at random and set ${g_1}^{a_1}$ as $A/({g_2}^{a_2})$. (Note that we do not derive a_1.)

1) **Hash queries $\mathcal{O}_\mathcal{H}$:** We use a ROM table for the hash queries $\mathcal{O}_\mathcal{H}$. The table has three columns I, H, r: for the input, the output and the masked exponent, respectively. On each query $\mathcal{O}_\mathcal{H}(I_i)$, the oracle checks, if the answer has been it already stored in the hash table - if so, then it returns the corresponding output H_i. Otherwise it chooses $r_i \leftarrow_R \mathbb{Z}_q$, computes $H_i = g^{r_i}$, places a new record (I_i, H_i, r_i) in the ROM table, and returns H_i.

2) **Commitment X:** When injected ephemerals $(\bar{x}_1, \bar{x}_2)$ we use it to compute $\tilde{X} = {g_1}^{\bar{x}_1} \cdot {g_2}^{\bar{x}_2}$. The value $\bar{X}$ is sent to the verifier $\tilde{\mathcal{V}}^{\mathcal{O}_\mathcal{H}}(\mathsf{pk}, (\bar{x}_1, \bar{x}_2))$ in the first message.

3) **Creating a proof for the verifier:** upon receiving $\tilde{c}$ from the verifier, we call $\mathcal{O}_\mathcal{H}(\bar{X}|\tilde{c})$. We check $\mathcal{O}_\mathcal{H}$ table for the input $\bar{X}|\tilde{c}$, locate and retrieve the corresponding g^r and r. We set $\hat{g} = g^r$. Then we compute

$$S_1 = {g_1}^{r\bar{x}_1}(A/{g_2}^{a_2})^{rc} \quad \text{and} \quad S_2 = {g_2}^{r\bar{x}_2 + a_2 rc}$$

Note that $S_1 = {g_1}^{r\bar{x}_1}({g_1}^{a_1})^{rc} = \hat{g}^{\bar{x}_1 + a_1 c}$ and $S_2 = \hat{g}^{\bar{x}_2 + a_2 c}$.

Now, the verification test on the prover's side yields the positive result: $\hat{e}(\tilde{S}_1, g_1) \cdot \hat{e}(\tilde{S}_2, g_2) = \hat{e}(\hat{g}, X \cdot A^{\tilde{c}})$ for $\hat{g} \leftarrow \mathcal{O}_\mathcal{H}(\bar{X}|\tilde{c})$. Moreover, the simulated transcript tuples $(\tilde{X}, \tilde{c}, \tilde{S}_1, \tilde{S}_2)$ and the real ones (X, c, S_1, S_2) are identically distributed. □

4.4 Security Analysis

We follow the same methodology as in the case of the original Okamoto IS. We show that a successful attack against our scheme can be applied to break the underlying GDH problem with a non-negligible probability. In order to use this attack given a GDH instance, we have to build an environment for execution of our scheme with the instance of the GDH problem injected. As not all private keys are given in this case, we have to use the simulation described in Sect. 4.3. Then we use the *rewinding technique*: we run the impersonation stage twice for the same fixed commitment X, but with different challenges c, c' resulting with different responses S_1, S_2, and S_1', S_2'. The resulting tuples (X, c, S_1, S_2), (X, c', S_1', S_2') will enable us to break the underlying GDH problem. Rewinding is possible, as we are in full control of the process breaking the scheme.

THEOREM 4.3. *Let* IS *denote the modified Okamoto identification scheme (as of Fig. 2). If* IS *is insecure (in the sense of Definition 2), i.e. the advantage* $\mathbf{Adv}(\mathcal{A}, \mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE},\lambda,\ell})$ *is non-negligible in* λ *for an algorithm* $\mathcal{A}$*, then there is an algorithm* $\mathcal{A}_{\mathsf{GDH}}$ *with similar computational complexity that breaks* GDH *with a non-negligible probability.*

PROOF SKETCH. Suppose there is an adversary algorithm $\mathcal{A} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ for which $\mathbf{Adv}(\mathcal{A}, \mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE},\lambda,\ell})$ is non-negligible. We use it as a subprocedure of an efficient algorithm $\mathcal{A}_{\mathsf{GDH}}$ that breaks the GDH: for a given instance g^α, g^β it computes $g^{\alpha\beta}$ also with a non-negligible probability.

Let us describe how we build an environment for running the adversary $\mathcal{A}$ breaking the modified Okamoto scheme and run it in order to break the GDH problem:

`Init stage` : We set params to be $\mathbb{G} = (q, g, G)$ from the GDH problem, and let (g^α, g^β) be an GDH instance in $\mathbb{G}$. We choose $(a_2, w) \leftarrow_R \mathbb{Z}_q$. We set

$$g_1 = g, \quad g_2 = g^w, \quad \mathsf{pk} = A = g^\alpha \, .$$

Thus we have ${g_1}^{a_1} = A/({g_2}^{a_2}) = A/(g^{a_2 \cdot w})$.

The adversary $\mathcal{A}$ is given the public key pk. We initialize a ROM table for hash queries $\mathcal{O}_\mathcal{H}$. The table has columns I, H, r for, respectively, the input, the output and the masked exponent. The hash queries will be answered in the following way: in the `Query stage` we will use the simulator $\mathcal{S}_{\mathsf{IS}}^{\mathsf{CPE},\pi}$

(according to the method described in the proof of Theorem 4.2). In the `Impersonation stage` the adversary gets the value $(g^{\beta})^{\zeta}$, where ζ is a random mask.

`Query stage` : We simulate in ROM a polynomial number ℓ of executions of the protocol $\pi(\mathcal{P}^{(\bar{x}_1,\bar{x}_2)_i}(\mathsf{pk}), \tilde{\mathcal{V}}^{\mathcal{O}_{\mathcal{H}}}(\mathsf{pk}, (\bar{x}_1,\bar{x}_2)_i)$ without the secret key, interacting with the active adversary verifier, which injects ephemerals $\bar{x}_i$ by running the simulator $\mathcal{S}_{\mathsf{IS}}^{\mathsf{CPE},\pi}$ given by Theorem 4.2.

Let $\mathsf{v}^{\mathcal{P},\tilde{\mathcal{V}},\vec{\bar{x}}(\ell)}$ be the view of the adversary collected in this stage, where $(\bar{x}_1,\bar{x}_2)_i \in \{(\bar{x}_1,\bar{x}_2)_1, \ldots, (\bar{x}_1,\bar{x}_2)_\ell\} = \vec{\bar{x}}(\ell)$ denotes the adaptive choices of $\tilde{\mathcal{V}}$ injected as ephemerals to the prover $\mathcal{P}^{(\bar{x}_1,\bar{x}_2)_i}$ during the ith execution.

`Impersonation stage` : In the ROM model we run $\pi(\tilde{\mathcal{P}}^{\mathcal{O}_{\mathcal{H}}}(\mathsf{pk}, \mathsf{v}^{\mathcal{P},\tilde{\mathcal{V}},\vec{\bar{x}}(\ell)}), \mathcal{V}(\mathsf{pk}))$ serving the role of the honest verifier. We use the *rewinding technique*: we fix the commitment X used by the algorithm $\tilde{\mathcal{P}}$, and let $\tilde{\mathcal{P}}$ interact twice with the verifier, choosing each time a different random challenge, c and c', such that neither $X|c$ nor $X|c'$ were the input to $\mathcal{O}_{\mathcal{H}}$ in the `Query stage`, and setting $\hat{g} = \mathcal{O}_{\mathcal{H}}(X|c) \leftarrow (g^{\beta})^{\eta}$, $\hat{g}' = \mathcal{O}_{\mathcal{H}}(X|c') \leftarrow (g^{\beta})^{\eta'}$ for $\eta, \eta' \leftarrow_R \mathbb{Z}_q$. These interactions will result with $(X, c, S_1, S_2, \hat{g}, \eta)$ and $(X, c', S_1', S_2', \hat{g}', \eta')$, accordingly.

We proceed, if $\mathcal{A}$ is successful, that is the verification tests yield the positive results: $\hat{e}(S_1, g_1)\cdot\hat{e}(S_2, g_2) = \hat{e}(\hat{g}, X\cdot A^c)$ and $\hat{e}(S_1', g_1)\cdot\hat{e}(S_2', g_2) = \hat{e}(\hat{g}, X\cdot A^{c'})$.

Let us notice that we can express S_1, S_2, A, X in the following form:

$$S_1 = \hat{g}^{s_1} = \hat{g}^{x_1+a_1\cdot c}, \quad S_2 = \hat{g}^{s_2} = \hat{g}^{x_2+a_2\cdot c},$$
$$A = g_1^{a_1}\cdot g_2^{a_2}, \quad X = g_1^{x_1}\cdot g_2^{x_2}$$

Indeed, given (A, s_1, s_2, c) we fix x_1 arbitrarily. Then we get a unique a_1 fulfilling $s_1 = x_1 + a_1 c \bmod q$, and a unique x_2 fulfilling $X = g_1^{x_1}\cdot g_2^{x_2}$. Then we may derive a unique a_2 for which $s_2 = x_2 + a_2 c \bmod q$. In this way all equations are fulfilled, may be except for $A = g_1^{a_1}\cdot g_2^{a_2}$. However, note that the test equation is equivalent to:

$$g_1^{s_1}\cdot g_2^{s_2} = g_1^{x_1}\cdot g_2^{x_2}\cdot A^c$$

which in turn yields $g_1^{a_1}\cdot g_2^{a_2} = A$. (Of course, we cannot assume that the adversary $\mathcal{A}$ is aware of these exponents.)

In a similar way we may express

$$S_1' = \hat{g}'^{x_1'+a_1'\cdot c'}, \quad S_2' = \hat{g}'^{x_2'+a_2'\cdot c'},$$
$$A = g_1^{a_1'}\cdot g_2^{a_2'}, \quad X = g_1^{x_1'}\cdot g_2^{x_2'}$$

We have that:
$S_1{}^{(\eta^{-1})}/S_1'{}^{(\eta'^{-1})} = (g^{\beta})^{(x_1+a_1\cdot c)-(x_1'+a_1'\cdot c')}$ and
$S_2{}^{(\eta^{-1})}/S_2'{}^{(\eta'^{-1})} = (g^{\beta})^{(x_2+a_2\cdot c)-(x_2'+a_2'\cdot c')}$.
So we have:

$$\begin{aligned}
&(S_1{}^{(\eta^{-1})}/S_1'{}^{(\eta'^{-1})})(S_2{}^{(\eta^{-1})}/S_2'{}^{(\eta'^{-1})})^w \\
&= (g^{\beta})^{(x_1+a_1\cdot c)-(x_1'+a_1'\cdot c')}(g^{\beta})^{w(x_2+a_2\cdot c)-w(x_2'+a_2'\cdot c')} \\
&= (g^{\beta})^{(x_1+w\cdot x_2)-(x_1'+w\cdot x_2')}(g^{\beta})^{c(a_1+w\cdot a_2)-c'(a_1'+w\cdot a_2')} \\
&= (g^{\beta})^{(c-c')\alpha} \quad .
\end{aligned}$$

Thus we have
$g^{\beta\alpha} = ((S_1{}^{(\eta^{-1})}/S_1'{}^{(\eta'^{-1})})(S_2{}^{(\eta^{-1})}/S_2'{}^{(\eta'^{-1})})^w)^{((c-c')^{-1})}$.

□

5. FINAL REMARKS

5.1 Security assumptions

The original invention of the Okamoto protocol was to enable a security proof that refers directly to a reduction to the Discrete Logarithm Problem. This is considered to be a much better guarantee than the one given for the Schnorr IS.

On the other hand, the effort to fine tune the formal security proof has negligible practical impact, if we forget about the critical issue of securing the basic implementation details such as malicious random number generator. In this light retreating to Random Oracle Model can be regarded as lesser evil than a strong argument in a model that makes risky assumptions about device dependent security features. Nevertheless, we hope that the Random Oracle Assumption may be eliminated with a different reduction proof, possibly for a different mutation of the Okamoto protocol.

5.2 Private key separation

One of the important features of the protocol proposed in this paper is that one can implement the private key in a dedicated high security unit and separated from the random number generation. In our case this unit has to execute only one operation: computes r^x for an element r contained in the request and the private key x stored in the unit. In the classical Okamoto protocol there is no way to effectively separate the private key as long as the ephemeral values are not encapsulated in the unit. Let us note that this kind of separation has been already proposed in [12], however on the level of internal protocol implementation.

Of course, the internal unit described above may serve as an oracle for raising to secret power x, which to some degree may ease cryptanalysis for the adversary that takes control over the whole system apart from the security unit. However, this is much better compared with the case when the secret key is present in the plain form in this compromised system.

In our opinion this mechanism of private key separation within the protocol might be a very handy mechanism enabling much easier and more secure practical implementation. This concerns in particular future PC architectures with the private key encapsulated in future TPM units. This applies also to cloud systems, where the processes run for the user would identify themselves on behalf of the user and interact with an external user's unit performing the operations on the secret key. In such an architecture the whole identification process would look the same on the side of the user's partners, regardless how the user organizes his processes in the cloud system.

5.3 Security situation for the eIDAS token

Let us shortly review the situation of the security of Pseudonymous Signature from eIDAS Token from [6]. The mechanism of this signature is based directly on the Okamoto identification scheme adjusted, via Fiat-Shamir heuristic, to the case of signatures. Each signature corresponds to a pseudonym. The pseudonyms are created ad hoc for any number of domains, so that for a given domain only one set of identifiers can be created by a user. The major advantage of the scheme is no need for certificates and white-lists as well as strong unlinkability across different domains.

On the dark side, the scheme is based on Schnorr-like signatures. In particular, there are two components of the signature which are linear transformations of the secret keys. Unfortunately, once the adversary learns the ephemeral values used to define these transformations, the private keys of the user can be immediately derived by the attacker. Unfortunately, this is only the beginning of problems. If the adversary succeeds to perform this attack for two different users, then it remains to solve a system of two independent linear equations in order to derive the system private keys. This in turn enables the attacker to create any number of fake identities accepted in the system.

Immunizing the Pseudonymous Signature against this kind of threats is a challenge, unless we agree to use the strategies followed in our paper – namely moving a signature component to the exponent. The price to be paid is, however, the use of pairings.

6. CONCLUSION

We have shown that it is possible to redesign the Okamoto identification scheme in a way that makes it immune against an attacker knowing the ephemeral values chosen by the prover during the protocol execution. Increasing the security level has its price – requires pairings to be used on the side of the verifier. Nevertheless, it might be plausible to design a version of the protocol without pairings. It is an open question whether we can base security argument in this case on such a firm assumption like GDH.

The proposed method can be regarded as a step in rethinking the cryptographic protocols to be deployed without encapsulating them completely in secure hardware units. Certainly, this may significantly reduce security level, however the software implementation might be the only available option. This involves in particular the user (devices) communicating with the cloud services. We show that at least some risks can be mitigated.

7. ACKNOWLEDGMENTS

The paper was partially supported by the Polish National Science Center, HARMONIA DEC-2013/08/M/ST6/00928. We would like to thank Shlomi Dolev for discussions on this paper during our visit to Ben Gurion University.

8. REFERENCES

[1] Alwen, J., Dodis, Y., and Wichs, D. Leakage-resilient public-key cryptography in the bounded-retrieval model. In CRYPTO 2009, S. Halevi, Ed., vol. 5677 of LNCS, Springer, pp. 36–54.

[2] Ateniese, G., Magri, B., and Venturi, D. Subversion-resilient signature schemes. In ACM CCS 2015, I. Ray, N. Li, and C. Kruegel, Eds., ACM, pp. 364–375.

[3] Becker, G. T., Regazzoni, F., Paar, C., and Burleson, W. P. Stealthy dopant-level hardware trojans: extended version. *J. Cryptographic Engineering 4*, 1 (2014), 19–31.

[4] Bellare, M., Fischlin, M., Goldwasser, S., and Micali, S. Identification protocols secure against reset attacks. In EUROCRYPT 2001, B. Pfitzmann, Ed., vol. 2045 of LNCS, Springer, pp. 495–511.

[5] Bender, J., Fischlin, M., and Kügler, D. The PACE|CA protocol for machine readable travel documents. In INTRUST 2013, R. Bloem and P. Lipp, Eds., vol. 8292 of LNCS, Springer, pp. 17–35.

[6] BSI. Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token 2.20. Technical Guideline TR-03110, 2015.

[7] Canetti, R., Goldreich, O., Goldwasser, S., and Micali, S. Resettable zero-knowledge (extended abstract). In ACM STOC 2000, F. F. Yao and E. M. Luks, Eds., ACM, pp. 235–244.

[8] Dziembowski, S., Faust, S. Leakage-Resilient Cryptography from the Inner-product Extractor. In ASIACRYPT '11, Lee D.H., Wang X., Ed., vol. 7073 of LNCS, Springer, pp. 702–721.

[9] Feige, U., Fiat, A., and Shamir, A. Zero-knowledge proofs of identity. *Journal of Cryptology 1*, 2, 77–94.

[10] Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In CRYPTO '86, A. M. Odlyzko, Ed., vol. 263 of LNCS, Springer, pp. 186–194.

[11] Guillou, L. C., and Quisquater, J. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In EUROCRYPT '88, C. G. Günther, Ed., vol. 330 of LNCS, Springer, pp. 123–128.

[12] Hanzlik, L., Krzywiecki, L., and Kutyłowski, M. Simplified PACE|AA protocol. In ISPEC 2013, R. H. Deng and T. Feng, Eds., vol. 7863 of LNCS, Springer, pp. 218–232.

[13] ISO/IEC JTC1 SC17 WG3/TF5 for the International Civil Aviation Organization. Supplemental Access Control for Machine Readable Travel Documents v1.1. Technical Report, April 15 2014.

[14] Krzywiecki, Ł. Schnorr-like identification scheme resistant to malicious subliminal setting of ephemeral secret. In Innovative Security Solutions for Information Technology and Communications - 9th International Conference, SECITC 2016, I. Bica and R. Reyhanitabar, Eds., vol. 10006 of LNCS, pp. 137–148.

[15] Kutyłowski, M., Hanzlik, L., and Kluczniak, K. Controlled randomness - a defense against backdoors in cryptographic devices. In *MYCRYPT'2016 -Paradigm-Shifting Cryptography (to appear), see http://kutylowski.im.pwr.wroc.pl/articles/ MYCRYPT2016PRNG-talk.pdf*.

[16] NIST. Random Number Generation, 2010.

[17] Okamoto, T. Provably secure and practical identification schemes and corresponding signature schemes. In CRYPTO '92, E. F. Brickell, Ed., vol. 740 of LNCS, Springer, pp. 31–53.

[18] Russell, A., Tang, Q., Yung, M., and Zhou, H. Cliptography: Clipping the power of kleptographic attacks. *IACR Cryptology ePrint Archive 2015* (2015), 695.

[19] Schnorr, C.-P. Efficient signature generation by smart cards. *J. Cryptology 4*, 3 (1991), 161–174.

[20] Young, A. L., and Yung, M. Kleptography from standard assumptions and applications. In *Security and Cryptography for Networks*, SCN 2010, J. A. Garay and R. D. Prisco, Eds., vol. 6280 of LNCS, Springer, pp. 271–290.