Efficient Cryptographic Password Hardening Services from Partially Oblivious Commitments

Jonas Schneider
CISPA, Saarland University
Saarland Informatics Campus

Nils Fleischhacker
CISPA, Saarland University
Saarland Informatics Campus

Dominique Schröder
Friedrich-Alexander-University,
Erlangen-Nürnberg

Michael Backes
CISPA, Saarland University &
MPI-SWS
Saarland Informatics Campus

ABSTRACT

Password authentication still constitutes the most widespread authentication concept on the Internet today, but the human incapability to memorize safe passwords has left this concept vulnerable to various attacks ever since. Affected enterprises such as Facebook now strive to mitigate such attacks by involving external cryptographic services that harden passwords. Everspaugh et al. provided the first comprehensive formal treatment of such a service, and proposed the PYTHIA PRF-Service as a cryptographically secure solution (Usenix Security'15). PYTHIA relies on a novel cryptographic primitive called partially oblivious pseudorandom functions and its security is proven under a strong new interactive assumption in the random oracle model.

In this work, we prove that this strong assumption is inherently necessary for the PYTHIA construction, i.e., it cannot be weakened without invalidating the security of PYTHIA. More generally, it is impossible to reduce the security of PYTHIA to any non-interactive assumptions. Hence any efficient, scalable password hardening service that is secure under weaker assumptions necessarily requires a conceptually different construction. To this end, we propose a construction for password hardening services based on a novel cryptographic primitive called partially oblivious commitments, along with an efficient secure instantiation based on simple assumptions. The performance and storage evaluation of our prototype implementation shows that our protocol runs almost twice as fast as PYTHIA, while achieving a slightly relaxed security notion but relying on weaker assumptions.

1. INTRODUCTION

Password-based authentication is still the most widespread means of authentication on the Internet today. In such a system, a user sends her username and password to a server, which compares both values against its stored record. Securely realizing such a system turns out to be difficult. In addition to numerous usability pitfalls [25, 17], attackers can steal password databases and mount efficient brute-force attacks, even in the face of known security properties of a password scheme [41]. Practical attacks against several enterprises, such as Target, Adobe, or AOL (an overview is given in [42]) have shown that storing passwords as salted hash values is not sufficient to ensure confidentiality in case of a breach. From this insufficiency arose a multitude of proposals for authentication mechanisms which avoid passwords altogether, e.g. [13, 30, 12]. However, if a password-based authentication system has already been established it is often more economically viable to harden it against possible attacks than to replace it from scratch.

One way to harden the security of password-based authentication which has recently seen adoption from large companies like Facebook [32] is to involve external cryptographic PRF-Services [6, 38]. The basic idea is to physically separate the database that stores the password records from the server which computes the corresponding hashes. This way, stealing a database of password records is not sufficient to mount offline attacks and the adversary has to compromise the cryptographic service as well. Failing to do this the adversary is forced to interact with the service, pretending to be a legitimate client. This interaction can then be detected via fine-grained monitoring of suspicious use of the API.

While cryptographic PRF-Services have received attention in industrial systems [32], a comprehensive formal treatment of such a service was only recently provided by Everspaugh et al. [19], who described the PYTHIA PRF-Service and gave the first formal consideration to its security. The PYTHIA PRF-Service is flexible and efficient enough to be deployed within an enterprise and it offers several security features that conventional PRF-Services do not offer. Among others, PYTHIA supports message privacy, rate-limiting, and individual key rotation. To achieve these properties, the authors introduced a novel cryptographic primitive called partially oblivious pseudorandom functions (PO-PRFs). A PO-PRF is a two-party protocol where one party, a cryptographic service provider, holds a key k for a function f_k and a client wishes to learn $y \leftarrow f_k(t, m)$, where t is called a tweak and m the message. The output of the function is pseudorandom and the service provider is partially oblivious about the input meaning that the value t is public and m remains hidden. The main motivation for revealing t is to enable rate-limiting of online attacks, as a specific t being requested over and over again is visible in the requests. Going beyond password hardening applications, [19] discusses additional applications of PO-PRFs such as Bitcoin brainwallets.

Partially oblivious PRFs are clearly an extremely powerful primitive and realizing them seems to require strong assumptions. Indeed, the construction of Everspaugh et al. is similar to the left-or-right constrained PRF of Boneh and Waters [9] and the non-interactive key-agreement protocol of Sakai, Ohgishi, and Kasahara [39], thus the scheme also requires similar assumptions. That is, the security of their scheme is proven under an interactive assumption and the proof is given in the random oracle model. The authors leave open the question if these (trusted) assumptions are necessary or if one could find a scheme based on simple assumptions in the standard model.

In this work, we show that this is not the case. We give an impossibility result that rules out any security proof to an arbitrary non-interactive assumption (using black-box techniques only). Driven by the idea to develop an efficient password hardening service based on simple assumptions, we develop a different cryptographic primitive in order to bypass this impossibility result. This primitive is general enough to capture all requirements for a modern password hardening system, yet it is not affected by the impossibility result. This is achieved by relaxing the requirement of pseudorandom protocol outputs. Pseudorandomness provides ample security for password hashes but is not ultimately necessary to achieve a sufficient level of security. We call this new primitive *Partially Oblivious Commitments* and propose an instance based on simple assumptions. The main difference to PO-PRFs is that the record output is not required to be pseudorandom, although it can be, and that there exists a verification algorithm to check whether a stored record matches (t, m) (rather than an equality test). Our comprehensive performance evaluations demonstrate the practicability of our approach and also show that our scheme is nearly twice as fast as the scheme suggested by Everspaugh et al. [19] in terms of latency in common settings.

1.1 Our Contribution

Our main contribution is the first efficient and scalable password hardening scheme that is provably secure under simple assumptions. The construction is based on a novel cryptographic primitive called *Partially Oblivious Commitment* (PO-COM). A PO-COM is an interactive two-party protocol which allows one party to commit to a message m and a tweak t. The commitment is binding in the usual sense and partially hiding meaning that t is public, but m remains hidden. In addition, there exists an efficient protocol that allows to verify whether a commitment contains (t, m). PO-COMs are similar to partially oblivious pseudorandom functions as introduced by Everspaugh et al. [19]. However, we show that the security of their scheme inherently relies on strong interactive assumptions and cannot be reduced to any simple assumptions in the standard model (using black-box techniques only). In fact, our result is more general as it also rules out achieving weaker security notions. This result answers a question explicitly left open by Everspaugh et al. [19].

Our definition of PO-COMs is general enough to capture all (functional) properties specified by the Pythia framework for password hardening mechanisms and yet remains impervious to the impossibility result. We give a simple and efficient instantiation of PO-COM that can be seen as a “twin-key” variant of the Pedersen commitment scheme [36]. The main idea of our twin-key technique is to derive a second key $s \leftarrow F_k(t)$ via a pseudorandom function and to use this key to blind its twin key x. The purpose of the twin key is to deterministically bind the stored record to the non-hidden tweak t.

We demonstrate the efficiency of our construction via comprehensive performance evaluations and compare it to the original implementation of Pythia. Our performance evaluations show the practicability of our scheme. In particular, our scheme is almost twice as fast as the construction Everspaugh et al. [19], which is somewhat surprising given that our construction is provably secure under simple assumptions and fulfills all properties put forward in the work of Everspaugh et al. [19].

1.2 Related Work

Primitives that are related to partially oblivious pseudorandom functions, and by extension to our generalization, include delegatable PRFs [29], fully oblivious PRFs, introduced in [33, 24] and variants of those which also provide verifiability [26]. These primitives are close in functionality to the requirements of our setting, but as noted in [19] do not allow (only partial) obliviousness.

The security properties introduced in [19] are similare to the security properties of blind signature schemes [14, 27, 37, 21, 23, 40, 34]. This similarity to blind signatures opens the possibility for a translation of results regarding the ability to instantiate the primitive under various assumptions.

Our analysis in this respect draws from the results of [22] which say that there cannot be a black-box reduction from the unforgeability of a three-move blind signature scheme to any non-interactive problem. Moreover, there are a number of separation results regarding “one-more” type assumptions and security notions, e.g. [10, 11] and there is recent progress on automatically analyzing certain classes of assumptions [4].

Our goal of offline attack resistance is similar to that of threshold password-authenticated key-exchange (t-PAKE) [31], where a client has to interact with a threshold k out of n available servers to authenticate. However, our primitive is not distributed as we only consider protocols between a client and a single server. Additionally, although the schemes presented in [31, 3, 16] support blinding, they do not support *partial* blinding and the rate-limiting enabled by it, as required for our primitive. The overall result of the protocol execution in our case is a binary decision on a provided piece of information, whereas PAKE, t-PAKE, and group key exchange [2] aim to derive a new key each time the protocol is executed.

Our primitive is similar to P-Signatures [5], in that there is a way to verify whether two commitments are for the same value. However, this capability is restricted to the client in our setting, whereas in P-Signatures, the client should convince the server that they hold a signature for a committed value.

2. PASSWORD AUTHENTICATION

In this section, we discuss the main security and functional properties that modern password authentication services should have, following the discussion in [19]. A password-based authentication scheme has two phases:

Registration. A user registers at a service providing username and password of her choice. The service stores the username together with an authentication token, such as a salted hash value.

Verification. A registered user sends her username and password to the server, which checks if this information matches the stored record.

The aim of a password registration and verification procedure is to ensure that legitimate users have access to the service. At the same time, an attacker should not be able to use/guess the credentials of an honest user. Sophisticated online and offline attacks have shown that the passwords should not be permanently stored in plain anywhere on the server. Instead, the server has to derive a check value, such as a salted hash, from the password upon registration, which can then be derived also from the offered password in the verification phase.

In offline brute-force attacks, an attacker has obtained the information used by the server to verify a password. This information typically consists of the hash of the original password together with a random salt per password, which is stored together with the hash. The attacker now tries to verify a large number of (precomputed) password guesses to determine the correct password.

While such attacks are hard to prevent in general, it is possible to inhibit their effectiveness, e.g., through the introduction of a third party, a cryptographic PRF Service, which was formalized in [19]. The third party manages a high entropy secret, which is used to compute high entropy authentication tokens, stored by the server. The separation of data and the secret used to compute the data introduces an additional barrier a brute-force adversary has to overcome. We describe the main functional and security properties that should be provided. Subsequently, we show how to build such a service based on a novel cryptographic primitive called partially oblivious commitments (PO-COM's).

Message Privacy. The cryptographic service should be oblivious about the password m. This ensures that a corrupted service provider does not leak any information about the password.

Tweak Visibility. The username t must be visible to the service to allow fine-grained rate-limiting of requests.

Secure Key Rotation. On compromise of the service master secret key there must a way to issue key updates to all clients which allow migration of client data to a fresh, uncompromised key. This key rotation should be secure in the sense that an attacker who obtains a client and service state of different rotation levels should not be able to make use of the client database.

In the subsequent sections, we propose our novel cryptographic primitive, PO-COMs, and show how to build a system as described above based on PO-COMs.

3. PO-COMMITMENTS

We introduce the notion of partially oblivious commitments (PO-COM). Loosely speaking, a PO-COM is a two-party protocol between a client $\mathcal{C}$ and a server $\mathcal{S}$. The input of $\mathcal{C}$ is a public key pk, a hidden message m, a public tweak t, and a client secret csk. The server's input is a secret key sk (that corresponds to pk). At the end of the protocol the client outputs a commitment T and we call T the *enrollment record* for t and m. Subsequently, the client can interact with the server and check that (t, m) is stored in T. We call this phase the *validation phase* and the client outputs either `accept` or `reject`.

The security properties of PO-COM follow the ones of commitment schemes. That is, the enrollment record T is partially hiding, meaning that T and also the interactive protocol to compute T hide all information about m, but t may be leaked. Moreover, PO-COM is binding in the sense that it is (computationally) hard to a find a second pair (t', m') such that the client outputs `accept` in the validation phase.

3.1 Definition of PO-COM

To define partially oblivious commitments formally, we introduce the following notations for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. Let $n \in \mathbb{N}$ denote the security parameter and X be a set. By $x \leftarrow_{\$} X$ we denote the uniform drawing of a random element x from set X. Unless stated otherwise, all algorithms run in probabilistic polynomial time (PPT). For an algorithm $\mathcal{A}$, let $x \leftarrow_{\$} \mathcal{A}(y)$ denote the event that $\mathcal{A}$ on input y outputs x. If $\mathcal{A}$ is a deterministic algorithm we write instead $x \leftarrow \mathcal{A}(y)$. For two PPT algorithms $\mathcal{X}, \mathcal{Y}$ we denote by $(a, b) \leftarrow_{\$} \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$ the event that the joint execution of $\mathcal{X}$ on input x and $\mathcal{Y}$ on input y produces local output a for $\mathcal{X}$ and b for $\mathcal{Y}$, respectively. If there is only one output, then it is assumed to be for $\mathcal{X}$. We denote with $\mathcal{M}$ the space of messages and with $\mathcal{T}$ the space of tweaks. We write $\mathcal{Y}^{\langle \mathcal{X}(x), \cdot \rangle}(y)$ if $\mathcal{Y}$ can invoke an unbounded number of executions of the interactive protocol with $\mathcal{X}$ in arbitrarily interleaved order.

Definition 1 (Partially Oblivious Commitments)**.** A partially oblivious commitment scheme consists of a tuple of efficient algorithms $\Pi = (\mathsf{Setup}, \langle \mathcal{C}, \mathcal{S} \rangle_{\texttt{enrl}}, \langle \mathcal{C}, \mathcal{S} \rangle_{\texttt{val}})$, in three phases:

Setup Phase. On input the security parameter, $\mathsf{Setup}(1^n)$ outputs a client secret csk as well as a pair (pk, sk). The client is provided with public key pk and client secret csk, while the server gets secret key sk.

Enrollment Phase. The joint execution $\langle \mathcal{C}(csk, pk, t, m), \mathcal{S}(sk) \rangle_{\texttt{enrl}}$ for a message $m \in \mathcal{M}$ and tweak $t \in \mathcal{T}$ generates an enrollment record T (the commitment) for the client.

Validation Phase. The joint execution $\langle \mathcal{C}(csk, pk, T, t, m), \mathcal{S}(sk) \rangle_{\texttt{val}}$ for an enrollment record T, a message $m \in \mathcal{M}$, and tweak $t \in \mathcal{T}$ generates a binary decision $\{\texttt{accept}, \texttt{reject}\}$ for the client.

A partially oblivious commitment scheme Π is *correct* if for all $n \in \mathbb{N}$, all keys $(csk, pk, sk) \leftarrow_{\$} \mathsf{Setup}(1^n)$, all pairs of messages and tweaks (t, m), and all $T \leftarrow_{\$} \langle \mathcal{C}(csk, pk, t, m), \mathcal{S}(sk) \rangle_{\texttt{enrl}}$, the probability that $\langle \mathcal{C}(csk, pk, T, t, m), \mathcal{S}(sk) \rangle_{\texttt{val}}$ outputs `accept` is 1.

Definition 2 (Key Rotation)**.** A PO-COM Π has *key rotation* if there exists a key rotation protocol between client and server such that the joint execution $\langle \mathcal{C}(csk, pk, T), \mathcal{S}(sk) \rangle_{\texttt{rot}}$ for an enrollment record T generates an updated record T', a new public key pk', and client secret csk' for the client as well as an updated secret key sk' for the server.

Key rotation is *correct*, if for all $(csk, pk, sk) \leftarrow_\$ \mathsf{Setup}(1^n)$ and tweak-message pairs (t, m) it holds that for an enrollment record $T \leftarrow_\$ \langle \mathcal{C}(csk, pk, t, m), \mathcal{S}(sk)\rangle_{\texttt{enrl}}$ after key rotation $((csk', pk', T'), sk') \leftarrow_\$ \langle \mathcal{C}(csk, pk, T), \mathcal{S}(sk)\rangle_{\texttt{rot}}$ the updated T' validates with the new key, i.e., $\texttt{accept} \leftarrow_\$ \langle \mathcal{C}(csk', pk', T', t, m), \mathcal{S}(sk')\rangle_{\texttt{val}}$.

3.2 Security of PO-COM

A partially oblivious commitment protocol allows to prove that a tweak-message pair (t, m) is contained in a jointly generated enrollment record T, which leaks no information about m. The enrollment record T is verifiable, meaning that there exists an interactive protocol to check whether (t, m) is contained in T. Based on the applications we have in mind, we derive the following security goals.

HIDING. In all interactions between both parties, the tweak is revealed to the server, while the message should stay *private to the client.*

BINDING. The enrollment record must be *binding*, which means that given some enrollment record T an attacker should not be able to produce a different, valid input pair other than the originally enrolled pair.

OBLIVIOUSNESS. Given only an enrollment record, an attacker should be oblivious about m, meaning that he should not able to *verify a guess* for the input message that was used to create the record.

In the following, we formalize these requirements in terms of cryptographic games.

3.2.1 Partial Hiding

Our notion of partial hiding says that the enrollment record T hides all information about the message m, but may leak information about the tweak t. This security notion is useful in settings like password-based authentication, where input t, the username, is revealed to the server, while input m, the password, should stay hidden.

Definition 3 (Partial Hiding)**.** A partially oblivious commitment protocol $\Pi = (\mathsf{Setup}, \langle \mathcal{C}, \mathcal{S}\rangle_{\texttt{enrl}}, \langle \mathcal{C}, \mathcal{S}\rangle_{\texttt{val}})$ is *partially hiding* if for any two-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(n)$ such that

$$\left|\Pr\left[\mathsf{Hiding}^0_{\Pi,\mathcal{A}}(1^n) = 1\right] - \Pr\left[\mathsf{Hiding}^1_{\Pi,\mathcal{A}}(1^n) = 1\right]\right| \le \mathsf{negl}(n)\,,$$

where the randomness is taken over the random coins of the experiments and the adversary. The two experiments are defined as follows:

```
$\mathsf{Hiding}^b_{\Pi,\mathcal{A}}(1^n)$
$(csk, pk, sk, t, m_0, m_1, T, \mathsf{st}) \leftarrow_\$ \mathcal{A}_1(1^n)$
$b' \leftarrow_\$ \mathcal{A}_2^{\langle \mathcal{C}(csk,pk,t,m_b),\cdot\rangle_{\texttt{enrl}},\langle \mathcal{C}(csk,pk,T,t,m_b),\cdot\rangle_{\texttt{val}}}(\mathsf{st})$
return $(b = b')$
```

3.2.2 Binding

A partially oblivious commitment protocol must be binding in the sense that it should be computationally hard to find two distinct pairs $(t, m), (t', m')$ for a record T, such that the validation protocol outputs $\texttt{accept}$ for both pairs. In the context of password-based authentication, this property ensures that the validation protocol will fail with overwhelming probability if the pair (t, m) does not have a matching enrollment record T generated by the enrollment protocol $T \leftarrow_\$ \langle \mathcal{C}(csk, pk, t, m), \mathcal{S}(sk)\rangle_{\texttt{enrl}}$.

Definition 4 (Binding)**.** A partially oblivious commitment protocol $\Pi = (\mathsf{Setup}, \langle \mathcal{C}, \mathcal{S}\rangle_{\texttt{enrl}}, \langle \mathcal{C}, \mathcal{S}\rangle_{\texttt{val}})$ is *binding* if for any PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(n)$ such that

$$\Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1\right] \le \mathsf{negl}(n)\,,$$

where the randomness is taken over the random coins of the experiment and the adversary. The experiment is defined as follows:

```
$\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n)$
$(csk, pk, sk) \leftarrow_\$ \mathsf{Setup}(1^n)$
$((t_0, m_0), (t_1, m_1), T) \leftarrow_\$ \mathcal{A}^{\langle \cdot, \mathcal{S}(sk)\rangle_{\texttt{enrl}}, \langle \cdot, \mathcal{S}(sk)\rangle_{\texttt{val}}}(csk, pk)$
$d_0 \leftarrow_\$ \langle \mathcal{C}(csk, pk, T, t_0, m_0), \mathcal{S}(sk)\rangle_{\texttt{val}}$
$d_1 \leftarrow_\$ \langle \mathcal{C}(csk, pk, T, t_1, m_1), \mathcal{S}(sk)\rangle_{\texttt{val}}$
if $(t_0, m_0) \neq (t_1, m_1)$ and $d_0 = d_1 = \texttt{accept}$
  return 1
else return 0
```

3.2.3 Obliviousness

An enrollment record T is oblivious in the sense that the adversary in unable to verify a guess for (t, m) without running the validation protocol with the server. We formalize this intuition in a game where the adversary chooses two messages, obtains an enrollment record T and can only guess which of the two messages is stored in T. In our application, this property prevents offline brute-force attacks at reproducing the values t and m from their enrollment record T.

Definition 5 (Obliviousness)**.** A partially oblivious commitment protocol $\Pi = (\mathsf{Setup}, \langle \mathcal{C}, \mathcal{S}\rangle_{\texttt{enrl}}, \langle \mathcal{C}, \mathcal{S}\rangle_{\texttt{val}})$ is *oblivious* if for any two-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{negl}(n)$ such that

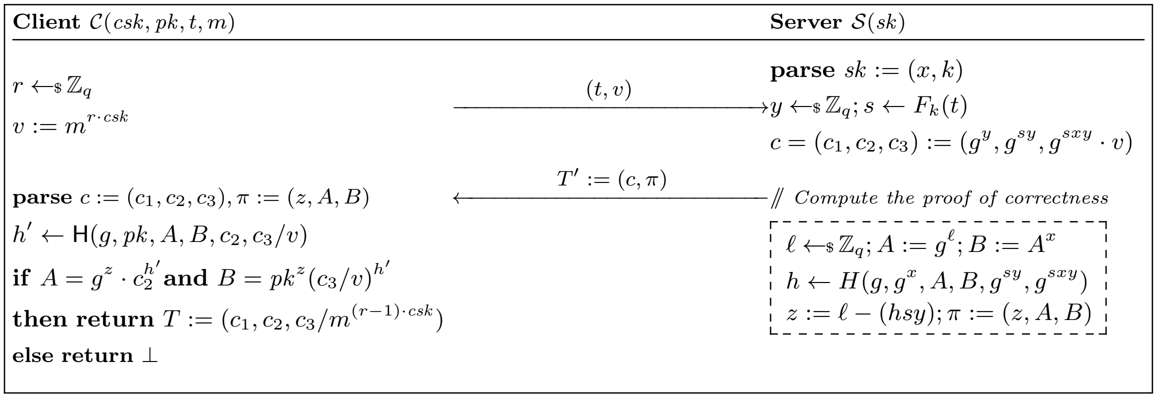
$$\left|\Pr\left[\mathsf{Obliv}^0_{\Pi,\mathcal{A}}(1^n) = 1\right] - \Pr\left[\mathsf{Obliv}^1_{\Pi,\mathcal{A}}(1^n) = 1\right]\right| \le \mathsf{negl}(n)\,,$$

where the randomness is taken over the random coins of the experiments and the adversary. The two experiments are defined as follows:

```
$\mathsf{Obliv}^b_{\Pi,\mathcal{A}}(1^n)$
$(csk, pk, sk) \leftarrow_\$ \mathsf{Setup}(1^n)$
$(t, m_0, m_1, \mathsf{st}) \leftarrow_\$ \mathcal{A}_1^{\langle \cdot, \mathcal{S}(sk)\rangle_{\texttt{enrl}}, \langle \cdot, \mathcal{S}(sk)\rangle_{\texttt{val}}}(csk, pk)$
$T \leftarrow_\$ \langle \mathcal{C}(csk, pk, t, m_b), \mathcal{S}(sk)\rangle_{\texttt{enrl}}$
$b' \leftarrow_\$ \mathcal{A}_2^{\langle \cdot, \mathcal{S}(sk)\rangle_{\texttt{enrl}}}(\mathsf{st}, T)$
return $(b = b')$
```

3.2.4 Secure Key Rotation

Key rotation should render an old client state useless to the adversary and further an old server secret key should not help in recovering information from an updated client database. All security properties must hold in the presence of key- rotations. Please refer to the full version of this paper for a formal definition of secure key-rotation and further discussions about hiding, binding, and obliviousness in the presence of key-rotations.



Client $\mathcal{C}(csk, pk, t, m)$

$r \leftarrow_{\$} \mathbb{Z}_q$

$v := m^{r \cdot csk}$

$\xrightarrow{(t, v)}$

$\xleftarrow{T' := (c, \pi)}$

parse $c := (c_1, c_2, c_3), \pi := (z, A, B)$

$h' \leftarrow \mathsf{H}(g, pk, A, B, c_2, c_3/v)$

if $A = g^z \cdot c_2^{h'}$ **and** $B = pk^z (c_3/v)^{h'}$

then return $T := (c_1, c_2, c_3/m^{(r-1) \cdot csk})$

else return $\perp$

Server $\mathcal{S}(sk)$

parse $sk := (x, k)$

$y \leftarrow_{\$} \mathbb{Z}_q; s \leftarrow F_k(t)$

$c = (c_1, c_2, c_3) := (g^y, g^{sy}, g^{sxy} \cdot v)$

// Compute the proof of correctness

$\ell \leftarrow_{\$} \mathbb{Z}_q; A := g^\ell; B := A^x$

$h \leftarrow H(g, g^x, A, B, g^{sy}, g^{sxy})$

$z := \ell - (hsy); \pi := (z, A, B)$

Figure 1: The enrollment protocol.

4. CONSTRUCTION OF PO-COM

In this section we present our construction of partially oblivious commitments. Due to space constraints, we omit the definition of pseudorandom function, non-interactive zero-knowledge proofs, and the definition of the decisional Diffie-Hellman (DDH) problem.

4.1 Intuition

The core part of our construction can be seen as a "twin key" variant of the Pedersen commitment scheme [36]. To illustrate this idea, let x be a secret key, g^x the corresponding public key, and k the key of a pseudorandom function F. To commit to a tweak t and a message m, the client sends a blinded version (t, m^r) to the server. The server derives a twin-key via the PRF as $s \leftarrow_{\$} F_k(t)$ and computes the enrollment record T (the commitment) choosing a random value y and using both keys s and x. That is, it computes T as $T = (g^y, g^{sy}, g^{sxy} \cdot m^r)$. Computing the twin key s via a PRF ensures that the record depends deterministically on the non-hidden tweak t and the PRF property ensures that the output is computationally indistinguishable from random. To enable client specific secure key rotation, each client keeps its own client secret csk, which is used to shift the message m.

In the validation phase, we perform two checks against a previously generated enrollment record T. First, we check whether the twin key was correctly generated from the tweak. If this is not the case, we abort, because it means the server is not following the protocol. The second check concerns the validity of the message compared to the one committed to in the record at hand.

Because of the randomization factor y, which is unique to every run of the enrollment and validation protocols, two enrollment records for the same message appear indistinguishable, unless one knows the enrolled message. This prevents an adversary from verifying message guesses by repeatedly modifying a given record and comparing against the server response for the guessed message.

4.2 Our Scheme

Construction 1. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q, and let g be a generator of $\mathbb{G}$. Let $F : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q$ be a pseudorandom function and let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function.

Setup Phase. The setup algorithms $\mathsf{Setup}(1^n)$ chooses random values $x, z \leftarrow_{\$} \mathbb{Z}_q$ and a random key $k \leftarrow_{\$} \{0,1\}^n$. It returns $(csk, pk, sk) := (z, g^x, (x, k))$.

Enrollment Phase. The full enrollment protocol is described in Figure 1 and consists of the following steps: The client chooses a random blinding factor $r \leftarrow_{\$} \mathbb{Z}_q$ and sends as its first message the blinded message $v = m^{r \cdot csk}$ and tweak t . Once the server receives (t, v) it computes the twin key $s \leftarrow F_k(t)$ corresponding to t and the preliminary record $T' = ((c_1, c_2, c_3), \pi)$ consisting of $(g^y, g^{sy}, g^{sxy} \cdot v)$, for some randomly chosen value $y \in \mathbb{Z}_q$, and a proof of correctness π. It sends the preliminary record T' to the client. The client parses the server's response and checks the proof of correctness. If the proof verifies, the client outputs the record $T := (c_1, c_2, c_2/m^{(r-1) \cdot csk})$.

Validation Phase. The full validation protocol is described in Figure 2 and consists of the following steps: The client parses the stored enrollment record $T = (T_1, T_2, T_3)$ and chooses a random blinding factor $r \leftarrow_{\$} \mathbb{Z}_q$. It then sends the blinded message $v = m^{r \cdot csk}$ together with tweak t and the value T_1 to the server. Once the server receives (T_1, t, v), it computes the twin key $s \leftarrow F_k(t)$ corresponding to t, the tuple $c = (c_1, c_2, c_3) := (g^y, g^{sy}, g^{sxy} \cdot v)$ and two proofs: The first proof shows that the server has used the same twin key $s \leftarrow F_k(t)$ which was also used during enrollment. The second proof indicates whether the message m is equal to the committed message in T. The server sends the preliminary record c and the proofs to the client. The client parses the server's response and checks the proofs of correctness. If both proofs verify, the client outputs `accept`, otherwise `reject`.

Both proofs resemble a Fiat-Shamir transformed [20] version of a protocol for proving the equivalence of discrete logarithms first given by Chaum and Pedersen [15]. In order to see why the scheme is correct, consider the case that we perform a validation run for the pair (t, m) and a record T originally generated for the pair (t, m').

The first check guarantees that

$$A_1 = T_1^{z_1} \cdot T_2^{h'_1} \quad \text{i.e.,} \quad T_1^{\ell_1} = T_1^{\ell_1 - h_1 s} \cdot T_1^{h'_1 s}$$

and that

$$B_1 = c_1^{z_1} \cdot c_2^{h'_1} \quad \text{i.e.,} \quad g^{y \ell_1} = g^{y(\ell_1 - h_1 s)} \cdot g^{y h'_1 s}.$$

Thus, both equations are fulfilled if $h_1 = h'_1$ implying that the server used the correct twin key s.

```
Client $\mathcal{C}(csk, pk, T, t, m)$                                   Server $\mathcal{S}(sk)$

parse $T = (T_1, T_2, T_3)$                                               parse $sk := (x, k)$
$r \leftarrow\!\!\$\, \mathbb{Z}_q$                  $(T_1, t, v)$ $\longrightarrow$          $y \leftarrow\!\!\$\, \mathbb{Z}_q; s \leftarrow F_k(t)$
$v := m^{r \cdot csk}$                                                    $c = (c_1, c_2, c_3) := (g^y, g^{sy}, g^{sxy} \cdot v)$

parse $c := (c_1, c_2, c_3)$          $\longleftarrow$ $T' := (c, \pi_1, \pi_2)$      // Compute the proof of correctness
parse $\pi_1 := (z_1, A_1, B_1), \pi_2 := (z_2, A_2, B_2)$                  $\ell_1 \leftarrow\!\!\$\, \mathbb{Z}_q; A_1 := T_1^{\ell_1}; B_1 := g^{y\ell_1}$
$h_1' \leftarrow \mathsf{H}(T_1, c_1, A_1, B_1, T_2, c_2)$                          $\ell_2 \leftarrow\!\!\$\, \mathbb{Z}_q; A_2 := T_1^{\ell_2}; B_2 := g^{y\ell_2}$
$h_2' \leftarrow \mathsf{H}(T_1, c_1, A_2, B_2, T_3/m^{csk}, c_3/v)$                $h_1 \leftarrow \mathsf{H}(T_1, g^y, A_1, B_1, T_1^s, g^{sy})$
if $A_1 = T_1^{z_1} T_2^{h_1'}$ and $B_1 = c_1^{z_1} c_2^{h_1'}$                          $h_2 \leftarrow \mathsf{H}(T_1, g^y, A_2, B_2, T_1^{sx}, g^{sxy})$
  and $A_2 = T_1^{z_2} (T_3/m^{csk})^{h_2'}$ and $B_2 = c_1^{z_2} (c_3/v)^{h_2'}$      $z_1 := \ell_1 - h_1 s; z_2 := \ell_2 - h_2 sx$
then return accept                                                        $\pi_1 := (z_1, A_1, B_1); \pi_2 \leftarrow (z_2, A_2, B_2)$
else return reject
```

Figure 2: The validation protocol.

The second check determines whether the submitted message is the same as the enrolled message in T. In detail, it works as follows:

$$A_2 = T_1^{z_2} \cdot \left(\frac{T_3}{m^{csk}}\right)^{h_2'} \quad \text{i.e., } T_1^{\ell_2} = T_1^{\ell_2 - h_2 sx} \cdot T_1^{h_2' sx} \cdot \left(\frac{m'}{m^{csk}}\right)^{h_2'}$$

and

$$B_2 = c_1^{z_2} \cdot \left(\frac{c_3}{m^{r \cdot csk}}\right)^{h_2'} \quad \text{i.e., } g^{y\ell_2} = g^{y(\ell_2 - h_2 sx)} \cdot g^{h_2' ysx}$$

The first equation is fulfilled if $m' = m \cdot csk$, which is the case for a correct enrollment of m, and the second equation is fulfilled if the validation record was created using the correct public key.

Key Rotation. Key Rotation in our construction consists of one round of interaction where the server draws a fresh random secret key $x' \leftarrow\!\!\$\, \mathbb{Z}_q$ and provides the client with an update token $\delta = x'/x$ where x is the old secret key. The client updates each record $T = (T_1, T_2, T_3)$ of the database by raising the third element of each record to the power of δ, such that the new record is $T' = (T_1, T_2, T_3^{\delta})$. Further, the client updates the stored public key and the client secret: $pk' = pk^{\delta}$ and $csk' = csk \cdot \delta$.

We defer the proof of correctness of this key rotation mechanism to the full version of this paper.

4.3 Proof of Security

In this section, we show that our construction is partially-hiding, binding, and oblivious. We prove the stronger statement that Construction 1 is secure with any perfectly sound non-interactive zero knowledge (NIZK) proof system for equality of discrete logarithms. In our construction, this NIZK is then instantiated with the Fiat-Shamir transformed protocol of Chaum and Pedersen. More formally, we prove the following theorem:

Theorem 1. If the DDH problem is hard in $\mathbb{G}$, the non-interactive proof system is zero-knowledge and perfectly sound, and H is modeled as a random oracle, then Construction 1 is partially hiding, binding, and oblivious.

Informally, the scheme is hiding, because the message m is information-theoretically blinded with an exponent drawn independently and uniformly at random for every interaction with the server.

For binding, consider that a binding adversary has to find a "forgery" pair (t^*, m^*), which validates correctly for a record enrolled from a different pair. Such a forgery amounts to finding a collision for the PRF on t and t^*, which cannot be efficient if F is pseudorandom.

Obliviousness is guaranteed under the DDH-assumption. We use a game-hopping approach with a series of intermediate experiments, where we first replace every occurrence of the PRF with a truly random function with negligible loss in security since F is pseudorandom. Then, we can reduce the indistinguishability of records for two differing messages to the DDH assumption. In the following, we give formal proofs for each property.

Partial Hiding.

Consider the view of an adversary $\mathcal{A}$ in one of the partial hiding experiments. In the first phase of the game, the view of $\mathcal{A}$ consists solely of its own outputs. In the second phase, $\mathcal{A}$ engages in executions of the enrollment and validation protocols. When $\mathcal{A}$ engages in an execution of the enrollment protocol, it receives a tuple (t, v), where $v = m_b^r$ for some uniformly random $r \in \mathbb{Z}_q$ and t is some fixed part of $\mathcal{A}$'s challenge output. When $\mathcal{A}$ engages in an execution of the validation protocol, it receives a tuple (T_1, t, v), where $v = m_b^r$ for some uniformly random $r \in \mathbb{Z}_q$ and t and T_1 some fixed parts of $\mathcal{A}$'s challenge output.

By construction t and T_1 in all these executions are identical and independent of the message and thus independent of b. Further, since the r in all protocol runs are chosen independently and uniformly at random, all v seen by the attacker are uniformly distributed elements of $\mathbb{G}$ and thus independent of b.

We thus can conclude that the two games are perfectly indistinguishable to an (even unbounded) attacker. I.e.,

$$\left|\Pr\left[\mathsf{Hiding}^0_{\Pi,\mathcal{A}}(1^n) = 1\right] - \Pr\left[\mathsf{Hiding}^1_{\Pi,\mathcal{A}}(1^n) = 1\right]\right| = 0,$$

and Construction 1 is thus perfectly partially-hiding. □

Binding.

Let $\mathcal{A}$ be an adversary, such that $\Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1\right] = \epsilon$. Let $((t_0, m_0), (t_1, m_1), T)$ denote the output of $\mathcal{A}$. Further let y_0 and y_1 denote the random values chosen by server in the validation protocol executions for t_0, m_0 and t_1, m_1 respectively.

We can split the probability of $\mathcal{A}$ succeeding into two cases, namely depending on whether $t_0 = t_1$ or not. I.e., we have

$$\begin{aligned}
&\Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1\right] \\
\leq &\Pr[t_0 \neq t_1] \cdot \Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1 \,\middle|\, t_0 \neq t_1\right] \qquad (1)\\
&+ \Pr[t_0 = t_1] \cdot \Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1 \,\middle|\, t_0 = t_1\right]
\end{aligned}$$

Consider the case where $t_0 \neq t_1$. Let a be an element of $\mathbb{Z}_q$ such that $T_1^a = T_2$, i.e., let a be the discrete logarithm of T_2 relative to base T_1. Observe, that the first zero-knowledge proof in the validation protocol proves that $g^{y_b a} = g^{y_b F_k(t_b)}$ for $b \in \{0, 1\}$. Therefore, by the perfect soundness of the NIZK, we have that $F_k(t_0) = a = F_k(t_1)$. I.e., the adversary has found a collision in the pseudorandom function. From this we can construct a distinguisher $\mathcal{D}$ against the security of the pseudorandom function as follows:

The distinguisher $\mathcal{D}$ gets as input the security parameter and access to an oracle that is either the PRF with a uniformly chosen key k or a uniformly random function. It then samples a random exponent $x \leftarrow_\$ \mathbb{Z}_q$ and invokes $\mathcal{A}$ on input $pk = g^x$. $\mathcal{A}$ may now invoke arbitrary enrollment and validation protocol executions. To simulate these, $\mathcal{D}$ honestly simulates the server as specified in the protocol, except that it uses its oracle to compute the values s.

Eventually, $\mathcal{A}$ outputs $((t_0, m_0), (t_1, m_1), T)$. If $t_0 = t_1$, the distinguisher outputs a random bit $b \leftarrow_\$ \{0, 1\}$. Otherwise, it queries t_0 and t_1 to its function oracle, receiving responses s_0 and s_1 respectively. If $s_0 = s_1$, then $\mathcal{D}$ outputs 1, otherwise 0.

We now analyze the advantage of $\mathcal{D}$. Consider first the case where the function oracle is the PRF F. In this case, $\mathcal{D}$ performs a perfect simulation of the binding challenger for $\mathcal{A}$. Hence, we get

$$\begin{aligned}
&\Pr\left[\mathcal{D}^{F_k(\cdot)}(1^n) = 1\right] \\
= \quad &\Pr[t_0 = t_1] \cdot \frac{1}{2} \\
+ \quad &\Pr[t_0 \neq t_1] \cdot \Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1 \,\middle|\, t_0 \neq t_1\right].
\end{aligned}$$

Now consider the case where the function oracle is a truly random function H. Since the adversary only makes a polynomial number of queries, the probability that it was able to find a collision in a truly random function is negligible. Hence, we get

$$\Pr\left[\mathcal{D}^{H(\cdot)}(1^n) = 1\right] = \Pr[t_0 = t_1] \cdot \frac{1}{2} + \Pr[t_0 \neq t_1] \cdot \mathsf{negl}(n).$$

We thus get

$$\begin{aligned}
&\left|\Pr\left[\mathcal{A}^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[\mathcal{A}^{H(\cdot)}(1^n) = 1\right]\right| \leq \mathsf{negl}(n) \\
\Longrightarrow &\left| \begin{array}{c} \Pr[t_0 \neq t_1] \cdot \Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1 \,\middle|\, t_0 \neq t_1\right] \\ - \Pr[t_0 \neq t_1] \cdot \mathsf{negl}(n) \end{array} \right| \leq \mathsf{negl}(n) \\
\Longrightarrow &\Pr[t_0 \neq t_1] \cdot \Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1 \,\middle|\, t_0 \neq t_1\right] \leq \mathsf{negl}(n) \qquad (2)
\end{aligned}$$

Now consider the case where $t_0 = t_1$. Let a_0, a_1 be elements of $\mathbb{Z}_q$ such that $T_1^{a_b} = T_3 / m_b^{csk}$, i.e., let a_b be the discrete logarithm of T_3 / m_b^{csk} relative to base T_1. Observe, that the second zero knowledge proof in the validation protocol proves that $g^{y_b a_b} = g^{y_b sk F_k(t_b)}$ for $b \in \{0, 1\}$. Therefore, by the perfect soundness of the NIZK, we have that $a_b = skF_k(t_b)$, however, since $t_0 = t_1$, this implies $a_0 = a_1$ and – by definition of a_b – it thus follows that $m_0 = m_b$.

Therefore, if $\mathcal{A}$ outputs $t_0 = t_1$, it never holds that $(t_0, m_0) \neq (t_1, m_1)$. Thus we can conclude that

$$\Pr[t_0 = t_1] \cdot \Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1 \,\middle|\, t_0 = t_1\right] = 0 \qquad (3)$$

and finally combining Equations (1) to (3), we get that

$$\Pr\left[\mathsf{Binding}_{\Pi,\mathcal{A}}(1^n) = 1\right] \leq \mathsf{negl}(n)$$

and Construction 1 is thus binding. □

Obliviousness.

Let $\mathcal{A}$ be an adversary, such that

$$\left|\mathsf{Obliv}^0_{\Pi,\mathcal{A}}(1^n) - \mathsf{Obliv}^1_{\Pi,\mathcal{A}}(1^n)\right| = \epsilon(n).$$

We will bound ϵ using a series of games.

Game 1. The first game is $\mathsf{Obliv}^0_{\Pi,\mathcal{A}}(1^n)$.

Game 2. The second game behaves exactly like the first game, except that the values s are no longer chosen via a PRF and are instead computed using a lazily sampled truly random function.

Game 3. The third game behaves exactly like the second game, except that all zero-knowledge proofs are now simulated.

Game 4. The fourth game behaves exactly like the third game, except that $b = 1$.

Game 5. The fifth game behaves exactly like the fourth game, except that zero-knowledge proofs are once again computed honestly.

Game 6. The sixth game is $\mathsf{Obliv}^1_{\Pi,\mathcal{A}}(1^n)$.

We will now bound the difference between each pair of consecutive games.

Let $|\Pr[\mathsf{Game}_1(1^n) = 1] - \Pr[\mathsf{Game}_2(1^n) = 1]| = \delta_1(n)$. Consider the distinguisher $\mathcal{D}$ against the pseudorandomness of F as follows: The distinguisher $\mathcal{D}$ gets as input the security parameter and access to an oracle that is either the PRF with a uniformly chosen key k or a uniformly random function. It then samples random elements $z, x \leftarrow_\$ \mathbb{Z}_q$ and invokes $\mathcal{A}_1$ on input $(csk = z, pk = g^x)$. $\mathcal{A}_1$ may now invoke arbitrary enrollment and validation protocol executions. To simulate these, $\mathcal{D}$ honestly simulates the server as specified in the protocol, except that it uses its oracle to compute the values s. Once $\mathcal{A}_1$ outputs $(t, m_0, m_1, \mathsf{st})$, $\mathcal{D}$ once again honestly computes $T \leftarrow_\$ \langle \mathcal{C}(csk, pk, t, m_b), \mathcal{S}(sk) \rangle_{\mathtt{enrl}}$ computing s using its oracle. $\mathcal{D}$ then invokes $\mathcal{A}_2$ on st and T, simulating the enrollment protocol executions as before. Eventually, $\mathcal{A}_2$ outputs a bit b' and $\mathcal{D}$ outputs the same bit.

It should be clear, that if the function oracle is the PRF F, then $\mathcal{D}$ perfectly simulates Game_1, while with a truly random oracle, $\mathcal{D}$ provides a perfect simulation of Game_2. Therefore, by assumption that F is a pseudorandom function, we get that

$$\delta_1(n) = \left|\Pr\left[\mathcal{D}^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[\mathcal{D}^{H(\cdot)}(1^n) = 1\right]\right| \leq \mathsf{negl}(n).$$

Let $|\Pr[\mathsf{Game}_2(1^n) = 1] - \Pr[\mathsf{Game}_3(1^n) = 1]| = \delta_2(n)$. Consider the reduction $\mathcal{B}$ against the zero-knowledge property of the NIZK. Let $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ be the simulator of the non-interactive zero-knowledge proof system. The reduction $\mathcal{B}$ gets as input the common reference string and access to an oracle that is either the Prove algorithm, or the simulator Sim'. It then samples random elements $z, x \leftarrow_{\$} \mathbb{Z}_q$ and invokes $\mathcal{A}_1$ on input $(csk = z, pk = g^x)$. $\mathcal{A}_1$ may now invoke arbitrary enrollment and validation protocol executions. To simulate these, $\mathcal{B}$ honestly simulates the server as specified in the protocol, except that it computes s using a lazily sampled truly random function and computes all proofs π, π_1 and π_2 by querying statement and witness to the oracle. Once, $\mathcal{A}_1$ outputs $(t, m_0, m_1, \mathsf{st})$, $\mathcal{B}$ once again honestly computes $T \leftarrow_{\$} \langle \mathcal{C}(csk, pk, t, m_b), \mathcal{S}(sk)\rangle_{\mathtt{enrl}}$ computing s using its lazily sampled truly random function and computing π by querying its oracle. $\mathcal{B}$ then invokes $\mathcal{A}_2$ on st and T, simulating the enrollment protocol executions as before. Eventually $\mathcal{A}_2$ outputs a bit b' and $\mathcal{B}$ outputs the same bit.

It should be clear, that if the oracle is the Prove algorithm, then $\mathcal{B}$ perfectly simulates Game_2. And if the oracle is in fact the simulator, then $\mathcal{B}$ provides a perfect simulation of Game_3. Therefore, by assumption that the NIZK is zero knowledge, we get that

$$\delta_2(n) = \left| \begin{array}{l} \Pr\left[\sigma \leftarrow \mathsf{ZKSetup}(1^n) : \mathcal{B}^{\mathsf{Prove}(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \\ - \Pr\left[(\sigma,\tau) \leftarrow \mathsf{Sim}_0(1^n) : \mathcal{B}^{\mathsf{Sim}'(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right] \end{array} \right| \leq \mathsf{negl}(n)\,.$$

Let $|\Pr[\mathsf{Game}_3(1^n) = 1] - \Pr[\mathsf{Game}_4(1^n) = 1]| = \delta_3(n)$. Consider the reduction $\mathcal{D}$ against the hardness of DDH. The reduction $\mathcal{D}$ gets as input a tuple $(\mathbb{G}, g, A, B, C)$ that is either a DDH tuple, or a random tuple. It draws a random element $z \leftarrow_{\$} \mathbb{Z}_q$ and invokes $\mathcal{A}_1$ on input $(csk = z, pk = B)$. $\mathcal{A}_1$ may now invoke arbitrary enrollment and validation protocol executions. In the protocol executions $\mathcal{D}$ follows the description of Game_3, except that c is computed as $(c_1, c_2, c_3) := g^y, g^{ys}, B^{ys} \cdot v$. Once, $\mathcal{A}_1$ outputs $(t, m_0, m_1, \mathsf{st})$, $\mathcal{D}$ samples a bit $b \leftarrow_{\$} \{0, 1\}$ at random, computes the s for t according to its lazily sampled random function, and computes $T = (T_1, T_2, T_3) := (A^{s^{-1}}, A, C \cdot m_b^{csk})$. $\mathcal{D}$ then invokes $\mathcal{A}_2$ on st and T, simulating the enrollment protocol executions as before. Eventually $\mathcal{A}_2$ outputs a bit b' and $\mathcal{D}$ outputs 1, if $b = b'$ and outputs a random bit otherwise.

Now consider the two cases. If (A, B, C) is a DDH tuple, then $\mathcal{D}$ perfectly simulates Game_3 if $b = 0$ and Game_4 if $b = 1$. It is easy to see that answers to the protocol invocations are well formed. But also the challenge is well formed. To see this, consider that we have $(A, B, C) = (g^a, g^b, g^{ab})$ and therefore $(g^{as^{-1}}, g^a, g^{ab} \cdot m_b^{csk})$ is a well-formed enrollment record for $y = as^{-1}$, which is a uniform random element. Therefore, $\Pr_{a,b}\left[\mathcal{D}(\mathbb{G}, g, g^a, g^b, g^{ab}) = 1\right] = \frac{1}{2} + \delta_3$.

Now consider the other case. If (A, B, C) is a random tuple, then the challenge enrollment record contains no information about the bit b, since m_b is blinded by a uniformly distributed element C. Therefore, $\Pr_{a,b,c}\left[\mathcal{D}(\mathbb{G}, g, g^a, g^b, g^c) = 1\right] = \frac{1}{2}$, and thus by the DDH assumption in $\mathbb{G}$,

$$\delta_3(n) = \left| \begin{array}{l} \Pr_{a,b}\left[\mathcal{D}(\mathbb{G}, g, g^a, g^b, g^{ab}) = 1\right] \\ - \Pr_{a,b,c}\left[\mathcal{D}(\mathbb{G}, g, g^a, g^b, g^c) = 1\right] \end{array} \right| \leq \mathsf{negl}(n)\,.$$

The fifth game simply reverts the changes introduced in game three and an almost identical reduction lets us bound δ_4 by the assumption that the NIZK is zero knowledge. Similarly, it is easy to see that the sixth game simply reverts the changes introduced in game two and an almost identical reduction lets us bound δ_5 by the assumption that F is a pseudorandom function. Finally, we conclude that

$$\epsilon(n) \leq \sum_{i=1}^{5} \delta_i(n) \leq \mathsf{negl}(n)\,,$$

and Construction 1 is thus oblivious. □

Secure Key Rotation.
We defer the proof of secure key rotation to the full version of this paper for brevity. Intuitively, the proof follows from the fact that each key rotation unconditionally hides the previous key material unless the initial client state is known.

4.4 Password Authentication Using Our Construction

In this section, we show how to build a password authentication system that enjoys all functional properties described in Section 2.

Setup. The client and the PO-COM-Service run the setup of the PO-COM protocol meaning that the client receives a public key and unique client secret of the PO-COM protocol and can now make enrollment and validation queries to the PO-COM-Service.

Enrollment. The user provides her `password` and `username` to the client. The client sets $t =$ `username` and $m =$ `password` and makes an enrollment query for (t, m), storing the resulting enrollment record T alongside t.

Password verification. If a user contacts the client providing a tentative `password'` for a `username`, the client sets again $t =$ `username` and $m =$ `password`$'$ and performs a validation query for the enrollment record stored for t and (t, m). If validation is successful, `password`$'$ was the correct password for `username`.

Key Rotation. To perform key rotation, each client executes the key rotation protocol with the server to obtain a new public key and an updated unique client secret.

Stretching. Orthogonal to the functionality of our construction, one could introduce an additional step of password stretching on the client side by not enrolling the message itself, but the result of a key derivation function such as PBKDF2 [28] or Argon2 [7]. Intuitively this step could further inhibit offline-brute force attacks.

5. EVALUATION

We have implemented a prototype of our construction using the Charm framework [1] for cryptographic prototyping and using NIST curve P-256 as the base group for our construction. This group choice reflects a security level of approximately 128 bits. The average runtime across 10,000 iteration of all the common group operations on our server machine is given on the right side of Table 1.

The server is implemented as an HTTPS web service using the falcon framework in Python and served via a standard configuration of the nginx web server.

Latency. Latency test were performed with an amazon web services t2.micro instance on the client side and an amazon web services c4.2xlarge instance on the server side. The client instance has access to 1 CPU core (Intel Xeon) and 1 GB of RAM. The server instance has access to 8 CPU cores (Intel Xeon) and 15 GB of RAM.

	Local		WAN		EC Operations(μs)	
HTTP-Keepalive	On	Off	On	Off		
Enrollment	7.28	11.70	24.54	64.62	Sampling	56.8
Validation	9.51	13.80	21.56	67.00	Group op.	3.2
Validation Failure	9.01	13.47	23.96	68.96	Mod. Exp.	191.1
Pythia-Eval	13.79	17.24	34.25	76.91	Hash to $\mathbb{G}$	65.4
Network RTT	0.338		6.656			

Table 1: Average latency in milliseconds of different client requests for 1000 requests each and average runtime in μs of different group operations.

For comparison purposes, we set up an instance of Pythia using the code provided at [18] and ran performance measurements using the same web server setup.

The results of the latency tests appear on the left side of Table 1 and show client latency for LAN and WAN settings as the average latency of 1000 client requests for an enrollment or a validation request. We provide measurements for HTTP-Keep-alive turned on and turned off.

In the WAN setting, which resembles a multi-tenant setup as envisioned in [19], we can achieve latency which is at least 1.4 times lower than that of Pythia if HTTP-Keep-alive is active. Given the additional overhead without HTTP-Keep-alive we achieve a speedup by a factor of 1.1.

In the LAN setting, which is similar to most local enterprise networks, our implementation outperforms the Pythia prototype with 1.4 times lower latency in the case of successful validation and even better 1.9 times lower latency for enrollment.

Throughput. We have used the automated HTTP web server load testing tool `autobench` to perform throughput testing for our test setup. We use a static web page served using the same web server as the baseline comparison for throughput measurements of all the protocol operations the web service offers For the static web page, the maximum connection rate is at 6050 connections per second (cps), parameter retrieval is similarly high at 6000 cps. For validation queries, the reply rate levels off at around 1100 cps while enrollment can withstand request rates up to 1900 cps.

Storage. We used our setup to test the typical storage requirements of our password-authentication scheme both on client side and server side. To this end, we performed 100,000 enrollment queries for randomly chosen tweaks and passwords to generate 100,000 enrollment records on the client side and 100,000 time-stamped tweaks on the server side. On both sides, the information is stored in an indexed mongoDB instance.

Server-side storage For rate-limiting purposes, the server stores a 32-byte hash of any tweak which is queried together with a 32-byte timestamp. The average size of one such entry in the database is 83 bytes, with the additional overhead resulting from the database index structure.

Client-side storage On the client side, the enrollment records are stored together with a 32-byte hash of the given tweak. The average size of one database entry is approximately 444 bytes bringing the total size of our record database of 100,000 successful enrollments to a mere 43MB. Extrapolating linearly suggests that a database of 20 million enrollment records could fit within 8.5GB of memory.

6. REVISITING PO-PRFS

Partially oblivious pseudorandom functions (PO-PRFs) allow the computation of pseudorandom function values in a protocol between two parties, such that one party holds the function key and the other party holds the function inputs. After the execution of the protocol, the party which has provided the inputs learns the function output. Furthermore, the evaluation is (partially) oblivious, meaning that the respective inputs of both parties stay hidden to the other party, except one part of the function inputs which is known to both parties. The possibility to partially reveal the function input separates the primitive from previous work on fully oblivious PRFs [33, 24], where no information on the inputs may be revealed.

In [19], the authors left open the question if the security of their scheme can be based on weaker assumption. In this section, we answer this question negatively.

6.1 Preliminaries

Partially Oblivious PRFs. We recall the formal definition of partially oblivious pseudorandom functions given in [19].

Definition 6 (Partially Oblivious Pseudorandom Functions)**.** A partially oblivious PRF protocol (PO-PRF) is a tuple of three algorithms $(\mathsf{KeyGen}, \mathsf{Client}, \mathsf{Server})$ and a keyed function $f_k : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ which behave as follows:

KeyGen. The key generation algorithm outputs a public and secret key pair (pk, sk).

Server. The server algorithm takes as input the secret key sk and a client request bit string. It outputs a response bit string.

Client. The client algorithm takes as input a tweak t and a message m. It can perform a single invocation of the server algorithm before outputting a value x.

A PO-PRF protocol is said to be *correct* if for all security parameters n and all key-pairs $(pk, sk) \leftarrow\!\!\$\, \mathsf{KeyGen}(1^n)$, the protocol $\langle \mathsf{Client}(t, m), \mathsf{Server}(sk)\rangle$ always results in client-side output $f_{sk}(t, m)$.

The primitive supports verifiability of outputs, i.e., the client is able to verify the correctness of the server computation. Additionally the server does not learn the clients input m. Leakage of the tweak t, however, is allowed.

One-More Unpredictability. We recall the formal security property of one-more unpredictability [19]. On a high-level, one-more unpredictability says that an adversary cannot predict the value of the function f associated with the scheme on new inputs, except with negligible probability. This is formalized in the following experiment.

Definition 7. A PO-PRF $\Pi = (\mathsf{KeyGen}, \mathsf{Client}, \mathsf{Server})$ is called *one-more unpredictable* if for any PPT adversary $\mathcal{A}$ there exists a negligible function such that

$$\Pr\left[\mathsf{Unpredictability}_{\Pi,\mathcal{A}}(1^n) = 1\right] \le \mathsf{negl}(n)\,,$$

where the randomness is taken over the random coins of the experiment and the adversary and the experiment is defined as follows:

```
$\mathsf{Unpredictability}_{\Pi,\mathcal{A}}(1^n)$
$(sk, pk) \leftarrow_\$ \mathsf{KeyGen}(1^n); c \leftarrow 0$
$((t_1, m_1, T_1), \ldots, (t_q, m_q, T_q)) \leftarrow_\$ \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(pk)$
if $q > c$ and $T_i = f_{sk}(t_i, m_i)$ for all $i \in \{1, \ldots, q\}$
    return 1
else return 0
```

where $\mathcal{O}(t, X)$ responds with $\mathsf{Server}(sk, t, X)$ and sets $c \leftarrow c + 1$.

The authors also defined a stronger property similar to one-more unpredictability, but where the outputs must be pseudorandom. Since we are interested in proving an impossibility result it is sufficient to only consider one-more unpredictability as this property is strictly weaker and thus our impossibility result is strictly stronger.

Hard Non-interactive Problem. We recall the definition of a hard non-interactive problem following [22].

Definition 8 (Hard Non-interactive Problem)**.** A non-interactive problem $P = (I, V)$ consists of two efficient algorithms:

Instance Generation $I(1^n)$. The instance generation algorithm takes as input the security parameter 1^n and outputs an instance $\mathbf{x}$.

Instance Verification $V(\mathbf{x}, \mathbf{y})$. The instance verification algorithm takes as input a value $\mathbf{y}$ as well as an instance $\mathbf{x}$ of a cryptographic problem, and outputs a decision bit.

We say that a problem is *hard* if for any PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ solves the problem, i.e. on input $\mathbf{x} \leftarrow_\$ I(1^n)$ outputs $\mathbf{y}$ such that with overwhelming probability $V(\mathbf{x}, \mathbf{y}) = 1$ is negligible.

6.2 The Pythia Protocol

In this section, we recall Pythia, the PO-PRF protocol given in [19]. We consider the definition of *one-more unpredictability* and show that there cannot be a black-box reduction of this property to any non-interactive assumption. To prove this formally, we use meta-reduction techniques as put forward in e.g., [35, 8, 22]. Meta-reduction techniques can be seen as building a "reduction against the reduction". Before showing our meta-reduction and explaining the main ideas, we recall the basics of bilinear maps and the PO-PRF from [19].

Bilinear Setting. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of order q and let $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ be generators, such that there is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and it holds $e(g_1^\alpha, g_2^\beta) = e(g_1, g_2)^{\alpha\cdot\beta}$ for all $\alpha, \beta \in \mathbb{Z}_q$. Furthermore, e is non-degenerate meaning that $e(g_1, g_2) \neq 1$ and $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

The Pythia PO-PRF. To describe the Pythia PO-PRF, we first recall the underlying language to verify the computations of the server. Let L_{Pythia} describe the following language of valid server responses:

$$L_{\text{Pythia}} = \{(g, h, x, y) \mid \exists sk.g^{sk} = h \wedge x^{sk} = y\}.$$

Based on this language, we describe the Pythia protocol.

Construction 2 (Pythia)**.** Let $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1, \mathsf{H}_2 : \{0,1\}^* \to \mathbb{G}_2$ be hash functions modeled as random oracles, with $\mathbb{G}_1, \mathbb{G}_2$ groups as described in the bilinear setting above and let $P = (\mathsf{P}, \mathsf{V})$ be a non-interactive zero-knowledge proof system for L_{Pythia}.

$\mathsf{Setup}(1^n)$. picks a random exponent $sk \leftarrow_\$ \mathbb{Z}_q$ and computes a public key $pk = g_1^{sk}$. The secret key sk is sent to the server, the public key to the client.

$\mathsf{Client}(pk, t, m)$. picks a random $r \leftarrow_\$ \mathbb{Z}_q$ and sends $(t, \mathsf{H}_2(m)^r)$ to the server. Upon response (y, π) from the server, the client verifies the proof π and if it is valid, outputs $y^{1/r}$.

$\mathsf{Server}(sk, t, x)$. computes $\hat{x} = e(\mathsf{H}_1(t), x)$ and $y = \hat{x}^{sk}$ as well as $\pi \leftarrow_\$ \mathsf{P}((g_1, pk, \hat{x}, y), sk)$. It sends (y, π) back to the client.

6.3 Impossibility Result

Our meta-reduction is similar to [22] and it shows that there does not exists (using black-box techniques) a reduction of the unpredictability of the protocol to any non-interactive problem. Note that the result from [22] applies to blind signature schemes and not to partially oblivious PRFs. For clarity and succinctness, we show the impossibility result for a version of the protocol which omits the non-interactive zero-knowledge proofs. However, the result also holds for the verifiable version presented above.

To show that no reduction to a non-interactive problem exists, we introduce a magic adversary, which has access to a computationally unbounded oracle Σ which allows the adversary to break the scheme. A reduction R using the magic adversary, could therefore solve the non-interactive problem. This alone does not lead to a contradiction, since the magic adversary has access to an unbounded oracle and can thus not be considered efficient. However, in our proof we show that this adversary can be simulated efficiently, proving that there cannot be an efficient reduction if the problem is indeed computationally difficult.

We construct the simulation of the adversary, the so-called meta-reduction M, in the same way as [22], by resetting the execution of the reduction at an appropriate point and thus fooling the reduction with the result of computations that have occurred before the reset.

Let Σ be an unbounded DLOG oracle such that on input two group elements (g, h), Σ returns x, such that $g^x = h$.

The magic adversary uses Σ to generate a valid output of the protocol by extracting the secret key from the public key and performing all computations locally.

Definition 9 (Magic Adversary)**.** The magic adversary is defined as follows: Choose messages $m_0, m_1 \leftarrow_\$ \{0,1\}^*$ and tweak $t \leftarrow_\$ \{0,1\}^*$ as well as randomness $r \leftarrow_\$ \{0,1\}^*$. Upon input pk from the challenger, compute $x = H_2(m_0)^r$ and send it along with tweak t to the challenger to obtain response y. Now invoke $\Sigma(g, pk)$ to obtain the secret key sk. Finally, compute $z_1 = e(t, H_2(m_1))^{sk}$ and $z_0 = y^{1/r}$ and output $(m_0, z_0), (m_1, z_1)$.

Now for the main result.

Theorem 2. The unpredictability of Pythia cannot be reduced to a non-interactive problem.

Proof. Assume R was a reduction of the unpredictability of Pythia to some non-interactive problem $P = (I, V)$.

Upon input a problem instance $\mathbf{x}$, the meta-reduction M forwards $\mathbf{x}$ to R to receive a public key pk generated by R. M draws uniformly and independently at random a tweak $t \leftarrow_\$ \{0,1\}^*$, two messages $m_0, m_1 \leftarrow_\$ \{0,1\}^*$ and $r_0, r_1 \leftarrow_\$ \{0,1\}^*$.

M executes one run of the protocol, behaving like an honest user instance $C(t, m_0)$ started on randomness r_0 by sending $(t, H_2(m_0)^{r_0})$ to R. It saves the result y_0 output by R.

Next, M resets R to the point where R had just output the public key pk.

M again executes a run of the protocol, emulating $C(t, m_1)$ on randomness r_1 by sending $(t, H_2(m_1)^{r_1})$ to R.

Upon receiving the result y_1 of this run, the meta-reduction sends $(t, m_0, y_0^{1/r_0}), (t, m_1, y_1^{1/r_1})$ to the reduction. When R outputs a tentative problem solution $\mathbf{y}'$, M also outputs $\mathbf{y}'$.

Assuming the reduction always let's the adversary compute a valid output on the first run, the pairs $(m_0, z_0), (m_1, z_0)$ output by the magic adversary and by the meta-reduction are distributed identically. To see this, observe that m_0 and m_1 are chosen independently in both cases, so their respective results z_0 and z_1 are also independent and we can consider the distributions of (m_0, z_0) and (m_1, z_1) separately in both cases. Furthermore, the transcripts resulting from m_0 and m_1 respectively are independent of the messages and uniformly distributed because the blinding is unconditional.

For (m_0, z_0) note, that in both cases z_0 is the result of a valid run of the protocol, so the pair is identically distributed for the magic adversary case and the meta-reduction case.

To obtain (m_1, z_1), the meta-reduction executes an honest run of the protocol, while the magic adversary does not use the protocol, but uses Σ to perform all computations locally. This does not affect the distribution of the result, as it is the same as if the protocol had run with input t, m_1 and the reduction as server. Thus the distributions are again the same. It follows that

$$\Pr[V(\mathbf{x}, \mathbf{y}') = 1 \mid \mathbf{y}' \leftarrow_\$ R^{\mathcal{A}^\Sigma}(\mathbf{x})] = \Pr[V(\mathbf{x}, \mathbf{y}') = 1 \mid \mathbf{y}' \leftarrow_\$ R^M(\mathbf{x})].$$

As our efficient meta-reduction can solve P by running R locally, we have shown, that the existence of R is in contradiction of the hardness of P. □

7. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable feedback. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA) (FKZ: 16KIS0345) and for the project PROMISE. Moreover, it was supported by the German Research Foundation (DFG) via the collaborative research center "Methods and Tools for Understanding and Controlling Privacy" (SFB 1223).

8. REFERENCES

[1] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[2] F. Armknecht and J. Furukawa. On the minimum communication effort for secure group key exchange. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 320–337, Waterloo, Ontario, Canada, Aug. 12–13, 2011. Springer, Heidelberg, Germany.

[3] A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu. Password-protected secret sharing. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM CCS 11*, pages 433–444, Chicago, Illinois, USA, Oct. 17–21, 2011. ACM Press.

[4] G. Barthe, E. Fagerholm, D. Fiore, J. Mitchell, A. Scedrov, and B. Schmidt. Automated analysis of cryptographic assumptions in generic group models. In *Advances in Cryptology–CRYPTO 2014*, pages 95–112. Springer, 2014.

[5] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374, San Francisco, CA, USA, Mar. 19–21, 2008. Springer, Heidelberg, Germany.

[6] T. Berson, D. Dean, M. Franklin, D. Smetters, and M. Spreitzer. Cryptography as a network service. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*. Citeseer, 2001.

[7] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson. The memory-hard Argon2 password hash and proof-of-work function. Internet-Draft draft-irtf-cfrg-argon2-00, Internet Engineering Task Force, 2016. Work in Progress.

[8] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.

[9] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300, Bengalore, India, Dec. 1–5, 2013. Springer, Heidelberg, Germany.

[10] E. Bresson, J. Monnerat, and D. Vergnaud. Separation results on the "one-more" computational problems. In T. Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 71–87, San Francisco, CA, USA, Apr. 7–11, 2008. Springer, Heidelberg, Germany.

[11] D. R. L. Brown. Irreducibility to the one-more evaluation problems: More may be less, 2007. dbrown@certicom.com 13850 received 23 Nov 2007, last revised 3 Dec 2007.

[12] H. Busch, S. Katzenbeisser, and P. Baecher. PUF-based authentication protocols - revisited. In H. Y. Youm and M. Yung, editors, *WISA 09*, volume 5932 of *LNCS*, pages 296–308, Busan, Korea, Aug. 25–27, 2009. Springer, Heidelberg, Germany.

[13] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 201–210, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press.

[14] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[15] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105, Santa Barbara, CA, USA, Aug. 16–20, 1993. Springer, Heidelberg, Germany.

[16] M. Di Raimondo and R. Gennaro. Provably secure threshold password-authenticated key exchange. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 507–523, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[17] J. Engler, C. Karlof, E. Shi, and D. Song. Is it too late for pake? *indicators*, 5(9):17, 2009.

[18] A. Everspaugh. Pythia server (prototype) implementation. https://github.com/ace0/pythia, 2015.

[19] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart. The pythia prf service. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 547–562, Washington, D.C., 2015. USENIX Association.

[20] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, Aug. 1987. Springer, Heidelberg, Germany.

[21] M. Fischlin and D. Schröder. Security of blind signatures under aborts. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 297–316, Irvine, CA, USA, Mar. 18–20, 2009. Springer, Heidelberg, Germany.

[22] M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[23] M. Fischlin and D. Schröder. Security of blind signatures under aborts and applications to adaptive oblivious transfer. *J. Mathematical Cryptology*, 5(2):169–204, 2012.

[24] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 303–324, Cambridge, MA, USA, Feb. 10–12, 2005. Springer, Heidelberg, Germany.

[25] A. Herzberg and R. Margulies. Forcing johnny to login safely - long-term user study of forcing and training login mechanisms. In V. Atluri and C. Díaz, editors, *ESORICS 2011*, volume 6879 of *LNCS*, pages 452–471, Leuven, Belgium, Sept. 12–14, 2011. Springer, Heidelberg, Germany.

[26] S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and t-pake in the password-only model. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 233–253. Springer Berlin Heidelberg, 2014.

[27] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164, Santa Barbara, CA, USA, Aug. 17–21, 1997. Springer, Heidelberg, Germany.

[28] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, RFC Editor, September 2000.

[29] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.

[30] E. Kiltz, K. Pietrzak, D. Cash, A. Jain, and D. Venturi. Efficient authentication from hard learning problems. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 7–26, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[31] P. D. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 385–400, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Heidelberg, Germany.

[32] A. Muffet. Facebook: Password hashing and authentication. https://video.adm.ntnu.no/pres/54b660049af94, 2015. Video.

[33] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467, Miami Beach, Florida, Oct. 19–22, 1997. IEEE Computer Society Press.

[34] T. Okamoto. Efficient blind and partially blind signatures without random oracles. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99, New York, NY, USA, Mar. 4–7, 2006. Springer, Heidelberg, Germany.

[35] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In B. K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20, Chennai, India, Dec. 4–8, 2005. Springer, Heidelberg, Germany.

[36] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140, Santa Barbara, CA, USA, Aug. 11–15, 1992. Springer, Heidelberg, Germany.

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

[38] P. Robinson. Cryptography as a service. *RSAConference Europe*, 2013.

[39] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystem based on pairing, 2000.

[40] D. Schröder and D. Unruh. Security of blind signatures revisited. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 662–679, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

[41] D. Wagner and I. Goldberg. Proofs of security for the Unix password hashing algorithm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 560–572, Kyoto, Japan, Dec. 3–7, 2000. Springer, Heidelberg, Germany.

[42] Wikipedia. List of data breaches — wikipedia, the free encyclopedia, 2016. [Online; accessed 14-August-2016].