Robust Authentication Using Physically Unclonable Functions*

Keith B. Frikken[1], Marina Blanton[2], and Mikhail J. Atallah[3]

[1] Computer Science and Software Engineering, Miami University
frikkekb@muohio.edu
[2] Department of Computer Science and Engineering, University of Notre Dame
mblanton@cse.nd.edu
[3] Department of Computer Science, Purdue University
mja@cs.purdue.edu

Abstract. In this work we utilize a physically unclonable function (PUF) to improve resilience of authentication protocols to various types of compromise. As an example application, we consider users who authenticate at an ATM using their bank-issued PUF and a password. We present a scheme that is provably secure and achieves strong security properties. In particular, we ensure that (i) the user is unable to authenticate without her device; (ii) the device cannot be used by someone else to successfully authenticate as the user; (iii) the device cannot be duplicated (e.g., by a bank employee); (iv) an adversary with full access to the bank's personal and authentication records is unable to impersonate the user even if he obtains access to the device before and/or after the setup; (v) the device does not need to store any information. We also give an extension that endows the solution with emergency capabilities: if a user is coerced into opening her secrets and giving the coercer full access to the device, she gives the coercer alternative secrets whose use notifies the bank of the coercion in such a way that the coercer is unable to distinguish between emergency and normal operation of the protocol.

1 Introduction

Recent work has demonstrated the existence and practicality of physically unclonable functions (PUFs), but many of their security implications remain to be explored. PUFs have both advantages and limitations compared to more traditional security devices. E.g., compared to a smartcard, a PUF has the advantage that it cannot be cracked and its secrets revealed, or replicated by an insider who has the blueprint. But unlike a smartcard, one can no longer assume the convenient existence of multiple copies that all contain the same key, nor can one assume any storage capacity within a device other than the PUF functionality.

The focus of this work is authentication, where a physically unclonable function (PUF) is used to provide superior resilience against various forms of compromise. A PUF is a function that is tied to a device and cannot be reproduced on another device, even another device from the same manufacturing batch. That is, a PUF is computed

* Portions of this work were supported by grants NSF-CNS-0627488 and AFOSR-FA9550-09-1-0223, and by sponsors of CERIAS.

P. Samarati et al. (Eds.): ISC 2009, LNCS 5735, pp. 262–277, 2009.

using unique physical characteristics of the device, and any attempts to tamper with the device change the behavior of the device and therefore destroy the PUF. This function is often assumed to be evaluated on a challenge c which is sent to the device. Upon receiving c, the response is computed as $r = PUF(c)$ and is assumed to be unpredictable to anyone without access to the device. Schemes exist for using in different contexts (e.g., for protection of intellectual property and authentication), where the inability to clone the function improves the properties of a solution.

Here we use PUFs for authentication in contexts such as bank ATMs, through the use of a device with a built-in PUF. The ATM communicates with the bank to establish authenticity of the user before any transaction. We are able to achieve strong security properties which are not simultaneously achieved by previous protocols. In particular, our protocol provably has the following properties (in the random oracle model):

- a user is unable to successfully authenticate without her device;
- a stolen device cannot be used to authenticate as the user;
- the device functionality cannot be duplicated (e.g., by an employee of the bank even if that employee has access to the card);
- an adversary with full access to the bank's data with user information and authentication records is unable to impersonate the user even if she obtains access to the device before and/or after the account is setup.

Furthermore, our design requirements are to avoid placing any sensitive information on the device, to eliminate any possibility of data compromise (i.e., the PUF, which measures a physical characteristic of the device, will be destroyed in the event of tampering with the device, while the data stored on the device might not be erased). In fact, our protocols do not require the device to store *any* information not related to the PUF functionality, which introduces a challenge in the protocol design.

Our Contributions

1. We provide a protocol for one-factor authentication with PUFs (Section 4.1). It provides only a weak form of security in that to authenticate the adversary needs to have had physical access to the PUF at some point in time. One limitation of this protocol (and any one-factor "what you have" authentication mechanism) is that in order to impersonate a user, the adversary only needs physical access to the device.
2. We provide a stronger protocol for two-factor authentication that combines PUFs with passwords (Section 4.2). The adversary must have had access to the PUF and to the user's password in order to impersonate the user, even if the adversary has compromised the bank's servers. A unique feature of this protocol is that the password is not stored in either the PUF or the bank, but is integrated into the PUF challenge, and thus in order to perform a dictionary attack one must have physical access to the PUF.
3. One limitation of the previous schemes is that an adversary can clone the PUF in software by having physical access to the PUF. That is, the adversary can obtain the PUFs response to a challenge, and then build a piece of software that impersonates the user. To mitigate this software cloning attack, we introduce a protocol which requires the authenticator to currently have physical access to the PUF in order to authenticate (Section 4.3). This protocol requires a stronger assumption than those

required by the previous schemes: We assume an integrated PUF (or computational PUF) where the device performs some computation with the PUF.

4. We give an extension which additionally improves robustness of the protocol when a user is coerced into giving her device and secret data (e.g., her password), which permits an adversary to authenticate on behalf of the user. We provide a mechanism for a user to give a false secret to the coercer that will lead to successful authentication, but will trigger an alarm at the bank. Solutions of this type are common in physical security systems, but do not appear in cryptographic protocols[1].

2 Related Work

Existing literature on PUF-based authentication is not extensive and can be divided into three categories: (i) implementation-based publications that consider the feasibility of reliably computing a PUF response to a challenge; (ii) PUF-based authentication for IP (intellectual property) protection; and (iii) enhancing properties of lightweight authentication solutions using PUF.

Publications from the first category include [2,3] and others and are complementary to our work. They also provide support for using public-key cryptography with PUF-based authentication. Publications from the second category (e.g., [4,5,6]) are also largely implementation-based, often implementing existing authentication protocols for reconfigurable FPGA and are not suitable for our purposes. The last category covers PUF-based protocols for RFID (Radio-frequency identification) systems [7,8,9] and human protocols HB [10,11]. The RFID publications are implementation-based realizing simple authentication constructions. Recent results [10,11] strengthen the HB protocol by using PUFs and are not suitable in our context (i.e., do not achieve the properties we seek).

Authentication protocols based on smart-cards can also be viewed as related to our framework. However, the nature of PUF-based authentication places unique requirements: For a smartcard protocol to fit our model, the smartcard must implement a PUF and have no other information stored, yet satisfy our security requirements – there are no such previous smartcard protocols.

Multi-factor authentication protocols, which often use a password and a mobile device, have been explored in prior literature (see, e.g., [12,13,14,15] among others – some have insufficient security analysis). Resilience to user impersonation in the event of database compromise (the "insider threat"), however, is not considered and not achieved in previous work. In our case both factors (i.e., the user password and the token) are inaccessible to the server in their plain form, so that an insider with full access to the server is unable to recover either of them.

Boyen [16] uses biometrics and fuzzy extractors (i.e., biometric-based key derivation) to provide zero-storage authentication that achieves insider security. Our solution then can be viewed as an authentication mechanism with similar security properties, but which is based on a different technique and type of device (instead of using a device that captures biometrics) and additionally includes passwords as the second security

[1] The only publication on panic passwords in computer systems we are aware of is [1] that treats the general framework of panic passwords and is discussed later in this section.

factor. Note that we desire the same level of security even when the PUF is misused (by either a bank employee who temporarily gets access to the PUF or the user herself). This means that, to ensure that the device is present during each authentication session, we would like to make the raw information output of a PUF inaccessible to the user and use computational capabilities of a PUF. This problem is not a threat in case of biometric-based authentication, when the user is interested in erasing her personal biometric data output by the device and used in the protocol.

Recent work of Clark and Hengartner [1] defines the framework for panic passwords, where any user has a regular password and another, panic, password which can be used when the user is coerced into giving her password to the adversary. They define the adversarial model in terms of the response the user receives from the authenticator upon using a panic password, and goals/capabilities of the adversary. Our solution was designed independently of this recent model, but in section 5 we briefly discuss how it fits the Clark-Hengartner framework.

3 Security Model

3.1 Problem Description

There are three principal entities: server $\mathcal{S}$ (or another entity authenticating the user on behalf of the server), user $\mathcal{U}$, and device $\mathcal{D}$. Before authentication can take place, the user obtains a device with a PUF built into it and participates in the registration or enrollment protocol with the server. Once the registration is complete, the user will be able to authenticate with the help of the device. Thus, we specify two procedures:

Enroll**:** is a protocol between $\mathcal{S}$ and $\mathcal{U}$, where the user $\mathcal{U}$ registers with the server with the aid of $\mathcal{D}$. If enrollment is successful, the server obtains and stores a token $\mathsf{cred}_{\mathcal{U}}$ that can be used in subsequent authentications.

Auth**:** is a protocol between $\mathcal{S}$ and $\mathcal{U}$, where $\mathcal{U}$ uses $\mathcal{D}$ and $\mathcal{S}$ uses its stored credentials $\mathsf{cred}_{\mathcal{U}}$ to make its decision to either accept or reject the user.

3.2 Modeling PUFs

Prior literature does not contain a lot of cryptographic constructions where PUFs are used in a provably secure scheme. We are aware of the following uses of such functions. In what follows, we will generically refer to the entity trying to authenticate (i.e., user, device, tag, etc.) as a client and to the entity verifying authentication as a server.

1. *Straightforward authentication.* This is the most common form found in the PUF literature, where the server sends a challenge c and the client responds with $r = PUF(c)$. At the enrollment phase, the server stores n challenges $c_1, \ldots, c_n$ and their corresponding responses $r_1, \ldots, r_n$ for each client. During authentication, the client is challenged on one of the c_i's at random and that (c_i, r_i) is removed from the database. If the server runs out of challenge-response pairs (CRPs), there are protocols for updating the server's database with new CRPs [17].
2. *PUF as a random oracle.* Modeling a PUF as a random oracle (as in [8]) might be unnecessary if the full features of the random oracle model are not used.

3. *PUF as a computable function.* Hammouri and Sunar [10] define a delay-based PUF that can be represented using a linear inequality. This means that the server does not need to store CRPs, but instead can compute the expected responses. While it might be possible to model the specific PUF used in the above paper, for general functions it is commonly assumed that the function cannot be modeled and its behavior cannot be predicted by any entity without physical access to it.
4. *PUF in previously published identification protocols.* Some papers gave implementations where a PUF response is used as a part of known identification protocols. E.g., Tuys and Batina [7] use PUFs in Schnorr's identification protocol, where the user's secret key is set to be PUF's response to a challenge. Similarly, Batina et al. [9] use Okamoto identification protocol with PUF-derived secrets. We, however, aim to design a PUF-based protocol specific to our security goals.

As in the previous PUF literature, we make the standard assumption that, without having the physical device, the behavior of a PUF is impossible to predict. Let PUF be a function $PUF : \{0,1\}^{\kappa_1} \rightarrow \{0,1\}^{\kappa_2}$ that on input of length κ_1 produces a string of length κ_2. Before giving the definition, let us first define the following *PUF response game*:

Phase 1: Adversary $\mathcal{A}$ requests and gets the PUF response r_i for any c_i of its choice.

Challenge: $\mathcal{A}$ chooses a challenge c that it has not queried thus far.

Phase 2: $\mathcal{A}$ is allowed to query the PUF for challenges other than c.

Response: Eventually, $\mathcal{A}$ outputs its guess for r' for PUF's response to $r = PUF(c)$.

$\mathcal{A}$ wins if $r = r'$. Let $Adv_{\mathcal{A}}^{puf}(\kappa_2) = \Pr[r = r']$ denote the probability of $\mathcal{A}$ winning.

Under different conditions and in different environments, PUF responses to the same challenge can contain noise resulting in non-perfect match. We measure such noise in terms of hamming distance between two binary strings x_1 and x_2 of equal length κ, i.e., $\mathsf{dist}(x_1, x_2)$ is the number of positions such that ith bit of x_1 is different from the ith bit of x_2. In what follows, let U_κ denote the set of strings chosen uniformly at random from $\{0,1\}^\kappa$. Now we are ready to define a PUF:

Definition 1. *A physically unclonable function $PUF_D : \{0,1\}^{\kappa_1} \rightarrow \{0,1\}^{\kappa_2}$ bound to a device D is a function with the following properties:*

1. Efficient: *PUF_D is easy to evaluate;*
2. Hard to characterize: *for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{puf}(\kappa_2)$ is negligible in κ_2;*
3. Bounded noise: *in a wide variety of environments, the distance between two responses from PUF_D on the same challenge is at most t, e.g.,* $\Pr[\mathsf{dist}(y,z) > t \mid x \leftarrow U_{\kappa_1}, y \leftarrow PUF_D(x), z \leftarrow PUF_D(x)] \leq \varepsilon_1$ *for a negligibly small ε_1;*
4. Unique: *the PUF_D is unique for each D (even those from the same manufacturing batch), e.g., for any other function $PUF_{D'}$,* $\Pr[\mathsf{dist}(y,z) \leq t \mid x \leftarrow U_{\kappa_1}, y \leftarrow PUF_D(x), z \leftarrow PUF_{D'}(x)] \leq \varepsilon_2$ *for sufficiently small ε_2.*

We call such a function a $(t, \varepsilon_1, \varepsilon_2)$ PUF (i.e., ε_1 and ε_2 are false rejection rate and false acceptance rate, respectively). Some of our constructions furthermore assume that a PUF is inseparatable from the device to which it is bound, i.e., our latter schemes make the strong assumption that the device (circuit) can do computation based on PUF responses. More specifically:

Definition 2. *An integrated PUF (I-PUF) has the following additional properties:*

1. *It is bound to the chip – any attempt to remove it changes its behavior.*
2. *Its communication with the chip cannot be accessed from outside the chip.*
3. *The output of the PUF cannot be accessed.*

I-PUFs have been used in prior literature (see, e.g., [7]), and the best known examples of them are silicon PUFs [17] and coating PUFs [18].

Because the output of a PUF is noisy, PUF-based authentication must either tolerate a certain threshold of errors at the protocol level or implement a mechanism for correcting the errors prior to using the response of the PUF. We choose the second option. Prior literature [7,6] already contains examples of using functions such as *fuzzy extractors* to remove the noise and extract responses that are close to uniform. Fuzzy extractors [19] can be defined for different metric spaces, and throughout this work we will assume we are dealing only with Hamming distance as the distance metric. The definition furthermore assumes a sufficient amount of uncertainty of the noisy string from which a random string is being extracted, defined in terms of min-entropy m (see [19] for more precise definitions). The construction generates a public helper string P that permits correction of errors and reconstruction of the extracted string and ensures that, even after releasing P, the statistical distance between the extracted string and a uniformly chosen string of the same length is less than a (negligibly small) threshold ε (likewise, we refer the reader to [19] for precise definitions).

Definition 3 ([19]). *An (m,ℓ,t,ε) fuzzy extractor is given by procedures* Gen *and* Rep*:*

Gen**:** *is a probabilistic algorithm that on input W outputs a string $R \in \{0,1\}^\ell$ and a helper string P, such that for any distribution of W with min-entropy m, if $(R,P) \leftarrow \mathsf{Gen}(W)$, then the statistical difference between (R,P) and (U_ℓ,R) is at most ε.*

Rep**:** *is a deterministic algorithm that, given P and W' such that $\mathsf{dist}(W,W') \le t$, allows to exactly reproduce R: if $(R,P) \leftarrow \mathsf{Gen}(W)$, then $\mathsf{Rep}(W',P) = R$.*

Our discussion does not necessitate the details of fuzzy extractors, as we refer to them only at a high level. They make possible the construction of an exact I-PUF having $(t,\varepsilon_1,\varepsilon_2)$ $PUF : \{0,1\}^{\kappa_1} \to \{0,1\}^{\kappa_2}$ such that:

1. An I-PUF bound to device D is associated with a (m,ℓ,t,ε_3) fuzzy extractor $(\mathsf{Gen},\mathsf{Rep})$, where Gen, Rep, and PUF_D are efficient procedures.
2. During the enrollment phase, given a challenge c, I-PUF computes $(R,P) \leftarrow \mathsf{Gen}(r)$, where $r \leftarrow PUF_D(c)$ and outputs P.
3. In a wide variety of environments, given a pair (c,P) where $c \leftarrow U_{\kappa_1}$ and P was produced by $\mathsf{Gen}(PUF_D(c))$, the exact extracted string can be recovered: $\Pr[x \neq y \mid x \leftarrow \mathsf{Rep}(PUF_D(c),P), y \leftarrow \mathsf{Rep}(PUF_D(c),P)] \le \varepsilon_1$.
4. Any PPT adversary $\mathcal{A}$ cannot distinguish I-PUF's output from a random value with more than a negligible probability, i.e., $Adv_{\mathcal{A}}^{puf\text{-}ind}(\ell) \le \varepsilon_3$ as defined below.

The last property can be viewed as a decisional version of the PUF response game, which we call *PUF response indistinguishability game* and it is defined as follows:

Enroll: $\mathcal{A}$ executes the enrollment phase on any values c_i of its choice receiving the corresponding P_i values from the PUF. Let CP be the set of these (c_i,P_i) pairs.

Phase 1: $\mathcal{A}$ requests and receives PUF response R_i for any $(c_i, P_i) \in CP$ of its choice.

Challenge: $\mathcal{A}$ chooses a challenge c that it queried in **Enroll** phase but not in **Phase 1**. A random bit b is chosen. If $b = 0$, $\mathcal{A}$ receives $R = \mathsf{Rep}(PUF_D(c), P)$ where $(c, P) \in CP$, otherwise it receives a string uniformly chosen from from $\{0,1\}^\ell$.

Phase 2: $\mathcal{A}$ is allowed to query the PUF for challenges in CP other than (c, P).

Response: Eventually, $\mathcal{A}$ outputs a bit b'.

$\mathcal{A}$ wins if $b = b'$. Let $Adv_{\mathcal{A}}^{puf\text{-}ind}(\ell) = \Pr[b = b']$ denote the probability of $\mathcal{A}$ winning the game. We assume that $Adv_{\mathcal{A}}^{puf\text{-}ind}(\ell) - 1/2$ is negligible [2].

In addition to the above properties, as before, we have that two I-PUFs will produce the same output on a (c, P) pair with probability at most ε_2. Finally, our protocols rest on the difficulty of the discrete logarithm in certain groups. That is, we assume that any PPT adversary $\mathcal{A}$, when given group G of large order q, group generator g, and element g^x for some $x \in \mathbb{Z}_q$, has a negligible change in outputting x.

3.3 Security Requirements

We place strict security requirements on the authentication process to achieve a solution robust to various types of misuse. In particular, we target to achieve the following properties beyond the traditional infeasibility to impersonate a user:

- Authentication by an adversary is not successful even with the possession of the device. Here we assume a powerful adversary who has access to all stored information at the server's side, including all information stored by the server during the enrollment phase such as recorded (c, P) pairs for the device and other user's information $\mathsf{cred}_{\mathcal{U}}$, as well as information belonging to other users. This strong notion of security is necessary in realistic scenarios, when, for example, the device originally resides with a bank, is consequently issued to a user, and a bank employee might later temporarily get access to the device and attempt to impersonate the user.
- Authentication by an honest user without the device is not successful with more than a negligible probability. This is important because, if this property holds, it is equivalent to a strong form of unclonability, i.e., even if an adversary knows all bank and user information, it cannot create a clone of the device in question. This adds resilience to a "what you have" form of authentication, because it guarantees that one must have the device during a successful login.

The use of I-PUFs ensures that the device cannot be duplicated or cloned, and tampering with the PUF effectively makes it unusable. Thus, the above requirements guarantee that both the original device and the user must be present at the time of authentication for authentication to succeed (except with negligible probability).

Furthermore, to ensure that tampering with the device does not reveal any sensitive information, our design stores no such information on the device. In fact we assume that the device does not store any information at all, and can be used for authentication with a number of servers. Thus, all necessary information is provided as input to the device, which makes the design of the protocols particularly challenging in presence of adversaries who can query the device's response on various inputs.

[2] We assume the PUF is built with a security parameter that allows tuning this probability.

4 Schemes

4.1 Preliminary Scheme

In this section we introduce a scheme that does not satisfy the security criteria, but that is a warm-up to the later schemes that do. In this and subsequent solutions we assume that the server $\mathcal{S}$ sets up and announces a group $\mathbb{G}_q$ of prime order q, in which the discrete logarithm problem is hard, and its generator g. That is, $\mathbb{G}_q$ could be a subgroup of the multiplicative group $\mathbb{Z}_p^*$ for a prime p. We assume either that the PUF is constructed to use $\mathbb{G}_q$ or the user submits the group to the PUF whenever it queries the PUF[3]

The authentication protocol given below uses a zero-knowledge proof of knowledge (ZKPK) of discrete logarithm. In a nutshell, a ZKPK of a discrete logarithm y to the base g allows the prover to convince the verifier that she knows x such that $y = g^x$ without revealing any information about x. Because standard and well-known solutions for several discrete logarithm based ZKPKs exist, we do not list their details in this work and refer the reader to, e.g., [20].

Enroll :

1. Server $\mathcal{S}$ sends challenge c to user $\mathcal{U}$.
2. $\mathcal{U}$ sends c to device $\mathcal{D}$ for Gen protocol.
3. $\mathcal{D}$ sends to $\mathcal{U}$ (r,P).
4. $\mathcal{U}$ sends (g^r,P) to $\mathcal{S}$ who stores the information along with c.

Auth :

1. $\mathcal{S}$ sends challenge (c,P) to the user $\mathcal{U}$.
2. $\mathcal{U}$ sends (c,P) to device $\mathcal{D}$ for Rep protocol.
3. $\mathcal{D}$ sends r to $\mathcal{U}$.
4. $\mathcal{U}$ and $\mathcal{S}$ engage in a ZKPK of discrete logarithm g^r to the base g.

Clearly, the above scheme does not satisfy either authentication goal. That is, if the adversary has access to the device and knows the server's challenge, then it can obtain the response and can impersonate the user without the device.

4.2 Preliminary Scheme Revisited

The problem with the previous scheme is that having the PUF (at any point in time) allows an adversary to impersonate the user. In this section we modify the previous scheme by adding a user password (and thus the adversary must have the device and guess the user's password). The password is integrated into the Enroll and Auth protocols so that the password is necessary to do authentication. Furthermore, the password is not stored anywhere. This prevents an adversary from being able to login in to the protocol even if it has the device. While this scheme still does not require that the user have the device, it does prevent a malicious outsider from impersonating the user (assuming that the user can choose a strong password). In what follows, $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a cryptographic hash function and $||$ denotes concatenation of strings.

[3] To prevent tampering with this value the PUF could take the hash of the description of this group with the challenge to form a modified challenge which it then responds to. So as not to clutter the exposition we have omitted this step from our scheme.

Note that in this scheme the password is bound to the PUF-challenge in order to get the PUF-response. Thus the password is not really stored anywhere, and thus in order to perform a dictionary attack the adversary must have physical access to the PUF.

Enroll :

1. Server $\mathcal{S}$ sends challenge c to user $\mathcal{U}$.
2. $\mathcal{U}$ sends $H(c||pwd)$, where pwd is the password, to device $\mathcal{D}$ for Gen protocol.
3. Device $\mathcal{D}$ sends (r,P) to $\mathcal{U}$..
4. $\mathcal{U}$ sends (g^r,P) to server $\mathcal{S}$ who stores the information along with c.

Auth :

1. Server $\mathcal{S}$ sends challenge c and P to the user $\mathcal{U}$.
2. User sends $(H(c||pwd),P)$ to device $\mathcal{D}$ for Rep protocol.
3. $\mathcal{D}$ sends r to $\mathcal{U}$.
4. User $\mathcal{U}$ and server $\mathcal{S}$ engage is ZKPK of discrete logarithm for g^r to the base g.

At a high level, this scheme requires that the adversary enter the user's password in order to the actual challenge sent to the PUF, thus this prevents an adversary with the PUF from being able to find the response r.

The proof of security of this approach has two parts. First, it is shown that if the response is generated independently from the PUF then breaking the above authentication scheme implies that the discrete log problem can be solved. Thus this implies (assuming discrete log problem is hard) that a computationally-bounded adversary has a negligible success probability (in the security parameter for the size of the prime q) in breaking the above scheme. The second part of the proof shows that if $\mathcal{A}$ can break the scheme with non-negligible probability when a real PUF is used, then this could be used to win the *PUF response indistinguishability game* with non-negligible probability. In other words, to determine if a specific response came from the PUF, the adversary uses $\mathcal{A}$ and if $\mathcal{A}$ succeeds then we assume that we are dealing with a real PUF (because if we are not the success probability is negligible).

Lemma 1. *If H is a random oracle and there exists an adversary, $\mathcal{A}$, that successfully authenticates with the above authentication protocol with probability $1/p(|q|)$ when given a randomly generated challenge (that is independent from the PUF), then $\mathcal{A}$ contains a knowledge extractor that can solve the discrete log problem with non-negligible probability (in the length of q).*

Proof. Assume that such an adversary $\mathcal{A}$ exists, and that an adversary $\mathcal{B}$ is given a discrete log problem instance g,q,g^r and is given access to a PUF. $\mathcal{B}$ sends the challenge c_s,g^r,P for a randomly chosen c_s and P to $\mathcal{A}$. To simulate H, $\mathcal{B}$ creates a set of tuples $HSET$ and initializes it by choosing a random password pwd and adding $(c_s||pwd,h_s)$ to $HSET$ for a randomly chosen h_s. When $\mathcal{A}$ queries H on a value x, $\mathcal{B}$ does the following: If there is a tuple (x,y) already in $HSET$, it responds with y; otherwise, it chooses a random r', adds (x,r') to $HSET$, and responds with r'. When $\mathcal{A}$ queries PUF with $(c_{\mathcal{A}},P_{\mathcal{A}})$, $\mathcal{B}$ does the following: If $c_A = h_s$ and $P_{\mathcal{A}} = P$, $\mathcal{B}$ outputs FAIL. Otherwise $\mathcal{B}$ queries its PUF with $(c_{\mathcal{A}},P_{\mathcal{A}})$ and receives r_A. $\mathcal{B}$ then sends to $\mathcal{A}$ the value r_A.

It is straightforward to show that if $\mathcal{B}$ does not output FAIL, then the above view is the same as the view when engaging in the protocol. In the following we show that: (i)

$\mathcal{B}$ outputs FAIL with negligible probability, and (ii) if $\mathcal{B}$ does not output FAIL and $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{B}$ can use $\mathcal{A}$ to obtain r.

$\mathcal{B}$ outputs FAIL only when $\mathcal{A}$ asks for the PUF response for a challenge (d,P) and $d = h_s$. There are two situations: (i) $\mathcal{A}$ queries H on $c||pwd$ or (ii) $\mathcal{A}$ does not query H on $c_s||pwd$. The first case implies that $\mathcal{A}$ knows pwd (which we assume is a negligible event), and the second case corresponds to $\mathcal{A}$ randomly guessing h_s which is negligible. Thus, $\mathcal{B}$ outputs FAIL with negligible probability.

Now if $\mathcal{B}$ does not output FAIL and $\mathcal{A}$ can create a ZKPK of the discrete log of g^r, then by the properties of zero-knowledge, there must a be knowledge extractor for $\mathcal{A}$ that produces the secret r. $\mathcal{B}$ uses this knowledge extractor to solve the discrete log problem. Notice that if $\mathcal{A}$ succeeds then so does $\mathcal{B}$, and therefore assuming discrete log problem is hard, an adversary $\mathcal{A}$ does not exist. □

We now utilize the above lemma to show that an adversary cannot break the above protocol if a real PUF is used (except with negligible probability). The lynchpin to this argument is that if such an adversary exists, then this adversary could be used to distinguish between a fake and real PUF, which violates the assumed security of the *PUF response indistinguishability game*.

Theorem 1. *Any polynomial-time adversary with access to the PUF (with security parameter ℓ) and server information has a negligible probability of passing the authentication protocol for a previously generated enrollment, assuming that H is a random oracle, the discrete log problem is hard, and the passwords are chosen from a large enough domain to make guessing the password succeed with negligible probability.*

Proof. Assume that such an adversary $\mathcal{A}$ exists, we then use this as a black-box to construct an adversary $\mathcal{B}$ for the *PUF response indistinguishability game* that succeeds with non-negligible probability. $\mathcal{B}$ proceeds as follows: it chooses a random challenge value c_s and a random password pwd. It computes $c' = H(c_s||pwd)$ and chooses c' as its challenge. $\mathcal{B}$ then receives a pair (r,P) where with probability $1/2$ the value r is $PUF_D(c)$ and is otherwise a randomly chosen value. $\mathcal{B}$ constructs server information c_s, g^r, P and invokes the adversary $\mathcal{A}$ on these values while $\mathcal{B}$ provides oracle access to the PUF and to the random oracle H in the exact same manner as in Lemma 1. Eventually $\mathcal{A}$ will output a proof of knowledge. If this proof of knowledge is correct, then $\mathcal{B}$ outputs 0; otherwise, $\mathcal{B}$ chooses a random guess for b' and outputs this value.

We now analyze the probability $\Pr[b = b']$. Let F be the event the $\mathcal{B}$ outputs FAIL. Since F was shown to be a negligible event in Lemma 1, we concentrate on $Pr[b = b'|\overline{F}]$. We condition it based on event $b = 0$ or $b = 1$, which gives us:

$$\Pr[b = b'|\overline{F}] = \frac{1}{2}\Pr[b = b'|\overline{F}, b = 0] + \frac{1}{2}\Pr[b = b'|\overline{F}, b = 1]$$

Let G be the event that $\mathcal{A}$ outputs a correct proof of knowledge. We condition both of the above cases on G. In case of $b = 1$:

$$\Pr[b = b'|\overline{F}, b = 1] = \Pr[b = b'|\overline{F}, b = 1, G]\Pr[G|\overline{F}, b = 1] + \Pr[b = b'|\overline{F}, b = 1, \overline{G}]\Pr[\overline{G}|\overline{F}, b = 1].$$

Here, $\Pr[b = b'|\overline{F}, b = 1, G] = 0$, $\Pr[b = b'|\overline{F}, b = 1, \overline{G}] = \frac{1}{2}$, and $\Pr[G|\overline{F}, b = 1]$ is negligible (by Lemma 1). This gives us:

$$\Pr[b = b' | \overline{F}, b = 1] > \frac{1}{2} - neg_2(\ell)$$

for some negligible function neg_2. Next, let us consider $b = 0$, in which case we have:

$$\Pr[b = b' | \overline{F}, b = 0] = \Pr[b = b' | \overline{F}, b = 0, G] \Pr[G | \overline{F}, b = 0] + \Pr[b = b' | \overline{F}, b = 0, \overline{G}] \Pr[\overline{G} | \overline{F}, b = 0].$$

Here, $\Pr[b = b' | \overline{F}, b = 0, G] = 1$, $\Pr[b = b' | \overline{F}, b = 0, \overline{G}] = \frac{1}{2}$, and $\Pr[G | \overline{F}, b = 0] > \frac{1}{f(\ell)}$ for some polynomial f (this follows from our assumption that $\mathcal{A}$ breaks the authentication with non-negligible probability). Putting all of this together, it is straightforward to show that $\Pr[b = b' | \overline{F}, b = 0] > \frac{1}{2} + \frac{1}{h(\ell)}$ for some polynomial h.

In summary, $\Pr[b = b' | \overline{F}] - \frac{1}{2}$ is non-negligible, hence so is $\Pr[b = b'] - \frac{1}{2}$. □

4.3 Final Scheme

Here we present the final scheme, where any user is required to currently possess the device in order to be able to successfully authenticate. The principal idea behind this approach is that the device does not reveal the response r in any protocol. The device produces only zero-knowledge proofs that it possesses the secret r. And since the proofs are zero-knowledge an adversary cannot learn r by observing the device. Unlike the previous two protocols, this protocol assumes that the PUF can also perform computation.

Enroll :

1. Server $\mathcal{S}$ sends challenge c to user $\mathcal{U}$ along with description of the group $\mathbb{G}_q$, denoted by $\langle \mathbb{G}_q \rangle$ and which could consists of a pair (p,q), and its generator g.
2. User $\mathcal{U}$ sends $H(c||pwd), \langle \mathbb{G}_q \rangle, g$, where pwd is a user password, to device $\mathcal{D}$ for a modified Gen protocol.
3. Device $\mathcal{D}$ calculates a challenge $d = H(H(c||pwd), \langle \mathbb{G}_q \rangle, g)$ and runs Gen on this value to obtain response r, P. $\mathcal{D}$ then sends to the user (g^r, P).
4. User forwards (g^r, P) to server $\mathcal{S}$, which stores the information along with $c, g, \langle \mathbb{G}_q \rangle$.

Auth :

1. Server $\mathcal{S}$ sends challenge $c, \langle \mathbb{G}_q \rangle, g, P$, and a nonce N to the user $\mathcal{U}$.
2. $\mathcal{U}$ sends $(H(c||pwd), \langle \mathbb{G}_q \rangle, g, P, N)$ to device $\mathcal{D}$ for Rep protocol.
3. Device $\mathcal{D}$ calculates a challenge $d = H(H(c||pwd), g, p)$ and runs Rep on this value to obtain response r. $\mathcal{D}$ chooses a random value $v \in \mathbb{Z}_q$ and calculates $t = g^v$. $\mathcal{D}$ then calculates $c' = H(g, g^r, t, N)$ and $w = v - c'r \bmod q$, and sends c', w to the $\mathcal{U}$.
4. User $\mathcal{U}$ sends these values to the server $\mathcal{S}$. $\mathcal{S}$ calculates $t' = g^w g^{rc'}$ and accepts the authentication if $c' = H(g, g^r, t', N)$, and otherwise rejects the value.

What is implicit in this and previous schemes is the step where the user provides its account number or some other identifying information that permits the server to locate the user's record with the corresponding helper data P and authentication verification information. What form this account information takes is not essential in our solution, and different mechanisms would be acceptable. For example, since we assume that the device does not store information permanently, the account number can be computed at the user side as a function of the user's identity and the bank name.

We first show security of a simpler (but very similar) system that uses an oracle. In this system, the oracle is initialized by obtaining the group setup $\langle \mathbb{G}_q \rangle$ and g, after which it chooses a random value r and publishes g^r. This operation is performed once, and all consecutive interactions with the oracle will use the same value r. After the setup stage, a user can query the oracle with a nonce N. On each query, the oracle chooses a random value $v \in \mathbb{Z}_q$ and calculates $t = g^v$. It then computes $c' = H(g, g^r, t, N)$ and $w = v - c'r \bmod q$, and replies with c', w to the querier. We denote this oracle by O_{auth}.

The difference between this system and the PUF-based system is that the value r is randomly chosen (rather than produced by the PUF) and the system cannot be used for authentications on different challenges c. Let an adversary be given black box access to oracle O_{auth} and a challenge nonce N. The adversary is allowed to query the oracle on all values except the challenge nonce. We now argue that, assuming that H is a random oracle, the adversary cannot forge a proof for the nonce N to the challenger.

The core of the computation performed by the above oracle (and the device in our Auth protocol) is basically a proof of knowledge of the exponent of g^r to the base g, where the proof uses a priori specified value of nonce N. The basic form of this proof of knowledge was used in different authentication protocols, including the standard Schnorr identification protocol [21]. It is well known that in Schnorr's protocol, if an adversary can produce the proof with a non-negligible probability, then there is a knowledge extractor that can produce r with non-negligible probability. That basic argument is that t must be chosen before c' and thus for a given value of t there must be a non-negligible portion of c' values for which an adversary can construct a proof. Furthermore, if the adversary can construct two proofs for the same t value but different c' values, then they can obtain r. We now argue that, if such a knowledge extractor exists, then assuming the random oracle model, there is a polynomial time adversary that can solve the discrete logarithm problem.

Lemma 2. *Any polynomial-time user has at most negligible probability of success authenticating in the above modified system with oracle O_{auth}.*

Proof. Let A be a proof generator with oracle access to O_{auth} that succeeds in answering the challenge for nonce N with non-negligible probability. Assume that algorithm B with access to A is given a value g^r and is asked to provide r. B provides A's access to random oracles H and O_{auth} and answers such queries as follows. Recall that in all queries to O_{auth} the same g, g^r are used.

1. B creates a list of values L_{OH} that will store queries to O_{auth} and H. The list is initially empty.
2. When A queries O_{auth} on a nonce value N, B does the following: it chooses a random response $w \in \mathbb{Z}_q$ and a random value $c \in \mathbb{Z}_q$. It sets $t = g^w g^{rc}$ and then adds the pair (t, N, c) to L_{OH}; but if there is already a tuple $(t, N, \hat{c})$ in L_H, $\hat{c} \neq c$, then B outputs FAIL. It returns w, c to A.
3. When A queries H on (g, g^r, t, N), B searches L_{OH} for a value of the form (t, N, c) for some c. If it exists, it responds with c. If not, B chooses a random value c, adds (t, N, c) to L_{OH}, and responds with c.

We first argue that B outputs FAIL with negligible probability. The only way it happens is if, when answering a query to O_{auth}, the chosen value t is already in L_{OH}. However, t

will be a randomly chosen value in $\mathbb{G}_q$ and, since there are at most a polynomial number of tuples in L_{OH}, the probability of an overlap is negligible. Therefore, if A succeeds with non-negligible probability, a knowledge extractor would exist that allows B to obtain r. Thus, no such A exists, assuming the discrete logarithm problem is hard. □

Now consider a challenger that provides an adversary with oracle access to a PUF. The adversary queries a challenger with either Enroll or Auth queries. The challenger answers all queries with the PUF. Eventually the adversary asks for a challenge and is given $c, \langle \mathbb{G}_q \rangle, g$, and a nonce N. The adversary can then continue to ask Enroll and Auth queries (but cannot ask for Auth on the specific nonce N and the specific challenge). The goal of the adversary is to be able to construct a response to the challenge that would pass the authentication verification at the server. The adversary wins if the authentication is successful.

Theorem 2. *Any polynomial-time adversary without the proper I-PUF device is unable to successfully authenticate with more than negligible probability in the* Auth *protocol.* Proof omitted due to page constraints.

5 Adding Emergency Capabilities

We would like to provide a user under duress with the possibility to lie about her secrets in such a way that a "silent alarm" is triggered at the server. The coercer should be unable to distinguish between an authentication protocol with real password and one with an emergency password; nor should it be detectable that the authentication protocol has provisions for using different secrets. More precisely, we consider an adversary who can record the user's communication with the server during successful authentication protocols, but does not have access to the communication between the user and the server at the enrollment stage. The adversary then forces to the user to reveal all information the user possesses in relation to authentication, including all secrets such as passwords, and also obtains physical access to the user's device. The adversary engages in an authentication protocol with the server on behalf of the user. We require that all information the adversary observes with full access to the user-provided data and the device does not allow it to distinguish its communication with the bank from the previously-recorded communication of the user with more than negligible probability. This means that all messages must follow exactly the same format and the distributions of data on different executions are not distinguishable.

We next present a scheme that has this capability. Often the above problem of coercion is addressed by letting the user choose two different passwords (or PINs), the first for normal operation and second for emergencies (i.e., it also sets off an alarm). This simple approach no longer works for PUFs because of the noisy nature of their responses. That is, the server will need to send the appropriate helper data P prior to knowing what password is being used; sending two helpers would be a tipoff to the coercer. We solve this problem by splitting each password (real and false) in two parts: the first part is identical in both passwords and it used by PUF to compute its challenge and response. The second halves are different, but the PUF is not queried on their values.

Enroll :

1. Server $\mathcal{S}$ sends challenge c to user $\mathcal{U}$ along with $\langle \mathbb{G}_q \rangle$ and generator g of $\mathbb{G}_q$.
2. User $\mathcal{U}$ sends $c, \langle \mathbb{G}_q \rangle, g, pwd_1, pwd_2, pwd_3$, where pwd_i's are three user passwords, to device $\mathcal{D}$ for a modified Gen protocol.
3. Device $\mathcal{D}$ calculates a challenge $d = H(H(c||pwd_1), \langle \mathbb{G}_q \rangle, g)$ and runs Gen on this value to obtain response r, P. $\mathcal{D}$ then sends to the user $(g^{H(r||pwd_2)}, g^{H(r||pwd_3)}, P)$.
4. User $\mathcal{U}$ forwards $(g^{H(r||pwd_2)}, g^{H(r||pwd_3)}, P)$ to server $\mathcal{S}$, which stores the information along with $c, g, \langle \mathbb{G}_q \rangle$.

Auth :

1. Server $\mathcal{S}$ sends challenge $c, \langle \mathbb{G}_q \rangle, g, P$, and a nonce N to the user $\mathcal{U}$.
2. $\mathcal{U}$ sends $(c, \langle \mathbb{G}_q \rangle, g, P, N, pwd)$ to device $\mathcal{D}$ for Rep protocol, where $pwd = pwd_1||pwd_2$ in a normal login and $pwd = pwd_1||pwd_3$ in an emergency login.
3. Device $\mathcal{D}$ splits pwd into two parts pwd' and pwd''. $\mathcal{D}$ then calculates its challenge $d = H(H(c||pwd'), g, p)$ and runs Rep on this value to obtain response r. $\mathcal{D}$ chooses a random value $v \in \mathbb{Z}_q$ and calculates $t = g^v$. $\mathcal{D}$ then calculates $c' = H(g, g^{H(r,pwd'')}, t, N)$ and $w = v - c'H(r||pwd'') \bmod q$, and sends c', w to the $\mathcal{U}$.
4. User $\mathcal{U}$ sends these values to the server $\mathcal{S}$. $\mathcal{S}$ calculates $t' = g^w g^{H(r||pwd_2)c'}$ and accepts the authentication as normal if $c' = H(g, g^{H(r||pwd_2)}, t, N)$. If this fails, then $\mathcal{S}$ calculates $t' = g^w g^{H(r||pwd_3)c'}$ and accepts the authentication as an emergency login if $c' = H(g, g^{H(r||pwd_3)}, t, N)$.

We now argue the coercer's inability distinguish between normal and emergency executions. As stated earlier, the coercer can record $\mathcal{U}$'s normal authentications prior to coercion. The communication between $\mathcal{U}$ and $\mathcal{S}$ during Auth consists of values $(c, \langle \mathbb{G}_q \rangle, g, P, N, c', w)$, where the first five are sent by the server as a challenge and the last two are the user's reply. Coercion gives the adversary the device $\mathcal{D}$ and user's password $pwd = pwd_1||pwd_3$, that he then uses in the protocol. We now formally state (the proof is omitted due to page constraints) that the adversary's view of the protocol after the coercion is indistinguishable from its view of previous invocations of the protocol by the user.

Theorem 3. *A polynomial-time coercer with access to private user data and I-PUF has negligible probability of distinguishing between normal and emergency executions.*

In the above solution, our goal was to provide an authentication mechanism where the communication during the protocol upon use of emergency password cannot be distinguished from normal communication, i.e., the observable response remains the same regardless of what password is used. The actions taken by the server, however, can be different depending on what password is used (e.g., in emergency, a silent alarm can sound at the bank and the ATM can issue marked bills). The work of Clark and Hengartner [1] ties the use of a panic password (or passwords) to the context in which this functionality is used, as well as the goals and capabilities of the adversary. It is assumed that the system design is open, in which case the adversary will be aware of the emergency capabilities of the system. The adversary is also capable of forcing the user to authenticate several times, possibly using different passwords. The adversary thus can force the user to open all (regular and panic) passwords he has. In our case,

the goals of the adversary can be to avoid detection (i.e., not trigger the alarm at the bank) or escape with unmarked money (i.e., authenticate at least once with the regular password). We refer the reader to [1] for more information on how such goals can be achieved with one or more panic passwords. The goal of this section is to provide a protocol to support emergency capabilities that can be combined with any policy the system wants to employ in terms of how to use and respond to panic passwords.

One weakness of our protocol is that the adversary could force a user to reveal two passwords, and then choose one of the passwords at random. Once the user reveals multiple passwords, the adversary would then either have a 50% chance of either catching the user in a lie (if the user provided a bad password) or a 50% chance of using the non-emergency password (if the user did not provide a bad password). We leave the mitigation of this problem for future work.

6 Conclusions

In this work we describe authentication solutions based on a PUF device that provide stronger security guarantees to the user than what previously could be achieved. In particular, in our solution each user is issued a device that aids in authentication and cannot be copied or cloned. We ensure that: (i) the device alone is not sufficient for authenticating; (ii) the user must have the device in order to successfully authenticate; (iii) anyone with complete access to the authentication data at the server side and the device itself is still unable to impersonate the user (even if the access to the device is possible prior to account setup). These guarantees hold in the random oracle model.

As another contribution of this work, we add protective mechanisms to the protocol that allow institutions to quickly recognize attacks when a user is coerced into revealing her secrets. We allow the user to have an alternative secret that triggers an alarm at the corresponding institution, but allows for successful authentication in such a way that the adversary is unable to distinguish between protocol executions that use the regular and alternative secrets.

A future direction of research is to achieve similar results, but without the random oracle model. More broadly, there is a need for a systematic investigation of the implications of PUFs for security functionalities other than authentication, such as fighting piracy, policy enforcement, tamper-resistance, and anti-counterfeiting.

Acknowledgments

The authors thank the anonymous reviewers for their comments and useful suggestions.

References

1. Clark, J., Hengartner, U.: Panic passwords: Authenticating under duress. In: USENIX Workshop on Hot Topics in Security, HotSec 2008 (2008)
2. Suh, G., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: DAC, pp. 9–14 (2007)

3. Ozturk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: IEEE International Conference on Pervasive Computing and Communications, pp. 170–178 (2008)
4. Simpson, E., Schaumont, P.: Offline hardware/software authentication for reconfigurable platforms. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 311–323. Springer, Heidelberg (2006)
5. Guajardo, J., Kumar, S., Schrijen, G.J., Tuyls, P.: Physical unclonable functions and public-key crypto for fpga ip protection. In: International Conference on Field Programmable Logic and Applications, pp. 189–195 (2007)
6. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic pUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
7. Tuyls, P., Batina, L.: RFID-tags for anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006)
8. Bolotnyy, L., Robins, G.: Physically unclonable function-based security and privacy in rfid systems. In: IEEE International Conference on Pervasive Computing and Communications (PerCom 2007), pp. 211–220 (2007)
9. Batina, L., Guajardo, J., Kerins, T., Mentens, N., Tuyls, P., Verbauwhede, I.: Public-key cryptography for rfid-tags. In: Pervasive Computing and Communications Workshops, pp. 217–222 (2007)
10. Hammouri, G., Sunar, B.: PUF-HB: A tamper-resilient HB based authentication protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 346–365. Springer, Heidelberg (2008)
11. Hammouri, G., Ozturk, E., Birand, B., Sunar, B.: Unclonable lightweight authentication scheme. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 33–48. Springer, Heidelberg (2008)
12. Park, Y.M., Park, S.K.: Two factor authenticated key exchange (TAKE) protocol in public wireless LANs. IEICE Transactions on Communications E87-B(5), 1382–1385 (2004)
13. Pietro, R.D., Me, G., Strangio, M.: A two-factor mobile authentication scheme for secure financial transactions. In: International Conference on Mobile Business (ICMB 2005), pp. 28–34 (2005)
14. Bhargav-Spantzel, A., Sqicciarini, A., Modi, S., Young, M., Bertino, E., Elliott, S.: Privacy preserving multi-factor authentication with biometrics. Journal of Computer Security 15(5), 529–560 (2007)
15. Stebila, D., Udupi, P., Chang, S.: Multi-factor password-authenticated key exchange. Technical Report ePrint Cryptology Archive 2008/214 (2008)
16. Boyen, X.: Reusable cryptographic fuzzy extractors. In: ACM Conference on Computer and Communications Security (CCS 2004), pp. 82–91 (2004)
17. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security (CCS 2002), pp. 148–160 (2002)
18. Skoric, B., Tuyls, P.: Secret key generation from classical physics. Philips Research Book Series (2005)
19. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
20. Chaum, D., Evertse, J.H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
21. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)