Reliable Secret Sharing With Physical Random Functions

Marten van Dijk, Daihyun Lim, Srinivas Devadas

2004

Computation Structures Group
Memo 475

Reliable Secret Sharing With Physical Random Functions[*]

Marten van Dijk[†] Daihyun Lim Srinivas Devadas
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
{marten,daihyun,devadas}@mit.edu

ABSTRACT

A Physical Random Function (PUF) is a random function that maps challenges to responses and that can only be evaluated with the help of a complex physical system. It stores key material as a combination of large amounts of hard to measure physical state. If a PUF can only be accessed via an algorithm that is physically linked to the PUF in an inseparable way (i.e., any attempt to circumvent the algorithm will lead to the destruction of the PUF), then it can be used to establish a shared secret key between a remote user and a physical device with the PUF. Once established, the secret key can be used for a wide range of applications including certified execution and software licensing.

A practical implementation of a PUF does not immediately lead to a function; the responses are noisy. To make the PUF reliable, together with the challenge, extra redundant information is provided to the PUF. The redundant information is used to correct the noise and generate the shared key. The input to the PUF is transmitted over a public channel, hence, any adversary learns the redundant information and may (in combination with information obtained by experiments with PUFs) distill knowledge about the shared key.

To make the PUF *securely* reliable, we introduce one-pass protocols which can be used for certified execution of a program with encrypted input. We show that the existence of such protocols is equivalent to the existence of fuzzy extractors. We present a practical example based on experiments with chip realizations of silicon PUFs (SPUFs). Finally, we show how responses can be reused in identifying and authenticating SPUFs while staying resistant against replay attacks.

[*]This work was funded by Acer Inc., Delta Electronics Inc., HP Corp., NTT Inc., Nokia Research Center, and Philips Research under the MIT Project Oxygen partnership.

[†]Visiting researcher from Philips Research, Prof Holstlaan 4, Eindhoven, The Netherlands.

Keywords

fuzzy extractors, secret key agreement, Reed-Solomon codes, pseudo random functions, physical random functions

1. INTRODUCTION

1.1 Physical Random Functions

Recently Gassend et al. [8] introduced the concept of a Physical Random Function (PUF) (which stands for Physical Unclonable Function) to be a function that maps challenges to responses, that is embodied by a physical device, and that verifies the following properties:

1. Easy to evaluate: The physical device is easily capable of evaluating the function in a short amount of time.

2. Hard to characterize: From a polynomial number of plausible physical measurements (in particular, determination of chosen challenge-response pairs), an attacker who no longer has the device, and who can only use a polynomial amount of resources (time, matter, etc...) can only extract a negligible amount of information about the response to a randomly chosen challenge.

In this definition, the terms short and polynomial are relative to the size of the device, which is the security parameter. In particular, short means linear or low degree polynomial. The term plausible is relative to the current state of the art in measurement techniques and is likely to change as improved methods are devised.

The PUF contains the key material as a combination of large amounts of hard to measure physical state as opposed to security based on the difficulty of reading out digital keys stored in registers. The key is determined by random manufacturing variations as opposed to a secret chosen by a trusted third party. How to securely guarantee the reliability of PUFs is an open problem (as opposed to the guaranteed reliability of digital computation).

In [20, 19] PUFs were referred to as Physical One Way Functions (POWFs), and realized using 3-dimensional microstructures and coherent radiation. This terminology is confusing because PUFs do not match the standard meaning of one way functions [16]. A PUF is a one-way function in the sense that it is hard to reconstruct the physical system from challenge-response pairs. However, unlike a one-way function, a PUF does not require going from the response to the challenge to be hard. For a PUF, all that matters is that going from a challenge to a response without using the device is hard.

In [8] silicon PUFs (SPUFs) are introduced. Individual integrated circuits (ICs) are identified based on a prior delay characterization of the IC. While IC's can be reliably mass-manufactured to have identical digital logic functionality, each IC is unique in its delay characteristics due to inherent variations in manufacturing across different dies, wafers, and processes. In [10] SPUFs are used to identify and authenticate keycards.

1.2 Controlled Physical Random Functions

Gassend et al. [9] defined a PUF to be controlled (CPUF) if it can only be accessed via an algorithm that is physically linked to the PUF in an inseparable way (i.e., any attempt to circumvent the algorithm will lead to the destruction of the PUF). In particular this algorithm can be used to avoid the man-in-the-middle attack by restricting the challenges that are presented to the PUF and by limiting the information about responses that is given to the outside world.

As explained in [9] control turns out to be the fundamental idea that allows PUFs to go beyond simple authenticated identification applications and allows secret sharing with a remote user. Through secret sharing, control can be used to enable applications such as trusted third party computation, certified execution and software licensing. In these applications only a single-chip processor with a PUF needs to be trusted, where the processor implements the control/protocols restricting the access to the PUF. Since attempts to probe a processor can be made to change the PUF, therefore no active monitoring circuits are needed.

In the certified execution application a remote user Alice, who has a challenge response pair (CRP), wants to share a secret with the processor with a PUF (see [9] for CRP management protocols resistant against man-in-the-middle attacks). To obtain a shared secret, Alice transmits a challenge from a CRP in her database to the processor with the PUF. Ideally, the PUF computes a corresponding response. Based on the shared response, Alice and the processor with the PUF distill a shared secret key.

A practical implementation of a PUF in a processor does not immediately lead to a function; the responses are noisy. To correct the noise, redundant information needs to be transmitted over a public (untrusted) network between Alice and the processor with the PUF. Besides correcting the noise, the redundant information may be used by an adversary Eve (who taps the public network) to distill information about the shared secret key. This paper solves the problem of reliable secret sharing; *reliable* in the sense that Alice and the processor with a PUF share a joint key by correcting noise and *secret* in the sense that their method does not leak information about the shared key to an adversary Eve.

Previous work on CPUFs has not addressed reliable secret sharing. The protocols we present in this paper solve the problem of how to achieve reliable secret key sharing with PUFs or CPUFs.

1.3 Contributions and Organization

We introduce one- and two-pass protocols for certified execution in Section 2. This motivates the need for reliable secret sharing. The noise model in Section 3 is based on silicon PUFs (SPUFs). In Section 4 we show that the existence of one-pass protocols is equivalent to the existence of fuzzy extractors [5]. We present a practical example based on experiments with chip realizations of SPUFs. In Section 5 we show how responses can be reused in identifying and authenticating SPUFs while staying resistant against replay attacks.

2. MOTIVATION

In certified execution, a certificate is produced that guarantees that the program was run without being tampered with on a processor. Certified execution can be performed with encrypted or unencrypted inputs.

We assume that Alice has a database of challenge-response pairs (CRPs) for the PUF obtained, for example, using the management protocols of [9].

2.1 Certified Execution with Encrypted Input

In case of certified execution of a program with encrypted input, Alice needs to know the shared secret key to encrypt the input before she transmits the program with the encrypted input to the processor with the PUF. This leads to a *one-pass protocol.* In a one-pass protocol,

1. Alice asks the processor with the PUF for a certified execution of her program:
   - Alice selects and generates a secret key K,
   - Alice encrypts the input of the program with K,
   - Alice takes a CRP (C, R_A) from her database of CRPs for the PUF,
   - Alice computes some redundant information I based on key K and response R_A (and possibly some randomness), and
   - Alice transmits to the processor with the PUF the program, the encrypted input data, the challenge C, and the redundant information I.
2. The processor with the PUF performs the certified execution:
   - the processor with the PUF measures a (noisy) response R_B corresponding to challenge C,
   - the processor with the PUF distills the secret key K from R_B and the redundant information I,
   - the processor with the PUF decrypts the encrypted input and runs the program,
   - the processor with the PUF encrypts and certifies the output of the execution of the program with K, and
   - the processor with the PUF transmits the encrypted and certified output to Alice.
3. Alice verifies the certificate and decrypts the output.

We call this a one-pass protocol since a key is shared in one communication step from Alice to the processor with the PUF.

To execute a program on the processor, Alice needs to transmit a request and receive output, requiring a one-pass protocol for certified execution with encrypted input.

Since the request to the processor is transmitted over a public channel, any adversary Eve learns the redundant information. The one-pass protocol should be designed such that (in combination with information obtained by experiments with the same or other PUFs) it is only feasible to distill negligible knowledge about the shared key. As soon as the processor with the PUF has recovered a secret key K', he needs to convince himself that $K = K'$. As in operating systems, which store hashes of passwords instead of storing passwords explicitly [6, 17], a straightforward solution is to include $f(K)$ in I, where $f(.)$ is a one-way function. $f(K)$ is a commitment of K. If R_B is close to R_A then $K = K'$ and $f(K) = f(K')$. If R_B has a large distance to R_A then, according to our requirement, $K \neq K'$ and $f(K) \neq f(K')$. Of course a simple CRC check instead of a check with $f(.)$ suffices for this purpose. However, a CRC check of K reveals information about K to any Eve who receives I over the public network.

2.2 Certified Execution without Encrypted Input

In case of certified execution of a program with unencrypted input, Alice does not need to know the shared secret key before she transmits the program with the encrypted input to the processor with the PUF. This allows a *two-pass protocol*.

1. Alice asks the processor with the PUF for a certified execution of her program:
   - Alice takes a CRP (C, R_A) from her database of CRPs,
   - Alice computes some redundant information I_A based on response R_A (she may use a probabilistic algorithm which uses some randomness X_A), and
   - Alice transmits to the processor with the PUF the program, the input data, the challenge C, and the redundant information I_A.
2. The processor with the PUF performs the certified execution:
   - the processor with the PUF measures a (noisy) response R_B corresponding to challenge C,
   - the processor with the PUF selects and generates a secret key K,
   - the processor with the PUF computes some redundant information I_B based on key K, response R_B, and redundant information I_A,
   - the processor with the PUF runs the program,
   - the processor with the PUF encrypts and certifies the output of the execution of the program with K, and
   - the processor with the PUF transmits the encrypted and certified output to Alice.
3. Alice verifies the certificate and decrypts the output:
   - Alice distills the secret key K from R_A, I_A (with the randomness X_A), and the redundant information I_B, and
   - Alice checks the certificate and decrypts the output with K.

We call this a two-pass protocol since a key is shared in two communication steps from Alice to the processor with the PUF and from the processor with the PUF to Alice.

To execute a program on the processor, Alice needs to transmit a request and receive output. In order to avoid additional interaction between Alice and the processor with the PUF, a one-pass or two-pass protocol is sufficient for certified execution with unencrypted input. In this paper we only discuss the realization of one-pass protocols.

3. NOISE MODEL BASED ON SPUFS

A silicon PUF (SPUF) can be implemented by using an *arbiter circuit* [8] [10] [13]. Figure 1 shows the structure and operation of the arbiter circuit. Two signals race through two complementary delay paths which are defined by the challenge input bits (the challenge is a binary vector). At the end of the circuit, an arbiter decides which signal arrives first. The arbiter has two inputs, which are both low initially. The arbiter waits for one of the inputs to go high, at which time its output indicates which input went high first.

An arbiter circuit with 64 switch blocks was fabricated in 37 different chips using the TSMC 0.18μ process [13] (notice that each challenge has 64 bits). Experiments show that response bits are equally likely a 0 or a 1. To reduce the measurement noise, the PUF computes each response bit as the majority out of 11 repeated measurements. To quantify the delay variation across ICs, we define the interchip variation τ as the probability that the first measured response bit for a given challenge on a first chip is different from the first measured response bit for the same challenge on a second chip. The first measured response bits are called reference response bits. In [13] experiments estimate $\tau \approx 23\%$.

Measurement noise μ is defined as the probability that a newly measured response bit is different from the corresponding reference response bit (on the same chip). In [13] experiments estimate $\mu \approx 0.7\%$. If the temperature increases by 40 degrees Celsius then $\mu \approx 4.8\%$. If the power supply voltage varies with $\pm 2\%$ then $\mu \approx 3.7\%$. The effect of aging is not yet available. By using majority voting over more measurements, by restricting the challenges to those which are robust against noise due to voltage variations, and by using other tricks to compensate noise due to temperature variations, we can reduce the noise even for the worst-case environment to $\mu \approx 1\%$. If the chip is housed in a stable environment, the noise can be reduced to 0.1%.

Our statistical analysis, which estimates the accuracy of the estimated parameters μ and τ, shows that the estimated parameters μ and τ hold for millions of chips.

We propose to use the arbiter circuit in the following way.

1. We use a challenge seed to generate n challenges of 64 bits.
2. The arbiter circuit produces a single response bit for each challenge after 11 measurements and a majority vote.
3. The n response bits are grouped in a final response $R = (r_1, \ldots, r_n)$ of n bits.

In [9], given a challenge, the corresponding response R is not allowed to be used directly by the program which is being executed by the processor with the PUF. The primitive GetSecret is used to access the PUF if a challenge is given

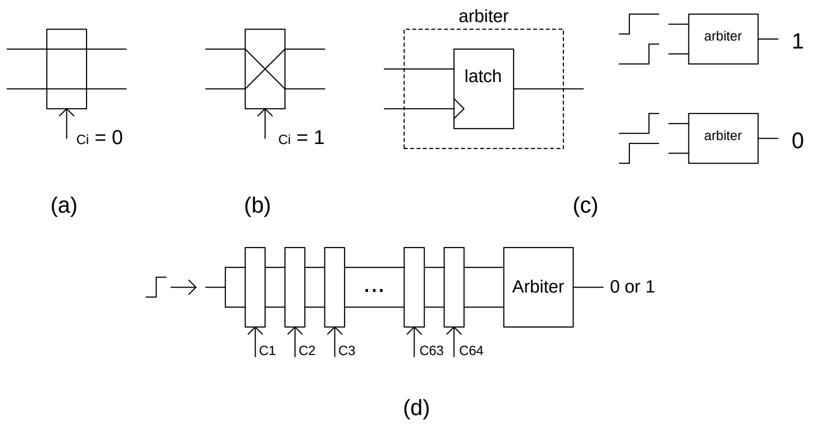

Figure 1: Each switch block defines different circuits for challenge bits of zero (a) or one (b). The arbiter decides which of the two signals reaches the arbiter first and outputs a bit (c). Multiple switch blocks together with the arbiter are combined into a long chain (d).

to the PUF (the primitive GetResponse is used to access the PUF if a so-called prechallenge is given to the PUF). In [9], the output of the GetSecret primitive is defined as $h(R, h(Program))$, where $h(.)$ is a pseudo random function. This means that a different program on the same or a different chip cannot produce information about the output of the GetSecret primitive used by the original program (this is used to avoid the man-in-the-middle attack). Also, the original program should not output information about the result of the GetSecret primitive by its design. This means that an adversary Eve has no way to predict responses by experimenting with (multiple) different arbiter chips. Also, due to the one-wayness of the pseudo random function, Eve cannot model the arbiter chip by using a database of CRPs and a linear classifier algorithm [22] as was possible in [10].

Concluding, Eve only learns the redundant information transmitted during the one- and two-pass protocols. The response $R_B = (b_1, \ldots, b_n)$ of the processor with the PUF is a noisy version of Alice's response $R_A = (a_1, \ldots, a_n)$, where the noise is binary symmetric with cross over probability μ. Alice's response R_A is uniformly distributed over $\{0,1\}^n$. The Hamming distance is defined by

$$d_H(R_A, R_B) = |\{i : a_i \neq b_i\}|.$$

Then,

$$Prob(R_B|R_A) = (1-\mu)^{n-d_H(R_A,R_B)}\mu^{d_H(R_A,R_B)}.$$

4. PROTOCOLS BASED ON FUZZY EXTRACTION

An unconditionally secure one-way protocol can be described in terms of secure sketches and fuzzy extractors. These are introduced and defined by Dodis et al. in [5]. They generalize concepts as developed in [21, 12, 7, 14, 24]. Let

$$H_\infty(A) = -\log \max_a Prob(A = a)$$

and

$$H_\infty(A|B) = -\log E_{b\leftarrow B} \max_a Prob(A = a|B = b)$$

measure the uncertainty about random variable A without and with knowledge of random variable B. Let $\mathcal{M}$ be a metric space with distance function d. An $(\mathcal{M}, m, m', t)$-secure sketch is a randomized map $SS : \mathcal{M} \rightarrow \{0,1\}^*$ such that

1. there exists a deterministic recovery function Rec such that, for all $w, w' \in \mathcal{M}$, if $d(w, w') \leq t$ then $Rec(w', SS(w))$ $= w$, and,
2. for all random variables W over $\mathcal{M}$, if $H_\infty(W) \geq m$ then $H_\infty(W|SS(W)) \geq m'$.

An $(\mathcal{M}, m, l, t, \epsilon)$ fuzzy extractor has two procedures Gen and Rep such that

1. Gen is a probabilistic generation procedure, which on input $w \in \mathcal{M}$ outputs an extracted string $r \in \{0,1\}^l$ and a public string p. For any random variable W over $\mathcal{M}$ with $H_\infty(W) \geq m$, if $(R, P) \leftarrow Gen(W)$ then the statistical distance $(\sum_{r,p} Prob(R = r, P = p) - Prob(U = r, P = p))/2 \leq \epsilon$, where U is the uniform random variable over $\{0,1\}^l$.
2. Rep is a deterministic reproduction procedure such that, for all $w, w' \in \mathcal{M}$ and $(r, p) \leftarrow Gen(w)$, if $d(w, w')$ $\leq t$ then $Rep(w', p) = r$.

The main contribution in [5] is that a $(\mathcal{M}, m, m', t)$-secure sketch can be used in combination with a pairwise independent hash function [18] (which implements privacy amplification [1, 2, 25]) to construct a $(\mathcal{M}, m, l, t, \epsilon)$ fuzzy extractor with $l = m' - 2\log(1/\epsilon)$. In this construction the public string p is equal to the possible randomness x used to select a pairwise independent hash function together with $SS(w)$,

$$p = (SS(w), x). \qquad (1)$$

We notice that the extracted string r can be used to share a (predetermined) arbitrary secret key K. We simply include the xor of r and K in the public string p (the statistical distance remains the same because K is uniformly distributed). This means that the randomness used in Gen contains the shared secret key K.

In our model $w \in \mathcal{M}$ and $w' \in \mathcal{M}$ play the role of responses R_A and R_B. The metric space $\mathcal{M} = \{0,1\}^n$ uses the Hamming distance. Since R_A is uniformly distributed over $\mathcal{M}$, $H_\infty(W) = n$, hence, $m = n$. The randomness used in *Gen* contains the shared secret key K. In our notation, on input $w = R_A$ it outputs $(r = K, p = I)$. Alice uses *Gen* to construct the redundant information. The processor uses *Rep* to produce $Rep(w' = R_B, p = I) = r = K$. In this way the fuzzy extractor models an adversary Eve, who cannot predict a response and who only knows the redundant information transmitted over the public network.

For the Hamming distance, [5] proves that given an $[n, k, 2t+1]$ code $\mathcal{C}$ and any m, ϵ, then an $(\mathcal{M}, m, l, t, \epsilon)$ fuzzy extractor can be constructed where

$$l \leq m + k - n - 2\log(1/\epsilon). \tag{2}$$

The reproduction procedure *Rep* is efficient if $\mathcal{C}$ allows decoding t errors in polynomial time. Their construction is based on a secure sketch which uses the coding technique of fuzzy commitments [12, 7]; $SS(w) = w + c$ where c is uniformly selected from $\mathcal{C}$. They shorten the length of $SS(w)$ by using syndrome decoding for linear codes; $SS(w)$ equals the syndrome wH with H the parity check matrix of $\mathcal{C}$. We notice that a similar construction can be based on [4]; for code words (w, u), $SS(w)$ equals the sequence of parity symbols u.

Let $\mathcal{C}$ be a binary BCH code of length $n \leq 2^v - 1$ and designed distance $\delta = 2t + 1$ [15]. Then the dimension k of $\mathcal{C}$ is at least equal to $2^v - 1 - vt = n - vt$ and the minimum distance of $\mathcal{C}$ is at least δ. For our SPUF application, we wish a security of $\epsilon = 2^{-160}$ and we want to generate a secret key of $l = 160$ bits. From (2) and $m = n$ we infer that we need $k \geq 480$. This is implied by $n - vt = 480$. Take $v = 11$, $t = 120$, and $m = n = 1800 \leq 2047 = 2^v - 1$. This gives a $(\{0,1\}^{1800}, 1800, 160, 120, 2^{-160})$ fuzzy extractor.

Let

$$Q(z) = \int_{x=z}^{\infty} \frac{e^{-x^2/2}}{\sqrt{2\pi}}\, \mathrm{d}x.$$

For $z > 0$, $Q(z) \approx e^{-z^2/2}/(\sqrt{2\pi}z)$. The processor with the PUF can reproduce K with *Rep* if $d_H(R_A, R_B) \leq t$. The probability that $d_H(R_A, R_B) \geq t + 1$ is equal to

$$\sum_{j=0}^{n-(t+1)} \binom{n}{j} (1-\mu)^j \mu^{n-j} \approx Q\left(\frac{(1-\mu)n - (n - (t+1))}{\sqrt{\mu(1-\mu)n}}\right). \tag{3}$$

In the worst case environment $\mu = 4.8\%$. Hence, the probability that the processor with the PUF cannot reproduce K with *Rep* is approximately $Q(3.78) \approx 10^{-4}$. The probability that a different processor with the PUF is able to reproduce K is equal to

$$\sum_{j=0}^{t} \binom{n}{j} (1-\tau)^{n-j} \tau^j \approx Q\left(\frac{\tau n - t}{\sqrt{\tau(1-\tau)n}}\right).$$

This equals $\approx 2^{-200}$ for $\tau = 23\%$. We achieve reliable secret key sharing with responses of 1800 bits or 225 bytes.

A multiple-bit response can be generated by a single-output arbiter by starting with a challenge seed and using a pseudo-random number generator such as a linear or non-linear feedback shift register to generate 1800 challenges from the starting seed. In this manner, we do not need to send 1800 different challenges to the PUF. For faster response generation, 16 or more arbiter circuits can be used in parallel.

We notice that the Guruswami-Sudan decoding algorithm [11] produces a list of closest code words. The correct code word can be picked from the list by checking the commitment talked about in Section 2.1. Hence, there exists a $t' > t$ such that the processor with the PUF can reproduce K for $d_H(R_A, R_B) \leq t'$. This allows us to choose δ/n smaller while keeping the same security and reliability. This leads to shorter responses.

As a final remark we mention that in our model a one-pass protocol is also achieved if we let w play the role of (K, R_A), the concatenation of the key K of b bits and response R_A, and w' play the role of $(?, R_A)$, the concatenation of b erasures with response R_B. In this case we need an t-errors and b-erasures decoding algorithm for $\mathcal{C}$ (we choose $\mathcal{M} = \{0,1\}^{b+n}$ and $\mathcal{C}$ an $[b + n, k, 2t + 1 + b]$ code, l stays the same and we increase m with b).

5. IDENTIFICATION AND AUTHENTICATION

If Alice wants to identify or check the authenticity of a smartcard or keycard with the PUF, then she may simply compare R_A and R_B and compute their Hamming distance. If the distance is small enough, Alice identifies and recognizes the card with the PUF. In order to compare R_A and R_B, Alice needs to receive R_B from the card with the PUF over an untrusted communication line. To avoid replay attacks Alice should not reuse the corresponding CRP (C, R_A). She should remove (C, R_A) from her database.

If the smartcard or keycard has some processing power then Alice can use certified execution to identify the card. This allows her to reuse her CRPs which is interesting for PUFs which do not possess many CRPs like optical PUFs. For example, Alice lets the card execute the following identification program.

```
K = shared key from the one-pass protocol

begin program
  Nonce = ...;
  Output(Certify(Nonce)with(K));
end program
```

To avoid replay attacks the nonce should be chosen differently for each certified execution based on the response R_A.

We notice that in related work [3] the privacy of noisy continuous data is analyzed.

5.1 Arbiter Circuit

In card applications we cannot expect much processing power. In particular, the GetSecret primitive may not be implemented. If not, an adversary Eve may predict responses, see Section 3. For example, Eve may use multiple cards which are in her possession. Each card generates a response bit corresponding to challenge C. By using a majority vote over all generated response bits, Eve predicts the response bit of the card she wants to impersonate. A more powerful method [10] uses a linear classifier algorithm [22] and a database of CRPs (obtained during a period in which Eve had control over the card, in this period Eve was not able to clone the card because PUFs are unclonable). This allows Eve to predict a response $R_E = (e_1, \ldots, e_n)$ with

$\eta = Prob(a_i \neq e_i) \approx 3\%$ for the arbiter circuit. This gives

$$Prob(R_E|R_A) = (1-\eta)^{n-d_H(R_A,R_E)}\eta^{d_H(R_A,R_E)}.$$

Let us assume that the measurement noise is $\mu = Prob(a_i \neq b_i) \approx 1\%$ in a stable environment.

Eve knows both the redundant information I as well as R_E. This can be modeled as public information $p = (I, R_E)$. We recall that $w \in \mathcal{M} = \{0,1\}^n$ plays the role of R_A, let $w'' \in \mathcal{M}$ play the role of R_E. See Eqn. (1), if we again wish to use a fuzzy extractor constructed from a secure sketch, then the redefined public information implies that the new $SS'(w)$ needs to be defined as the (old) $SS(w)$ together with w'', where $Prob(w''|w) = (1-\eta)^{n-d_H(w,w'')}\eta^{d_H(w,w'')}$. Recall that $w = R_A$ is uniformly distributed on $\mathcal{M}$, hence, $m = H_\infty(W) = n$.

Let us consider the construction of a secure sketch with $SS(w) = w+c$ where c is uniformly selected from a $[n, k, 2t+1]$ code $\mathcal{C}$. We need to compute the parameter m' of the new $(\mathcal{M}, m = n, m', t)$-secure sketch defined by $SS'(w)$. By definition, m' is chosen as a lower bound on $H_\infty(W|SS'(W)) = H_\infty(W|SS(W), W'')$. In Appendix A we prove that

$$m' = k - n(1 + (1-\alpha)\log(1-\eta) + \alpha \log \eta) \qquad (4)$$

gives a lower bound where

$$h(\alpha) \geq 1 - k/n \text{ and } \eta \geq \alpha \qquad (5)$$

with $h(x) = -(1-x)\log(1-x) - x\log x$ defined as the binary entropy function. The new secure sketch leads to a $(\mathcal{M}, n, l, t, \epsilon)$ fuzzy extractor where

$$l \leq m' - 2\log(1/\epsilon). \qquad (6)$$

If the rate k/n of code $\mathcal{C}$ is large enough, then we may choose $\alpha = \eta$ and $m' = k - (1 - h(\eta))n$. This corresponds to the intuition that R_E is the output of a binary symmetric channel with cross over probability η and capacity $(1-h(\eta))n$. If $\eta = 1/2$ then $m' = k = m + k - n$ and R_E is just a random string. This corresponds to the intuition that if R_E is an independent random string then the new secure sketch $SS'(w)$ reduces to $SS(w)$.

Let $\mathcal{C}$ be a binary BCH code of length $n \leq 2^v - 1$ and designed distance $\delta = 2t+1$. Then the dimension k of $\mathcal{C}$ is at least equal to $k \geq n - vt$. We want a security of $\epsilon = 2^{-160}$ and we want to generate a secret key K of $l = 160$ bits. From (6) we infer that we need $m' \geq 480$. From $k \geq n - vt$, (4), and (5), we infer that $m' \geq 480$ is implied by

$$-((1-\alpha)\log(1-\eta) + \alpha\log\eta)n - vt = 480$$

with $-h(\alpha) \geq vt/n$ and $\eta = 3\% \geq \alpha$. Take $\alpha = \eta$ (we have not computed the optimal value α), then $0.194n - vt = 480$, that is, $n = 2474.2 + 5.15vt$. From (3) we infer that the card with the PUF cannot reproduce K with probability

$$Q\left(\frac{(1-\mu)n - (n-(t+1))}{\sqrt{\mu(1-\mu)n}}\right) = Q\left(\frac{t(1-0.0515v) - 23.74}{\sqrt{0.0510vt + 23.50}}\right).$$

Take $v = 16$, $t = 608$, and $n = 52574 \leq 65536 = 2^v - 1$. This gives a $(\{0,1\}^{52574}, 52574, 160, 608, 2^{-160})$ fuzzy extractor with a probability of $Q(3.64) \approx 10^{-4}$ that the card with the PUF cannot reproduce K. We achieve reliable identification and authentication with responses of 52574 bits or 6572 bytes. For card applications the required length of the responses is too large and the decoding complexity of such long BCH codes is too much. For this reason we investigate new circuits which are harder to predict and lead to larger η in the next subsection.

5.2 Non-Linear Circuits

The arbiter circuit is too predictable by using a linear classifier algorithm [22] and a database of CRPs. For this reason we investigate circuits which are less linear. Figure 2 depicts the feed-forward arbiter circuit of [13], where one or more challenge bits are determined by the racing result in an intermediate stage instead of being provided by Alice. The linear classifier algorithm does not work at all, that is Eve cannot predict a response of the feed-forward arbiter circuit based on such an algorithm. In [13] experimental results estimate $\tau = 40\%$ and $\mu = 2.2\%$ for a stable environment.

Even though a linear modeling attack is not possible, Eve may use multiple cards which are in her possession. Each card generates a response bit corresponding to challenge C. By using a majority vote over all generated response bits, Eve predicts the response bit of the card she wants to impersonate. This allows Eve to predict a response $R_E = (e_1, \ldots, e_n)$. Let $\eta = Prob(a_i \neq e_i)$.

Suppose that Eve uses $2j+1$ cards to generate $2j+1$ response bits. Let P_i be the probability that i of the $2j+1$ response bits are 0, hence, the other $2j+1-i$ response bits are 1. If i bits are 0 and if $i \leq j$, then a majority vote mispredicts each of these i bits. If i bits are 0 and if $i \geq j+1$, then a majority vote mispredicts each of other $2j+1-i$ bits. This means that

$$\eta \approx \sum_{i=0}^{j} P_i \frac{i}{2j+1} + \sum_{i=j+1}^{2j+1} P_i \frac{2j+1-i}{2j+1} = \sum_{i=1}^{j}(P_i + P_{2j+1-i})\frac{i}{2j+1}.$$

If i bits are 0, then exactly $i(2j+1-i)$ pairs of bits correspond to a pair in which the bits differ/vary. Since the total number of pairs is equal to $(2j+1)(2j)/2$,

$$\tau \approx \sum_{i=0}^{2j+1} P_i \frac{i(2j+1-i)}{(2j+1)j} = \sum_{i=1}^{j}(P_i + P_{2j+1-i})\frac{i}{2j+1}\frac{2j-(i-1)}{j}$$

$$\leq 2\eta.$$

This proves that $\eta \geq \tau/2 = 20\%$. Experimental results with 21 feed-forward arbiter circuits showed $\eta \geq 24.6\%$.

By using the same method as explained in the previous subsection, we obtain $n = 664.8 + 1.39vt$ (for $\alpha = \eta = 20\%$) and

$$Q\left(\frac{(1-\mu)n - (n-(t+1))}{\sqrt{\mu(1-\mu)n}}\right) = Q\left(\frac{t(1-0.0306v) - 14.63}{\sqrt{0.0299vt + 14.30}}\right).$$

Take $v = 11$, $t = 55$, and $n = 1506 \leq 2048 = 2^v - 1$. This gives a $(\{0,1\}^{1506}, 1506, 160, 55, 2^{-160})$ fuzzy extractor with a probability of $Q(3.80) \approx 10^{-4}$ that the card with the PUF cannot reproduce K. We achieve reliable identification and authentication with responses of 1506 bits or 189 bytes. This is a practical solution for card applications. The required length of the responses is small enough and the decoding complexity is not too large.

6. CONCLUDING REMARKS

We have introduced one- and two-pass protocols for certified execution. For PUFs and CPUFs we showed how the theory of fuzzy extractors can be used to reliably share a

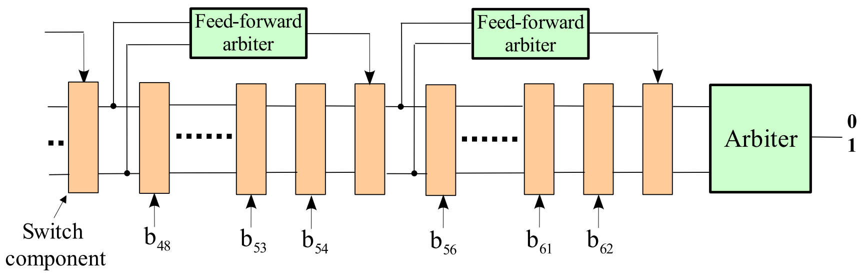

Figure 2: Feed-forward arbiter circuit.

key. For a processor with the PUF we argued that an adversary cannot predict responses. Based on experimental results with SPUFs based on the arbiter circuit under extreme environmental variations, we constructed a practical protocol in which the responses are 225 bytes.

For the identification and authentication of a smartcard with the PUF we want to be able to reuse responses. In order to do so, we use certified execution. However, in this scenario it is reasonable to assume that an adversary can predict responses. This leads to extra side information from which information about the secret key can be distilled. To reliably identify and authenticate a smartcard with an SPUF, the SPUF should be based on the feed-forward arbiter circuit. Based on experimental results with SPUFs based on the feed-forward arbiter circuit in a stable environment, we constructed a practical protocol in which the responses are 189 bytes.

The protocol for identification also works in biometrics applications where, in stead of a response of a PUF, we measure a biometric template, for example, a fingerprint. The lower bound presented in the appendix computes how much information is leaked from, for example, a latent fingerprint on a glass [23]. This leads to schemes which are provably secure against leaked fingerprints.

7. REFERENCES

[1] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer. Generalized Privacy Amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[2] A. B. Carleial and M. E. Hellman. A note on Wyner's wiretap channel. *IEEE Transactions on Information Theory*, IT-23:387–390, 1977.

[3] L. Csirmaz and G. O. H. Katona. Geometrical cryptography. In *Proc. International Workshop on Coding and Cryptography*, 2003.

[4] G. I. Davida, Y. Frankel, and B. J. Matt. On Enabling Secure Application Through Off-Line Biometric Identification. In *IEEE 1998 Symposium on Research in Security and Privacy*, 1998.

[5] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - Eurocrypt 2004*, 2004.

[6] D. C. Felmeier and P. R. Karn. UNIX password security - ten years later. In *Advances in Cryptology - Crypto'89*, volume LNCS 435, pages 44–63, 2004.

[7] N. Frykholm and A. Juels. Error-tolerant password recovery. In *Proceedings of the* 8th *ACM Conference on Computer and Commmunications Security (CCS'01)*, pages 1–8, 2001.

[8] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon Physical Random Functions . In *Proceedings of the Computer and Communication Security Conference*, November 2002.

[9] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *Proceedings of* 18th *Annual Computer Security Applications Conference*, December 2002.

[10] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Delay-based circuit authentication and applications. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, March 2003. Extended version to appear in Concurrency and Computation: Practice and Experience.

[11] V. Guruswami and M. Sudan. Improved Decoding of Reed-Solomon Codes and Algebraic-Geometry Codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[12] A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In *Proceedings of the* 6th *ACM Conference on Computer and Commmunications Security*, pages 28–36, 1999.

[13] J.-W. Lee, D. Lim, B. Gassend, E. G. Suh, M. van Dijk, and S. Devadas. A Technique to Build a Secret Key in Integrated Circuits with Identification and Authentication Applications. In *Proceedings of the IEEE VLSI Circuits Symposium*, June 2004.

[14] J.-P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Proceedings of the* 4th *International Conference on Audio- and Video-Based Biometric Person Authentication*, 2003.

[15] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* Elsevier, 1977.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[17] R. Morris and K. Thompson. Password Security: A Case History. *Communications of the ACM*, 22:594–597, 1979.

[18] N. Nissan and D. Zuckerman. Randomness is Linear in Space. *Journal for Computer and System Sciences (JCSS)*, 52(1):43–52, 1996.

[19] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical One-Way Functions. *Science*, 297:2026–2030, 2002.

[20] P. S. Ravikanth. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, 2001.

[21] V. Shoup. A Proposal for an ISO Standard for Public Key Encryption, 2001.

[22] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 2003.

[23] U. Uludag and A. Jain. Attacks on Biometric Systems: a Case Study in Fingerprints. In *Proc. SPIE-EI 2004, Security, Seganography and Watermarking of Multimedia Contents VI*, 2004.

[24] E. Verbitskiy, P. Tuyls, D. Denteneer, and J.-P. Linnartz. Reliable Biometric Authentication with Privacy Protection. In *Proceedings of the IEEE Benelux Symp. on Information Theory*, 2003.

[25] S. Wolf. *Information-theoretically and computationally secure key agreement in cryptography*. PhD thesis, ETH dissertation No. 13138, ETH Zürich, 1999.

APPENDIX

A. LOWER BOUND

In this appendix we provide a lower bound on

$$H_\infty(W|SS'(W)) = H_\infty(W|SS(W;C),W''),$$

where

- W is uniformly distributed on $\mathcal{M} = \{0,1\}^n$,
- $SS(w;c) = w + c$ with c uniformly selected from a (not necessary linear) code $\mathcal{C} \subseteq \mathcal{M}$ with 2^k code words, and
- W'' is a random variable defined by

$$Prob(W'' = w''|W = w) = (1-\eta)^{n-d_H(w,w'')}\eta^{d_H(w,w'')}. \quad (7)$$

Since W is uniformly distributed on $\mathcal{M}$, (7) implies that W'' is also uniformly distributed on $\mathcal{M}$ and statistically independent of $W - W''$, in particular,

$$Prob(W - W'' = w - w'') = Prob(W'' = w''|W = w).$$

Notice that

$$\begin{aligned} & Prob(W = w|SS(W;C) = v, W'' = w'') \\ = \;& Prob(W = w|W = v - C, W'' = w'') \end{aligned}$$

is equal to

$$Prob\left(W - W'' = w - w'' \;\middle|\; \begin{array}{l} W - W'' = v - w'' - C, \\ W'' = w'' \end{array}\right). \quad (8)$$

The random variables W'' and $W - W''$ are statistically independent such that we may ommit $W'' = w''$ in (8). Together with Bayes rule, probability (8) is equal to

$$\frac{Prob(W - W'' = w - w'', W - W'' = v - w'' - C)}{Prob(W - W'' = v - w'' - C)}. \quad (9)$$

Notice that

$$\begin{aligned} & Prob(W - W'' = w - w'', W - W'' = v - w'' - C) \\ = \;& Prob(W - W'' = w - w'')Prob(C = v - w) \end{aligned}$$

in (9) and that

$$Prob(C = v - w) = \begin{cases} 2^{-k}, & \text{if } v - w \in \mathcal{C}, \\ 0, & \text{if } v - w \notin \mathcal{C}. \end{cases}$$

This proves that probability (9) is maximized for w if $v - w$ minimizes

$$d_H(C, v - w'') = \min_{c \in C} d_H(c, v - w'').$$

The maximizing probability is equal to

$$\frac{2^{-k}(1-\eta)^{n-d_H(C,v-w'')}\eta^{d_H(C,v-w'')}}{Prob(W - W'' = v - w'' - C)}. \quad (10)$$

This proves that $H_\infty(W|SS(W;C),W'')$ equals $-\log$ of the expectation $E_{v \leftarrow SS(W;C), w'' \leftarrow W''}$ of (10).

Notice that

$$\begin{aligned} & Prob(SS(W;C) = v, W'' = w'') \\ = \;& Prob(W = v - C, W'' = w'') \\ = \;& Prob(W - W'' = v - w'' - C, W'' = w''). \end{aligned} \quad (11)$$

Since the random variables W'' and $W - W''$ are statistically independent, probability (11) is equal to the product

$$Prob(W - W'' = v - w'' - C)Prob(W'' = w'').$$

This proves that the expectation

$$E_{v \leftarrow SS(W;C), w'' \leftarrow W''} \frac{(1-\eta)^{n-d_H(C,v-w'')}\eta^{d_H(C,v-w'')}}{Prob(W - W'' = v - w'' - C)}$$

is equal to

$$\begin{aligned} & \sum_{v \in \mathcal{M}, w'' \in \mathcal{M}} \begin{array}{l} Prob(W'' = w'')\cdot \\ \quad (1-\eta)^{n-d_H(C,v-w'')}\eta^{d_H(C,v-w'')} \end{array} \\ = \;& \sum_{x \in \mathcal{M}} (1-\eta)^{n-d_H(C,x)}\eta^{d_H(C,x)}. \end{aligned} \quad (12)$$

Combining (10) and (12) gives

$$\begin{aligned} & H_\infty(W|SS(W;C),W'') \\ = \;& -\log 2^{-k} \sum_{x \in \mathcal{M}} (1-\eta)^{n-d_H(C,x)}\eta^{d_H(C,x)} \\ = \;& k - \log \sum_{x \in \mathcal{M}} (1-\eta)^{n-d_H(C,x)}\eta^{d_H(C,x)}. \end{aligned}$$

Notice that $H_\infty(W|SS(W;C),W'')$ is minimized if C is a perfect code. Such a perfect code has minimum distance $2t' + 1$, where $2^k \sum_{i=0}^{t'} \binom{n}{i} = 2^n$ or equivalently

$$t' = \alpha n \text{ with } h(\alpha) = 1 - k/n,$$

where $h(x) = -x\log x - (1-x)\log(1-x)$ is the binary entropy function. For a perfect code with $\eta > \alpha$,

$$\begin{aligned} & \sum_{x \in \mathcal{M}} (1-\eta)^{n-d_H(C,x)}\eta^{d_H(C,x)} \\ = \;& 2^k \sum_{i=0}^{\alpha n} \binom{n}{i} (1-\eta)^{n-i}\eta^i \\ \approx \;& 2^n(1-\eta)^{(1-\alpha)n}\eta^{\alpha n}. \end{aligned}$$

We obtain

$$H_\infty(W|SS'(W)) \geq k - n(1 + (1-\alpha)\log(1-\eta) + \alpha\log\eta).$$

This inequality also holds for $h(\alpha) \geq 1 - k/n$ with $\eta \geq \alpha$.