PUF-Based UC-Secure Commitment without Fuzzy Extractor

Huanzhong Huang
Department of Computer Science, Brown University
Joint work with Feng-Hao Liu
Advisor: Anna Lysyanskaya

May 1, 2013

Abstract

Cryptographic protocol constructions based on hardware-assisted tokens is one fairly new topic of research in recent years. Physically Uncloneable Functions (PUFs) are hardware tokens having interesting properties such as unpredictable and non-programmable. Previous works have shown PUFs can be used to construct secure computation protocols such as Oblivious Transfer and Commitment in the Universal Composable Framework. In this work, we propose a UC-secure commitment scheme which has a uniqueness of not relying on Fuzzy Extractors, which can be seen in every construction in previous literature.

1 Introduction

Designing cryptographic protocols that simultaneously achieve high efficiency and strong security requirements has always been an important goal in the crypto community. In recent years, a cluster of research suggest cryptographic protocol designs based on various hardware components and have achieved fruitful results.

Physically Uncloneable Functions (PUFs) are another type of hardware component that have received much attention in the community. Roughly speaking, a PUF is a hardward token that is derived through a complex physical manufacturing process that makes its behavior being unpredictable and hard to clone. By performing measurement based on physical stimuli, a PUF provides unpredictable and noisy responses and can be treated as a certain source of randomness.

The Universal Composable Security Framework (UC framework) was proposed by Cannetti [2] which aims to capture cryptographic protocol executions in complex environments such as in the real world, and provides a framwork of analysis which supports the decomposition of cryptographic tasks into basic building blocks. Roughly speaking, if a protocol $\pi^{\mathcal{F}}$ UC-realizes an ideal functionality $\mathcal{G}$ in the hybrid model with access to another ideal functionality $\mathcal{F}$, and if there is a protocol ρ which UC-realizes $\mathcal{F}$, then the composed protocol π^{ρ}, which replace the access to functionality $\mathcal{F}$ by invoking protocol ρ, UC-realizes $\mathcal{G}$.

One main contribution in [1] is that Brzuska et al. modeled PUFs in the UC framwork by giving an ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ that captures the properties of PUFs. The ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ only allows the party in possession of PUF to retrieve response, thus ensuring restricted access. PUFs can be hand overed to other parties , and the adversary is allowed a temporary access before the PUF is delivered. They also made assumptions regarding PUFs as being temper-evidence as the temper of a PUF can be detected by the receiver upon receiving it. Also in [1] Brzuska et al. present PUF-based protocols for Oblivious Transfer, Commitments, and Key Exchange. All protocols are efficient as well as UC-secure, and the security of the protocols do not rely on additional cryptographic assumptions other than those regarding PUFs.

As mentioned previously, PUF has the property of produce noisy responses, which means if we query a PUF twice based on the same stimulus, it may respond with distinct outputs. Nevertheless, the noise can be bounded, so the two responses will be close in terms of distance. In order to overcome such inconsistency in response as to make PUF as a mathematical function, Fuzzy Extractors [3] are used along with PUFs in to guarantee response consistency, as to be part of the design of the protocols in [1] and every subsequent literature.

In this work, we present a UC-secure PUF-based commitment scheme without fuzzy extractors. This result is somewhat surprising since because commitment scheme is equivalent to other secure computation schemes such as oblivious transfer, zero-knowledge proof, and coin tossing, our result implies the existence of secure computations in the UC-framework depending only on a hardware token that produces inconsistent noisy output. Also, the absence of a fuzzy extractor in the protocol design lessen the computation cost and thus improves efficiency. Another characteristic of our proposed scheme is its efficiency both in terms of communication bandwidth and in terms of the number of rounds needed for a protocol execution. In this paper we also do an investigation into the possibility of having an even more efficient scheme.

2 Background: Physically Uncloneable Functions

In this section we review the definitions of Physically Uncloneable Functions in [1]. A Physically Uncloneable Function (PUF) is a type of hardware token that is fabricated in a way that is uncontrollable even for the manufacturer which can be used as a source of randomness. A PUF evaluation involves querying the physical system with a stimulus, or a *challenge*, and in return the PUF output a noisy *response*. We call a pair of stimulus and corresponding output a *challenge/response pair* (CRP). It is worth noting that the outputs of a PUF being noisy means a PUF does not implement a mathematical function where the same output is guaranteed when performing two evaluations on the same input. However, the noise can be bounded so that the two responses are still close in terms of Hamming distance.

2.1 Definition and Security of PUFs

A PUF-family $\mathcal{P}$ consists of two not necessarily efficient algorithms Sample and Eval. The Sample algorithm does the index sampling by returning an index id on input of a security parameter. The evaluation algorithm Eval takes a challenge c and reponds with output r corresponds to PUF evaluation.

Definition 1 (Physically Uncloneable Functions) *Let rg be length of the range of the PUF respnses, let d_{noise} be an upperbound on noise in the number of bits of PUF responses. $\mathcal{P} = (\mathsf{Sample}, \mathsf{Eval})$ is a family of (rg, d_{noise})-PUFs if it satisfies the following properties:*

Index Sampling. *Let $\mathcal{I}_\lambda$ be an index set. The sampling algorithm Sample takes input a security parameter 1^λ, outputs an index $\mathsf{id} \in \mathcal{I}_\lambda$. Each $\mathsf{id} \in \mathcal{I}_\lambda$*

corresponds to a set of distributions $\mathcal{D}_{\mathsf{id}}$. For each challenge $c \in \{0,1\}^\lambda$, $\mathcal{D}_{\mathsf{id}}(c)$ is a distribution on $\{0,1\}^{rg(\lambda)}$ in $\mathcal{D}_{\mathsf{id}}$. Neither do we require the index sampling is efficient, nor do we require elements in $\mathcal{D}_{\mathsf{id}}$ can be efficiently sampled.

Evaluation. *The evaluation algorithm* Eval *takes input $(1^\lambda, \mathsf{id}, c)$, where $c \in \{0,1\}^\lambda$ is a challenge, outputs $r \in \{0,1\}^{rg(\lambda)}$, according to the distribution $\mathcal{D}_{\mathsf{id}}(c)$, as a response.* Eval *need not to be efficient.*

Bounded Noise. *For all $\mathsf{id} \in \mathcal{I}_\lambda$, for all challenges $c \in \{0,1\}^\lambda$, we have that when running $\mathsf{Eval}(1^\lambda, \mathsf{id}, c)$ twice, then the Hamming distance of the respective outputs r_1, r_2 is bounded by $d_{\mathsf{noise}}(\lambda)$.*

The main security definition of PUFs is unpredictability. Namely, on input a new challenge c, it should be hard to predict the corresponding response. The notion can be captured by requiring the response to have some significant amount of intrinsic entropy. More formally, when one has measured a PUF on a challenges $c_1, ..., c_l$, as long as a new challenge c is not close to each measured challenges, the response corresponds to c from the PUF will have a certain average min-entropy.

Definition 2 (Unpredictability) *We call a (rg, d_{noise})-PUF family $\mathcal{P} = (\mathsf{Sample}, \mathsf{Eval})$ is $(d_{\mathsf{min}}(\lambda), m(\lambda))$-unpredictable if for any $c \in \{0,1\}^\lambda$ and any challenge list $\mathcal{C} = (c_1, ..., c_l)$, if $\mathsf{dis}(c, c_k) \geq d_{\mathsf{min}}(\lambda)$ for all $c_k \in \mathcal{C}$, then the average min-entropy satisfies $\tilde{H}_\infty(\mathsf{PUF}(c)|\mathsf{PUF}(\mathcal{C})) \geq m(\lambda)$, where $\tilde{H}_\infty(\mathsf{PUF}(c)|\mathsf{PUF}(\mathcal{C}))$ is the average min-entropy of $\mathsf{PUF}(c)$ conditioned on the measurements of challenge list $\mathcal{C}$. Such a PUF-family is called a $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$-PUF family.*

2.2 PUFs in UC framework

Same as the definition of PUFs, we do not alter the modeling of PUFs in the UC framework in [1]. Basically, the ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ handles the operations of (1) issuing PUFs, (2) evaluating a PUF on some specified input only for the right holder, (3) the transfer of a PUF to another specified party, and (4) allows the adversary to query the PUF during the transition. The reader can refer to [1] for more detailed and formal definition of the $\mathcal{F}_{\mathsf{PUF}}$ functionality. We note that the definition requires that PUFs are temper-evidence, so that the adversary cannot replace a PUF by a fake or malicious one.

3 PUF-based Commitement Scheme

A commitment scheme is a two-party protocol between a sender (or committer) and a receiver which consists two phases. In the first phase, called the commitment phase, the sender first sends (possibly through some interaction with the receiver) a commitment of some value to the receiver. Subsequently, in the second phase, called the decommitment (or opening) phase, the sender reveals the committed value by sending to the receiver some opening. We require that: 1. the commitment reveals nothing about the value, which is called the property of hiding. 2. it is infeasible for the sender to come up with another opening so that the commitment can be opened to another value, which is also called the property of binding.

3.1 The Commitment Scheme Ideal Functionality

The ideal functionality $\mathcal{F}_{\mathsf{com}}$ is defined as to emulate the aforementioned notion of a commitment scheme: $\mathcal{F}_{\mathsf{com}}$ first receives input $(\mathsf{commit}, \mathsf{sid}, \mathsf{ssid}, \mathsf{msg})$ from committer P_i where msg is the value that it wishes to commit to. After some verification of the validity of the identities and the session identifiers, $\mathcal{F}_{\mathsf{com}}$ records msg, sends to the receiver P_j a delayed output $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid})$, and thus completes the commitment phase.

In the decommitment phase, P_i sends $(\mathsf{open}, \mathsf{sid}, \mathsf{ssid})$ to $\mathcal{F}_{\mathsf{com}}$. Upon receiving the message from P_i, $\mathcal{F}_{\mathsf{com}}$ first checks there indeed exists a value msg, then sends a delayed output $(\mathsf{open}, \mathsf{sid}, \mathsf{ssid}, \mathsf{msg})$ to P_j.

The adversary can corrupts the committer by sending $(\mathsf{corrupt} - \mathsf{committer}, \mathsf{sid}, \mathsf{ssid})$ to $\mathcal{F}_{\mathsf{com}}$. Upon receiving the instruction, $\mathcal{F}_{\mathsf{com}}$ reveals the recorded value msg to the adversary $\mathcal{S}$. Furthermore, $\mathcal{F}_{\mathsf{com}}$ allows the adversary to modify the committed value if the $\mathsf{receipt}$ message has not yet delivered to P_j.

The specific ideal functionality for commitment is given in Figure 1.

3.2 Commitment Scheme

Our commitment scheme depends on a PUF and an authentication channel and does not depend on a fuzzy extractor. In the setup phase, the sender evaluates the PUF for a set of randomly chosen challenges and stores every CRPs in a list $\mathcal{L}$. The sender then hand over the PUF to the receiver.

$\mathcal{F}_{\mathsf{com}}$ is parameterized by an integer N as the maximum number of legitimate commit executions, and runs with parties P_i, P_j, and adversary $\mathcal{S}$. Once it sets P_i and P_j be the corresponding sender and receiver by receiving the first commit-input from P_i, it ignores any following input in which P_i and P_j are not the corresponding sender and receiver.

- Upon receiving input $(\mathsf{commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{msg})$ from party P_i, $\mathcal{F}_{\mathsf{com}}$ records msg, sends a delayed output $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid})$ to party P_j.
- Upon receiving input $(\mathsf{open}, \mathsf{sid}, \mathsf{ssid})$ from party P_i, $\mathcal{F}_{\mathsf{com}}$ checks if a value msg has been recorded. If the answer is positive, it sends to P_j a delayed output $(\mathsf{open}, \mathsf{sid}, \mathsf{ssid}, \mathsf{msg})$. Otherwise it does nothing.
- Upon receiving the input $(\mathsf{corrupt} - \mathsf{committer}, \mathsf{sid}, \mathsf{ssid})$ from the adversary $\mathcal{S}$, $\mathcal{F}_{\mathsf{com}}$ sends the recorded msg to $\mathcal{S}$. Furthermore, if 1. $\mathcal{S}$ provides a value msg' and 2. the $\mathsf{receipt}$ output has not yet sent to P_j, $\mathcal{F}_{\mathsf{com}}$ will change the recorded value to msg'.

Figure 1: The ideal functionality for commitment

The receiver initializes each of the protocol executions by sending two randomly generated values x_0, x_1 to the sender. The sender, upon receiving x_0 and x_1, arbitrarily picks from $\mathcal{L}$ a challenge/response pair (c, r), computes $v = c \oplus x_b$ based on the bit b the sender would like to commit to, then sends v as a commitment of b to the receiver. It can be seen that, since c is randomly chosen, x_b is statistically hidden and thus the sender's bit b is protoected by the hiding property of the protocol.

In an opening phase, the sender disclose the committed bit b, along with the a PUF response r, are both sent to the receiver. The receiver verify the validity of the decommitment by basically checking whether $v \oplus x_b$ recovers c. This can be achieved by evaluating the PUF on challenge $c' = v \oplus x_b$, and compare the response r' with r from the sender. Although the fact that PUF outputs are noisy implies r and r' are unlikely to be equal, but fortunately the noise can be bounded, and thus the receiver accepts the decommitment if $\mathsf{dis}(r, r') < d_{\mathsf{noise}}(\lambda)$. The sender can break the binding property if he can come up with a response $\bar{r}$ close enough to $\mathsf{PUF}(v \oplus x_{\bar{b}})$. By the intuitive idea of unpredictability, the only way that the sender can have $\bar{r}$ is to obtain it through evaluation of the PUF, and the probability that the sender has indeed measured $v \oplus x_{\bar{b}}$ or close enough values can be argued to be negligible.

The specific scheme is given in Figure 2. Now we give a formal proof of security of the proposed commitment scheme.

Sender P_i	session sid	Receiver P_j
$(\mathsf{init}_{\mathsf{PUF}}, \mathsf{sid}, P_i, \lambda)$		
$k = 1, ..., N : c_k \leftarrow \{0,1\}^\lambda$		
$r_k \leftarrow (\mathsf{eval}_{\mathsf{PUF}}, \mathsf{sid}, P_i, c_k)$		
$\mathcal{L} := (c_1, r_1, ..., c_l, r_l)$		
$\mathcal{C} := \emptyset$	$\xrightarrow{(\mathsf{handover}_{\mathsf{PUF}}, \mathsf{sid}, P_i, P_j)}$	$\mathcal{C} := \emptyset$
	Repeat at most N times with new ssid	
	(commitment phase)	
Input: $b \in \{0,1\}, \mathsf{sid}$		Input: sid
	$\xleftarrow{(x_0, x_1)}$	$x_0, x_1 \xleftarrow{\$} \{0,1\}^\lambda$
Draw $(c, r) \xleftarrow{\$} \mathcal{L}$		
$v := c \oplus x_b$		
$\mathsf{dis}(c, \mathcal{C}) \geq d_{\mathsf{min}}$?	$\xrightarrow{v}$	$\mathsf{dis}(v \oplus x_0, \mathcal{C}) \geq d_{\mathsf{min}}$?
$\mathsf{dis}(c \oplus x_0 \oplus x_1, \mathcal{C}) \geq d_{\mathsf{min}}$?		$\mathsf{dis}(v \oplus x_1, \mathcal{C}) \geq d_{\mathsf{min}}$?
Add c, $c \oplus x_0 \oplus x_1$ to $\mathcal{C}$		Add $v \oplus x_0$, $v \oplus x_1$ to $\mathcal{C}$
Delete (c, r) in $\mathcal{L}$		Output: $\mathsf{receipt}$
	(opening phase)	
	$\xrightarrow{(b, r)}$	$c' = v \oplus x_b$
		$r' \leftarrow (\mathsf{eval}_{\mathsf{PUF}}, \mathsf{sid}, P_j, c')$
		$\mathsf{dis}(r, r') < d_{\mathsf{noise}}(\lambda)$?
		Ouput: b

Figure 2: Commitment scheme with PUFs

Theorem 1 *Assuming* $\mathsf{PUF} = (\mathsf{Sample}, \mathsf{Eval})$ *is a family of* $(rg, d_{\mathsf{noise}}) -$ *PUFs, the proposed commitment scheme securely realizes the ideal functionality* $\mathcal{F}_{\mathsf{com}}$ *in the* $\mathcal{F}_{\mathsf{PUF}}$*-hybrid model.*

Proof: We prove the theorem by giving simulations based on separate cases involving different sets of corrupt parties. In general, for every real world PPT adversary $\mathcal{A}$, we have a simulator $\mathcal{S}$, which runs a black-box simulation of $\mathcal{A}$, simulates the transcript of honest parties from only the limited information provided by the functionality in the ideal world, so that no PPT environment $\mathcal{Z}$ can distinguish whether it is a real world execution or an ideal one. In essence, the simulator needs to come up with a legit transcript of execution when both parties are honest. Furthermore, it needs to be able to extract the committed value from a commitment when the sender is corrupt, and it has to be able to equivocate when the receiver is corrupt. We consider the same setting as in [1], where the simulator faithfully initialize a PUF and allow the environment to access the PUF when the PUF is in possession of the simulator.

Simulating the case in which both parties are honest. In this case the simulator $\mathcal{S}$ needs to come up with the transcript of an execution. In particular, it needs to come up with a real world commitment v before knowing the bit to be committed, and later comes up with a decommitment (b, r) after knowing the committed bit b. This is easy because actually $\mathcal{S}$ can just pick random strings as v as well as r. The reason why it is okay to just use random strings is simple: by the unclonability and unpredictability of PUF, the only way to verify the validity of a commitment is through a PUF measurement. However, since the environment has only limited access to PUF in this case where both parties are honest, the environment cannot, without the access of the PUF, distinguish random strings from a valid commitment/decommitment pair with non-negligible advantage over 1/2.

When the sender is corrupt. In the case where the sender P_i is corrupt whereas the receiver P_j is honest, The simulator $\mathcal{S}$ observes P_i's PUF querries (made by $\mathcal{A}$ and $\mathcal{Z}$) in the setup phase and stores all the challenge-respnse pairs in a list $\mathcal{L}$. In order to transform whatever happens in the real world into the ideal world under current corruption setting, $\mathcal{S}$ should be able to extract the commited bit b from the real world protocol execution. During the simulation, $\mathcal{S}$ draws a pair of random values (x_0, x_1) from $\{0,1\}^\lambda$ and sends them to the P_i (which is instructed by $\mathcal{A}$) in the simulation. After that, $\mathcal{A}$ will instruct P_i to send v to the receiver. At this point, the simulator looks for querries $v \oplus x_0$ and $v \oplus x_1$ in the list $\mathcal{L}$. If there exists a CRP pair $(c, r) \in \mathcal{L}$ such that $\mathsf{dis}(c, v \oplus x_0) < d_{\mathsf{min}}$, the simulator sets $b = 0$, for the case that it is $\mathsf{dis}(c, v \oplus x_1) < d_{\mathsf{min}}$, $\mathcal{S}$ sets $b = 1$. If neither of them appear on $\mathcal{L}$, $\mathcal{S}$ just picks a random b. Afterwards $\mathcal{S}$ sends $(\mathsf{commit}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, b)$

on behalf of P_i to $\mathcal{F}_{\mathsf{com}}$. It is clear the simulation only fails when later it turns out $\mathcal{S}$ had picked the wrong b. We argue that this only happens with negligible probability, in the sense that in this case, the dishonest sender P_i, intructed by $\mathcal{A}$ and $\mathcal{Z}$, has to be able to come up with a decommitment without performing a corresponding PUF measurement, which is aginst the assumption of PUF being unpredictable.

First we establish the fact that it can only happen with negligible probability that there exists a CRP pair $(c, r) \in \mathcal{L}$ such that $\mathsf{dis}(c, v \oplus x_0) < d_{\mathsf{min}}$ and $\mathsf{dis}(c, v \oplus x_1) < d_{\mathsf{min}}$, as it implies $\mathsf{dis}(v \oplus x_0, v \oplus x_1) < 2d_{\mathsf{min}}$ and $\mathsf{dis}(x_0, x_1) < 2d_{\mathsf{min}}$, which can only happen negligibly with randomly chosen x_0 and x_1. Next we establish the fact that with only negligible probability, there exists two challenge-response pair $(c_0, r_0), (c_1, r_1)$ such that $\mathsf{dis}(c_0, v \oplus x_0) < d_{\mathsf{min}}$ and $\mathsf{dis}(c_1, v \oplus x_1) < d_{\mathsf{min}}$. Because $\mathsf{dis}(c_0, v \oplus x_0) < d_{\mathsf{min}}$ and $\mathsf{dis}(c_1, v \oplus x_1) < d_{\mathsf{min}}$ implies $\mathsf{dis}(c_0 \oplus x_0 \oplus x_1, c_1) < 2d_{\mathsf{min}}$, or $\mathsf{dis}(x_0 \oplus x_1, c_0 \oplus c_1) < 2d_{\mathsf{min}}$. Since x_0 and x_1 are randomly chosen after the setup phase, it can be seen that, the probability of making a polynomial number of querries and two of them happen to be related to a specific random number is negligible, as $C(p(\lambda), 2) \times 2d_{\mathsf{min}} = (1/2)p(\lambda)(p(\lambda) - 1) \times 2d_{\mathsf{min}}$ is a negligible fraction of 2^{λ} if d_{min} is in $o(\lambda / \log \lambda)$. Based these two facts, it follows that the simulation fails when P_i, instructed by $\mathcal{Z}$ and $\mathcal{A}$, has the ability to produce a PUF output without querried the corresponding input, which only happens negligibly under the unpredictability of PUFs.

When the receiver is corrupt. The last case of the analysis is when the sender P_i is honest whereas the receiver P_j is dishonest. In this case, the simulator has to be able to produce an equivocal commiment that can be later opened to either 0 or 1. As shown later, the simulator can achieve equivocality by making use its permanent PUF access in the simulation.

There will be at some point in the ideal world such that 1. $\mathcal{A}$ instructs P_j to send the challenge (x_0, x_1) in the simulation and 2. $\mathcal{F}_{\mathsf{com}}$ writes $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ on $\mathcal{S}$'s communication tape. The simulator $\mathcal{S}$ then draw a random string v from $\{0, 1\}^{\lambda}$, sends v to the simulated P_j, and give $\mathcal{F}_{\mathsf{com}}$ the permission of sending the opening to P_j. After learning the committed bit b, $\mathcal{S}$ computes $v \oplus x_b$ and use the permanent PUF access to obtain corresponding r, and sends the decommitment (b, r) to the simulated P_j. It is clear from the fact that v is uniformly random regardless of x_0 and x_1, that the simulation is perfect, and thus the environment cannot distinguish a real world execution from an ideal one.

3.3 Possibility of Getting Fewer Rounds

Our commitment scheme consists one round for the setup phase, followed by two rounds of challenge-and-response for the commitment phase. One natural question is whether the number of rounds can be further reduced, while the scheme itself still retains to be a UC-secure. In this section we investigate this problem and our answer to this question tends to be a negative one: under a mild assumption that committer with PUF access while generatiing the commitment can equivocate, there exists no UC-secure bit commitment scheme with fewer rounds of communication. The observation is that, once we reduce the number of rounds, there will always be one party, be it either the sender or the receiver, can run a simulator $\mathcal{S}$ as a subroutine and make use of $\mathcal{S}$'s power as either being able to extract a committed bit from a commitment, or being able to produce an equivocal commitment, to contradict the hiding or binding property of the scheme.

Theorem 2 *Under the assumption based on the observation from protocol design that if the committer has the PUF access upon generating the commitment, the committer can equivocate, there exists no PUF-based commiment scheme securely realizes the $\mathcal{F}_{\mathsf{com}}$ functionality with fewer rounds in communication.*

Proof: First we recall that, for a commitment scheme being UC-secure, it is required that there exist a simulator $\mathcal{S}$ able to extract a commitment when the sender is corrupt, and another $\mathcal{S}$ that is able to equivocate when the receiver is dishonest. Next we observe that any scheme with fewer rounds than three-round design as ours, must be one of the two cases: 1. the receiver doesn't need to send "challenge" to the sender, or 2. the PUF transfer in the setup phase can either be eliminated, or be included into one of the two rounds in the commitment phase.

In each of the two cases above, we observe that one of the following must be true: either 1. the sender has PUF access when performing the computation of the commitment v, or 2. the receiver has PUF access all along the protocol execution. In the first case, the sender can simply equivocate by making use of the PUF access, thus breaks the binding property. In the other case where the receiver has the PUF all along, it goes without question that he/she can run the simulator and use the PUF as the PUF initialized by the simulator. By making use of $\mathcal{S}$'s ability, the receiver can extract the commitment from the sender, thus breaks the hiding property.

4 Conclusion

As mentioned earlier, by the modeling of PUFs in [1], Brzuska et al. made two assumptions about physically uncloneable functions. The first one is temper-evidence, that is, adversaries are assumed to be unable to produce fake or malicious PUFs. The other assumption is that PUFs can only be accessed in a prescribed way, which is implicitly suggested from the construction of simulators in the security proof. One immediate question would be whether the two aforementioned assumptions can be relaxed. In [5] Ostrovsky et al. gave a positive answer to the question through providing two protocol constructions, each fulfills UC-security based on one of the two relaxed assumptions. Subsequent research results such as [4] also aims to provide secure protocol construction based on relaxed assumptions. One common characteristic that shared among those protocols is that the constructions are somewhat tedious and unsatisfactory regarding efficiency. In this work we adopt the definition in [1] and provide a secure construction which is also highly efficient. Undoubtly, to design efficient PUF-based schemes in the malicious PUF model would be fascinating problem to consider and a challenging goal to achieve.

References

[1] Christina Brzuska, Marc Fischlin, Heike Schroder, Stefan Katzenbeisser *Physically Uncloneable Functions in the Universal Composition Framework.* In CRYPTO 2011. http://eprint.iacr.org/2011/681

[2] Ran Canetti *Universally composable security: A new paradigm for cryptographic protocols.* In FOCS, pages 136-145, 2001.

[3] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, Adam Smith *Fuzzy extractors: How to generate strong keys from biometrics and other noisy data.* In SIAM J. Comput., 38(1):97-139, 2008.

[4] Ivan Damgard and Alessandra Scafuro *Unconditionally Secure and Universally Composable Commitments from Physical Assumptions.* http://eprint.iacr.org/2013/108

[5] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, Akshay Wadia *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions.* In EUROCRYPT 2012. http://eprint.iacr.org/2012/143