Feasibility and Infeasibility of Secure Computation with Malicious PUFs

Dana Dachman-Soled[1], Nils Fleischhacker[2], Jonathan Katz[1], Anna Lysyanskaya[3], and Dominique Schröder[2]

[1] University of Maryland
danadach@ece.umd.edu,jkatz@cs.umd.edu
[2] Saarland University
{fleischhacker,schroeder}@cs.uni-saarland.de
[3] Brown University
anna_lysyanskaya@brown.edu

Abstract. A recent line of work has explored the use of *physically uncloneable functions (PUFs)* for secure computation, with the goals of (1) achieving universal composability without (additional) setup, and/or (2) obtaining unconditional security (i.e., avoiding complexity-theoretic assumptions). Initial work assumed that all PUFs, even those created by an attacker, are honestly generated. Subsequently, researchers have investigated models in which an adversary can create *malicious* PUFs with arbitrary behavior. Researchers have considered both malicious PUFs that might be stateful, as well as malicious PUFs that can have arbitrary behavior but are guaranteed to be stateless.

We settle the main open questions regarding secure computation in the malicious-PUF model:

- We prove that unconditionally secure oblivious transfer is impossible, even in the stand-alone setting, if the adversary can construct (malicious) *stateful* PUFs.
- We show that universally composable two-party computation is possible if the attacker is limited to creating (malicious) *stateless* PUFs. Our protocols are simple and efficient, and do not require any cryptographic assumptions.

1 Introduction

A *physically uncloneable function* (PUF) [19,20,17,1,15] is a physical object generated via a process that is intended to create "unique" objects with "random" (or at least random-looking) behavior. PUFs can be probed and their response measured, and a PUF thus defines a function. (We ignore here the possibility of slight variability in the response, which can be corrected using standard techniques.) At an abstract level, this function has two important properties: it is *random*, and it *cannot be copied* even by the entity who created the PUF.

Since their introduction, several cryptographic applications of PUFs have been suggested, in particular in the area of secure computation. PUFs are especially interesting in this setting because they can potentially be used (1) to obtain *universally composable* (UC) protocols [6] without additional setup, thus bypassing known impossibility results that hold for universal composition in the "plain" model [7,8], and (2) to construct protocols with *unconditional* security, i.e., without relying on any cryptographic assumptions.

Initial results in this setting [21,22] showed constructions of oblivious transfer with stand-alone security based on PUFs. Brzuska et al. [5] later formalized PUFs within the UC framework, and showed UC constructions of bit commitment, key agreement, and oblivious transfer (and hence secure computation of arbitrary functionalities) with *unconditional* security. The basic feasibility questions related to PUFs thus seemed to have been resolved.

Ostrovsky et al. [18], however, observe that the previous results implicitly assume that all PUFs, including those created by the attacker, are honestly generated. They point out, correctly, that this may not be a reasonable assumption: nothing forces the attacker to use the recommended process for manufacturing PUFs and it is not clear, in general, how to "test" whether a PUF sent by some party was generated correctly or not. (Assuming a trusted entity who creates the PUFs is not a panacea, as one of the goals of using PUFs is to avoid reliance on trusted parties.) Addressing this limitation, Ostrovsky et al. define a model in which an attacker can create *malicious* PUFs having arbitrary, adversary-specified behavior. The previous protocols can be easily attacked in this new adversarial setting, but Ostrovsky et al. show that it is possible to construct universally composable protocols for secure computation in the malicious-PUF model under additional, number-theoretic assumptions. They explicitly leave open the question of whether unconditional security is possible in the malicious-PUF model. Recently, Damgård and Scafuro [9] have made partial progress on this question by presenting a commitment scheme with unconditional security in the malicious-PUF model.

Stateful vs. stateless (malicious) PUFs. Honestly generated PUFs are stateless; that is, the output of an honestly generated PUF is independent of its computation history. Ostrovsky et al. note that maliciously generated PUFs might be stateful or stateless. Allowing the adversary to create stateful PUFs is obviously more general. (The positive results mentioned earlier remain secure even against an attacker who can create malicious, stateful PUFs.) Nevertheless, the assumption that the adversary is limited to producing stateless PUFs is meaningful; indeed, depending on the physical technology used to implement the PUFs, incorporating dynamic state in the PUF may simply be infeasible.

1.1 Our Results

Spurred by the work of Ostrovsky et al. and Damgård and Scafuro, we reconsider the possibility of unconditionally secure computation based on malicious PUFs and resolve the main open questions in this setting. Specifically, we show:

1. Unconditionally secure oblivious transfer (and thus unconditionally secure computation of general functions) is impossible when the attacker can create malicious *stateful* PUFs. Our result holds even with regard to stand-alone

security, and even for indistinguishability-based (as opposed to simulation-based) security notions.

2. If the attacker is limited to creating malicious, but *stateless*, PUFs, then universally composable oblivious transfer (OT) and two-party computation of general functionalities are possible. Our oblivious-transfer protocol is efficient and requires each party to create only a single PUF for polynomially many OT executions. The protocol is also conceptually simple, which we view as positive in light of the heavy machinery used in [18].

1.2 Other Related Work

Hardware tokens have also been proposed as a physical assumption on which to base secure computation [14]. PUFs are incomparable to hardware tokens since they are more powerful in one respect and less powerful in another. PUFs have the property that a party cannot query an honestly generated PUF when it is out of that party's possession, whereas in the token model parties place known functionality in the token and can simulate the behavior of the token at any point. On the other hand, tokens can implement arbitrary code, whereas honestly generated PUFs just provide a random function. In any case, known results (such as the fact that UC oblivious transfer is impossible with stateless tokens [11]) do not directly translate from one model to the other.

Impossibility results for (malicious) PUFs are also not implied by impossibility results in the random-oracle model (e.g., [3]). A random oracle can be queried by any party at any time, whereas (as noted above) an honestly generated PUF can only be queried by the party who currently holds it. Indeed, we show that oblivious transfer *is* possible when malicious PUFs are assumed to be stateless; in contrast, oblivious transfer is impossible in the random-oracle model [12].

Ostrovsky et al. [18] consider a second malicious model where the attacker can *query* honestly generated PUFs in a non-prescribed manner. They show that secure computation is impossible if both this and maliciously generated PUFs are allowed. We do not consider the possibility of malicious queries in this work.

In other work, van Dijk and Rührmair [23] show impossibility results in a malicious-PUF model very different from the one considered in [18,9] and here. It is not clear to us how their model corresponds to attacks that could feasibly be carried out in the real world.

2 Physically Uncloneable Functions

A physically uncloneable function (PUF) is a physical device with "random" behavior introduced through uncontrollable manufacturing variations during their fabrication. When a PUF is queried with a stimulus (i.e., a challenge), it produces a physical output (the response). The output of a PUF can be noisy; i.e., querying the PUF twice with the same challenge may yield distinct, but close, responses. Moreover, the response need not be uniform; it may instead only have high min-entropy. Prior work has shown that, by using fuzzy extractors, one can eliminate the noisiness of a PUF and make its output effectively uniform. For simplicity, we assume this in the definition that follows.

Formally, a PUF family is defined by two algorithms S and E. The index-sampling algorithm S, which corresponds to the PUF-fabrication process, takes as input the security parameter 1^λ and returns as output an index id. The evaluation algorithm E takes as input an index id and a challenge[1] c, and generates as output the corresponding response r.

We do not require that S or E can be evaluated efficiently. In fact, these are meant to represent *physical* processes that generate a physical object and measure this object's behavior under various conditions. The index id is simply a formal placeholder that refers to a well-defined physical object; it does not in itself represent any meaningful information about how this object works.

Following [5], we define the two main security properties of PUFs: *unpredictability* and *uncloneability*. As noted earlier, for simplicity we consider only a strong form of unpredictability where the output of the PUF is uniform. Intuitively, uncloneability means that only one party can evaluate a PUF at a time. This is formally modeled using an ideal functionality, $\mathcal{F}_{\mathsf{PUF}}$, that enforces this. Details of this ideal functionality are given in the full version of this work.

Finally, we also allow for the possibility of a maliciously generated PUF whose behavior does not necessarily correspond to (S, E) as described above. We consider two possibilities here: The first possibility is a *malicious-but-stateless PUF* that may use an $\mathsf{E}_{\mathtt{mal}}$ procedure of the adversary's choice in place of the honest algorithm E. Whenever a party in possession of this PUF evaluates it, it receives $\mathsf{E}_{\mathtt{mal}}(c)$ instead of $\mathsf{E}_{\mathsf{id}}(c)$. (As noted in prior work, care must be taken to ensure that the adversary cannot use $\mathsf{E}_{\mathtt{mal}}$ to perform arbitrary exponential-time computation; formally, we restrict $\mathsf{E}_{\mathtt{mal}}$ to be a polynomial-time algorithm with oracle access to E.) The second possibility is a *malicious-and-stateful PUF* that may use a *stateful* $\mathsf{E}_{\mathtt{mal}}$ procedure of the adversary's choice in place of E. Again, $\mathsf{E}_{\mathtt{mal}}$ is limited to polynomial-time computation with oracle access to E.

To simplify notation throughout the rest of the paper, we write $\mathsf{PUF} \leftarrow \mathsf{S}(1^\lambda)$ to denote the fabrication of a PUF, and then write $r := \mathsf{PUF}(c)$.

3 Impossibility Result for Malicious, Stateful PUFs

We prove that any PUF-based oblivious-transfer (OT) protocol is insecure when the attacker has the ability to generate malicious, *stateful* PUFs. Formally:

Theorem 1. *Let Π be a PUF-based OT-protocol where the sender $\mathcal{S}$ and receiver $\mathcal{R}$ each make at most $m = \text{poly}(\lambda)$ PUF queries. Then at least one of the following holds:*

1. *There is an unbounded adversary $\mathcal{S}^*$ that uses malicious, stateful PUFs and makes only $\text{poly}(\lambda)$ queries to honestly generated PUFs, and computes the choice bit of $\mathcal{R}$ (when $\mathcal{R}$'s input is uniform) with probability $1/2 + 1/\text{poly}(\lambda)$.*

[1] We assume the challenge space is just a set strings of a certain length. For some classes of PUFs, this is naturally satisfied (see [17]). For others, this can be achieved using appropriate encoding.

2. *There is an unbounded adversary $\mathcal{R}^*$ that uses malicious, stateful PUFs and makes only* $\mathrm{poly}(\lambda)$ *queries to honestly generated PUFs, and correctly guess both secrets of $\mathcal{S}$ (when $\mathcal{S}$'s inputs are uniform) with probability at least* $2/3$.

3.1 Overview

The starting point for our impossibility result is the impossibility of constructing oblivious transfer in the random-oracle model. The fact that OT is impossible in the random-oracle model follows from the fact that key agreement is impossible in the random-oracle model [12,3,2], and the observation that OT implies key agreement. However, a direct proof ruling out OT in the random-oracle model is also possible, and we sketch such a proof here.

Consider an OT protocol in the random-oracle model between a sender $\mathcal{S}$ and receiver $\mathcal{R}$, where $\mathcal{S}$'s two input bits are uniform and $\mathcal{R}$'s selection bit is uniform. We show that either $\mathcal{S}$ or $\mathcal{R}$ can attack the protocol. Consider the case where both parties run the protocol honestly, and then at the end of the protocol they each run a variant of the Eve algorithm from [3,2] to obtain a set Q of queries/answers to/from the random oracle. This set Q contains all "intersection queries" between $\mathcal{S}$ and $\mathcal{R}$, which are queries made by both parties to the random oracle. However, note that the setting here is different from the key-agreement setting in which a third party (the eavesdropper) runs the Eve algorithm. In fact, in our setting, finding intersection queries is trivial for $\mathcal{S}$ and $\mathcal{R}$: all intersection queries are, by definition, already contained in the view of either of the parties. Thus, the point of running the Eve algorithm is for both parties to reconstruct the *same* set of queries Q that contains all intersection queries. As in [3,2], conditioned on the transcript of the protocol and this set Q, the views of $\mathcal{S}$ and $\mathcal{R}$ are independent. The property of the Eve algorithm we use is that with high probability over random coins of the protocol and the choice of random oracle, the distribution over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Q is statistically close to the distribution over $\mathcal{R}$'s view conditioned on only the transcript and Q.

To use the above to obtain an attack, we first consider the distribution over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Q. We argue that with roughly $1/2$ probability over this distribution, $\mathcal{R}$'s view must be consistent with selection bit 0, and with $1/2$ probability is must be consistent with selection bit 1. (If not, then $\mathcal{S}$ can compromise $\mathcal{R}$'s security by guessing that $\mathcal{R}$'s selection bit is the one which is more lilkely.) Next, we consider the distribution over $\mathcal{R}$'s view conditioned on only the transcript and Q. Note that $\mathcal{R}$ can sample from this distribution, since $\mathcal{R}$ knows the transcript and can compute the same set Q. Since this distribution is statistically close to the distribution over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Eve queries, we have that $\mathcal{R}$ can with high probability sample a view consistent with selection bit 0 and $\mathcal{S}$'s view *and* a view consistent with selection bit 1 and $\mathcal{S}$'s view. But correctness of the protocol then implies that $\mathcal{R}$ can with high probability discover both of $\mathcal{S}$'s input bits.

From random oracles to PUFs. The problem with extending the above to the PUF model is that, unlike a random oracle, a PUF can only be queried by the party who currently holds the PUF. This mean that the above attack, as described, will not work. In fact, this property is what allows us to *construct* an OT protocol in the case where malicious PUFs are assumed to be stateless! To overcome this difficulty, we will need to use the fact that malicious parties can create *stateful* PUFs.

To illustrate the main ideas, consider a protocol in which four PUFs are used. $\mathsf{PUF}_{\mathcal{S}}$ and $\mathsf{PUF}'_{\mathcal{S}}$ are created by $\mathcal{S}$, with $\mathsf{PUF}_{\mathcal{S}}$ held by $\mathcal{S}$ at the end of the protocol and $\mathsf{PUF}'_{\mathcal{S}}$ held by $\mathcal{R}$ at the end of the protocol. Similarly, $\mathsf{PUF}_{\mathcal{R}}, \mathsf{PUF}'_{\mathcal{R}}$ are created by $\mathcal{R}$, with $\mathsf{PUF}_{\mathcal{R}}$ held by $\mathcal{R}$ at the end of the protocol and $\mathsf{PUF}'_{\mathcal{R}}$ held by $\mathcal{S}$ at the end of the protocol. We now want to provide a way for both parties to be able to obtain a set of queries/answers Q for all the PUFs that contains the following "intersection queries":

1. Any query that both parties made to $\mathsf{PUF}'_{\mathcal{S}}$ or $\mathsf{PUF}'_{\mathcal{R}}$ (as in [3,2]).
2. All queries that $\mathcal{R}$ made to $\mathsf{PUF}_{\mathcal{S}}$.
3. All queries that $\mathcal{S}$ made to $\mathsf{PUF}_{\mathcal{R}}$.

The first of these can be achieved by having $\mathcal{S}$ (resp., $\mathcal{R}$) construct $\mathsf{PUF}'_{\mathcal{S}}$ (resp., $PUF'_{\mathcal{R}}$) with known code, such that $\mathcal{S}$ (resp., $\mathcal{R}$) can effectively query $\mathsf{PUF}'_{\mathcal{S}}$ (resp., $PUF'_{\mathcal{R}}$) at any time. Formally, we have each party embed a randomly chosen t-wise independent function in the PUF they create, where t is large enough so that the behavior of the PUF is indistinguishable from a random function as far as execution of the protocol (and the attack) is concerned. At the end of the protocol, both parties can then run the Eve algorithm with access to $\mathsf{PUF}'_{\mathcal{S}}$: $\mathcal{R}$ has access because it holds $\mathsf{PUF}'_{\mathcal{S}}$, and $\mathcal{S}$ has access because it knows the code in $\mathsf{PUF}'_{\mathcal{S}}$. An analogous statement holds for $\mathsf{PUF}'_{\mathcal{R}}$.

To handle the second set of queries, above, we rely on the ability of $\mathcal{S}$ to create stateful PUFs. Specifically, we have $\mathcal{S}$ create $\mathsf{PUF}_{\mathcal{S}}$ in such a way that it records (in an undetectable fashion) all the queries that $\mathcal{R}$ makes to $\mathsf{PUF}_{\mathcal{S}}$, in such a way that $\mathcal{S}$ can later recover these queries once $\mathsf{PUF}_{\mathcal{S}}$ is back in its possession. (This is easy to do by hardcoding in the PUF a secret challenge, chosen in advance by $\mathcal{S}$, to which the PUF responds with the set of all queries made to the PUF.) So, at the end of the protocol, it is trivial for $\mathcal{S}$ to learn all the queries that $\mathcal{R}$ made to $\mathsf{PUF}_{\mathcal{S}}$. Of course, $\mathcal{R}$ knows exactly the set of queries it made to $\mathsf{PUF}_{\mathcal{S}}$ throughout the course of the protocol. Queries that $\mathcal{S}$ makes to $\mathsf{PUF}_{\mathcal{R}}$ are handled in a similar fashion.

To complete the proof, we then show that the set of intersection queries as defined above is enough for the analysis from [3,2] to go through.[2]

[2] In order for our proof to go through, it is crucial to find intersection queries immediately after each message is sent, as opposed to waiting until the end of the protocol. This is necessary in order to ensure the *independence* of the views of $\mathcal{S}$ and $\mathcal{R}$. Therefore, we define a variant of the Eve algorithm which, after each protocol message is sent, makes queries to a particular set of PUFs, determined by the sets of PUFs currently held by each party. For example, if immediately after message i is sent $\mathcal{S}$ holds $\{\mathsf{PUF}_{\mathcal{S}}, \mathsf{PUF}'_{\mathcal{R}}\}$ and R holds $\{\mathsf{PUF}_{\mathcal{R}}, \mathsf{PUF}'_{\mathcal{S}}\}$, then our Eve variant will make queries only to $\mathsf{PUF}'_{\mathcal{R}}$ and $\mathsf{PUF}'_{\mathcal{S}}$.

3.2 Proof Details

Oblivious transfer. Oblivious transfer (OT) is a protocol between a sender $\mathcal{S}$ with input bits (s_0, s_1) and a receiver $\mathcal{R}$ with input bit b. Informally, the receiver wishes to retrieve s_b from $\mathcal{S}$ in such a way that (1) $\mathcal{S}$ does not "learn" anything about $\mathcal{R}$'s choice and (2) $\mathcal{R}$ learns nothing about s_{1-b}.

We note that our impossibility holds even for protocols that do not enjoy perfect correctness, i.e., it holds for protocols where correctness holds (over choice of inputs, randomness, and PUFs) with probability $1 - 1/\operatorname{poly}(\lambda)$.

Protocols based on PUFs. We consider a candidate PUF-based OT protocol $\mathit{\Pi}$ with ℓ rounds that has 2ℓ *passes* and where in each *pass* a party sends a message. We assume w.l.o.g. that $\mathcal{S}$ sends the first message of the protocol and $\mathcal{R}$ sends the final message. Let $z = z(\lambda)$ be the total number of PUFs used in protocol $\mathit{\Pi}$ with security parameter λ. We model the set of all PUFs $\{\mathsf{PUF}_1, \ldots, \mathsf{PUF}_z\}$ utilized by $\mathit{\Pi}$ as a single random oracle. W.l.o.g. we assume that each query q to a PUF has the form $q = (j, q')$, where j denotes the identity of the PUF being queried, and q' denotes the actual query to this PUF. Note that responses to unique queries $q = (j, q')$ are independent and uniform. We further assume w.l.o.g. that a party can only send a PUF back and forth along with some message m_i of the protocol $\mathit{\Pi}$. In particular, we denote by S^i_{back} the set of indices $j \in [z]$ such that PUF_j is sent by $\mathcal{S}$ (resp. $\mathcal{R}$) to $\mathcal{R}$ (resp. $\mathcal{S}$) immediately after message m_i of $\mathit{\Pi}$ is sent, and PUF_j was created by $\mathcal{R}$ (resp. $\mathcal{S}$). We define S^i_{PUF} to be the set of indices j such that either:

- PUF_j is held by $\mathcal{S}$ immediately after message m_i is sent and PUF_j was created by $\mathcal{R}$.
- PUF_j is held by $\mathcal{R}$ immediately after message m_i is sent and PUF_j was created by $\mathcal{S}$.

Augmented transcripts. A full (augmented) transcript of protocol $\mathit{\Pi} = \langle \mathcal{S}, \mathcal{R} \rangle$ is denoted by $\widetilde{\mathsf{M}}$. The "augmented transcript" consists of the transcript $\mathsf{M} = m_1, \ldots, m_{2\ell}$ of protocol $\mathit{\Pi}$ with a set ψ^i appended after each message m_i. If message m_i is sent by $\mathcal{S}$ (resp. $\mathcal{R}$), then ψ^i contains all queries made by $\mathcal{S}$ (resp. $\mathcal{R}$) up to this point in the protocol to all PUF_j, $j \in S^i_{\mathsf{back}}$. Specifically, $\mathsf{M} = \{m_1, \ldots, m_{2\ell}\}$ and $\widetilde{\mathsf{M}} = \{m_1 || \psi^1, \ldots, m_{2\ell} || \psi^{2\ell}\}$. Note that $\widetilde{\mathsf{M}}$ can be computed by both a malicious $\mathcal{S}$ and a malicious $\mathcal{R}$ participating in $\mathit{\Pi}$. Intuitively, this is because both malicious $\mathcal{S}$ and $\mathcal{R}$ can program each of their PUFs to record all queries made to it. The following claim formalizes the fact that malicious, stateful PUFs can be used to extract sets of queries made by the opposite party:

Claim 2 *Consider a PUF-based ℓ-round OT protocol, $\mathit{\Pi}$. By participating in an execution of $\mathit{\Pi}$ while using maliciously constructed PUFs, we have that, for all odd $i \in [2\ell]$,* both *a malicious $\mathcal{S}$ and a malicious $\mathcal{R}$ can find the set of queries ψ^i made by $\mathcal{S}$ up to this point in the protocol to all PUF_j, $j \in S^i_{\mathsf{back}}$. The same claim holds for even $i \in [2\ell]$, with the roles of $\mathcal{S}, \mathcal{R}$ reversed.*

Proof. The malicious $\mathcal{R}$ will create stateful PUFs which record all queries made to them and such that this record can later be retrieved by the creator of the PUF. Since at the end of the i-th pass, for odd $i \in [2\ell]$, $\mathcal{R}$ holds PUF_j, $j \in S^i_{\mathsf{back}}$, we have that the malicious $\mathcal{R}$ can recover the ordered set of queries made to that PUF, and can therefore deduce the set of queries made by $\mathcal{S}$ to that PUF thus far. On the other hand, $\mathcal{S}$ knows the queries it made itself to PUF_j, $j \in S^i_{\mathsf{back}}$. An analogous argument holds for even $i \in [2\ell]$.

We also define the set Ψ^i which is the union of the ψ^j sets for $j \leq i$. Specifically, $\Psi^i = \psi^1 \cup \cdots \cup \psi^i$.

Queries and views. By $\mathrm{V}^i_{\mathcal{S}}$ (resp. $\mathrm{V}^i_{\mathcal{R}}$) we denote the view of $\mathcal{S}$ (resp. $\mathcal{R}$) until the end of round i. This includes $\mathcal{S}$'s (resp. $\mathcal{R}$'s) randomness $r_{\mathcal{S}}$ (resp. $r_{\mathcal{R}}$), exchanged messages M^i as well as oracle query-answer pairs known to $\mathcal{S}$ (resp. $\mathcal{R}$) so far. We use $\mathcal{Q}(\cdot)$ as an operator that extracts the set of queries from a set of query-answer pairs or views.

Executions and distributions. A (full) *execution* of $\mathcal{S}$, $\mathcal{R}$, Eve in protocol Π can be described by a tuple $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$ where H is a random function. We denote by $\mathcal{E}$ the distribution over (full) executions that is obtained by running the algorithms for $\mathcal{S}, \mathcal{R}, \mathsf{Eve}$ with uniformly chosen random tapes and a sampled oracle H. For a sequence of i (augmented) messages $\widetilde{\mathsf{M}}^i = [\widetilde{m}_1, \ldots, \widetilde{m}_i]$ and a set of query-answer pairs P, by $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ we denote the joint distribution over the views $(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}})$ of $\mathcal{S}$ and $\mathcal{R}$ in their (partial) execution of Π up to the point in the system in which the i-th message is sent (by $\mathcal{S}$ or $\mathcal{R}$) conditioned on: The transcript of messages in the first i passes equals $\widetilde{\mathsf{M}}^i$ and $H(j, q') = a$ for all $((j, q'), a) \in \mathsf{P}$ made to H (recall that a query (j, q') to H corresponds to a query q' made to PUF_j). For $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ such that $\Pr_{\mathcal{E}}(\widetilde{\mathsf{M}}^i, \mathsf{P}) > 0$, the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ can be sampled by first sampling $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$ uniformly at random conditioned on being consistent with $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ and then deriving $\mathcal{S}$ and $\mathcal{R}$ views $\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}$ from the sampled $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$.

For $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ such that $\Pr_{\mathcal{E}}(\widetilde{\mathsf{M}}^i, \mathsf{P}) > 0$, the event $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is defined over the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ and holds if and only if $\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}}) \cap \mathcal{Q}(\mathrm{V}^i_{\mathcal{R}}) \subseteq \mathsf{P}^+$, where $\mathsf{P}^+ = \mathsf{P} \cup \Psi^i$. For $\Pr_{\mathcal{E}}(\widetilde{\mathsf{M}}^i, \mathsf{P}) > 0$ we define the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ to be the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ conditioned on $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$.

For complete transcripts $\widetilde{\mathsf{M}}$, the distributions $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$ and $\mathsf{Good}(\widetilde{\mathsf{M}}, \mathsf{P})$ are defined similarly.

Transforming the protocol. We begin by transforming the OT protocol Π into one that has the following properties:

Semi-normal form: We define a semi-normal form for OT protocols, following [3,2]. A protocol is in semi-normal form if it fulfills the following two properties: (1) $\mathcal{S}$ and $\mathcal{R}$ ask at most one query in each protocol round, and (2) the receiver of the last message uses this message to compute its output and it does not query the oracle. We start by converting our OT protocol Π

into its semi-normal version. Note that any attack on the semi-normal version of Π can be translated into an attack on the original Π that makes the *same* number of queries [3,2]. Thus, in the following we present our attacks and analysis w.r.t. the semi-normal version of Π.

Using t-wise independent functions: Instead of creating honestly generated PUFs, a malicious $\mathcal{R}$ (resp. $\mathcal{S}$) will create stateful PUFs which behave as t-wise independent hash functions. We define the distribution $\mathcal{V}^t(\widetilde{\mathsf{M}}, \mathsf{P})$ exactly like $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$ except some subset of PUFs are instantiated with t-wise independent hash functions for some $t = \text{poly}(m/\varepsilon)$, instead of with random oracles. Since we choose t such that the malicious sender and honest receiver (resp., malicious receiver and honest sender) make a total of at most t queries to all PUFs then: For every setting of random variables $(\widetilde{\mathsf{M}}^i, \mathsf{P}^i)$, the distributions $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P}^i)$ and $\mathcal{V}^t(\widetilde{\mathsf{M}}^i, \mathsf{P}^i)$ are identical. Thus, from now on, even when $\mathcal{R}$ or $\mathcal{S}$ are malicious (and create t-wise independent PUFs), we consider only the distribution $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$.

Random inputs: In the last step we change the protocol such that both sender and receiver choose their input(s) uniformly at random. Thus, in the following, we consider execution of $\Pi = \langle \mathcal{S}(1^\lambda), \mathcal{R}(1^\lambda) \rangle$ where the parties use their random tapes to choose their inputs.

The Eve algorithm. Recall that we have converted the protocol Π into semi-normal form. We now present the attacking algorithm, Eve, which will be run by both the malicious sender ($\mathcal{S}'$) and malicious receiver ($\mathcal{R}'$) defined later:

Construction 1 *Let $\varepsilon < 1/100$ be an input parameter. After each message m_i is sent,* Eve *generates the augmented transcript $\widetilde{\mathsf{M}}^i$ (note that by Claim 2, $\widetilde{\mathsf{M}}^i$ can always be reconstructed by* Eve*, since* Eve *is launched by either the malicious $\mathcal{S}$ or $\mathcal{R}$). Given $\widetilde{\mathsf{M}}^i$,* Eve *attacks the ℓ-round two-party protocol $\Pi = \langle \mathcal{S}, \mathcal{R} \rangle$ as follows. During the attack* Eve *updates a set P of oracle query-answer pairs as follows: Suppose $\mathcal{S}$ (alternatively $\mathcal{R}$) sends the i-th message in $\widetilde{\mathsf{M}}^i$ which is equal to $\widetilde{m}_i = m_i || \psi^i$. For $i \in [2\ell]$,* Eve *does the following: As long as the total number of queries made by* Eve *is less than $t - 2m$ and there is a query $q = (j, q') \notin \mathsf{P}^+$, where $\mathsf{P}^+ := \Psi^i \cup \mathsf{P}$, such that one of the following holds:*

$$\Pr_{(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}) \leftarrow \mathcal{G}\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[q' \in \mathcal{Q}(\mathrm{V}^i_{\mathcal{S}}) \wedge j \in S^i_{\mathsf{back}}] \geq \frac{\varepsilon^2}{100m}$$

$$\textit{or} \qquad \Pr_{(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}) \leftarrow \mathcal{G}\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[q' \in \mathcal{Q}(\mathrm{V}^i_{\mathcal{R}}) \wedge j \in S^i_{\mathsf{back}}] \geq \frac{\varepsilon^2}{100m}.$$

Eve *queries the lexicographically first such $q = (j, q')$ to H, adds $(q, H(q))$ to* P.

Properties of the Eve algorithm. We summarize some properties of the Eve algorithm that can be verified by inspection:

Symmetry of* Eve*: Both $\mathcal{S}$ and $\mathcal{R}$ can run the Eve algorithm, making the same set of queries P to the PUFs. In particular, at the point where message m_i

is sent, a party requires only the augmented transcript $\widetilde{\mathsf{M}}^i$ and oracle access to PUF_j for $j \in S^i_{\mathsf{PUF}}$. Note that for each $j \in S^i_{\mathsf{PUF}}$ a party either holds PUF_j (and so can query it directly) or created PUF_j dishonestly and thus knows the code of PUF_j (and so can simulate responses to queries to PUF_j).

Determinism of Eve***:*** The Eve algorithm is deterministic and so for a fixed transcript $\widetilde{\mathsf{M}}$ and a fixed set of PUFs, both parties will recover the same set of queries when running Eve.

Number of queries: The number of queries made by Eve is at most $t - 2m$. Thus, since $\mathcal{S}$ and $\mathcal{R}$ each make at most m number of PUF queries, the total number of queries made by $\mathcal{S}$, $\mathcal{R}$ and Eve is at most $(t - 2m) + 2m = t$.

Breaking oblivious transfer. Recall that we assume that the honest $\mathcal{S}$ chooses its inputs (s_0, s_1) at random and that the honest $\mathcal{R}$ chooses its input bit at random. Thus, we may consider an execution of OT protocol $\Pi = \langle \mathcal{S}(1^\lambda), \mathcal{R}(1^\lambda)\rangle$ where the parties use their random tapes to choose their inputs.

We now state an alternative version of Theorem 1:

Theorem 3. *Let $\Pi = \langle \mathcal{S}(1^\lambda), \mathcal{R}(1^\lambda)\rangle$ be a PUF-based OT-protocol in which the sender and receiver each ask at most m queries total to the set of $z = \mathrm{poly}(\lambda)$ PUFs, $\{\mathsf{PUF}_1, \ldots, \mathsf{PUF}_z\}$. Then, at least one of the following must hold:*

1. *There exists an adversarial $\mathcal{S}$ that uses malicious, stateful PUFs to compute the choice bit of $\mathcal{R}$ with advantage $1/\mathrm{poly}(\lambda)$ and makes $\mathrm{poly}(\lambda)$ queries to the PUFs.*
2. *For $\varepsilon < 1/100$, there exists an adversarial $\mathcal{R}$ that uses malicious, stateful PUFs to correctly guess both secrets of $\mathcal{S}$ with probability $1 - O(\sqrt{\varepsilon})$ and makes $\mathrm{poly}(\lambda)$ queries to the PUFs.*

By choosing constant ε sufficiently small, we may obtain the parameters of Theorem 1.

Proof. We begin with some notation. For a view $\mathrm{V}_\mathcal{R}$ (resp. $\mathrm{V}_\mathcal{S}$), we denote by $\mathrm{In}(\mathrm{V}_\mathcal{R})$ (resp. $\mathrm{In}(\mathrm{V}_\mathcal{S})$) the input of the corresponding party implicitly contained in its view. We denote by $\mathrm{Out}(\mathrm{V}_\mathcal{R})$ the output of $\mathcal{R}$ implicitly contained in its view. For a distribution $\mathcal{D}$ and random variables $X_1, \ldots, X_n$, we denote by $\mathcal{D}(X_1, \ldots, X_n)$ the distribution $\mathcal{D}$ conditioned on $X_1, \ldots, X_n$.

Let $p(\cdot)$ be some sufficiently large polynomial. We consider two cases.

Case 1*:* With probability $1/p(\lambda)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_\mathcal{S})$ generated by a run of $\tilde{\Pi}$ we have that either

$$\begin{aligned} &\Pr_{\mathcal{V}_{\mathrm{V}_\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_\mathcal{S})}[\mathrm{In}(\mathrm{V}_\mathcal{R}) = 0 \wedge \mathrm{Out}(\mathrm{V}_\mathcal{R}) = s_0] \leq 1/2 - \varepsilon \\ \text{or } &\Pr_{\mathcal{V}_{\mathrm{V}_\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_\mathcal{S})}[\mathrm{In}(\mathrm{V}_\mathcal{R}) = 1 \wedge \mathrm{Out}(\mathrm{V}_\mathcal{R}) = s_1] \leq 1/2 - \varepsilon \end{aligned}$$

holds, where $(s_0, s_1) = \mathrm{In}(\mathrm{V}_\mathcal{S})$.

Case 2*:* With probability $1 - 1/p(\lambda)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})$ generated by a run of $\tilde{\Pi}$ we have that both

$$\begin{aligned} &\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathrm{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] \geq 1/2 - \varepsilon \\ \text{and} \quad &\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathrm{Out}(\mathrm{V}_{\mathcal{R}}) = s_1] \geq 1/2 - \varepsilon \end{aligned}$$

hold, where $(s_0, s_1) = \mathrm{In}(\mathrm{V}_{\mathcal{S}})$.

Clearly, for any PUF-based OT protocol, either Case 1 or Case 2 must hold. We show that if Case 1 holds then a malicious sender may attack receiver privacy making $\mathrm{poly}(m/\varepsilon)$ queries and succeeding with advantage $\varepsilon/4p(\lambda)$, and if Case 2 holds then a malicious receiver may attack sender privacy making $\mathrm{poly}(m/\varepsilon)$ queries and succeeding with probability $1 - O(\sqrt{\varepsilon})$. This is sufficient to prove the theorem.

We next present the attacks on Receiver and Sender privacy. We defer the analysis of the attacks to the full version.

Sender's attack (denoted $\mathcal{S}'$) on receiver privacy:

1. Participate in protocol Π where the PUFs constructed by $\mathcal{S}$ are instantiated with t-wise independent hash functions and maliciously constructed to record $\mathcal{R}$ queries.
2. Convert the resulting transcript M to the augmented transcript $\widetilde{\mathsf{M}}$.
3. Run the Eve algorithm on augmented transcript $\widetilde{\mathsf{M}}$ to generate the set P.
4. Compute the probabilities

$$\begin{aligned} &P_0 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathrm{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] \\ \text{and} \quad &P_1 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathrm{Out}(\mathrm{V}_{\mathcal{R}}) = s_1]. \end{aligned}$$

5. If $P_0 \geq 1/2 + \varepsilon/2$, output 0, if $P_1 \geq 1/2 + \varepsilon/2$, output 1. Otherwise, output 0 or 1 with probability 1/2.

Receiver's attack (denoted $\mathcal{R}'$) on sender privacy:

1. Participate in protocol Π where the PUFs constructed by $\mathcal{R}$ are instantiated with t-wise independent hash functions and are maliciously constructed to record $\mathcal{S}$ queries.
2. Convert the resulting transcript M to the augmented transcript $\widetilde{\mathsf{M}}$.
3. Run the Eve algorithm on augmented transcript $\widetilde{\mathsf{M}}$ to generate the set P.
4. Compute the probabilities

$$P_0 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 0] \quad \text{and} \quad P_1 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 1].$$

5. If $P_0 = 0$ or $P_1 = 0$ then abort.
6. Otherwise, draw two views $\mathrm{V}_R(0)$ and $\mathrm{V}_R(1)$ from $\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 0)$ and $\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{In}(\mathrm{V}_{\mathcal{R}}) = 1)$, respectively.
7. Output $s'_0 = \mathrm{Out}(\mathrm{V}_R(0)), s'_1 = \mathrm{Out}(\mathrm{V}_R(1))$.

4 Feasibility Result for Malicious, Stateless PUFs

We show that universally composable two-party computation is possible if the adversary is limited to creating *stateless* malicious PUFs. The core of our result is a construction of an unconditionally secure, universally composable, oblivious-transfer protocol in this model; we describe the protocol here, and defer its proof of security to the full version. In Section 4.2 we briefly discuss how the oblivious-transfer protocol can be used to obtain the claimed result.

Sender $\boldsymbol{\mathcal{S}}$	session `sid`	Receiver $\boldsymbol{\mathcal{R}}$
$\mathsf{PUF}_{\mathcal{S}} \leftarrow \mathsf{S}$		$\mathsf{PUF}_{\mathcal{R}} \leftarrow \mathsf{S}$
	$\xrightarrow{\mathsf{PUF}_{\mathcal{S}}}$	
		$i = 1, ..., N$:
		$c_i \leftarrow \{0,1\}^{\lambda}$
		$r_i := \mathsf{PUF}(i \Vert c_i)$
		store $(c_1, r_1), \ldots, (c_N, r_N)$
	$\xleftarrow{\mathsf{PUF}_{\mathcal{S}}, \mathsf{PUF}_{\mathcal{R}}}$	

For $i = 1, \ldots, N$ do:

Input: $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}^{\lambda}$		Input: $b \in \{0,1\}$
$x_0, x_1 \leftarrow \{0,1\}^{\lambda}$	$\xrightarrow{x_0, x_1}$	
		$v := c_i \oplus x_b$
	$\xleftarrow{v}$	
$\hat{r}_0 := \mathsf{PUF}\,(i \Vert (v \oplus x_0))$		
$\hat{r}_1 := \mathsf{PUF}\,(i \Vert (v \oplus x_1))$		
$\mathsf{S}_0 := \mathsf{s}_0 \oplus \hat{r}_0$		
$\mathsf{S}_1 := \mathsf{s}_1 \oplus \hat{r}_1$		
	$\xrightarrow{\mathsf{S}_0, \mathsf{S}_1}$	
		Output: $\mathsf{s}_b := \mathsf{S}_b \oplus r_i$

Fig. 1. Oblivious transfer protocol. We define $\mathsf{PUF}(c) \overset{\text{def}}{=} \mathsf{PUF}_{\mathcal{S}}(c) \oplus \mathsf{PUF}_{\mathcal{R}}(c)$.

4.1 Universally Composable Oblivious Transfer

Our OT protocol adapts the protocol of Brzuska et al. [5], which is secure against attackers limited to honestly generated PUFs. Roughly, we replace the single PUF—generated by one of the parties—in their protocol with a "combined PUF" generated by both parties. Specifically, this "combined PUF" PUF is constructed by having each party generate their own PUFs $\mathsf{PUF}_{\mathcal{S}}$, $\mathsf{PUF}_{\mathcal{R}}$, and then defining $\mathsf{PUF}(c) \overset{\text{def}}{=} \mathsf{PUF}_{\mathcal{S}}(c) \oplus \mathsf{PUF}_{\mathcal{R}}(c)$. Intuitively, as long as one of the parties generates their PUF honestly, the combined PUF is still unpredictable (the output is random) and uncloneable (without physical access to *both* $\mathsf{PUF}_{\mathcal{S}}$ and $\mathsf{PUF}_{\mathcal{R}}$, it is impossible to evaluate PUF).

In our description of the protocol in Figure 1, we have the parties exchange PUFs once, after which they can subsequently execute any pre-determined number N of oblivious-transfer executions. We remark that it is necessary to prevent a malicious $\mathcal{R}$ from substituting a PUF of its own for $\mathsf{PUF}_{\mathcal{S}}$; this can be done by having $\mathcal{S}$ probe a random point before sending $\mathsf{PUF}_{\mathcal{S}}$ and then checking it again later. We omit this check from the figure.

Theorem 4. *The protocol in Figure 1 securely realizes* $\mathcal{F}_{\mathsf{OT}}$ *in the* $(\mathcal{F}_{\mathsf{PUF}}, \mathcal{F}_{\mathsf{auth}})$*-hybrid model, where malicious parties are limited to generated stateless PUFs (with arbitrary behavior).*

Security holds against an unbounded cheating $\mathcal{S}$, and an unbounded cheating $\mathcal{R}$ limited to making polynomially-many queries to $\mathsf{PUF}_{\mathcal{S}}$. We provide a proof of Theorem 4 in the full version.

4.2 From UC Oblivious Transfer to UC Two-Party Computation

We observe that our UC oblivious-transfer protocol can be used to obtain UC two-party computation of any functionality. The main idea is to first construct a semi-honest secure two-party computation protocol using Yao's garbled-circuit protocol, and to then apply the compiler of Ishai, Prabhakaran, and Sahai [13].

Semi-honest secure two-party computation. Lindell and Pinkas presented a proof for Yao's two-party secure-computation protocol [16]. They show how to instantiate the garbling part of the protocol with a private-key encryption scheme having certain properties. In addition, the authors show that any pseudorandom function is sufficient to instantiate such a private-key encryption scheme. Our main observation is that we can replace the pseudorandom function with a PUF.[3] This has already been observed before by Brzuska et al. [5] in a different context. With this observation, we can apply the result of [16] to obtain a protocol for semi-honest secure two-party computation based on PUFs only (and no computational assumptions).

Theorem 5. *Let f be any functionality. Then there is a (constant-round) protocol that securely computes f for semi-honest adversaries in the* $(\mathcal{F}_{\mathsf{PUF}}, \mathcal{F}_{\mathsf{OT}})$*-hybrid model.*

We omit the proof since it follows easily from prior work.

Universally composable two-party computation. In the next step we apply the IPS compiler [13], a black-box compiler that takes

- An "outer" MPC protocol Π with security against a constant fraction of malicious parties.

[3] Note also that if the circuit generator is malicious, then he cannot violate the circuit evaluator's privacy by generating a malicious PUF.

- An "inner" two-party protocol ρ, in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, where the security of ρ only needs to hold against semi-honest parties.

and transforms them into a two-party protocol $\Phi_{\Pi,\rho}$ which is secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model against malicious corruptions.

In our setting, we must be careful to give information-theoretic instantiations of the "outer" and "inner" protocols so that our final protocol $\Phi_{\Pi,\rho}$ will be unconditionally secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model. Fortunately, we may instantiate the "outer" protocol, Π, with the seminal BGW protocol [4] and may instantiate the "inner" protocol, ρ, with the protocol from the previous section. Alternatively, the "inner" protocol can be instantiated with the semi-honest version of the two-party GMW protocol [10] in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model.

Let ψ denote the OT-protocol described in Figure 1 and let $\Phi^{\psi}_{\Pi,\rho}(f)$ denote the IPS-compiled protocol which makes subroutine calls to ψ instead of $\mathcal{F}_{\mathsf{OT}}$ and computes the functionality f. Using Theorems 4 and 5, along with the UC composition theorem, we obtain the following result:

Theorem 6. *For any functionality f, protocol $\Phi^{\psi}_{\Pi,\rho}(f)$ securely computes f in the $(\mathcal{F}_{\mathsf{PUF}}, \mathcal{F}_{\mathsf{auth}})$-hybrid model.*

Acknowledgments

Work of Nils Fleischhacker and Dominique Schröder done in part while visiting the University of Maryland. Their work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy, and Accountability (CISPA; see `www.cispa-security.org`). The visit of Nils Fleischhacker was supported by the Saarbrücken Graduate School of Computer Science funded by the German National Excellence Initiative. Dominique Schröder is also supported by an Intel Early Career Faculty Honor Program Award. The work of Jonathan Katz, as well the visit of Dominique Schröder, was supported in part by NSF award #1223623. Anna Lysyanskaya is supported by NSF awards #0964379 and #1012060.

References

1. Armknecht, F., Maes, R., Sadeghi, A.R., Standaert, F.X., Wachsmann, C.: A formalization of the security features of physical functions. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press (2011) 397–412
2. Barak, B., Mahmoody, M.: Merkle's key agreement protocol is optimal: An $o(n^2)$ attack on any key agreement from a random oracle. Manuscript (2013) `http://www.cs.virginia.edu/~mohammad/files/papers/MerkleFull.pdf`.
3. Barak, B., Mahmoody-Ghidary, M.: Merkle puzzles are optimal—An $o(n^2)$-query attack on any key exchange from a random oracle. In: Advances in Cryptology—Crypto 2009. Lecture Notes in Computer Science, Springer-Verlag (2009) 374–390
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computations. In: 20th Annual ACM Symposium on Theory of Computing, ACM Press (1988) 1–10

5. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. In: Advances in Cryptology—Crypto 2011. Volume 6841 of Lecture Notes in Computer Science, Springer-Verlag (2011) 51–70
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press (2001) 136–145
7. Canetti, R., Fischlin, M.: Universally composable commitments. In Kilian, J., ed.: Advances in Cryptology—Crypto 2001. Volume 2139 of Lecture Notes in Computer Science, Springer-Verlag (2001) 19–40
8. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. Journal of Cryptology **19**(2) (2006) 135–167
9. Damgård, I., Scafuro, A.: Unconditionally secure and universally composable commitments from physical assumptions. Asiacrypt 2013. Available at http://eprint.iacr.org/2013/108
10. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In Aho, A., ed.: 19th Annual ACM Symposium on Theory of Computing, ACM Press (1987) 218–229
11. Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In: Advances in Cryptology—Crypto 2010. Lecture Notes in Computer Science, Springer-Verlag (2010) 173–190
12. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st Annual ACM Symposium on Theory of Computing, ACM Press (1989) 44–61
13. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer—efficiently. In Wagner, D., ed.: Advances in Cryptology—Crypto 2008. Lecture Notes in Computer Science, Springer-Verlag (2008) 572–591
14. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Advances in Cryptology—Eurocrypt 2007. Volume 4515 of Lecture Notes in Computer Science, Springer-Verlag (2007) 115–128
15. Katzenbeisser, S., Koccabas, Ü., Rozic, V., Sadeghi, A.R., Verbauwhede, I., Wachsmann, C.: PUFs: Myth, fact, or busted? a security evaluation of physically uncloneable functions (PUFs) cast in silicon. In: Cryptographic Hardware and Embedded Systems—CHES 2012. Volume 7428 of Lecture Notes in Computer Science, Springer-Verlag (2012) 283–301
16. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology **22**(2) (2009) 161–188
17. Maes, R., Verbauwhede, I.: Physically unclonable functions: A study on the state of the art and future research directions. In: Towards Hardware-Intrinsic Security. Springer-Verlag (2010) 3–37
18. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. In Johansson, T., Nguyen, P.Q., eds.: Advances in Cryptology—Eurocrypt 2013. Volume 7881 of Lecture Notes in Computer Science, Springer-Verlag (2013) 702–718
19. Pappu, R.S.: Physical One-Way Functions. Phd thesis, Massachusetts Institute of Technology (2001)
20. Pappu, R.S., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297** (2002) 2026–2030

21. Rührmair, U.: Oblivious transfer based on physical uncloneable functions. In: TRUST. Volume 6101 of Lecture Notes in Computer Science, Springer-Verlag (2010) 430–440
22. Rührmair, U., Katzenbeisser, S., Busch, H.: Strong PUFs: Models, constructions, and security proofs. In: Towards Hardware-Intrinsic Security. Springer-Verlag (2010) 79–96
23. van Dijk, M., Rührmair, U.: PUFs in security protocols: attack models and security evaluations. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press (2013) 286–300