J Cryptol (2020) 33:595–617
https://doi.org/10.1007/s00145-019-09329-9

Journal of
CRYPTOLOGY

Feasibility and Infeasibility of Secure Computation with Malicious PUFs

Dana Dachman-Soled
University of Maryland, College Park, USA
danadach@ece.umd.edu

Nils Fleischhacker
Ruhr University Bochum, Bochum, Germany
mail@nilsfleischhacker.de

Jonathan Katz
University of Maryland, College Park, USA
jkatz@cs.umd.edu

Anna Lysyanskaya
Brown University, Providence, USA
anna_lysyanskaya@brown.edu

Dominique Schröder
Friedrich-Alexander-University Erlangen-Nürnberg, Erlangen, Germany
dominique.schroeder@fau.de

Communicated by Serge Fehr.

Received 15 January 2018 / Revised 27 June 2019
Online publication 9 July 2019

Abstract. A recent line of work has explored the use of *physically unclonable 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 if the attacker is limited to creating (malicious) *stateless* PUFs, then universally composable two-party computation is possible, unconditionally.

Keywords. Secure computation, Oblivious transfer, Physically unclonable functions.

1. Introduction

A *physically unclonable function* (PUF) [1,16,19,20] is a physical object generated via a process that is intended to create "unique" objects with "random" behavior. PUFs can be probed and their response measured, and a PUF thus defines a function. (We ignore for now 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. The latter implies that the PUF can only be queried by the party currently holding the PUF, something that distinguishes a PUF from a random oracle.

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 [8,9], and (2) to construct protocols with *unconditional* security [5], 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 (OT)—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, observed that the previous results implicitly assume that all PUFs, including those created by the attacker, are honestly generated. They point out 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 satisfying option, as one of the goals of using PUFs is to avoid reliance on trusted parties.) Addressing this limitation, Ostrovsky et al. considered attackers who can create *malicious* PUFs with arbitrary, adversary-specified behavior. Previous protocols can be easily attacked by such adversaries, but Ostrovsky et al. showed that it is possible to construct UC protocols for secure computation in the malicious-PUF model under specific complexity assumptions. They explicitly left open the question of whether *unconditional* security is possible in this model. Damgård and Scafuro [10] made partial progress on this question by presenting a UC commitment scheme with unconditional security in the malicious-PUF model. Prior work is summarized in Table 1.

Stateful versus stateless (malicious) PUFs Honestly generated PUFs are stateless; that is, the output of an honestly generated PUF on some input is independent of previous inputs to the PUF. Ostrovsky et al. noted that maliciously generated PUFs might be stateful or stateless. (The positive results mentioned earlier remain secure even against attackers

Table 1. Our results in relation to prior work.

Reference	Malicious PUFs?	Stateful PUFs?	Unconditional?	Secure computation?
[5]	No	N/A	Yes	Yes
[18]	Yes	Yes	No	Yes
[10]	Yes	Yes	Yes	No
Here	Yes	Yes	Yes	Impossible
Here	Yes	No	Yes	Yes

who can create malicious, stateful PUFs.) Allowing the adversary to create stateful PUFs is obviously more general. Nevertheless, assuming the adversary is limited to producing stateless PUFs may be reasonable; indeed, depending on the physical technology used to implement the PUFs, incorporating dynamic state in a PUF may 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 remaining open questions in this setting. (We remark that, technically speaking, our model of malicious PUFs differs from the model considered in prior work [5,10,18]. See Sect. 2 for further discussion.) 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) definitions of security.
2. If the attacker is limited to creating malicious, but *stateless*, PUFs, then UC oblivious transfer and two-party computation of general functionalities are possible, unconditionally. Our oblivious transfer protocol is efficient and conceptually simple, which we view as positive in light of the heavy machinery used in prior work [18].

Comparison to the previous version of this work We improve on the proceedings version of this work in several respects. First, we provide a simpler and corrected version of an ideal functionality corresponding to PUFs. We also provide a full proof for our impossibility result. Finally, we show a simpler protocol for our positive result; concretely, we show how to realize oblivious transfer using a single PUF rather than two PUFs.

1.2. *Other Related Work*

Hardware tokens have also been proposed as a physical assumption on which to base secure computation [15]. PUFs are incomparable to hardware tokens: on the one hand, hardware tokens can implement arbitrary code whereas PUFs can only implement a "random function"; on the other hand, an honest party creating a token knows the code inside the token, which is not the case for a PUF. For this reason, known results (in particular the fact that UC oblivious transfer is impossible with stateless tokens [12]) 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 an honestly generated PUF can only be queried by the party who currently holds it. Indeed, we show that unconditionally secure oblivious transfer *is* possible when malicious PUFs are assumed to be stateless; in contrast, this is impossible in the random-oracle model [3,13].

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 and leave this as an open question for future work.

Rührmair and van Dijk [23] show impossibility results in a malicious-PUF model that differs significantly from (and appears to correspond to a stronger adversary than in) the models considered in prior work [10,18] and here. In other work [24], van Dijk and Rührmair informally discussed the idea of using the Impagliazzo–Rudich technique [13] to prove impossibility in the context of PUFs, but did not give any formal proofs.

In recent work [2] subsequent to our own, Badrinarayanan et al. show unconditional UC protocols for secure computation based on alternate models of malicious PUFs where either (1) a malicious PUF can encapsulate other (previously generated) PUFs or (2) the attacker is limited to creating stateful PUFs with *bounded* state.

2. Formalizing Physically Unclonable Functions

Our goal in this section is to provide ideal functionalities for PUFs that correspond to the models used in most of the recent proofs of security for PUF-based protocols. We also provide some high-level intuition as well as justification for our model.

We begin in Fig. 1 with an ideal functionality corresponding to honest PUFs, i.e., where it is assumed that all parties are only able to generate PUFs according to some mandated specification. An honest PUF behaves as a random oracle; in particular, the first time some input c is queried to the PUF, a random value y is returned; if c is queried again, the same value y is returned. The key difference between a PUF and a random oracle is that *only one party can access the PUF at any given time*; this models the fact that, at any time, only one party can be in possession of the physical object corresponding

$\mathcal{F}_{\mathsf{HPUF}}$ is parameterized by security parameter 1^λ and runs with parties P_1, P_2, and adversary Sim. It begins with a list $\mathcal{L}$ initialized to $\emptyset$. It supports the following interface:

- If a party P sends init, choose uniform identifier $\mathsf{id} \in \{0,1\}^\lambda$ and set $\mathcal{L}_{\mathsf{id}} = \emptyset$. Return id to P, and store $(\mathsf{id}, P, \mathcal{L}_{\mathsf{id}})$ in $\mathcal{L}$.
- If a party P sends $(\mathsf{eval}, \mathsf{id}, c)$ then check whether there is an entry $(\mathsf{id}, P, \mathcal{L}_{\mathsf{id}}) \in \mathcal{L}$ and return $\perp$ if not. Otherwise, if there is an entry $(c, y) \in \mathcal{L}_{\mathsf{id}}$, return y to P; if not, choose uniform $y \in \{0,1\}^\lambda$, store (c, y) in $\mathcal{L}_{\mathsf{id}}$, and return y to P.
- If a party P sends $(\mathsf{handover}, \mathsf{id}, P')$ then check whether there is an entry $(\mathsf{id}, P, \mathcal{L}_{\mathsf{id}}) \in \mathcal{L}$ and return $\perp$ if not. Otherwise, replace the tuple $(\mathsf{id}, P, \mathcal{L}_{\mathsf{id}})$ in $\mathcal{L}$ with the tuple $(\mathsf{id}, P', \mathcal{L}_{\mathsf{id}})$, and send $(P, \mathsf{handover}, \mathsf{id}, P')$ to P' and Sim.

Fig. 1. The ideal functionality $\mathcal{F}_{\mathsf{HPUF}}$ for honestly generated PUFs.

to the PUF. The functionality addresses this by keeping track of the current holder of the PUF. The party in possession of the PUF has the ability to transfer the PUF to another party at any time; this is modeled in the ideal functionality via the handover query. (As in prior work, we assume the PUF is transferred from one party to another over an authenticated channel, so that an external adversary cannot modify the PUF in transit.)

Our ideal functionality for honest PUFs is inspired by the original specification given by Brzuska et al. [5], though we have chosen to simplify it in various ways. Some of the simplifications do not sacrifice anything significant; in particular:

- Physical PUFs are typically *noisy*; that is, when queried on the same input twice, they will return close—but not identical—outputs. Moreover, the output of a PUF may not be uniform, but may instead only have high entropy. Some prior work has explicitly modeled these features as part of the ideal functionality defining PUFs. Brzuska et al. [5] have shown that both these issues can be handled using *fuzzy extractors* [11], and therefore, one can assume non-noisy, uniform outputs without loss of generality. Looking ahead, we note that this holds also in the malicious-PUF setting.
- Prior work has allowed the adversary to directly query the PUF during the point in time when the PUF is transferred from one party to another. In the setting of two-party protocols where at least one party is corrupted by the adversary, this does not give the adversary any additional power. For our positive results, one can verify that if *no* party is corrupted then our protocols remain secure against an eavesdropping attacker even if it can query the PUF when it is transferred between the honest parties.

We also choose to model PUFs via a standard functionality in the UC framework that can thus be simulated (and programmed) by the simulator as part of a security proof. In contrast, Brzuska et al. model PUFs via a *global* functionality (cf. [7]) that is outside the control of the simulator. Although this difference technically weakens our positive result, we observe that the proof of security for our positive result does not rely on programmability, and we believe that our result would also hold for malicious PUFs modeled as a global functionality.

The practical difference between modeling PUFs via a standard or a global functionality is unclear to us. However, a reviewer suggested that a difference could manifest itself if attackers are allowed to *physically tamper with* an existing PUF PUF to generate a modified PUF PUF'. In the former case the original PUF could still be accessible to the simulator, whereas in the latter case it would not be. We consider such tampering attacks to be outside the scope of our paper.

Malicious PUFs Ostrovsky et al. [18] initiated the formal study of malicious PUFs. They considered two different types of malicious behavior: the first modeling the case where the attacker can produce a PUF that behaves arbitrarily, and the second addressing the case where the attacker can query an honest PUF in a disallowed manner. In this work we only consider the first type of malicious behavior. Figure 2 specifies the ideal functionality we use in that case. Note that honest users will not query the functionality with the initmal query.

Our ideal functionality allows corrupted parties to create PUFs having behavior specified by an arbitrary circuit C. (Of course, if the attacker is limited to running in polyno-

$\mathcal{F}_{\mathsf{PUF}}^{\ell}$ is parameterized by security parameter 1^λ and length parameter ℓ, and runs with parties P_1, P_2, and adversary Sim. It begins with a list $\mathcal{L}$ initialized to $\emptyset$. It supports the following interface:

- If a party P sends init, choose uniform identifier $\mathsf{id} \in \{0,1\}^\lambda$ and set $\mathcal{L}_{\mathsf{id}} = \emptyset$. Return id to P, and store $(\mathsf{id}, 0, P, \mathcal{L}_{\mathsf{id}})$ in $\mathcal{L}$.
- If a party P sends $(\mathsf{initmal}, C)$, where C is a circuit, choose uniform identifier $\mathsf{id} \in \{0,1\}^\lambda$ and set $\mathsf{state} = \epsilon$. Return id to P, and store $(\mathsf{id}, 1, P, (C, \mathsf{state}))$ in $\mathcal{L}$. (The circuit C may have special *oracle gates*; see the text for further discussion.)
- If a party P sends $(\mathsf{eval}, \mathsf{id}, c)$ then check whether there is an entry $(\mathsf{id}, b, P, \star) \in \mathcal{L}$ and return $\perp$ if not. Otherwise:

  $b = 0$**:** Say there is an entry $(\mathsf{id}, 0, P, \mathcal{L}_{\mathsf{id}}) \in \mathcal{L}$. If there is an entry $(c, y) \in \mathcal{L}_{\mathsf{id}}$, return y to P; if not, choose uniform $y \in \{0,1\}^\lambda$, store (c, y) in $\mathcal{L}_{\mathsf{id}}$, and return y to P.

  $b = 1$**:** Say there is an entry $(\mathsf{id}, 1, P, (C, \mathsf{state})) \in \mathcal{L}$. Compute $(y, \mathsf{state}') := C(c, \mathsf{state})$ and return y to P. If $\mathsf{state}' \in \{0,1\}^\ell$ then replace the tuple $(\mathsf{id}, 1, P, (C, \mathsf{state}))$ in $\mathcal{L}$ with the tuple $(\mathsf{id}, 1, P, (C, \mathsf{state}'))$.
- If a party P sends $(\mathsf{handover}, \mathsf{id}, P')$ then check whether there is an entry $(\mathsf{id}, b, P, X) \in \mathcal{L}$ and return $\perp$ if not. Otherwise, replace the tuple (id, b, P, X) in $\mathcal{L}$ with the tuple (id, b, P', X), and send $(P, \mathsf{handover}, \mathsf{id}, P')$ to P' and Sim.

Fig. 2. The ideal functionality $\mathcal{F}_{\mathsf{PUF}}^{\ell}$ that allows for maliciously generated PUFs. The parameter ℓ determines the size of the state; $\ell = 0$ corresponds to stateless PUFs, and $\ell = *$ corresponds to stateful PUFs with unbounded state.

mial time then C must have polynomial size; we return to this point below.) Although not explicitly included in the description of the functionality, the circuit is allowed to have *oracle gates* that enable queries to a (freshly created) PUF that cannot be directly accessed by any party; this ensures that malicious PUFs are at least as powerful as honest PUFs. The functionality encompasses both the case where the attacker can create *stateful* PUFs maintaining arbitrary state (by setting $\ell = *$), as well as the case where the attacker is limited to creating *stateless* PUFs (by setting $\ell = 0$).

In contrast to the formulation of malicious PUFs by Ostrovsky et al. (as well as the original formulation by Brzuska et al.), we restrict malicious PUFs to polynomial-time computation. Fundamentally, we do not consider it realistic to assume that an attacker can create a PUF performing superpolynomial-time computation. We also believe that allowing PUFs to perform superpolynomial-time computation does not add much to the problem at hand beyond technical complications; for example, Ostrovsky et al. anyway restrict malicious PUFs to computing "admissible" functions that do not violate certain computational hardness assumptions. Finally, in our case (where PUFs are modeled as standard functionalities) allowing malicious PUFs to perform superpolynomial-time computation seems to inherently require superpolynomial-time simulation, something we prefer to avoid.

We have also given what we believe is a more natural definition that allows the attacker to exactly specify the behavior of the PUF. In contrast, Ostrovsky et al. assume a fixed *distribution* over PUFs, and have the ideal functionality sample a malicious PUF from this distribution. Since the attacker can simulate any such distribution on its own, the attacker in our formulation is at least as strong in this regard as the attacker in their formulation.

Finally, we note that our functionality allows parties to identify a PUF via its identifier id. In particular, this ensures that if an honest P sends an honestly generated PUF to a (potentially malicious) party P', and then at some later point in time P' is supposed to send that same PUF back to P, then P can verify that the PUF sent by P' is indeed the correct one. Although a physical PUF may not obviously have this property (since it may not have an obvious identifier), a trivial way to achieve the same effect is for P to *mark* a PUF by querying it at a random point and recording the response; when the PUF is later returned, P can verify that the same response is returned for the same challenge. In our protocol descriptions, we implicitly assume that the honest party generating a PUF uses this marking approach.

3. Impossibility Result for Malicious, Stateful PUFs

In this section we prove the impossibility of constructing unconditionally secure oblivious transfer when the attacker is able to create malicious, *stateful* PUFs. Our impossibility result applies even for an indistinguishability-based, stand-alone definition of security for OT. We assume OT protocols with perfect correctness for simplicity; however, this is not essential. Formally, we show:

Theorem 1. *Let Π be a PUF-based OT protocol where the sender $\mathcal{S}$ and receiver $\mathcal{R}$ each make polynomially many (in the security parameter λ) PUF queries. Consider executions where the sender $\mathcal{S}$ is given uniform inputs $s_0, s_1 \in \{0, 1\}$, the receiver $\mathcal{R}$ is given uniform input $b \in \{0, 1\}$, and the receiver learns s_b. Then at least one of the following holds:*

1. *There is an unbounded sender $\mathcal{S}^*$ that uses malicious, stateful PUFs, makes polynomially many PUF queries, and outputs the bit b of $\mathcal{R}$ with probability at least* $1/2 + 1/\mathsf{poly}(\lambda)$.
2. *There is an unbounded receiver $\mathcal{R}^*$ that uses malicious, stateful PUFs, makes polynomially many PUF queries, and outputs both bits s_0, s_1 of $\mathcal{S}$ with probability at least* $1/2 + 1/\mathsf{poly}(\lambda)$.

We begin with a high-level overview of our proof before giving the details in the following sections. The starting point for our result is the impossibility of constructing OT in the random-oracle model. Such a result is implied by the impossibility of key agreement in the random-oracle model [3,13], but we sketch a direct proof here.

Consider an OT protocol in the random-oracle model between a sender $\mathcal{S}$ and receiver $\mathcal{R}$. We show that either $\mathcal{S}$ or $\mathcal{R}$ can attack the protocol in the same sense as in Theorem 1. Imagine that both parties run the protocol honestly and then at the end of the protocol they each run the Eve algorithm from [3] to obtain a set Q of queries to (and corresponding answers from) the random oracle. This set Q contains all *intersection queries* between $\mathcal{S}$ and $\mathcal{R}$, namely, all queries that were made by both parties to the random oracle. (Note that the setting here is different from the key agreement setting in which a third party—i.e., an eavesdropper—runs the Eve algorithm; in our setting, finding a set containing all the intersection queries is trivial for $\mathcal{S}$ and $\mathcal{R}$ since all intersection queries are, by definition, already contained in the view of each party. The point of running the Eve algorithm is for both parties to reconstruct the *same* set Q containing all intersection queries.) As in [3], conditioned on the transcript of the protocol and this set Q, the views of $\mathcal{S}$ and $\mathcal{R}$ are independent. We conclude, following [3], that with high probability 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 only on the transcript and Q.

To use the above to obtain an attack, we first consider the distribution D over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Q. If the protocol is secure against a malicious sender, then the probability that $b = 0$ (resp., $b = 1$) in that view must be roughly $1/2$, and in particular it is possible to sample views of $\mathcal{R}$ consistent with both $b = 0$ and $b = 1$. Next consider the distribution D' over $\mathcal{R}$'s view conditioned on the transcript and Q. Since this distribution is statistically close to the previous distribution, it must again be possible to sample views of $\mathcal{R}$ consistent with both $b = 0$ and $b = 1$. But since $\mathcal{R}$ can sample from distribution D', this means that a malicious $\mathcal{R}^*$ can with high probability sample a view consistent with $b = 0$ and the protocol transcript *and* a view consistent with $b = 1$ and the given transcript. Correctness of the protocol then implies that $\mathcal{R}$ can with high probability learn both of $\mathcal{S}$'s inputs. We remark that this argument shows that even unconditionally secure *semi-honest* OT cannot be constructed in the random-oracle model.

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 it. This means that the above attack, as described, will not work; in particular, since at most one party holds each PUF at the end of the protocol, it is not clear that both parties can compute Q. In fact, as we show in Sect. 4, there *does* exist an unconditionally secure OT protocol in the *stateless* malicious-PUF model. This means that any impossibility result must exploit the fact that the malicious party 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. Consider the set Q that contains the following queries:

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

The set Q of queries described above is a *superset* of all the intersection queries made by $\mathcal{S}$ and $\mathcal{R}$ because it contains any query made by both parties to $\mathsf{PUF}'_{\mathcal{S}}$ or $\mathsf{PUF}'_{\mathcal{R}}$, all queries $\mathcal{R}$ made to $\mathsf{PUF}_{\mathcal{S}}$ (which is a superset of the queries made by both parties to $\mathsf{PUF}_{\mathcal{S}}$), and all queries made by $\mathcal{S}$ to $\mathsf{PUF}_{\mathcal{R}}$ (which is a superset of the queries made by both parties to $\mathsf{PUF}_{\mathcal{R}}$). We now want to provide a way for both parties to compute Q.

For queries of the first type, this can be achieved using the Eve algorithm from [3] if we can provide a way for $\mathcal{S}$ to query $\mathsf{PUF}'_{\mathcal{S}}$ (resp., for $\mathcal{R}$ to query $\mathsf{PUF}'_{\mathcal{R}}$) at the end of the protocol. This, in turn, can be done if $\mathcal{S}$ constructs $\mathsf{PUF}'_{\mathcal{S}}$ with known code, so that $\mathcal{S}$ can effectively "query" $\mathsf{PUF}'_{\mathcal{S}}$ even when $\mathsf{PUF}'_{\mathcal{S}}$ is no longer in its possession (and analogously for $\mathsf{PUF}'_{\mathcal{R}}$). Specifically, we have each party embed a 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.

For queries of the second type, we rely on the ability of $\mathcal{S}$ and $\mathcal{R}$ to create *stateful* PUFs. Specifically, we have $\mathcal{S}$ create $\mathsf{PUF}_{\mathcal{S}}$ in such a way that the PUF records (in an undetectable fashion) all the queries that $\mathcal{R}$ makes to $\mathsf{PUF}_{\mathcal{S}}$; this allows $\mathcal{S}$ to later recover those queries once $\mathsf{PUF}_{\mathcal{S}}$ is in its possession. 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 queries Q as defined above is enough for the analysis from [3] to apply. Here it is crucial for the parties to find queries in Q immediately after each protocol message is sent (as opposed to waiting until the end of the protocol) in order to handle the fact that the PUFs are exchanged between the parties.

3.1. *Proof Details*

Let Π be a PUF-based OT protocol with ℓ rounds, where a round involves each party alternately sending a message, and possibly transferring PUFs, to the other party. (So, we have 2ℓ messages overall.) We assume without loss of generality that $\mathcal{S}$ sends the first message of the protocol, and $\mathcal{R}$ sends the final message. We also assume without loss of generality that Π is in *normal form*, meaning that (1) $\mathcal{S}$ and $\mathcal{R}$ each ask at most one PUF query in each round, and (2) the party receiving the final message of the protocol does not query any PUFs after receiving this message. Any protocol can be transformed to one in normal form without affecting its security.

Let z be (a bound on) the total number of PUFs used in the protocol, and let N be a bound on the total number of queries made by each party to those PUFs. To simplify notation, we treat the PUFs $\{\mathsf{PUF}_1, \ldots, \mathsf{PUF}_z\}$ as being defined by a single random function H; the query $q = (j, q')$, which corresponds to the query $\mathsf{PUF}_j(q')$ and can only be made by the party who holds PUF_j, is answered with $H(j, q')$.

For $i \in [2\ell]$ we denote by S^i_{back} the set of $j \in [z]$ such that PUF_j is sent back to its creator along with the ith message. We let S^i_{other} be the set of indices j such that PUF_j is not held by its creator after the ith message is sent. (Note that there can be PUFs that are not in either set.)

We let $\mathsf{M} = m_1, \ldots, m_{2\ell}$ denote the messages exchanged in a protocol execution, and let $\mathsf{M}^i = m_1, \ldots, m_i$ denote the first i messages sent. An *augmented message* $\tilde{m}_i$ consists of m_i along with a set ψ^i that contains all queries made (up to that point in the protocol) to $\{\mathsf{PUF}_j\}_{j \in S^i_{\mathsf{back}}}$ by the party who sent message m_i. We let $\widetilde{\mathsf{M}}, \widetilde{\mathsf{M}}^i$ denote augmented transcripts, and set $\Psi^i \overset{\text{def}}{=} \cup_{j \le i}\ \psi^j$.

We let $\mathcal{V}^i_{\mathcal{S}}$ denote the view of $\mathcal{S}$ up to the moment after the ith message is sent, which includes $\mathcal{S}$'s randomness $r_{\mathcal{S}}$, the partial transcript M^i, and all PUF query/answer pairs known to $\mathcal{S}$ so far. $\mathcal{V}^i_{\mathcal{R}}$ is defined analogously for $\mathcal{R}$. We let $\mathcal{Q}(\cdot)$ be an operator that extracts the set of queries from a set of query/answer pairs or a view.

Executions and distributions A (full) execution of protocol Π can be described by a tuple $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$ where $r_{\mathcal{S}}$ denotes $\mathcal{S}$'s randomness, $r_{\mathcal{R}}$ denotes $\mathcal{R}$'s randomness, and H is a random function that determines the behavior of the PUFs used throughout the protocol. (The parties' randomness is also used to determine their inputs to the protocol.) We denote by $\mathcal{E}$ the experiment in which uniform $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$ are chosen and then the protocol is executed.

Fix a sequence of augmented messages $\widetilde{\mathsf{M}}^i = [\widetilde{m}_1, \ldots, \widetilde{m}_i]$ and a set of query/answer pairs P for which $\Pr_{\mathcal{E}}[(\widetilde{\mathsf{M}}^i, \mathsf{P})] > 0$. We denote by $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ the joint distribution over $(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}})$ (as generated by $\mathcal{E}$) conditioned on the augmented transcript of the first i messages being equal to $\widetilde{\mathsf{M}}^i$ as well as $H(j, q') = a$ for all $((j, q'), a) \in \mathsf{P}$. The event $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ holds in this distribution if and only if $\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}}) \cap \mathcal{Q}(\mathcal{V}^i_{\mathcal{R}}) \subseteq \mathsf{P}^+$, where $\mathsf{P}^+ \overset{\text{def}}{=} \mathsf{P} \cup \Psi^i$, and we define $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ 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 $\mathcal{GV}(\widetilde{\mathsf{M}}, \mathsf{P})$ are defined similarly.

The Eve algorithm We now define a deterministic algorithm Eve. We imagine Eve as running in 2ℓ steps, where in step i it is given the next value $\tilde{m}_i$ in the augmented transcript of a protocol execution and is assumed to have access to the PUFs in S^i_{other}.

Construction 1. *Let $\varepsilon < 1/100$ be a parameter.* Eve *begins with a set* P *of query/answer pairs initialized to $\emptyset$. In step i, given $\tilde{m}_i$, do: as long as there is a query $q = (j, q') \notin \mathsf{P}^+$ such that*

$$\Pr_{(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[q \in \mathcal{Q}(\mathcal{V}^i_{\mathcal{S}}) \wedge j \in S^i_{\mathsf{other}}] \geq \frac{\varepsilon^2}{100m} \quad \textit{or}$$

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

query the lexicographically first such $q = (j, q')$ to H and add $(q, H(q))$ to P.

The output of Eve *is the final set* P.

Let $\Delta(A, B)$ denote the statistical difference between distributions A and B.

Lemma 2. *Construction 1 satisfies the following:*

1. *The expected number of PUF queries made by* Eve (*where the expectation is taken over uniform choice of* $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$) *is* $\mathsf{poly}(N/\varepsilon)$.
2. *Let* P *be the set output by* Eve. (*This is a random variable that depends on* $\widetilde{\mathsf{M}}$, *which in turn depends on* $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$.) *Then with probability at least* $1 - \varepsilon/2$,

$$\Delta(\mathcal{V}_{\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P}),\ \mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})) \leq \varepsilon/2,$$

where $\mathcal{V}_{\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P})$ (*resp.,* $\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P})$) *is the distribution of $\mathcal{S}$'s view* (*resp., $\mathcal{R}$'s view*) *in* $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$.

The proof of this lemma is very similar to the analogous proof by Barak and Mahmoody [3], and we have based our notation and our description of the Eve algorithm on their work. The main difference—besides the fact that we consider augmented transcripts—is that we prove (near) independence of $\mathcal{S}$'s and $\mathcal{R}$'s views, conditioned on the augmented transcript and the queries of Eve, even though Eve can only make queries to a (different) *subset* of the PUFs (namely, those in S^i_{other}) after each message m_i of the protocol is sent. Briefly, this is sufficient since for any PUF not in S^i_{other}, it must be the case that the augmented transcript contains all queries that *one* of the two parties made to that PUF up to the point when m_i is sent. Since S^i_{other} can change after each message is sent, it is important that Eve proceeds in an *online* fashion, asking queries each time a message is sent, rather than delaying its queries to the end of the protocol as in [3].

We prove Lemma 2 in Sect. 3.2; readers willing to take the lemma on faith may skip to Sect. 3.3, where we use the lemma to prove Theorem 1.

3.2. *Analysis of the Eve Algorithm*

Here, we prove Lemma 2. Recall that $H : [z] \times \{0,1\}^n \to \{0,1\}^n$ is sampled uniformly. For any partial function F with domain $D = [z] \times \{0,1\}^n$, we denote by $\Pr_H[F]$ the probability (over choice of H) that H is consistent with F. We rely on the following lemma [3]:

Lemma 3. *For consistent partial functions F_1, F_2 it holds that*

$$\Pr_H[F_1 \cup F_2] = \frac{\Pr_H[F_1] \cdot \Pr_H[F_2]}{\Pr_H[F_1 \cap F_2]}.$$

We first present a product characterization of the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$.

Lemma 4. *For any $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ there exists a distribution* $\mathbf{S}$ *(resp.,* $\mathbf{R}$*) over $\mathcal{S}$'s (resp., $\mathcal{R}$'s) view such that the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is identical to the product distribution* $(\mathbf{S} \times \mathbf{R})$ *conditioned on the event* $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$. *Namely,*

$$\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P}) \equiv \big((\mathbf{S} \times \mathbf{R}) \mid \mathcal{Q}(\mathbf{S}) \cap \mathcal{Q}(\mathbf{R}) \subseteq \mathsf{P}^+\big),$$

where $\mathsf{P}^+ \overset{\text{def}}{=} \mathsf{P} \cup \Psi^i$ *(note that Ψ^i is implicit in $\widetilde{\mathsf{M}}^i$).*

Proof. Let $(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}}) \leftarrow \mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ be such that $\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}}) \cap \mathcal{Q}(\mathcal{V}^i_{\mathcal{R}}) \subseteq \mathsf{P}^+$. We show that

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}})] = \alpha(\widetilde{\mathsf{M}}^i, \mathsf{P}) \cdot \alpha_{\mathcal{S}} \cdot \alpha_{\mathcal{R}},$$

where $\alpha(\widetilde{\mathsf{M}}^i, \mathsf{P})$ depends only on $(\widetilde{\mathsf{M}}^i, \mathsf{P})$, where $\alpha_{\mathcal{S}}$ depends only on $\mathcal{V}^i_{\mathcal{S}}$, and where $\alpha_{\mathcal{R}}$ depends only on $\mathcal{V}^i_{\mathcal{R}}$. This means that if we let $\mathbf{S}$ be the distribution over $\mathrm{Supp}(\mathcal{V}^i_{\mathcal{S}})$ such that $\Pr_{\mathbf{S}}[\mathcal{V}^i_{\mathcal{S}}]$ is proportional to $\alpha_{\mathcal{S}}$ and let $\mathbf{R}$ be the distribution over $\mathrm{Supp}(\mathcal{V}^i_{\mathcal{R}})$ such that $\Pr_{\mathbf{R}}[\mathcal{V}^i_{\mathcal{R}}]$ is proportional to $\alpha_{\mathcal{R}}$, then $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is proportional (and hence equal to) the distribution $\big((\mathbf{S} \times \mathbf{R}) \mid \mathcal{Q}(\mathbf{S}) \cap \mathcal{Q}(\mathbf{R}) \subseteq \mathsf{P}^+\big)$.

Since $\mathcal{Q}(\mathbf{S}) \cap \mathcal{Q}(\mathbf{R}) \subseteq \mathsf{P}^+$, we have

$$\begin{aligned}\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}})] &= \Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}}) \wedge \mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})] \\ &= \Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})] \cdot \Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}})]. \qquad (1)\end{aligned}$$

On the other hand, by the definition of conditional probability we have

$$\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}})] = \frac{\Pr_{\mathcal{E}}[(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})]}{\Pr_{\mathcal{E}}[(\widetilde{\mathsf{M}}^i,\mathsf{P})]}\,. \tag{2}$$

Therefore, by Eqs. (1) and (2) we have

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}})] = \frac{\Pr_{\mathcal{E}}[(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})]}{\Pr_{\mathcal{E}}[(\widetilde{\mathsf{M}}^i,\mathsf{P})]\cdot\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}[\mathsf{Good}(\widetilde{\mathsf{M}}^i,\mathsf{P})]}\,. \tag{3}$$

The denominator of Equation (2) only depends on $(\widetilde{\mathsf{M}}^i,\mathsf{P})$ and so we can take $\beta(\widetilde{\mathsf{M}}^i,\mathsf{P}) = \Pr_{\mathcal{E}}[(\widetilde{\mathsf{M}}^i,\mathsf{P})]\cdot\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}[\mathsf{Good}(\widetilde{\mathsf{M}}^i,\mathsf{P})]$. In the following we analyze the numerator.

We claim:

$$\Pr_{\mathcal{E}}[(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})] = \Pr[\mathbf{r}_{\mathcal{S}}=r_{\mathcal{S}}]\cdot\Pr[\mathbf{r}_{\mathcal{R}}=r_{\mathcal{R}}]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\cup\mathsf{P}].$$

The reason is that $(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})$ occurs in an execution iff when we sample a uniform $(r_{\mathcal{S}},r_{\mathcal{R}},H)$ it holds that $r_{\mathcal{S}}$ equals $\mathcal{S}$'s randomness in $\mathcal{V}^i_{\mathcal{S}}$, $r_{\mathcal{R}}$ equals $\mathcal{R}$'s randomness in $\mathcal{V}^i_{\mathcal{R}}$, and H is consistent with $\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\cup\mathsf{P}$. These conditions imply that $\mathcal{S}$ and $\mathcal{R}$ will produce transcript $\widetilde{\mathsf{M}}^i$ as well.

By Lemma 3, the fact that $\Psi^i\subseteq\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})$, and the fact that $\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cap\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\subseteq\mathsf{P}\cup\Psi^i=\mathsf{P}^+$, we have

$$\begin{aligned}\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\cup\mathsf{P}] &= \Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\cup\mathsf{P}^+]\\ &= \Pr_H[\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}[\big(\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\big)\setminus\big(\mathsf{P}^+\big)]\\ &= \Pr_H[\mathsf{P}^+]\cdot\frac{\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\backslash\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\backslash\mathsf{P}^+]}{\Pr_{\mathcal{E}}[\big(\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\cap\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\big)\setminus\mathsf{P}^+]}\\ &= \Pr_H[\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\backslash\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\backslash\mathsf{P}^+].\end{aligned}$$

Therefore,

$$\begin{aligned}\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}[(\mathcal{V}^i_{\mathcal{S}},\mathcal{V}^i_{\mathcal{R}})] &= \Pr[\mathbf{r}_{\mathcal{S}}=r_{\mathcal{S}}]\cdot\Pr[\mathbf{r}_{\mathcal{R}}=r_{\mathcal{R}}]\cdot\Pr_H[\mathsf{P}^+]\\ &\quad\times\frac{\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\backslash\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\backslash\mathsf{P}^+]}{\beta(\widetilde{\mathsf{M}}^i,\mathsf{P})},\end{aligned}$$

and so we can take

$$\alpha_{\mathcal{S}}=\Pr[\mathbf{r}_{\mathcal{S}}=r_{\mathcal{S}}]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{S}})\backslash\mathsf{P}^+],\quad \alpha_{\mathcal{R}}=\Pr[\mathbf{r}_{\mathcal{R}}=r_{\mathcal{R}}]\cdot\Pr_{\mathcal{E}}[\mathcal{Q}(\mathcal{V}^i_{\mathcal{R}})\backslash\mathsf{P}^+],$$

and $\alpha(\widetilde{\mathsf{M}}^i,\mathsf{P})=\frac{\Pr_H[\mathsf{P}^+]}{\beta(\widetilde{\mathsf{M}}^i,\mathsf{P})}$. □

In the remainder of this section, let $\varepsilon_1 = \varepsilon^2/100$.

Lemma 5. *Let* $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ *be the augmented partial transcript and the set of oracle query/answer pairs made by* Eve *when the last message in* $\widetilde{\mathsf{M}}^i$ *is sent, with* $\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}[\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})] > 0$. *For every such* $(\widetilde{\mathsf{M}}^i, \mathsf{P})$*, there is a bipartite graph* G *with vertex sets* $(\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}})$ *and edges* E *such that:*

1. *Every vertex* u *in* $\mathcal{U}_{\mathcal{S}}$ *has a corresponding view* $\mathcal{S}_u$ *for* $\mathcal{S}$ *and a set* $\mathcal{Q}_u = \mathcal{Q}(\mathcal{S}_u)\backslash\mathsf{P}^+$. *The same holds for vertices in* $\mathcal{U}_{\mathcal{R}}$ *with* $\mathcal{R}$ *in place of* $\mathcal{S}$.
2. *There is an edge between* $u \in \mathcal{U}_{\mathcal{S}}$ *and* $v \in \mathcal{U}_{\mathcal{R}}$ *if and only if* $\mathcal{Q}_u \cap \mathcal{Q}_v = \emptyset$.
3. *Every vertex is connected to at least a* $(1 - 2\varepsilon_1)$*-fraction of vertices in the other component.*
4. *The distribution* $(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ *is identical to sampling a uniform edge* $(u, v) \leftarrow E$ *and taking* $(\mathcal{S}_u, \mathcal{R}_v)$ *(i.e., the views corresponding to* u *and* v*).*

Proof. For fixed $(\widetilde{\mathsf{M}}^i, \mathsf{P})$, the bipartite graph $G = (\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}}, E)$ is defined as follows. Every node $u \in \mathcal{U}_{\mathcal{S}}$ will have a corresponding partial view $\mathcal{S}_u$ of $\mathcal{S}$ that is in the support of the distribution $\mathbf{S}$ from Lemma 4. We let the number of nodes corresponding to a view $\mathcal{V}^i_{\mathcal{S}}$ be proportional to $\Pr_{\mathbf{S}}[\mathbf{S} = \mathcal{V}^i_{\mathcal{S}}]$, meaning that $\mathbf{S}$ corresponds to the uniform distribution over the vertices in $\mathcal{U}_{\mathcal{S}}$. Similarly, every node $v \in \mathcal{U}_{\mathcal{R}}$ will have a corresponding partial view $\mathcal{R}_v$ such that $\mathbf{R}$ corresponds to the uniform distribution over $\mathcal{U}_{\mathcal{R}}$.

For $u \in \mathcal{U}_{\mathcal{S}}$, we define $\mathcal{Q}_u = \mathcal{Q}(\mathcal{S}_u)\backslash\left(\mathsf{P} \cup \Psi^i\right) = \mathcal{Q}(\mathcal{S}_u)\backslash\mathsf{P}^+$ to be the set of queries *outside* P^+ that were asked by $\mathcal{S}$ in the view $\mathcal{S}_u$. We define $\mathcal{Q}_v = \mathcal{Q}(\mathcal{R}_v)\backslash\mathsf{P}^+$ similarly. We put an edge between nodes u and v in G (denoted by $u \sim v$) if and only if $\mathcal{Q}_u \cap \mathcal{Q}_v = \emptyset$.

It can be seen that the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is equal to the distribution obtained by choosing a uniform edge $u \sim v$ of G and then outputting the views $(\mathcal{S}_u, \mathcal{R}_v)$. It is thus immediate that properties 1, 2, and 4 hold. It remains to show property 3. To show this, we will argue that the graph G is *dense* as formalized in the next claim. □

Claim 6. *For every* $u \in \mathcal{U}_{\mathcal{S}}$*,* $d(u) \geq |\mathcal{U}_{\mathcal{R}}| \cdot (1 - 2\varepsilon_1)$ *and for every* $v \in \mathcal{U}_{\mathcal{R}}$*,* $d(v) \geq |\mathcal{U}_{\mathcal{S}}| \cdot (1 - 2\varepsilon_1)$*, where* $d(w)$ *is the degree of vertex* w.

To prove the claim, we first show that for every $w \in \mathcal{U}_{\mathcal{S}}$, it holds that $\sum_{v\in\mathcal{U}_{\mathcal{R}}, w\not\sim v} d(v) \leq \varepsilon_1 \cdot |E|$. The reason is that the probability of vertex v being chosen when we choose a uniform edge is $d(v)/|E|$, and if $\sum_{v\in\mathcal{U}_{\mathcal{R}}, w\not\sim v} d(v)/|E| > \varepsilon_1$ it means that $\Pr_{(u,v)\leftarrow E}[\mathcal{Q}_w \cap \mathcal{Q}_u \neq \emptyset] \geq \varepsilon_1$. Moreover, note that $\mathcal{Q}_w \cap \mathcal{Q}_u$ can only contain queries of the form $q = (j, q')$ where $j \in S^i_{\mathsf{back}}$ since, for any $j \notin S^i_{\mathsf{back}}$, the set P^+ contains all queries made by at least one of the parties to PUF_j by the time the ith message was sent. Hence, by the pigeonhole principle (since $|\mathcal{Q}_w| \leq m$), there must exist $q = (j, q')$ where $j \in \mathsf{back}^i$ such that $\Pr_{(u,v)\leftarrow E}[q \in Q_v] \geq \varepsilon_1/m$. But this is a contradiction, because if that holds then q should have been in P by definition of Eve (and hence q could not be in $\mathcal{Q}_w$). The same argument shows that for every $w \in \mathcal{U}_{\mathcal{R}}$, $\sum_{u\in\mathcal{U}_{\mathcal{S}}, w\not\sim v} d(u) \leq \varepsilon_1 \cdot |E|$. Thus, for every vertex $w \in \mathcal{U}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{R}}$, $|E^{\not\sim}(w)| \leq \varepsilon_1|E|$ where $E^{\not\sim}(w)$ denotes the set of edges that do not contain any neighbor of w (i.e., $E^{\not\sim}(w) = \{(u, v) \in E \mid u \not\sim w \wedge w \not\sim\}$).

The following claim was proved in [3]:

Claim 7. *For $\varepsilon_1 \le 1/2$, let $G = (\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}}, E)$ be a nonempty bipartite graph where $|E^{\not\sim}(w)| \le \varepsilon_1 \cdot |E|$ for all vertices $w \in \mathcal{U}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{R}}$. Then $d(u) \ge |\mathcal{U}_{\mathcal{R}}| \cdot (1 - 2\varepsilon_1)$ for all $u \in \mathcal{U}_{\mathcal{S}}$ and $d(v) \ge |\mathcal{U}_{\mathcal{S}}| \cdot (1 - 2\varepsilon_1)$ for all $v \in \mathcal{U}_{\mathcal{R}}$.*

This completes the proof of Claim 6 and therefore the proof of Lemma 5. □

We say that event Fail holds if and only if for some $i \in [2\ell]$, immediately after the ith message is sent, $\mathcal{S}$ or $\mathcal{R}$ makes a query q that was made already by the other party but is not contained in $\mathsf{P}^+ := \mathsf{P} \cup \Psi^i$. If the $(i+1)$st query is the first for which Fail happens and i is odd, we say event RFail_i occurs, and if i is even we say event SFail_i occurs.

Lemma 8. *For odd $i \in [2\ell]$ and every $(\mathcal{V}^i_{\mathcal{R}}, \widetilde{\mathsf{M}}^i, \mathsf{P})$ sampled by executing the system it holds that*

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}[\mathsf{RFail}_i \mid \mathcal{V}^i_{\mathcal{R}}] \le \frac{3\varepsilon_1}{2m}.$$

A symmetric statement holds for even $i \in [2\ell]$ and $\mathcal{S}$.

Proof. Let $q = (j, q')$ be the $(i+1)$st query of the protocol, made by $\mathcal{R}$ immediately after the last message $\widetilde{m}_i$ in $\widetilde{\mathsf{M}}^i$ is sent from $\mathcal{S}$ to $\mathcal{R}$. By Lemma 5, the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ conditioned on getting $\mathcal{V}^i_{\mathcal{R}}$ as $\mathcal{R}$'s view is the same as sampling a uniform edge $u \sim v$ in the graph G conditioned on $\mathcal{R}_v = \mathcal{V}^i_{\mathcal{R}}$. We prove Lemma 8 even conditioned on choosing any vertex v such that $\mathcal{R}_v = \mathcal{V}^i_{\mathcal{R}}$. For such fixed v, the distribution of $\mathcal{S}$'s view $\mathcal{S}_v$ when we choose a uniform edge $u \sim v'$ conditioned on $v = v'$ is the same as choosing a uniform neighbor $u \leftarrow N(v)$ of the node v and then selecting $\mathcal{S}$'s view $\mathcal{S}_u$ corresponding to node u. Let $S = \{u \in \mathcal{U}_{\mathcal{S}} \mid q \in \mathcal{Q}_u\}$. Note that if $q = (j, q')$ is such that $j \notin S^i_{\mathsf{other}}$ then we have

$$\Pr_{u \leftarrow N(v)}[q \in \mathcal{Q}_u] = 0.$$

This is because $\mathcal{R}$ can only query a PUF that it holds at the point right after the ith message is sent. However, if PUF_j is such that $\mathcal{R}$ currently holds it, and $\mathsf{PUF}_j \notin S^i_{\mathsf{other}}$, then PUF_j must have been created by $\mathcal{R}$. Thus, by definition of the augmented transcript $\widetilde{\mathsf{M}}^i$, all queries made by $\mathcal{S}$ to PUF_j up to this point in the protocol are included in $\Psi^i \subseteq \mathsf{P}^+$ and thus cannot be in $\mathcal{Q}_u = \mathcal{Q}(\mathcal{S}_u) \backslash \mathsf{P}^+$.

We can therefore focus our attention on queries $q = (j, q')$ such that $j \in \mathcal{S}^i$. We have

$$\begin{aligned}\Pr_{u \leftarrow N(v)}[q \in \mathcal{Q}_u] &\le \frac{|S|}{d(v)} \le \frac{|S|}{(1-2\varepsilon_1) \cdot \mathcal{U}_{\mathcal{S}}} \le \frac{|S| \cdot |\mathcal{U}_{\mathcal{R}}|}{(1-2\varepsilon_1) \cdot |E|} \\ &\le \frac{\sum_{u \in S} d(u)}{(1-2\varepsilon_1)^2 \cdot |E|} \le \frac{\varepsilon_1}{(1-2\varepsilon_1)^2 \cdot m} \le \frac{3\varepsilon_1}{2m}.\end{aligned}$$

The second and fourth inequalities are by Lemma 5. The third inequality is because $|E| \leq |\mathcal{U}_{\mathcal{S}}| \cdot |\mathcal{U}_{\mathcal{R}}|$, and the sixth inequality is because $\varepsilon_1 < \varepsilon < 1/100$. The fifth inequality is by definition of Eve, who asks high-probability queries (j, q'), for $j \in \mathcal{S}^i$, as long as such queries exist. Namely, when we choose a uniform edge $u \sim v$ (which by Lemma 5 is the same as sampling $(\mathcal{V}^i_{\mathcal{S}}, \mathcal{V}^i_{\mathcal{R}}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$), it holds that $u \in S$ with probability $\sum_{u \in S} d(u)/|E|$. But for all $u \in S$ it holds that $q \in \mathcal{Q}_u$, and so if $\sum_{u \in S} d(u)/|E| > \varepsilon_1/m$ the query q would have been made by Eve already, and by property 2 of Lemma 5 it cannot be the case that q is in any set $\mathcal{Q}_u$. □

Given Lemma 8, Lemma 2 holds via the same analysis as in [3].

3.3. *Attacking the Protocol*

We now show how to use the Eve algorithm from Construction 1 to derive an attack on protocol Π. First note that although Eve, as defined, makes an *expected* polynomial number of queries to the PUFs, we can modify Eve in the standard way so that it makes at most $t = \mathsf{poly}(N/\epsilon)$ queries to the PUFs and such that with probability at least $1 - \varepsilon$ over the augmented transcript $\widetilde{\mathsf{M}}$ and the output P of Eve it holds that

$$\Delta(\mathcal{V}_{\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P}),\ \mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})) \leq \varepsilon. \tag{4}$$

We assume this Eve is used in all that follows.

A second, crucial observation is that *both a malicious $\mathcal{S}$ and a malicious $\mathcal{R}$* can create PUFs so that they can run Eve during an execution of Π. Here, we rely critically on the fact that the parties can create malicious, stateful PUFs. We describe how a malicious $\mathcal{S}$ can run Eve, but note that the situation is symmetric from the point of view of a malicious $\mathcal{R}$.

- Let $t^* \overset{\text{def}}{=} t + 2N$. All PUFs created by $\mathcal{S}$ will be modified in the following two ways: first, instead of being created (honestly) as a random function, each PUF will be defined by choosing an independent key k for a t^*-wise independent function h. Second, each PUF will use its state to keep track[1] of the queries made to it by $\mathcal{R}$.
- $\mathcal{S}$ otherwise runs the protocol honestly, running Eve after each message of the protocol is sent. We describe how this is done both when $\mathcal{S}$ sends and when it receives a message:

  – Consider the case after $\mathcal{S}$ sends the ith message m_i. Observe that $\mathcal{S}$ knows ψ^i because it knows S^i_{back} and it certainly knows the queries it made to those PUFs. Moreover, we show that it can access the PUFs in S^i_{other}. The PUFs in S^i_{other} are of two types: those created by $\mathcal{R}$ but currently held by $\mathcal{S}$, and those created by $\mathcal{S}$ but currently held by $\mathcal{R}$. PUFs of the first type can be directly accessed by $\mathcal{S}$. PUFs of the second type can still be computed by $\mathcal{S}$ because

[1] This is easy to do by having $\mathcal{S}$ choose a random "trapdoor" td, and then create a circuit with the following behavior: on input $x \neq \mathsf{td}$, return $h_k(x)$ and concatenate x to the state; on input td, return the current state. Note that creating PUFs in this way will have only a negligible effect on the output of an honest execution of any PUF-based protocol, since the probability that the PUF is queried with td during execution of the protocol is negligible.

they are defined by a t^*-wise independent hash function whose key is known by $\mathcal{S}$. We conclude that $\mathcal{S}$ is able to run Eve in this case.

– Consider the case when $\mathcal{S}$ receives the ith message m_i. Along with that message, $\mathcal{S}$ receives a set of PUFs S^i_{back} from $\mathcal{R}$. Since (by definition) all those PUFs were created by $\mathcal{S}$, it can extract all the queries made by $\mathcal{R}$ to those PUFs and hence compute ψ^i. Exactly as before, $\mathcal{S}$ can also access all the PUFs in S^i_{other}. We conclude that $\mathcal{S}$ is able to run Eve in this case as well.

The total number of queries made to any PUF throughout the entire experiment, whether by $\mathcal{S}$ or $\mathcal{R}$ as part of the protocol or by Eve, is at most $t + 2N = t^*$. This means that the resulting distribution is identical to the one analyzed in Lemma 2 (with the exception that we bound the number of queries made by Eve, as discussed above).

With the above in place, we show that either $\mathcal{S}$ or $\mathcal{R}$ can carry out a successful attack by running Eve. We let $\tilde{\Pi}$ denote the experiment in which $\mathcal{S}$ and $\mathcal{R}$ run an honest execution of the protocol and Eve is additionally run with $\epsilon < 1/400$. We let $\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathcal{V}_{\mathcal{S}})$ denote the distribution on the view of $\mathcal{R}$ in $\tilde{\Pi}$ conditioned on the augmented transcript of the first i messages being equal to $\widetilde{\mathsf{M}}^i$, the view of $\mathcal{S}$ being equal to $\mathcal{V}_{\mathcal{S}}$, and $H(j, q') = a$ for all $((j, q'), a) \in \mathsf{P}$. Given a view $\mathcal{V}_{\mathcal{R}}$ of $\mathcal{R}$, we let $\mathsf{in}(\mathcal{V}_{\mathcal{R}})$ denote the input bit of $\mathcal{R}$ in that view, and let $\mathsf{out}(\mathcal{V}_{\mathcal{R}})$ be the output bit of $\mathcal{R}$ implicit in that view (i.e., as dictated by Π). We define $\mathsf{in}(\mathcal{V}_{\mathcal{R}})$ analogously for $\mathcal{S}$.

Let $p(\cdot)$ be a sufficiently large polynomial. Clearly, one of the following must hold:

Case 1 For infinitely many λ, with probability at least $1/p(\lambda)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathcal{V}_{\mathcal{S}})$ generated by a run of $\tilde{\Pi}$ either

$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}}) = 0] < 0.45 \quad \text{or} \quad \Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}}) = 1] < 0.45.$$

Case 2 For infinitely many λ, with probability at least $1 - 1/p(\lambda)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathcal{V}_{\mathcal{S}})$ generated by a run of $\tilde{\Pi}$

$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}}) = 0] \geq 0.45 \quad \text{and} \quad \Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}}) = 1] \geq 0.45.$$

We show that if case 1 holds then a malicious sender can successfully attack the protocol, whereas if case 2 holds then a malicious receiver can successfully attack the protocol.

3.3.1. *Attack by a Malicious Sender*

Assume case 1 holds. The attack by a malicious sender $\mathcal{S}^*$ proceeds as follows. It runs the protocol with the honest receiver, additionally running Eve after each message is sent (as described previously). At the end of the execution, it has an augmented transcript $\widetilde{\mathsf{M}}$, the set of queries P output by Eve, and its own view $\mathcal{V}_{\mathcal{S}}$. It then computes the probabilities

$$p_0 \overset{\text{def}}{=} \Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}}) = 0] \quad \text{and} \quad p_1 \overset{\text{def}}{=} \Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}}) = 1].$$

Finally, if $p_0 < 0.45$ it outputs 1; if $p_1 < 0.45$ it outputs 0; and otherwise it outputs a random bit. (Note that $p_0 + p_1 = 1$. Therefore, if $p_0 < 0.45$ we must have $p_1 \geq 0.55$ and vice versa.)

We now analyze the probability with which the output of $\mathcal{S}^*$ is equal to $\mathcal{R}$'s input bit. Since case 1 holds, we know that for infinitely many λ the probability that $p_0 < 0.45$ or $p_1 < 0.45$ is at least $1/p(\lambda)$. When that happens, $\mathcal{S}^*$ correctly predicts $\mathcal{R}$'s output with probability at least 0.55; otherwise, $\mathcal{S}^*$ guesses $\mathcal{R}$'s output with probability 0.5. Overall, then, $\mathcal{S}^*$ outputs $\mathcal{R}$'s input with probability at least

$$0.55 \cdot \frac{1}{p(\lambda)} + 0.5 \cdot \left(1 - \frac{1}{p(\lambda)}\right) = \frac{1}{2} + \frac{1}{20 \cdot p(\lambda)}$$

for infinitely many λ.

3.3.2. *Attack by a Malicious Receiver*

Assume case 2 holds. The attack by a malicious receiver $\mathcal{R}^*$ proceeds as follows. It runs the protocol with the honest sender, additionally running Eve after each message is sent (as described previously). At the end of the execution, it has an augmented transcript $\widetilde{\mathsf{M}}$ and the set of queries P output by Eve. It then computes the probabilities

$$p_0' \overset{\text{def}}{=} \Pr_{\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{in}(\mathcal{V}_\mathcal{R}) = 0] \quad \text{and} \quad p_1' \overset{\text{def}}{=} \Pr_{\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{in}(\mathcal{V}_\mathcal{R}) = 1].$$

If $p_0' = 0$ or $p_1' = 0$ then it outputs $\perp$ and terminates. Otherwise, it samples views $\mathcal{V}_\mathcal{R}^0$ and $\mathcal{V}_\mathcal{R}^1$ from $\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{in}(\mathcal{V}_\mathcal{R}) = 0)$ and $\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{in}(\mathcal{V}_\mathcal{R}) = 1)$, respectively (where $\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{in}(\mathcal{V}_\mathcal{R}) = b)$ denotes that the distribution is additionally conditioned on $\mathsf{in}(\mathcal{V}_\mathcal{R}) = b$). Finally, it outputs $s_0' = \mathsf{out}(\mathcal{V}_\mathcal{R}^0)$ and $s_1' = \mathsf{out}(\mathcal{V}_\mathcal{R}^1)$.

We are interested in the probability with which s_0', s_1' correspond to the inputs bits of $\mathcal{S}$. Toward this, we first prove the following lemma:

Lemma 9. *With probability at least* $1 - \sqrt{\varepsilon} - \varepsilon > 1 - 2\sqrt{\varepsilon}$ *over* $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathcal{V}_\mathcal{S})$ *generated in* $\widetilde{\Pi}$*:*

$$\Delta\left(\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}),\ \mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathcal{V}_\mathcal{S})\right) \leq \sqrt{\varepsilon}.$$

Proof. Note that for any fixed $(\widetilde{\mathsf{M}}, \mathsf{P})$,

$$\begin{aligned}&\Delta\left(\mathcal{V}_\mathcal{S}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}),\ \mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})\right)\\ &\quad = \mathbf{Exp}_{\mathcal{V}_\mathcal{S} \sim \mathcal{V}_\mathcal{S}(\widetilde{\mathsf{M}},\mathsf{P})}\left[\Delta\left(\mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}),\ \mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathcal{V}_\mathcal{S})\right)\right].\end{aligned}$$

Moreover, from Eq. (4) we know that with probability at least $1 - \varepsilon$ over $(\widetilde{\mathsf{M}}, \mathsf{P})$

$$\Delta\left(\mathcal{V}_\mathcal{S}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_\mathcal{R}(\widetilde{\mathsf{M}}, \mathsf{P}),\ \mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})\right) \leq \varepsilon.$$

The lemma follows using Markov's inequality. □

Since case 2 holds, we know that for infinitely many λ the probability that both

$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=0]\geq 0.45 \quad\text{and}\quad \Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=1]\geq 0.45$$

is at least $1-1/p(\lambda)$. By perfect correctness, we also have

$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=0\wedge \mathsf{out}(\mathcal{V}_{\mathcal{R}})\neq s_0]=0 \quad\text{and}$$
$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=1\wedge \mathsf{out}(\mathcal{V}_{\mathcal{R}})\neq s_1]=0.$$

Thus, using Lemma 9 and the fact that $\varepsilon\leq 1/400$ (so $2\sqrt{\varepsilon}\leq 0.1$), and taking p large enough so that $1/p(\lambda)\leq 0.1$, we have that (for infinitely many λ) with probability at least $1-2\sqrt{\varepsilon}-1/p(\lambda)\geq 0.8$ over $(\widetilde{\mathsf{M}},\mathsf{P},\mathcal{V}_{\mathcal{S}})$ generated in $\tilde{\Pi}$ all the following hold:

- $\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=0]\geq 0.4$ and $\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=1]\geq 0.4$.
- $\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=0\wedge \mathsf{out}(\mathcal{V}_{\mathcal{R}})\neq s_0]\leq 0.05$.
- $\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{in}(\mathcal{V}_{\mathcal{R}})=1\wedge \mathsf{out}(\mathcal{V}_{\mathcal{R}})\neq s_1]\leq 0.05$.

Returning to our analysis of $\mathcal{R}^*$, we thus see that, for infinitely many λ, with probability at least 0.8 it holds that $p_0'\neq 0$, $p_1'\neq 0$, and

$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{out}(\mathcal{V}_{\mathcal{R}})\neq s_0\mid \mathsf{in}(\mathcal{V}_{\mathcal{R}})=0]\leq 1/8 \quad\text{and}$$
$$\Pr_{\mathcal{V}_{\mathcal{R}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{out}(\mathcal{V}_{\mathcal{R}})\neq s_1\mid \mathsf{in}(\mathcal{V}_{\mathcal{R}})=1]\leq 1/8.$$

So with probability at least $0.8\cdot\left(1-\frac{1}{8}-\frac{1}{8}\right)=0.6$, the output of $\mathcal{R}^*$ is equal to the input of $\mathcal{S}$.

4. Feasibility Results 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 an unconditionally secure construction of a universally composable OT protocol in this model, described in Sect. 4.1. In Sect. 4.2 we briefly discuss how this protocol can be used to obtain the claimed result.

4.1. *Universally Composable Oblivious Transfer*

Our OT protocol adapts the protocol of Brzuska et al. [5], which was proven secure against attackers limited to honestly generated PUFs. It is easy to see that their protocol is not secure against attackers who can create malicious stateless PUFs. We show that having the sender create the PUF instead of the receiver is sufficient to obtain security in that case.

The protocol, described in Fig. 3, consists of a preprocessing phase run by the sender $\mathcal{S}$ and receiver $\mathcal{R}$, followed by a predetermined number N of oblivious transfers. In the

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

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

Input: $s_0, s_1 \in \{0,1\}^\lambda$		Input: $b \in \{0,1\}$
$x \leftarrow \{0,1\}^\lambda$	$\xrightarrow{x}$	
		$v := c_i \oplus (b \cdot x)$
	$\xleftarrow{v}$	
$\hat{r}_0 := \mathsf{PUF}(i, v)$		
$\hat{r}_1 := \mathsf{PUF}(i, (v \oplus x))$		
$S_0 := s_0 \oplus \hat{r}_0$		
$S_1 := s_1 \oplus \hat{r}_1$		
	$\xrightarrow{S_0, S_1}$	
		Output: $s_b := S_b \oplus r_i$

Fig. 3. Our OT protocol. Following a preprocessing phase, the parties can execute N instances of oblivious transfer.

preprocessing phase, $\mathcal{S}$ first creates a PUF PUF and sends it to the receiver. The receiver then chooses N uniform values $c_1, \ldots, c_N$ and, for each one, computes $r_i := \mathsf{PUF}(i, c_i)$. It then sends PUF back to $\mathcal{S}$. (Recall from Sect. 2 that we assume $\mathcal{S}$ can verify that $\mathcal{R}$ sent back the same PUF that $\mathcal{S}$ created.)

When the parties want to execute the ith oblivious transfer, they proceed as follows. $\mathcal{S}$ begins by sending a uniform value x. Then $\mathcal{R}$, with choice bit b, computes $v := c_i \oplus (b \cdot x)$ and sends v back to $\mathcal{S}$. The sender then computes $\hat{r}_0 := \mathsf{PUF}(i, v)$ and $\hat{r}_1 := \mathsf{PUF}(i, v \oplus x)$ and uses these values to "mask" its inputs s_0, s_1. Since $\mathcal{R}$ knows $\mathsf{PUF}(i, v \oplus (b \cdot x)) = \mathsf{PUF}(i, c_i)$, it can recover s_b.

We prove that this protocol is secure even if a malicious $\mathcal{S}$ can create a malicious (but stateless) PUF. Our proof of security is unconditional, but assumes that the malicious party is limited to polynomially many queries to the PUF.

Theorem 10. *The protocol in Fig. 3 securely realizes $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}^0_{\mathsf{PUF}}$-hybrid model.*

Proof. For simplicity we assume $N = 1$, but the proof extends in a straightforward way for $N > 1$. The case where both $\mathcal{S}$ and $\mathcal{R}$ are honest is trivial, and so we focus on the case where one of the parties is corrupted.

Receiver is corrupted We take the corrupted receiver $\mathcal{R}^*$ to be the dummy adversary who simply forwards messages to/from the environment $\mathcal{Z}$. We describe an ideal-world simulator Sim that plays the role of the receiver in an interaction with the ideal-world OT functionality $\mathcal{F}_{\mathsf{OT}}$. We then argue that no environment $\mathcal{Z}$ can distinguish an interaction between an honest $\mathcal{S}$ and Sim in the ideal world from an execution of our protocol between $\mathcal{S}$ and $\mathcal{R}^*$ in the $\mathcal{F}^0_{\mathsf{PUF}}$-hybrid world.

Sim is defined as follows:

- Sim simulates a copy of an honestly generated PUF PUF sent by $\mathcal{S}$, i.e., when $\mathcal{Z}$ makes a query $\mathsf{PUF}(i, c)$ that has not been made before, Sim chooses a uniform value and returns it to $\mathcal{Z}$. At some point, $\mathcal{Z}$ indicates that PUF should be returned to $\mathcal{S}$. (Recall from Sect. 2 that we assume $\mathcal{S}$ can verify that its PUF is returned, and the protocol does not proceed until this is done.) Let Q denote the set of queries of the form $\mathsf{PUF}(1, \star)$ made in this step.
- When the second phase of the protocol is initiated, Sim chooses a uniform value x and sends it to $\mathcal{Z}$ (as if sent by $\mathcal{S}$). Next, $\mathcal{Z}$ specifies a message v to be sent to $\mathcal{S}$. Then:

  – If $v \in Q$ and $v \oplus x \in Q$ then Sim aborts.
  – If $v \in Q$ but $v \oplus x \notin Q$ then Sim sets $b = 0$ and $\hat{r} = \mathsf{PUF}(1, v)$ (i.e., $\hat{r}$ is the value returned previously in response to the same query).
  – If $v \notin Q$ but $v \oplus x \in Q$ then Sim sets $b = 1$ and $\hat{r} = \mathsf{PUF}(1, v \oplus x)$.
  – If $v \notin Q$ and $v \oplus x \notin Q$ then Sim sets $b = 1$ and chooses uniform $\hat{r}$.

  Sim sends b to $\mathcal{F}_{\mathsf{OT}}$ and receives in return a bit s_b. It then sets $S_b := s_b \oplus \hat{r}$ and chooses uniform $S_{\bar{b}}$. Finally, it sends S_0, S_1 to $\mathcal{Z}$.

It is not hard to see that the simulation is perfect unless $v \in Q$ and $v \oplus x \in Q$. We show that this event, which we call Bad, occurs with negligible probability. Let $p = p(\lambda)$ be a bound on the number of queries to PUF made by $\mathcal{Z}$, so $|Q| \leq p$. Event Bad can only possibly occur if there exist $q_1, q_2 \in Q$ such that $q_1 \oplus q_2 = x$. Since there are at most p^2 pairs $q_1, q_2 \in Q$ and x is uniform, this means that the probability of Bad is at most $p^2/2^\lambda$, which is negligible.

Sender is corrupted We take the corrupted sender $\mathcal{S}^*$ to be the dummy adversary who simply forwards messages to/from the environment $\mathcal{Z}$. We describe an ideal-world simulator Sim that plays the role of the sender in an interaction with the ideal-world OT functionality $\mathcal{F}_{\mathsf{OT}}$. We then argue that no environment $\mathcal{Z}$ can distinguish an interaction between an honest $\mathcal{R}$ and Sim in the ideal world from an execution of our protocol between $\mathcal{R}$ and $\mathcal{S}^*$ in the $\mathcal{F}^0_{\mathsf{PUF}}$-hybrid world.

Sim is defined as follows:

- We assume without loss of generality that $\mathcal{Z}$ requests creation of a malicious PUF PUF^*, defined via a circuit C. Any queries by $\mathcal{Z}$ to PUF^*, whether in this phase or the next phase, are answered in the obvious way by Sim. (Note that C may require oracle access to an honestly generated PUF, but that can be handled by Sim in the natural way.)

Sim simulates the sending of PUF^* to $\mathcal{R}$ as well as its return.

- $\mathcal{Z}$ specifies a message x (to be sent to $\mathcal{R}$), and in response Sim chooses a uniform v and sends it to $\mathcal{Z}$. Next, $\mathcal{Z}$ specifies messages S_0, S_1 (to be sent to $\mathcal{R}$). At this point, Sim locally computes $\hat{r}_0 := \mathsf{PUF}^*(1, v)$ and $\hat{r}_1 := \mathsf{PUF}^*(1, v \oplus x)$, sets $s_b = S_b \oplus \hat{r}_b$ for $b \in \{0, 1\}$, and sends (s_0, s_1) to $\mathcal{F}_{\mathsf{OT}}$.

It is immediate that the simulation is perfect. (Here we crucially rely on the fact that PUF^* is stateless, so it always returns the same response to the same challenge and contains no information about $\mathcal{R}$'s queries.) □

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 protocol with semi-honest security based on Yao's garbled-circuit protocol (using a PUF to implement a pseudorandom function) and then to apply the compiler of Ishai, Prabhakaran, and Sahai [14].

Semi-honest secure two-party computation Lindell and Pinkas presented a proof for Yao's two-party secure computation protocol [17]. They show how to instantiate the garbling part of the protocol with a private-key encryption scheme having certain properties. In addition, they 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. Specifically, the circuit generator in Yao's protocol will create a PUF PUF and define $F_k(x) = \mathsf{PUF}(k, x)$. If the circuit generator is honest, then this defines a good pseudorandom function; if the circuit generator is malicious (and in particular if the PUF is malicious), it cannot violate privacy of the other party. (We remark that Brzuska et al. [5] also observed that PUFs can be used to implement a pseudorandom function, though in a different context and assuming honest PUFs.)

Since we showed in the previous section that $\mathcal{F}_{\mathsf{OT}}$ can be realized in the $\mathcal{F}^0_{\mathsf{PUF}}$-hybrid model, we thus have:

Theorem 11. *Let f be any functionality. There is a (constant-round) protocol that securely computes f for semi-honest adversaries in the $\mathcal{F}^0_{\mathsf{PUF}}$-hybrid model.*

Universally composable two-party computation In the next step we apply the IPS compiler [14]. This is a black-box compiler that relies on protocols of the following types:

1. An "outer" multi-party computation protocol Π with security against a constant fraction of malicious parties.
2. An "inner" two-party protocol ρ, in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, secure against semi-honest parties.

The result of the IPS compiler is a two-party protocol, in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, that is universally composable for malicious adversaries.

In our setting, we instantiate the "outer" protocol with the BGW protocol [4], which is unconditionally secure in the presence of a malicious minority. We instantiate the "inner" protocol with the protocol from Theorem 11. Using Theorems 10 and 11, along with the UC composition theorem, we thus obtain the following result:

Theorem 12. *Let f be any functionality. There is a protocol that securely computes f for malicious adversaries in the $\mathcal{F}^{0}_{\mathsf{PUF}}$-hybrid model.*

Acknowledgements

Work of Nils Fleischhacker and Dominique Schröder was done in part while at Saarland University and 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, and the visit of Dominique Schröder was supported by NSF Award #1223623. Work of Dominique Schröder was also supported by an Intel Early Career Faculty Honor Program Award. Work of Jonathan Katz was supported in part by NSF Award #1223623, as well as by a Humboldt Award. Work of Anna Lysyanskaya was supported by NSF Awards #0964379 and #1012060.

References

[1] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, C. Wachsmann, A formalization of the security features of physical functions, in *IEEE Symposium on Security and Privacy* (IEEE, Washington, 2011), pp. 397–412

[2] S. Badrinarayanan, D. Khurana, R. Ostrovsky, I. Visconti, Unconditional UC-secure computation with (stronger-malicious) PUFs, in *Advances in Cryptology—Eurocrypt 2017, Part I, Volume 10210 of LNCS* (Springer, Berlin, 2017), pp. 382–411

[3] B. Barak, M. Mahmoody-Ghidary, Merkle puzzles are optimal—an $O(n^2)$-query attack on any key exchange from a random oracle. *J. Cryptol.* **30**(3), 699–734 (2017)

[4] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for noncryptographic fault-tolerant distributed computations, in *20th Annual ACM Symposium on Theory of Computing (STOC)* (ACM Press, London, 1988), pp. 1–10

[5] C. Brzuska, M. Fischlin, H. Schröder, S. Katzenbeisser, Physically uncloneable functions in the universal composition framework, in *Advances in Cryptology—Crypto 2011, Volume 6841 of LNCS* (Springer, Berlin, 2011), pp. 51–70

[6] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols, in *42nd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, Washington, 2001), pp. 136–145. Full version available at http://eprint.iacr.org/2000/067/

[7] R. Canetti, Y. Dodis, R. Pass, S. Walfish, Universally composable security with global setup, in *4th Theory of Cryptography Conference—TCC 2007, Volume 4392 of LNCS* (Springer, Berlin, 2007), pp. 61–85

[8] R. Canetti, M. Fischlin, Universally composable commitments, in *Advances in Cryptology—Crypto 2001, Volume 2139 of LNCS* (Springer, Berlin, 2001), pp. 19–40

[9] R. Canetti, E. Kushilevitz, Y. Lindell, On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptol.* **19**(2), 135–167 (2006)

[10] I. Damgård, A. Scafuro, Unconditionally secure and universally composable commitments from physical assumptions, In *Advances in Cryptology—Asiacrypt 2013, Part II, Volume 8270 of LNCS* (Springer, Berlin, 2013), pp. 100–119

[11] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **38**(1), 97–139 (2008)

[12] V. Goyal, Y. Ishai, M. Mahmoody, A. Sahai, Interactive locking, zero-knowledge PCPs, and unconditional cryptography, in *Advances in Cryptology—Crypto 2010, Volume 6223 of LNCS* (Springer, Berlin, 2010), pp. 173–190

[13] R. Impagliazzo, S. Rudich, Limits on the provable consequences of one-way permutations, in *21st Annual ACM Symposium on Theory of Computing (STOC)* (ACM Press, New York, 1989), pp. 44–61

[14] Y. Ishai, M. Prabhakaran, A. Sahai, Founding cryptography on oblivious transfer—efficiently, in *Advances in Cryptology—Crypto 2008, Volume 5157 of LNCS* (Springer, Berlin, 2008), pp. 572–591

[15] J. Katz, Universally composable multi-party computation using tamper-proof hardware, in *Advances in Cryptology—Eurocrypt 2007, Volume 4515 of LNCS* (Springer, Berlin, 2007), pp. 115–128

[16] S. Katzenbeisser, Ü. Koçabas, V. Rozic, A.-R. Sadeghi, I. Verbauwhede, C. Wachsmann, PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon, in *Cryptographic Hardware and Embedded Systems—CHES 2012, Volume 7428 of LNCS* (Springer, Berlin, 2012), pp. 283–301

[17] Y. Lindell, B. Pinkas, A proof of security of Yao's protocol for two-party computation. *J. Cryptol.* **22**(2), 161–188 (2009)

[18] R. Ostrovsky, A. Scafuro, I. Visconti, A. Wadia, Universally composable secure computation with (malicious) physically uncloneable functions, in *Advances in Cryptology—Eurocrypt 2013, Volume 7881 of LNCS* (Springer, Berlin, 2013), pp. 702–718

[19] R.S. Pappu, *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology (2001)

[20] R.S. Pappu, B. Recht, J. Taylor, N. Gershenfeld, Physical one-way functions. *Science* **297**, 2026–2030 (2002)

[21] U. Rührmair, Oblivious transfer based on physical uncloneable functions, in *Trust and Trustworthy Computing, Volume 6101 of LNCS* (Springer, Berlin, 2010), pp. 430–440

[22] U. Rührmair, S. Katzenbeisser, H. Busch. Strong PUFs: models, constructions, and security proofs, in *Towards Hardware-Intrinsic Security* (Springer, Berlin, 2010), pp. 79–96

[23] U. Rührmair, M. van Dijk, PUFs in security protocols: attack models and security evaluations, in *IEEE Symposium on Security and Privacy* (IEEE, Washington, 2013), pp. 286–300

[24] M. van Dijk, U. Rührmair, Physical unclonable functions in cryptographic protocols: security proofs and impossibility results. Cryptology ePrint Archive, Report 2012/228 (2012)