The Limits of Composable Crypto with Transferable Setup Devices

Ioana Boureanu
Akamai Technologies Limited
London
UK
icarlson@akamai.com

Miyako Ohkubo
NICT
Tokyo
Japan
m.ohkubo@nict.go.jp

Serge Vaudenay
EPFL
Lausanne
Switzerland
serge.vaudenay@epfl.ch

ABSTRACT

UC security realized with setup devices imposes that single instances of these setups are used. In most cases, UC-realization relies further on other properties of the setups devices, like tamper-resistance. But what happens in stronger versions of the UC framework, like EUC or JUC, where multiple instances of these setups are allowed? Can we formalise what it is about setups like these which makes them sometimes hinder UC, JUC, EUC realizability?

In this paper, we answer this question. As such, we formally introduce *transferable setups*, which can be viewed as setup devices that do not (publicly) disclose if they have been maliciously passed on. Further, we prove the *general result* that one cannot realize oblivious transfer (OT) or any "interesting" 2-party protocol using transferable setups in the EUC model.

As a by-product, we show that physically unclonable functions (PUFs) themselves are transferable devices, which means that one cannot use PUFs as a global setups; this is interesting because non-transferability is a weaker requirement than locality, which until now was the property informally blamed for UC-impossibility results regarding PUFs as global setups.

If setups are transferable (i.e., they can be passed on from one party to another without explicit disclosure of a malicious transfer), then they will not intrinsically leak if a relay attack takes place. Indeed, we further prove that if relay attacks are possible then oblivious transfer cannot be realized in the JUC model.

Linked to the prevention of relaying, authenticated channels have historically been an essential building stone of the UC model. Related to this, we show how to strengthen some existing protocols UC-realized with PUFs, and render them not only UC-secure but also JUC-secure.

Categories and Subject Descriptors

E.3 [**Data encryption**]: [Public key cryptosystems]

Keywords

universal comparability; relay attacks; physically unclonable functions

ASIA CCS'15, April 14–17, 2015, Singapore..

http://dx.doi.org/10.1145/2714576.2714591.

1. INTRODUCTION

UC-composable [5] cryptographic primitives give us the very sought-after secure guarantees when it comes to the design of cryptographic protocols. And, the skeptical can even go beyond and aim for stronger models, like the JUC (joint-state UC), EUC (externalized UC), GUC (generalized UC) [6], where realizability is even harder. However there is a fundamentally limiting impossibility result by Canetti *et al.* [7]: two-party secure computations are UC-impossible in the UC plain model. To bypass this, the UC models are augmented with helping building blocks, also called (UC) setups.

In this paper, we look at designing composable cryptographic primitives, especially in stronger UC frameworks (like JUC, EUC) and at how relaying comes in the way of that. We link relay attacks to whether they are apparent/visible at the level of the crypto setups used therein, via the notion of *transferable setups*. An instance of such setups is given by the much spoken-of PUFs (physical unclonable functions) [17, 24, 16, 10, 13, 25]. We then look at how the often-assumed authenticated channels have come to defeat relaying within the UC framework.

For completeness, we will first summarize the general settings of UC models with and without setups, underlining certain important details on the communication model and the legitimate use of setups.

A UC overview.

Plain UC. In the UC framework, we consider several participants and an environment. They all run their purported algorithms. Some participants may be corrupted, in which case they behave as the adversary instructs them to. Participants receive inputs from the environment and send their final outputs to the environment. It is assumed that all algorithms are probabilistic polynomial-time algorithms. The UC models comprises two symbolic worlds that should behave similarly, i.e., a real world where the protocol π is run, and an ideal world where the idealized version of the protocol is executed; this idealized version is usually denoted via an ideal (target) functionality $\mathcal{F}_{target}$. Let us detail slightly. In the real world, a set of honest participants is assumed to run their corresponding part of π. In the ideal world, honest participants are "dummies" who just forward inputs/outputs to the ideal functionality $\mathcal{F}_{target}$. We say that this functionality is UC-realized by the protocol π if for every real adversary, there exists an ideal adversary (also called simulator) such that for every environment $\mathcal{Z}$, the real and the ideal worlds are indistinguishable in terms of the output of $\mathcal{Z}$. This definition is accompanied by a composition theorem: every complex experiment using one instance of π as a subroutine emulates a corresponding experiment using (the dummy protocol based on) $\mathcal{F}_{target}$ instead. This substitution can of course be used recursively.

Communication Models in UC. In the original UC papers [5], it was assumed that the channels were secure; we will henceforth refer to that UC-model as the *secure-channel UC*. However, this assumption was consequently [6] dropped; we will henceforth refer to this later UC-model as the *insecure-channel UC*. The latter means that in the case of honest real-world executions, one can imagine man-in-the-middle adversaries mounting attacks. To bypass this issue, most UC-secure constructions assume or intrinsically require authenticated channels between the participants[1]. Participants can identify where an incoming message comes from and where an outgoing message goes to. Note that the UC models also assume that such authenticated channels are not visible to the adversary.

Hybrid UC. Unfortunately, virtually no interesting ideal functionalities can be realized by protocols in the plain UC setting [7]. So, one turns to the hybrid model, where participants can further access a setup functionality $\mathcal{F}_{\texttt{setup}}$, in both the real and the ideal worlds. This defines the $\mathcal{F}_{\texttt{setup}}$-UC-realization. Roughly speaking, a protocol π is said to be realized in the $\mathcal{F}_{\texttt{setup}}$-hybrid model if π is now built using calls to the ideal functionality $\mathcal{F}_{\texttt{setup}}$, in such a way that one instance of the corresponding "physical" setup will only be used in one protocol session of π. One clear disadvantage is that composition only works for *one single* session of π which is using a "physical" instance of the setup given via $\mathcal{F}_{\texttt{setup}}$ at one point in time. Namely, such a $\mathcal{F}_{\texttt{setup}}$ UC-functionality is always so that it models the following: "duplicates" of the (physical) UC setup-device that $\mathcal{F}_{\texttt{setup}}$ is modelling will be issued for every new session of π. Moreover, no other process should access $\mathcal{F}_{\texttt{setup}}$ when π is already using it (i.e., the environment $\mathcal{Z}$ cannot directly access $\mathcal{F}_{\texttt{setup}}$). In UC terms, for π to be UC-composable using $\mathcal{F}_{\texttt{setup}}$, π needs to be "subroutine respecting" [6].

EUC and GUC. From UC, one moves to the EUC formalization [6] by allowing the environment to access the $\mathcal{F}_{\texttt{setup}}$ as well. EUC-realization and EUC-composition are simply UC-realization and UC-composition respectively where the environment is empowered in this way. I.e., EUC still refers to *one* session of a protocol, but –in EUC– all parties have access to a *global* setup, even if it is still only *one (physical) instance of such a setup*. In the GUC framework [6], the environment is even more powerful as he has access to other protocols running simultaneously in the network. Also, in GUC, the target protocol has several simultaneously running sessions, which the GUC-environment can have adaptive (I/O) access to. GUC-realization is defined with respect to this all-powerful environment. A result of equivalence between some type of GUC-composition and some type of EUC-composition exists [6]. For this equivalence to hold, the communication is restricted again, in that it still only allows "subroutine-respecting" protocols. A protocol π is $\mathcal{F}_{\texttt{setup}}$-subroutine respecting if no subroutine of a party in π takes I/O from any non-subroutine of π (other than a single instance of $\mathcal{F}_{\texttt{setup}}$ and the environment). With such a restriction in place, a weaker EUC-realization/EUC-composition is defined, i.e., EUC-realization that applies to $\mathcal{F}_{\texttt{setup}}$-subroutine respecting protocols, which was proven equivalent to the GUC-realization.

JUC. There also exists a model in between the hybrid-UC and EUC, namely JUC [9]. In the definition of the $\mathcal{F}_{\texttt{setup}}$-JUC realization, we consider several groups of participants, each running their own instance of π in the real world, and "playing" with their own instance of $\mathcal{F}_{target}$ in the ideal world. The protocol participants can all jointly access $\mathcal{F}_{\texttt{setup}}$, but the environment cannot. The environment must also provide all inputs at once, i.e., the simultaneous sessions of π cannot be initialized adaptively. This notion makes feasible a complex composition in some limited way. Namely, for each set of sessions of π which can only be initialized with inputs in parallel, we can substitute each session with an instance of $\mathcal{F}_{target}$, and use the same instance of $\mathcal{F}_{\texttt{setup}}$ for this set (and this set only).

UC setups of interest. UC hybrid models, with setups of tamper-evident and tamper-resistant devices, have been employed to UC-realize bit-commitments (COM), oblivious transfers (OT), coin flipping, polling schemes [15, 18, 14, 19, 20, 22, 21, 2, 3], etc. More recently, a new type of tamper-resilient devices has been modelled as setups to create UC-secure OT and COM [4, 23], namely PUFs (physically unclonable functions). Different assumptions [4] have been taken to model these PUFs: e.g., super-polynomial sizes of the input domains, well-spread domains, freshness of usage, honest behavior, etc. In [23], malicious PUFs are also modelled.

Contributions.

I. In this paper, we first look at how UC setups can in fact lead to impossibility results in certain flavors of the UC model. To this end, we formalize *transferable setups*, which can be viewed as devices that do not (publicly) disclose if they have been maliciously passed on. For instance, transferable setups will not publicly output, e.g., the identifiers of the parties who are physically accessing them. We show that OT is not EUC-realizable if transferable setups are to be used. So, transferable setups cannot be global.

We will show that many of the well-known tamper-resilient devices, PUFs included, can be modelled as what we call *transferable setups*. More specifically, we show that those used in in [14, 19, 20, 22, 21, 2, 4, 23] can, while those used in [15, 18, 2] cannot.

II. We also formalize *relay attacks* in the context of using UC setups. Using this formalism, we then show that an OT protocol subject to a relay attack cannot be JUC-secure.

III. We then look at how relaying and PUFs were handled in the latest UC protocols using these setups. We observe that if PUFs are reusable amongst sessions, or that if we do not assume channel authentication within UC, then the OT protocol in [4] is not UC-secure in the $\mathcal{F}_{\texttt{PUF}}$-hybrid model.

We then show a manner to render the OT protocol in [4] even JUC-secure, by strengthening the protocol.

We discuss other related protocols and setups in the literature and obtain similar results with the protocols from [11, 12, 23].

Organization. We define and observe transferable setup in Section 2. The EUC-impossibilities with transferable setups, as well as other related JUC-imposibilities are discussed in Section 3. We give several examples of transferable setups in Section 4. Section 5 shows how to make JUC-secure OT protocol out of from UC-secure OT protocol that uses transferable setup. We conclude in Section 6.

2. UC SETUPS, TRANSFERABILITY AND RELAYING

Intuitive description of transferability.

This section will give a mathematical expression for the following scenario.

Scenario for transferability. Assume that parties A, B and C are engaged in a communication protocol involving some setup-device. Assume further that B and C are malicious, and A thinks

[1]In the UC literature, it is not always clear when authenticated channels are assumed, but most of the UC protocols do implicitly rely on this assumption of the model. There are a few feasibility results without it [1].

she is communicating with B when in fact she is communicating with C. Roughly speaking, the setup used therein is transferable if the fact that A is mis-lead w.r.t. its interlocutor is not prevented by/via the outputs of this setup.

By separating the notions of party/participant, algorithm, and setup being used, this section will formalise the idea behind transferable setups. This is needed in order to formulate and prove a formal (E)UC impossibility result.

Participants, algorithms and transferability.

We distinguish between a physical *participant* (e.g., a computer, a HSM) and the actual *algorithm* that is run by this physical participant (e.g., the client-side SSH installed on a computer). We will now formalise how these participants and algorithms interact together within a protocol execution. E.g., pieces of software can interact together on the same machine, and each piece can independently deliver several outputs: an SSL client algorithm can send some messages directly on port 443 and some other messages to the network configuration daemon of the same physical machine.

Participants can be viewed as identifiable entities/parties taking part in real/idealized execution: e.g., a physical machine with an identifiable IP address. Participants, functionalities, and the environment run *algorithms*. They are connected with each others by some bi-directional channels. For simplicity, we assume only two ends for every channel. We assume that algorithms have different input/output communication ports which can be "plugged" to the port of other algorithms, or to the communication channel of a participant. I.e., in a real-world setting, this would be akin to one algorithm being called by another algorithm and/or used by a device/party. Also, algorithms are not aware how their communication ports are plugged. At the same time, honest participants are correctly "plugged". I.e., the SSL algorithm does not "care" on which port messages are delivered, but a correctly/standardly configured SSL server will have port 443 open and it will be listening on it.

Participants, functionalities, and the environment have an identifier. Algorithms could use the identifier of participants as part of their communication. I.e., the SSL-handshake algorithm will verify a certificate against a given hostname that identifies the server. Indeed, algorithms could receive such identifiers as input, e.g., to instruct them which role they are supposed to "play" and with whom they are supposed to communicate. This is akin to instantiated an object of a generic class with specific parameters, or calling a function with specific arguments, e.g., instantiate the server-side (object) of a program with a server-identifier called "server1" to speak to a client instantiated as "client3". To denote that an algorithm $\mathtt{alg}$ runs with some given identifiers $id_1,\ldots,id_n$, we simply write $alg(id_1,\ldots,id_n)$. These identifiers can also be hard-coded in the algorithm. For instance, we can assume that the algorithm of a UC setup functionality $\mathcal{F}_{\mathtt{setup}}$ is correctly "plugged" and that all identifiers are hard-coded. In this way, $\mathcal{F}_{\mathtt{setup}}$ always knows from which participant a message comes in, and to which participant a message is to be sent to. However, for participants and their algorithms, there is no guarantee that the "plugging" and/or the input-manipulation runs correctly. This is so since inputs can be (maliciously) handled, e.g., if the algorithm is run by a corrupted participant.

Transferability.

In the context of communicating parties, using setups, we will also formalize how an unobservable relay can be achieved, i.e., a relay that no party/setup can be aware of. To do so, we will introduce a series of experiments (i.e., interactive games of machines/participants and algorithms) depicted in Fig. 1 and denoted $\mathbf{exp_1}$, $\mathbf{exp_2}$, $\mathbf{exp}'_1$ and $\mathbf{exp}'_2$.

The transferability/relay-defining experiments in Fig. 1. Assume an arbitrary 2-party protocol defined by two algorithms *sdr* (sender) and *rcv* (receiver). We assume that *sdr* is designed to be run by party P_1 and to communicate with party P_2, and that *rcv* is designed to be run by P_2 and to communicate with P_1 (I.e., this is as if the previously mentioned identifiers, here denoted P_1 and P_2, were hard-coded in *sdr* and *rcv*). The participant denoted with $\mathcal{Z}$ is executing an arbitrary algorithm with two communication ports, no input (port) and one output (port). Both algorithms *sdr* and *rcv* have an input/output port (connected to $\mathcal{Z}$), a setup port (connected to $\mathcal{F}_{\mathtt{setup}}$), and an extra communication port to communicate with their counterpart. This is depicted in the upper, leftmost part of Fig.1, denoted $\mathbf{exp_1}$. In this experiment denoted $\mathbf{exp_1}$, P_1 indeed executes *sdr* (sender), P_2 executes *rcv* (receiver) with correctly connected communication ports. A second experiment is shown on the upper right part of Fig. 1 and is denoted $\mathbf{exp_2}$. In this second experiment exp_2, P_3 is running two concurrent algorithms: the *rcv* (receiver) algorithm and $\mathtt{tr}^{P_1P_2}_{P_2P_3}$ (some "relay-accommodating" translator).

The aim of experiments $\mathbf{exp_1}$ and $\mathbf{exp_2}$ in Fig. 1. This is here is as follows: $\mathbf{exp_2}$ will appear (to the outside world) just like $\mathbf{exp_1}$., only that –in $\mathbf{exp_2}$– the outputs by the *sdr* algorithm are relayed from P_1 to the P_3 party, who is the one running the *rcv* algorithm in $\mathbf{exp_2}$; All of this is done transparently to Z and all other parties involved. For this transparency to be achieved, we introduced the $\mathtt{tr}^{P_1P_2}_{P_2P_3}$: a mechanisation facilitating whatever translation (of identifiers) needs to take place. Intuitively, via $\mathtt{tr}^{P_1P_2}_{P_2P_3}$'s "translation", the *rcv* (receiver) algorithm behaves as if it was run by P_2 communicating with P_1 in the upper part of $\mathbf{exp_2}$'s last block, while in its lower part, *rcv* behaves as if it is run by P_3 communicating with P_2.

We define similar experiments $\mathbf{exp}'_1$ and $\mathbf{exp}'_2$, in a symmetric way.

The first communication port of $\mathcal{Z}$ is connected to P_1 in both experiments. The second port of $\mathcal{Z}$ is connected to (the channel of) P_2 in the first experiment, and to P_3 in the second experiment. The algorithm $\mathtt{tr}^{P_1P_2}_{P_2P_3}$ uses two ports: the upper and the lower one. The input/output port of *rcv* is connected to $\mathcal{Z}$, its communication port is connected to P_2, and its setup port is connected to the upper port of $\mathtt{tr}^{P_1P_2}_{P_2P_3}$. The lower port of $\mathtt{tr}^{P_1P_2}_{P_2P_3}$ is connected (to the channel) to $\mathcal{F}_{\mathtt{setup}}$. In both experiments, we assume that P_1 executes *sdr* with correctly connected communication ports.

Let us now formalize the requirements of the setups, in order for the above (informal) transparent relaying to be the case.

Definition 1. (Transferable Setup) Assume some participants and identifiers P_1,P_2,P_3. Let $\mathcal{F}_{\mathtt{setup}}$ be a setup functionality and $\mathtt{relay}(P_1,P_2,P_3)$ be an algorithm. We say that $\mathcal{F}_{\mathtt{setup}}$ is a $\mathtt{relay}$-transferable setup if there exists some algorithms $\mathtt{tr}=\mathtt{tr}^{P_1P_2}_{P_2P_3}$ and $\mathtt{tr}'=\mathtt{tr}^{P_2P_3}_{P_1P_2}$ such that for any 2-party protocol $\pi=(\mathtt{alg}_1,\mathtt{alg}_2)$ as per the above and any $\mathcal{Z}$, the above experiments in $(\mathtt{exp}_1,\mathtt{exp}_2)$ and $(\mathtt{exp}'_1,\mathtt{exp}'_2)$ produce indistinguishable outcomes in each respective tuple.

In Section 4, we are going to show which "modern" UC setups are transferable and which are not; in anticipation, PUFs [4] are transferable, but tamper-resistant hardware [15] are not.

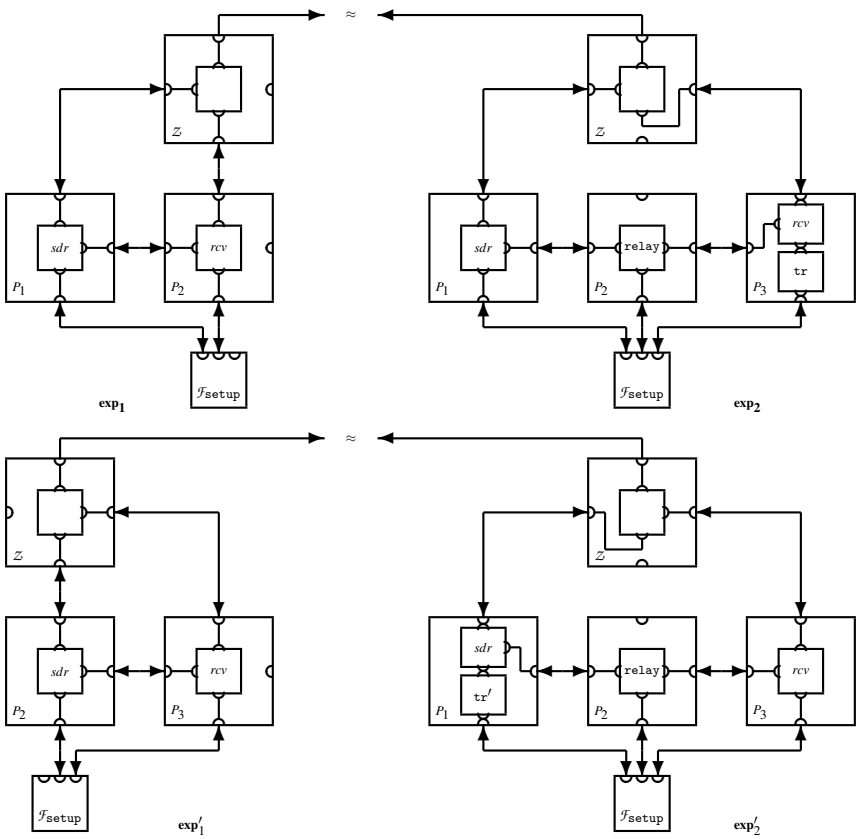

Figure 1: Transferable Setup.

We are now going to formalize the notion of relay attacks, in presence of transferable setups.

Definition 2. (Relay Attack (Fig. 2)) Let $\mathcal{F}_{\texttt{setup}}$ be a `relay`-transferable setup, for $\texttt{relay}(P_1,P_2,P_3)$, P_1, P_2, P_3 as before. We consider a 2-party protocol $\pi = (\texttt{alg}_1(P_1,P_2), \texttt{alg}_2(P_1,P_2))$ based on the functionality $\mathcal{F}_{\texttt{setup}}$ in which *sdr* and *rcv* receive two hard-coded identifiers. We say that π is subject to relay attacks if for any $\mathcal{Z}$, the two following experiments are indistinguishable to any environment: 1. participant P_1 runs $sdr(P_1,P_3)$ with the communication port connected (to the channel) to participant P_3, participant P_3 runs $rcv(P_1,P_3)$ with the communication port connected (to the channel) to participant P_1; 2. participant P_1 runs $sdr(P_1,P_2)$ with the communication port connected (to the channel) to participant P_2, participant P_3 runs $rcv(P_2,P_3)$ with the communication port connected (to the channel) to participant P_2, participant P_2 runs $\texttt{relay}(P_1,P_2,P_3)$, as per the definition of `relay`-transferable setups.

3. IMPOSSIBILITY RESULTS IN UC WITH TRANSFERABLE SETUPS & RELAYING

In this section, we will see how transferability affects EUC-realization.

3.1 Transferable Setups & EUC-Impossibility

THEOREM 1. *If $\mathcal{F}_{\texttt{setup}}$ is a* `relay`*-transferable setup for some symmetric* `relay` *algorithm, then there is no protocol which $\mathcal{F}_{\texttt{setup}}$-EUC-realizes $\mathcal{F}_{OT}$.*

The intuition of the proof.

The idea of the proof is classical. Start from the premise that an arbitrary protocol $\pi = (\texttt{alg}_s, \texttt{alg}_r)$ would $\mathcal{F}_{\texttt{setup}}$-EUC-realize $\mathcal{F}_{OT}$. Then, create a series of runs of this protocol and of the $\mathcal{F}_{OT}$ functionality respectively, such that these runs are pair-wise and/or transitively indistinguishable, yet you arrive at some (distinguishability-based) contradiction. Thus, conclude that no such protocol $\pi = (\texttt{alg}_s, \texttt{alg}_r)$ could have in fact $\mathcal{F}_{\texttt{setup}}$-EUC realized $\mathcal{F}_{OT}$.

To get to this contradiction, we will basically use the existence of the ideal-world simulator $\mathcal{S}im$ (given the initial presumption that $\pi = (\texttt{alg}_s, \texttt{alg}_r)$ would $\mathcal{F}_{\texttt{setup}}$-EUC-realize $\mathcal{F}_{OT}$ say running with one of the inputs as a bit b). Then, this $\mathcal{S}im$ must be able to "extract" the correct bit b to send to $\mathcal{F}_{OT}$. We will then make $\mathcal{S}im$ internally available to a corrupt participant in a new (real-world) run of the protocol. The executions run via such a corrupt party (in the real world) will not be indistinguishably simulatable (in an ideal counterpart). To build the series of indistinguishable experiments we apply transformation steps provided either by the (supposed) $\mathcal{F}_{\texttt{setup}}$-EUC realization of $\mathcal{F}_{OT}$ or by the definition of transferable setups.

Note1.

It seems intuitive that if a setup is transferable, then it loses "globality", hence it cannot serve as an EUC-setup. However, the separation between "globality" and non-transferability is not known. I.e., if a setup is non-global, it does not mean it is transferable,

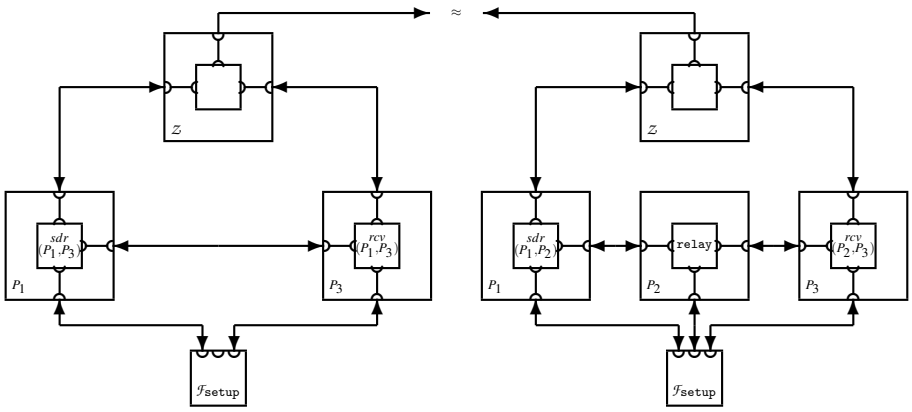

Figure 2: Relay Attack.

nor do we know how many other well-distinguishable properties may lie between transferable and non-global, or between global and non-transferable. Moreover, this is the first time a compact class of global setups, which lead to lack of realizability, has been formally characterised.

Note2.

This proof strategy to be seen below (but not the methodology of symmetric algorithms and relay-transferable setups) is akin to that of splitting adversaries in [8]. Thus, our result above seems to also be an extension from UC-impossibility results to EUC-impossibility results, when the class of relay-transferable setups are used. Along similar lines, the above OT-impossibility result should extend and, like in [8], apply to all same-output non-trivial 2-party protocols.

PROOF. We start with a protocol $\pi = (\texttt{alg}_s, \texttt{alg}_r)$ and assume contrary to our current theorem that π would $\mathcal{F}_{\texttt{setup}}$-EUC realize $\mathcal{F}_{OT}$. If this is the case then for every adversary that will corrupt π (in the real world), there exists a simulator running an algorithm $\mathcal{S}im$ that would mimic it (in the ideal world) in such a way that no (EUC) environment $\mathcal{Z}$ can tell the difference (between the two worlds).

In what follows we consider an environment and three participants P_1, P_2, P_3, together with $\mathcal{F}_{\texttt{setup}}$.

In the first three experiments, the environment is running and algorithm G generating two strings s_0 and s_1 and a bit b, sending s_0 and s_1 to a left participant and b to a right one, getting from the right participant some s, and giving as an outcome 1 if and only if $s = s_b$.

In the first experiment, denoted $\mathbf{exp_1}$ in Fig. 3, the left participant is connected to P_1 who is running $\texttt{alg}_s$ with P_2. The right one is connected to P_2 who is running $\texttt{alg}_r$ with P_1. So, this experiment always outputs 1.

At the higher level, we can simply view $\mathbf{exp_1}$ *as* P_1 *running an honest sender-side algorithm and* P_2 *running an honest receiver-side algorithm within an 1-out-of-2 two-party OT protocol* π.

Using the transferability assumption (marked with "(1)" in Fig. 3), a direct consequence of Def. 1 shows that this is equivalent to a second experiment in which the left participant is still connected to P_1, still running $\texttt{alg}_s$ with P_2, but the right one is connected to P_3, now running $\texttt{alg}_r$ and $\texttt{tr}^{P_1P_2}_{P_2P_3}$ with P_2. The participant P_2 is now running $\texttt{relay}(P_1, P_2, P_3)$. Please see the top two experiments of Fig. 3.

At a higher level, this second experiment $\mathbf{exp_2}$ *shows how mainly the receiver-side is getting corrupted (and, above all, impersonated). I.e., the receiver-side is now being run by some* P_3*, but* P_1 *thinks he is sending messages to some* P_2*; meanwhile, the receiver-side messages produced inside* P_3 *will be received by the man-in-the-middle* P_2 *and then fed by* P_2 *back to* P_1.

We can define a new environment putting the old one and P_3 together (see the doted line on the second diagram in Fig. 3, $\mathbf{exp_2}$ and $\mathbf{exp_3}$).

Thus, in $\mathbf{exp_2}$*, we simply have an EUC environment (within* $\mathbf{exp_2}$*'s dotted line) which runs the receiver-side inside himself, and we have an adversary in the person of* P_2 *who simply relays between this environment and an honest sender-side algorithm run inside* P_1*. This EUC real-world execution now becomes fundamental to the proof (because we presumed it would be simulatable).*

Let us apply the EUC-realizability assumption (marked with "(2)" in Fig. 3). Then, this second, real-life EUC experiment $\mathbf{exp_2}$ is equivalent to a third, ideal-life EUC experiment $\mathbf{exp_3}$. In $\mathbf{exp_3}$, P_1 is a dummy sender forwarding messages to $\mathcal{F}_{\texttt{OT}}$, P_2 is running a simulator exchanging messages with $\mathcal{F}_{OT}$, and the environment is as before, i.e., the (old, G-based) environment and P_3 are still being run as in $\mathbf{exp_2}$.

According to the assumption of EUC-realizability assumption, $\mathcal{S}im$ *is able to produce a bit* b' *to send to* $\mathcal{F}_{\texttt{OT}}$*,* $\mathcal{F}_{\texttt{OT}}$ *gives him back a state* $s_{b'}$ *such that, with a high probability,* $b' = b$ *and then* $s_{b'}$ *is equal to* s_b *(i.e., with s).*

In the next experiments, the environment is still generating s_0, s_1, b, sending b to a right participant (who is still connected to P_3 at this time), waiting for a bit b' from a left participant (that we connect to P_2 for the moment) and sending him back $s_{b'}$. The environment expects some s or s_b from the right participant and outputs 1 if and only if $s_{b'} = s_b$.

By redirecting the simulator's port to the environment (marked with "(3)" in Fig. 3), we obtain an experiment which is obviously equivalent to the previous one. We denote by G' the algorithm of the environment.

The simple step from $\mathbf{exp_3}$ *to* $\mathbf{exp_4}$ *was first to include a simulated version of* $\mathcal{F}_{OT}$ *inside the run done by* $\mathcal{Z}$*. Then, (algorithm of) the dummy* P_1 *was also silently "pushed inside"* $\mathcal{Z}$*. This is why we renamed his internal algorithm from G to* G'*. This is possible without distinguishability repercussions since* $\mathcal{F}_{OT}$ *does not state who queries what, i.e., it makes no public announcements that would hinter its simulated "inclusion" within* $\mathcal{Z}$*, and* P_1 *did practically nothing.*

We now apply the transferability with (P_3, P_2, P_1) (marked with "(4)" in Fig. 4). We obtain that P_2 runs $\texttt{relay}(P_3, P_2, P_1)$ with the

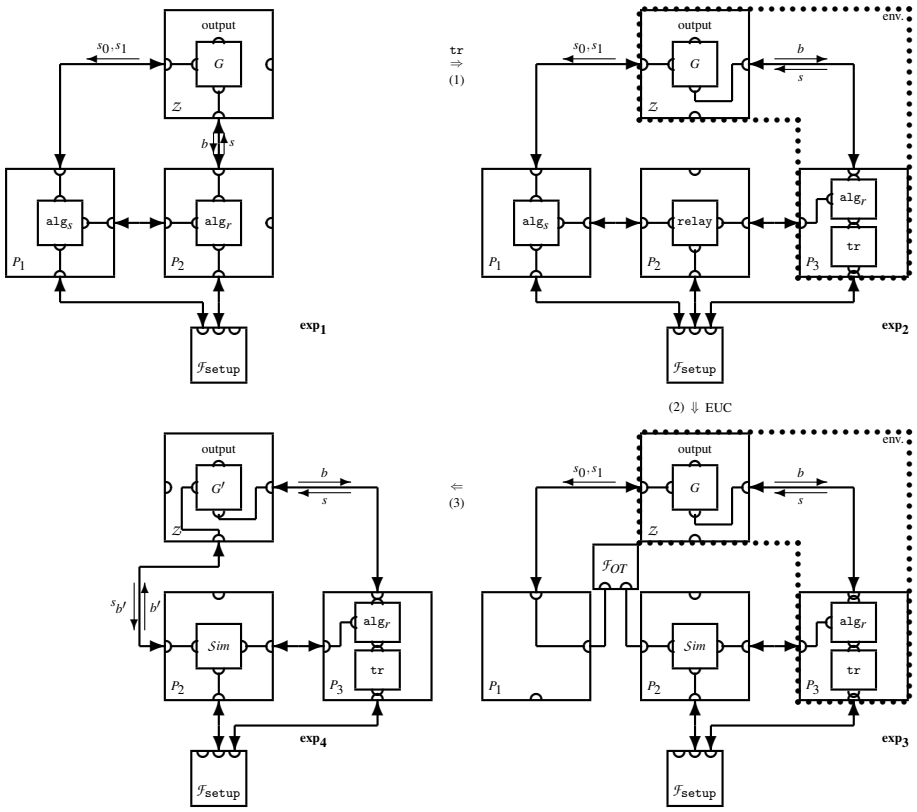

Figure 3: Reductions (1) to (3).

left port connected to P_3 and the right port connected to P_1. But, due to the symmetry, this is equivalent to $\texttt{relay}(P_1, P_2, P_3)$ with the left port connected to P_1 and the right port connected to P_3. Now, P_1 is running the simulator connected to a translator.

We apply again (marked with "(5)" in Fig. 4) the transferability (backwards) to move back to the situation where P_2 is running $\texttt{alg}_r$ correctly connected, and P_3 unused.

We have now arrived to some clearer, new real-world (UC) executions depicted in $\mathbf{exp_6}$. *We can see it in the following way.* $\mathcal{Z}$ *is running an OT session called* π_1 *(on* s_0, s_1, b*) with a receiver in the person of* P_2. *Out of this,* $\mathcal{Z}$ *gets* $s = s_b$ *from* P_2. *And,* P_1 *interacts as a sender with the honest receiver* P_2 *according to the OT protocol* π*, in a session called* π_3. *At the same time,* $\mathcal{Z}$ *is running another OT session called* π_2 *(on* b'*) with a (corrupt) party in the person of* P_1. P_1 *is acting as an OT-receiver for* $\mathcal{Z}$ *inside* π_2. *Out of this,* $\mathcal{Z}$ *should send* $s_{b'}$ *to* P_1*, where* b' *is provided by* P_1 *via* $\mathcal{S}im$. *Since* $\mathcal{S}im$ *should be able to extract* b *out of the run in* π_3*, then with an overwhelming probability,* b' *should be equal to* b *and then* $s_{b'}$ *should be equal to* s_b *with an overwhelming probability. So, in these real-world executions the environment with output* 1*, with an overwhelming probability.*

Finally, we apply (marked with "(6)" in Fig. 4) the EUC assumption again (actually, the UC assumption is enough in this case, since the environment is not using $\mathcal{F}_{\texttt{setup}}$). Due to our reductions, in the ideal-world execution depicted in $\mathbf{exp_7}$, the output must also be 1 (except with negligible probability).

In $\mathbf{exp_7}$, in the session π_1, we obtain that P_2 is honestly receiving a bit b and giving it to $\mathcal{F}_{\texttt{OT}}$, P_1 sends some s'_0, s'_1 to $\mathcal{F}_{\texttt{OT}}$, $\mathcal{F}_{\texttt{OT}}$ sends s'_b back to P_2, P_2 then sends this s'_b as s back to $\mathcal{Z}$. In the session π_2, P_1 is running a simulator $\mathcal{S}im'$ to send some b'' to the environment $\mathcal{Z}$, to get $s_{b''}$ in return from $\mathcal{Z}$.

The question is whether $s_{b''}$ is still equal to s (like $s_{b'}$ was in $\mathbf{exp_6}$). Clearly, if $b = b''$, then this holds with probability 1 (i.e., then $s_{b''} = s_b \equiv s$). But if $b \neq b''$ (which happens with probability $\frac{1}{2}$), this can only work if the simulator $\mathcal{S}im'$ can correctly guess s'_b. Given that there is at least one bit of information in s'_b, the outcome of the experiment is 1 with probability lower than $\frac{3}{4}$, which contradicts the initial assumption (i.e., $\mathbf{exp_7}$ cannot perfectly emulate $\mathbf{exp_6}$, which goes back to $\mathbf{exp_4}$ not perfectly emulating $\mathbf{exp_3}$, which refutes the EUC-realizability in the first place). □

Other EUC impossibilities with transferable setups.

We believe that this can be taken further, to some other types of protocols (like the impossibility in [8] was). For instance, we can easily show with a similar proof the same result for commitment.[2] A natural question, given our results, is whether we can EUC-realize key exchange $\mathcal{F}_{KE}$ (since practice likes to do key exchange with PUFs), based on transferable setups. The answer is that we cannot.

THEOREM 2. *If* $\mathcal{F}_{\texttt{setup}}$ *is a* relay*-transferable setup for some symmetric* relay *algorithm, then there is no protocol which* $\mathcal{F}_{\texttt{setup}}$*-EUC-realizes* $\mathcal{F}_{COM}$ *or* $\mathcal{F}_{KE}$.

[2]Care is to be taken at the fact that commitment-transactions are done in two phases.

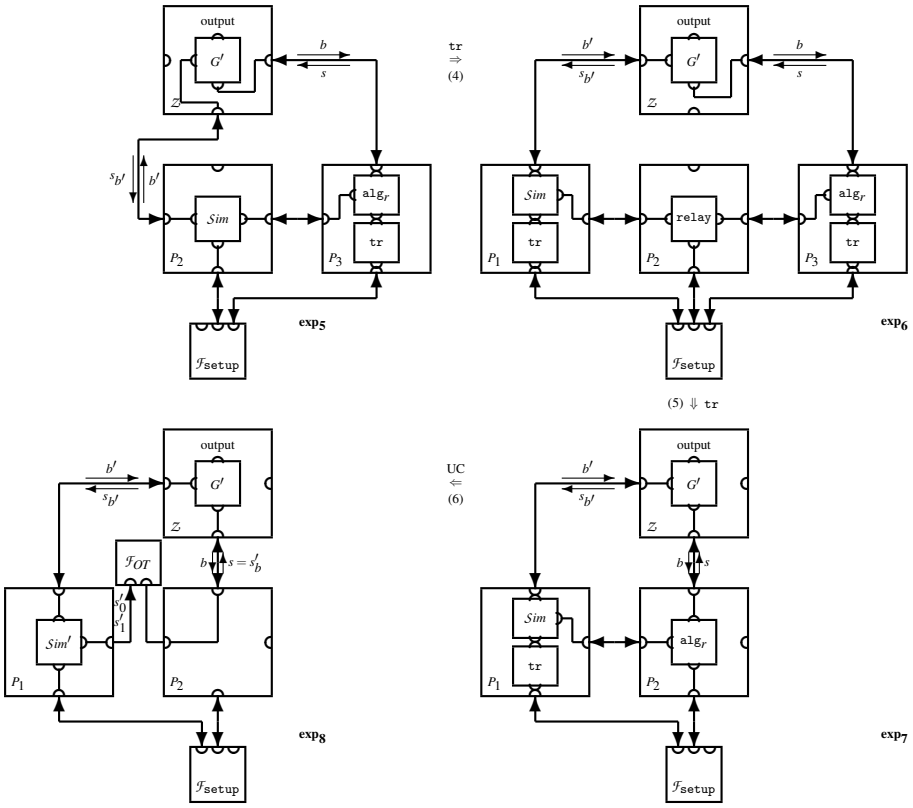

Figure 4: Reductions (4) to (6).

The intuition of the proof.

If we have a protocol EUC-realizing $\mathcal{F}_{KE}$, we can consider an experiment with Alice and Bob, where Alice is malicious and just forwards everything to the environment. Due to transferability, this is the same as Alice and Bob running the protocol honestly. Due to EUC-security, there must be an ideal adversary who as follows. First, this ideal adversary makes Alice and dummy Bob run $\mathcal{F}_{\mathrm{KE}}$ and get the exchanged key K. Then, it can simulate Bob's protocol to the environment running Alice's protocol in such a way that the protocol terminate with the key K. Now, take the algorithm of this ideal adversary and consider another experiment where Bob is malicious and running this algorithm. This will make a honest Alice terminate the KE protocol on a key K chosen by the environment and given to Bob. But now, it is clear that the ideal world cannot force $\mathcal{F}_{\mathrm{KE}}$ to make Alice receive some K chosen by the environment. So, there is a contradiction with EUC security.

3.2 Relaying & JUC-Impossibility

We now move to JUC-security and show that it is hindered by relay attacks, formalized in Def. 2.

THEOREM 3. *If $\pi = (\mathtt{alg}_s, \mathtt{alg}_r)$ is an OT protocol based on $\mathcal{F}_{\mathtt{setup}}$ and π is subject to relay attacks, then π does not $\mathcal{F}_{\mathtt{setup}}$-JUC-realizes $\mathcal{F}_{OT}$.*

PROOF. Consider a JUC (real-world) experiment of two sessions of the OT protocol in $\mathcal{F}_{\mathtt{setup}}$ hybrid model, with a sender S_1 and a receiver R_1, and a sender S_2 and receiver R_2, in the two respective sessions. The environment $\mathcal{Z}$ chooses $s_0^{(i)}$ and $s_1^{(i)}$ and sends them to the honest participant S_i, where $i = 1,2$. $\mathcal{Z}$ also selects a bit $b^{(i)}$ and sends it to the honest participant R_i, where $i = 1,2$. The outcome s coming for R_2 will be given back to the environment. The environment outputs 1 if and only if $s = s_b^{(1)}$. We consider the following execution of this real-world experiment where R_1 and S_2 are corrupted. In first session π_1 of π, using a setup function $\mathcal{F}_{\mathtt{setup}}$, S_1 runs with $(s_0^{(1)}, s_1^{(1)})$ with the adversary $\mathcal{A}$ who does the following: $\mathcal{A}$ relays the protocol sessions from S_1 to R_2 (as well as the physical setup instance, as per the transferability assumption). The above real-world execution is illustrated in Fig. 5.

Due to the transferability assumption, this is indistinguishable from S_1 and R_2 running the protocol together. So, in this real-world execution, the environment outputs 1 with probability 1 (as he would do if S_1 and R_2 running the protocol together).

Let us now describe the corresponding ideal-world execution. The ideal adversary $\mathcal{S}im$ (namely $\mathcal{S}im(R_1)$) learns $s_\beta^{(1)}$, but has no information about $s_{\bar{\beta}}$. Equally, $\mathcal{S}im$ (namely $\mathcal{S}im(S_2)$) knows nothing about $b^{(2)}$. So, with the output by the dummy R_2, in this ideal-world execution, the environment outputs 1 with a probability close to $\frac{1}{2}$.

Thus, we exemplified a JUC real-world execution of an arbitrary 1-out-of-2 2-party OT that is distinguishable from its ideal-world counterpart. Thus, OT is not JUC-secure in $\mathcal{F}_{\mathtt{setup}}$-hybrid model, when $\mathcal{F}_{\mathtt{setup}}$ is transferable and relaying is possible. □

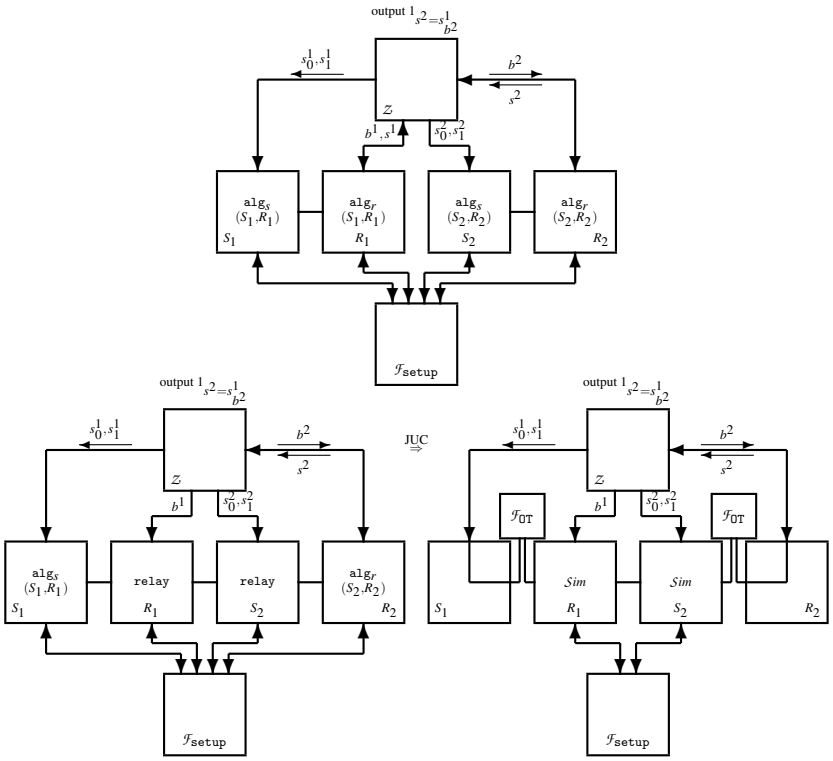

Figure 5: Relay Attack against JUC Security.

4. ACTUAL TRANSFERABLE SETUP DEVICES

Examples of transferable setups.

We will show prove/discuss the transferability of certain known UC setup devices. One of these is $\mathcal{F}_{\mathtt{PUF}}$, modelling PUFs, as they were formalised in [4].

THEOREM 4. *$\mathcal{F}_{\mathtt{PUF}}$ defined in [4] is* `relay`*-transferable setup.*

PROOF. Let us consider a 2-party protocol π run using PUFs (i.e., $\mathcal{F}_{\mathtt{PUF}}$). We presume some hard-coded identifiers P_1, P_2, and two algorithms *sdr*, *rcv* respectively used by parties with these identifiers. Let the run of this protocol by the parties above denote an experiment called $\mathbf{exp_1}$ (as in the left hand side of Fig. 1). Let P_3 be another hard-coded identifier, to be used by a third party. We will define two more algorithms, `relay` and `tr` and we will prove that — with $P_1, P_2, P_2, \mathtt{alg}_1, \mathtt{alg}_2, \mathcal{F}_{\mathtt{PUF}}$ and `relay` and `tr` — we can build an execution of π, in an experiment $\mathbf{exp_2}$, indistinguishable from $\mathbf{exp_1}$. The same holds for $\mathbf{exp}'_1$ and $\mathbf{exp}'_2$. Consequently, $\mathcal{F}_{\mathtt{PUF}}$ is `relay`-transferable setup as per Def. 1.

A `relay` *algorithm:* This is relaying messages in an intuitive way: whatever comes from the left port is forwarded to the right port and vice versa. Messages from $\mathcal{F}_{\mathtt{PUF}}$ are ignored except the notifications of a received handover: if `relay` receives $(\mathtt{handover}_{\mathtt{PUF}}, \mathtt{sid}, P_i)$ for $i = 1, 3$, it sends back $m = (\mathtt{handover}_{\mathtt{PUF}}, \mathtt{sid}, P_j)$ with j such that $\{i, j\} = \{1, 3\}$ to forward the PUF to the other participant. Since we consider 2-party protocols, whenever `relay` receives a message from $\mathcal{F}_{\mathtt{PUF}}$, it could only have originated from *sdr* or *rcv*. So, other handed over PUFs are just ignored by `relay`.

A $\mathtt{tr}^{P_1P_2}_{P'_1P'_2}$ *algorithm:* This translates identifiers in an intuitive way: if it receives $m = (\mathtt{init}_{\mathtt{PUF}}, \mathtt{sid}, P_j)$ from the top port, then $\mathtt{tr}^{P_1P_2}_{P'_1P'_2}$ sends $(\mathtt{init}_{\mathtt{PUF}}, \mathtt{sid}, P'_j)$ to the bottom one. The response of form $(\mathtt{initialized}_{\mathtt{PUF}}, \mathtt{sid})$ from the bottom port is forwarded to the top one. If $m = (\mathtt{eval}_{\mathtt{PUF}}, P_j, c)$ comes from the top, then $\mathtt{tr}^{P_1P_2}_{P'_1P'_2}$ sends $(\mathtt{init}_{\mathtt{PUF}}, P'_j, c)$ to the bottom. Then, the response of form $(\mathtt{eval}'\mathtt{ed}_{\mathtt{PUF}}, \mathtt{sid}, c, r)$ from the bottom is forwarded to the top. Finally, after it receives the message $m = (\mathtt{handover}_{\mathtt{PUF}}, \mathtt{sid}, P_j, P_i)$ from the top, $(\mathtt{handover}_{\mathtt{PUF}}, \mathtt{sid}, P'_j, P'_i)$ is sent by $\mathtt{tr}^{P_1P_2}_{P'_1P'_2}$ to the bottom, and $m = (\mathtt{handover}_{\mathtt{PUF}}, \mathtt{sid}, P'_i)$ coming from the bottom makes $\mathtt{tr}^{P_1P_2}_{P'_1P'_2}$ send the message $(\mathtt{handover}_{\mathtt{PUF}}, \mathtt{sid}, P_i)$ to the top.

For $P'_1 = P_3$ and $P'_2 = P_3$, we show by induction that any step in $\mathbf{exp_1}$ matches one (or several) steps in $\mathbf{exp_2}$. So, the simulation is perfect. □

Let us now consider other setup assumptions. Tamper-proof hardware is used in [15] and is described using $\mathcal{F}_{wrap}$. According to the definition, $\mathcal{F}_{wrap}$ is not a transferable setup since it is assumed that the creator of the tamper-proof device intends to have this device be used only by one user, whose id is recorded in the device, and the thus-wise recorded id cannot be changed. Tamper-resistant devices $\mathcal{F}_{TA}$ are building blocks in a model called the trusted agent model [18]; like $\mathcal{F}_{wrap}$, $\mathcal{F}_{TA}$ is not a transferable setup. This is because there is no opportunity for handing it over as per the device's definition in [18]. Tamper-evident envelopes presented in [21] can be considered relay-transferable setups, because such an envelope includes only the sender's id (and a value), and the envelope sent to a user can be diverted to another user. However, purported tamper-evident envelopes $\mathcal{F}^{DE}_{Oneseal}$ (a variant of the above envelopes), proposed by [2], are not transferable. It is because such an envelope is aimed only for an intended recipient and cannot be diverted to another recipient.

5. COMPOSABILITY & TRANSFERABLE DEVICES – FURTHER ASPECTS

We will now report on further issues with the UC-security of recent protocols that were particularly based on PUFs, which we showed above to be transferable devices. (These discussions could extend to other transferable devices.)

Then we will propose some solutions to bypass the issues highlighted in the first part of the section.

To set the basis of the discussions, in Fig. 6, we first recall the *PUFOT* protocol from [4], an OT protocol based on PUFs.[3] We simplified the description of the protocols as follows: 1). we are assuming non-noisy PUFs (which allows to get rid of the fuzzy extraction); 2). we removed the multiple OT capabilities. It is fairly easy to check that our observations extend to the original PUFOT protocol. In this way, we treat PUFs as access-controlled random oracles. (Malicious PUFs could later be introduced as arbitrary oracles.) Basically, the receiver initializes a PUF, gets one random

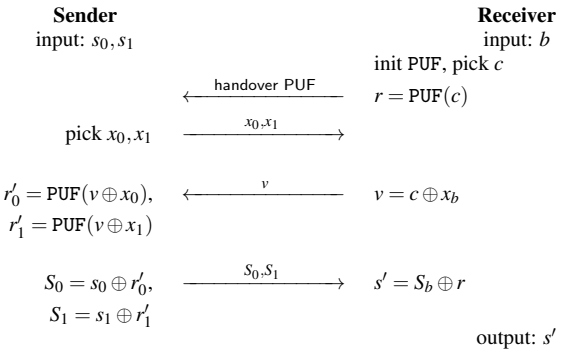

Figure 6: The (Simplified) PUFOT Protocol.

$(c, \mathtt{PUF}(c))$ pair, and hands over the PUF to the sender.[4] The sender selects two values x_0 and x_1 and sends them to the receiver. The receiver responds by $v = c \oplus x_b$ which perfectly hides b. The sender computes $r'_0 = \mathtt{PUF}(v \oplus x_0)$ and $r'_1 = \mathtt{PUF}(v \oplus x_1)$ as one of these must be the r known by the receiver. The receiver cannot know both since he cannot predict $x_0 \oplus x_1$. Finally, the sender used r'_0 and r'_1 to encrypt s_0 and s_1 respectively. The receiver is then able to decrypt s_b.

5.1 Two Issues in UC-security: Reusing PUFs and Authenticating Channels

Honest reuse of PUFs in PUFOT.

We now observe that the PUF instances/devices cannot be handed over again after one session of PUFOT. Indeed, if a PUF-instance is used in one session of PUFOT and then handed-over by the sender, then the first session is insecure. To see that, let us assume that a PUF instance can be reused after a session of the PUFOT with sender S and receiver R. As we know, when the session finishes, the sender S will hold the PUF. We assume that S hands over the PUF after the session. During this hand-over, the adversary that overheard the session is able to query the PUF on the value $c' = c \oplus x_0 \oplus x_1$ and learn r'_{1-b}. Then, he can decrypt s_{1-b} and break the OT requirements.

[3]Note that [26, 27] have shown a weakness of this protocol.

[4]On Fig. 6, "handover PUF" is a shorthand to mean that the receiver sends a hand-over command to $\mathcal{F}_{\mathtt{PUF}}$ and that the sender waits for being granted access to the PUF.

This observation states a version of the PAM-impossibility results in [28]. Our formulation/proof above is much simpler, and in fact natural.

Consequently, PUFs must be destroyed after a protocol is completed. Note that the PUFOT protocol allows multiple OTs with the same sender and receiver. But since the PUF is not be handed over, this imposes that the OT sessions are from the same sender to the same receiver. After all sessions completed, the PUF cannot be used anymore and must be destroyed. So, attacks based on honest reuse of PUFs can be avoided.

Malicious reuse of PUFs in PUFOT.

The PUFOT protocol is subject to relay attacks as per Def. 2, which could be seen as a malicious reuse of a PUF. So, due to Th. 3, it does not $\mathcal{F}_{\mathtt{PUF}}$-JUC realize $\mathcal{F}_{OT}$.

Actually, there is no practical way to prevent an adversary from reusing a PUF. This entails that the PUC model [4] does not compose in the classical sense, even to the weak level of JUC.

Malleability in PUFOT and its successors.

The PUFOT protocol has another shortcoming. Namely, it makes the oblivious transfer malleable: when messages are not authenticated, a man-in-the-middle in the OT between two honest participants can transform the $sdr(s_0, s_1)$ into $sdr(s_1, s_0)$ by exchanging S_0 and S_1 and by replacing v by $v \oplus x_0 \oplus x_1$: $rcv(b)$ will obtain s_{1-b} instead of s_b. This only requires the capability to corrupt the communication between the two participants. So, this property shall be a concern for the practicality of PUFOT.

When using PUFOT to implement a bit commitment [4], this translates into an man-in-the-middle attack to transform the commitment to b to a commitment to $1 - b$.

A similar problem occurs in the [23] protocol. (See Appendix A for details.) The same goes for the commitment protocol of [11, 12]. Although these protocols are UC-secure, this observation shows that it is not the case without the authenticated channel assumption. This is clearly stated in [4, 11, 12], but it is not mentioned in [23].

There could be another subtle form of malleability which could be a concern in practice: assume a man-in-the-middle adversary who could substitute the PUF which is handed over by R to S. Concretely, the adversary substitutes the PUF in a way that he runs independently a receiver protocol with S using a random $\bar{b}$, finishing by giving $s_{\bar{b}}$, and a sender protocol with R, using two strings $\bar{s}_0$ and $\bar{s}_1$ such that $\bar{s}_{\bar{b}} = f(s_{\bar{b}})$ and the other string $s_{1-\bar{b}}$ is random. Finally, R receives $f(s_b)$ if $b = \bar{b}$ and something random otherwise. This is still a malleability problem. It applies to all protocols using PUFs. Fortunately, the PUF formalisation does not allow substituting the PUF during the hand-over process. Indeed, when P_i hands over a PUF to P_j, upon reception, $\mathcal{F}_{\mathtt{PUF}}$ tells P_j that the PUF was sent by P_i, and $\mathcal{F}_{\mathtt{PUF}}$ also delivers a receipt to P_i saying that the PUF was well received by P_j. So, the PUF which is sent by R must be the one received by S, even in a protocol under attack. So, there is an *intrinsic authentication* of PUFs in the handover process of $\mathcal{F}_{\mathtt{PUF}}$ and this can be used to avoid the authenticated channel assumption. Even assuming a weaker PUF formalisation where the PUF delivery could be corrupted, the authors of [23] proposed a way to avoid this attack based on the unpredictability of PUFs.

5.2 Strengthening PUFOT (still using authenticated channels)

We will avoid relay attacks by strengthening the PUFOT protocol. Our *Strengthened PUF Oblivious Transfer* protocol (SPOT) is depicted in Fig. 7. Essentially, we bind the PUF query to the sender

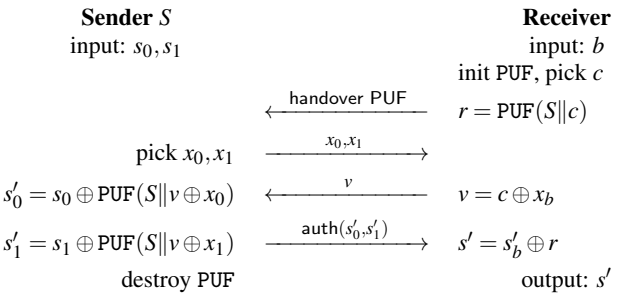

Figure 7: The (Simplified) SPOT Protocol.

S by adding S in the query. As one can see, this is a minor change, and it brings more security.

THEOREM 5. *Assuming authenticated channels, the SPOT protocol on Fig. 7 $\mathcal{F}_{\mathtt{PUF}}$-JUC-realizes $\mathcal{F}_{\mathtt{OT}}$.*

PROOF SKETCH. Following the JUC setting, we have several participants S_j (corrupted or not) receiving (s_0^j, s_1^j) from $\mathcal{Z}$ at the beginning and running the sender protocol with several participants R_i (corrupted or not) receiving b^i from $\mathcal{Z}$ at the beginning and running the receiver protocol, respectively. At the end, they send there output s^i to $\mathcal{Z}$ who produces the outcome of the experiment. Of course, $\mathcal{Z}$ may interact with corrupted participants in between sending the inputs and receiving the outputs.

We consider an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$ in a JUC setting, and we want to construct an ideal-world adversary $\mathcal{S}im$. This $\mathcal{S}im$ runs a simulation $\bar{\mathcal{A}}$ of $\mathcal{A}$ to simulate the corrupted participants as in the real world. $\bar{\mathcal{A}}$ will also have to interact with $\mathcal{F}_{\mathtt{PUF}}$. Since neither any honest participant nor the environment is assumed to interact with $\mathcal{F}_{\mathtt{PUF}}$, $\mathcal{S}im$ needs not hand over any PUF. He just uses $\mathcal{F}_{\mathtt{PUF}}$ for simulating the interaction with real PUFs. The hand-over processing is simulated. $\mathcal{S}im$ also has to interact with $\mathcal{F}_{\mathtt{OT}}$ on behalf of corrupted participants and to feed $\bar{\mathcal{A}}$ with communications between honest participants (since we assume the authenticated channel UC framework).

To run $\bar{\mathcal{A}}$, $\mathcal{S}im$ needs to simulate some honest participants $\bar{S}_j$ and $\bar{R}_i$ while the ideal participants S_j and R_i (with inputs unknown to $\mathcal{S}im$) are interacting with $\mathcal{F}_{\mathtt{OT}}$. In what follows, we consider

$$Views_{real}(\mathcal{Z}, \mathcal{A}, S, R) =$$

$$\langle View_{real}(\mathcal{Z}), View_{real}(\mathcal{A}), (View_{real}(S_j))_j, (View_{real}(R_i))_i \rangle$$

the collection of the views of $\mathcal{Z}$, $\mathcal{A}$, all honest S_j and R_i in the real world, and

$$Views_{ideal}(\mathcal{Z}, \mathcal{S}im) = \langle View_{ideal}(\mathcal{Z}), View_{ideal}(\mathcal{S}im) \rangle$$

the collection of the views of $\mathcal{Z}$ and $\mathcal{S}im$ in the ideal world. (Since the ideal S_j and R_i are just forwarding inputs from $\mathcal{Z}$, we can ignore their views which are redundant with the one of $\mathcal{Z}$.) We denote by $V_{\mathcal{Z}}$ the first view in a collection V. So, $Views_{real}(\mathcal{Z}, \mathcal{A}, S, R)_{\mathcal{Z}} = View_{real}(\mathcal{Z})$. Below, we construct $Views_{sim}$ as a function $Views_{sim} = f(Views_{ideal}(\mathcal{Z}, \mathcal{S}im))$ such that $Views_{real}(\mathcal{Z}, \mathcal{A}, S, R)$ and $Views_{sim}$ are indistinguishable and that $View_{ideal}(\mathcal{Z}) = (Views_{sim})_{\mathcal{Z}}$. We deduce from this construction that $View_{ideal}(\mathcal{Z})$ is indistinguishable from $View_{real}(\mathcal{Z})$, which is what we had to prove.

To prove indistinguishability, we just have to say that the inputs and responses from $\mathcal{F}_{\mathtt{PUF}}$ have a correct distribution and that running the real world on the same input produces the same views.

To define $Views_{sim} = f(Views_{ideal}(\mathcal{Z}, \mathcal{S}im))$, we just concatenate $View_{ideal}(\mathcal{Z})$, $View_{ideal}(\bar{\mathcal{A}})$, and the (reconstructed) views of each honest $\bar{S}_j$ and $\bar{R}_i$. All is computed from $View_{ideal}(\mathcal{S}im)$ by running $\bar{\mathcal{A}}$ and the simulator for the honest $\bar{S}_j$ and $\bar{R}_i$, except for $View_{ideal}(\mathcal{Z})$ which is already part of $Views_{ideal}(\mathcal{Z}, \mathcal{S}im)$. As we will see, $\bar{S}_j$ and $\bar{R}_i$ are simulated but we can reconstruct a meaningful view for S_j and R_i afterwards.

To make the simulation work, $\mathcal{S}im$ has to simulate the honest participants $\bar{S}_j$ and $\bar{R}_i$ interacting with $\bar{\mathcal{A}}$. One problem is that $\mathcal{S}im$ is not always aware of which inputs S_j and R_i receive from $\mathcal{Z}$. Another issue for $\mathcal{S}im$ is to simulate the interaction with $\mathcal{F}_{\mathtt{OT}}$ on behalf of corrupted participants interacting with honest ones. The interaction between two malicious participants is fully taken care of by $\bar{\mathcal{A}}$.

Incidentally, we notice that $\mathcal{S}im$ is aware of all interaction (queries and answers) with all PUFs.

To define $\mathcal{S}im$ and f, we maintain two sets Q and Q' of $(\mathtt{sid}_{\mathtt{PUF}}, q)$ queries. The set Q includes actual queries to the PUF while the set Q' contains undecided queries, which could have been made by some honest receiver. These two sets are originally empty. Whenever $\bar{\mathcal{A}}$ is making a query to PUF, the corresponding pair is added in Q. During the simulation for a honest receiver $\bar{R}_i$, some undetermined queries will be added in Q' as detailed below.

To simulate a honest $\bar{R}_i$, $\mathcal{S}im$ simply takes a fresh PUF, waits for some x_0, x_1, and selects a random string v. Then, $\mathcal{S}im$ queries both $\mathtt{PUF}(S\|v \oplus x_0)$ and $\mathtt{PUF}(S\|v \oplus x_1)$ and inserts both queries in Q'. To reconstruct the view of R_i in $Views_{sim} = f(Views_{ideal}(\mathcal{Z}, \mathcal{S}im))$, we take $c = v \oplus x_b$ as the random coins of R_i (where b is computed from $View_{ideal}(\mathcal{Z})$) and the response from PUF to the query corresponding to b. Clearly, the coins have a correct distribution and the simulation is perfect. If S is corrupted, $\mathcal{S}im$ further has to submit some s_0, s_1 to $\mathcal{F}_{\mathtt{OT}}$ for R_i. After receiving s'_0, s'_1, $\mathcal{S}im$ can get both s_0 and s_1 and submit them to $\mathcal{F}_{\mathtt{OT}}$, and the ideal R_i will get s_b like in the real world.

To simulate a honest $\bar{S}_j$, we pick x_0 and x_1 normally and wait for v. The corresponding PUF is now handed by a honest sender and not supposed to be queried any more by $\bar{\mathcal{A}}$. We can check if the queries $\mathtt{PUF}(S\|v \oplus x_0)$ and $\mathtt{PUF}(S\|v \oplus x_1)$ exist in Q or Q'. Then, there are two cases to analyse.

In the first case, there is at most one of the two queries in $Q \cup Q'$ and we can set b to the value such that $\mathtt{PUF}(S\|v \oplus x_b)$ is the existing query. If there is none, we just set b to a random value. So, $\mathcal{S}im$ can make the corrupted R_i send b to $\mathcal{F}_{\mathtt{OT}}$ to deduce s_b. Then, we construct s'_b based on s_b and set s'_{1-b} to a random string. Whatever $\bar{A}$ is doing is equivalent to running the real world with a correct s_b and some random s_{1-b}. But $\bar{A}$ cannot have any access to the correct value of s_{1-b} due to the non-existent query to the PUF. So, the executions produce equivalent results. In the reconstructed view of S_j, the non-queried PUF value is set to $s'_{1-b} \oplus s_{1-b}$ where s_{1-b} is taken from $View_{ideal}(\mathcal{Z})$. Since the PUF output is random and the PUF is no longer used, this reconstructed view perfectly simulates S_j running with input s_0 and s_1.

In the second case, both possible queries are in $Q \cup Q'$. Note that the PUF handed by S cannot have been queried after $\bar{S}_j$ selected x_0 and x_1. Since the probability that $x_0 \oplus x_1$ matches the XOR between two existing queries is negligible, both queries cannot be in $Q \cup Q'$ at this time and none can later be added in Q: it can only be the case that some queries were added in Q' afterwards. If only one is added in Q' and the other is already existing, then it means that a simulation for a honest receiver managed to select some random vector hitting an existing query. This occurs with negligible probability. So, the two values must have been added

in Q'. Clearly, this comes from the above simulation of a honest receiver $\bar{R}_i$ who is undecided about b and who received some $\bar{x}_0$ and $\bar{x}_1$. The queries have form $S\|v \oplus \bar{x}_b$. Due to this structure, it must be the case that these two participants are really interacting with each other and have seen some protocol messages $x_0, x_1, \bar{v}$ (for the sender) and $\bar{x}_0, \bar{x}_1, v$ (for the receiver) leading to the same pairs $\{\bar{v} \oplus x_0, \bar{v} \oplus x_1\}$ and $\{v \oplus \bar{x}_0 = v \oplus \bar{x}_1\}$. So, $x_0 \oplus x_1 = \bar{x}_0 \oplus \bar{x}_1$. Since $\mathtt{sid}_{\mathtt{PUF}}$ is the same, they also have seen the same PUF. By looking at the receiver protocol, we can see that the adversary cannot know any $v \oplus \bar{x}_i$ before he got v from the receiver. This occurs after giving $\bar{x}_0$ and $\bar{x}_1$. If either of the possible values of $\bar{v} \oplus x_i$ is known before $\bar{S}_j$ selects x_0 and x_1, then v must have been released by the receiver before. So, it is unlikely that $x_0 \oplus x_1 = \bar{x}_0 \oplus \bar{x}_1$. Therefore, the probability that the adversary queries the PUF with either of the possible values of $\bar{v} \oplus x_j$, before x_0 and x_1 have been released, is negligible. Similarly, he has no chance to query the PUF after learning these values. So, he knows nothing about the decryption keys for s'_0 and s'_1. The simulator $\bar{S}_j$ can thus use some random s'_0 and s'_1. The reconstructed view for S_j works as above, by changing the table value of the PUF. Since s'_0 and s'_1 cannot be corrupted, the receiver $\bar{R}_i$ must receive them unchanged. In this situation, we have to change slightly the simulator $\bar{R}_i$ which was described before: indeed, we cannot extract s_0 and s_1 from these random values. But fortunately, we do not need to submit s_0 and s_1 to $\mathcal{F}_{\mathtt{OT}}$. This was done by the ideal S_j already. □

Note that the above theorem (and proof) only holds in JUC, but does not hold in GUC. This is because, in JUC, all sessions and the PUF usages inter-sessions will have one single, all-encompassing interface with the environment, as if they were all part of one macro-session; in GUC, by contrast, the environment (and thus the combined adversarial body) has fine-tuned, inter-session access to the PUF and has separate access/control to and over each individual session. Thus, the simulation in the proof above would default in GUC.

6. CONCLUSIONS

It is important to know not only what we can realize with (UC-)setup devices, but also what shortfalls they bring to the crypto world. In that sense, in this paper, we looked at UC-setup devices that may hinder EUC/JUC realization. We singled these out and characterized them formally. We thuswise denote the class of transferable setups; informally, these are setup devices which do not (publicly) disclose if they have been maliciously passed on. A subset of these are the well-known physical unclonable functions (PUFs).

We proved that one cannot realize oblivious transfer (OT) or any other "interesting" 2-party protocol using transferable setups in the EUC model.

Furthermore, if relay attacks are possible then oblivious transfer cannot be realized in the JUC model either. As a by-product, we show that the protocols proposed built with PUFs by Brzuska *et al.* at CRYPTO 2011, by Ostrovsky *et al.* at EUROCRYPT 2013, by Damgård and Scafuro at ASIACRYPT 2013, are subject to relaying thus not being JUC secure. In addition to this, they all need authenticated channels. We also discuss this need, to some extend.

We showed how to strengthen one of these PUF-based protocols to make them JUC-secure.

7. REFERENCES

[1] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. *Journal of Cryptology*, 24(4):720–760, October 2011.

[2] Ioana Boureanu and Serge Vaudenay. Several weak bit-commitments using seal-once tamper-evident devices. In *Proceedings of the 6th International Conference on Provable Security*, ProvSec'12, pages 70–87, Berlin, Heidelberg, 2012. Springer-Verlag.

[3] Ioana Boureanu and Serge Vaudenay. Compact and efficient uc commitments under atomic-exchanges. In *Proceedings of the 17th International Conference on Information Security and Cryptology*, ICISC 2014, Berlin, Heidelberg, 2015. Springer-Verlag. to appear.

[4] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In *CRYPTO*, pages 51–70, 2011.

[5] R. Canetti. A Unified Framework for Analyzing Security of Protocols. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(16), 2001.

[6] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. Cryptology ePrint Archive, Report 2006/432, 2006. http://eprint.iacr.org/.

[7] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology, Proceedings of the 22nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86. Springer, 2003.

[8] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[9] Ran Canetti and Tal Rabin. Universal composition with joint state. Cryptology ePrint Archive, Report 2002/047, 2002. http://eprint.iacr.org/.

[10] Cheun Ngen Chong, Dan Jiang, Jiagang Zhang, and Long Guo. Anti-counterfeiting with a random pattern. In *Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*, SECURWARE 2008, pages 146–153, Washington, DC, USA, 2008. IEEE Computer Society.

[11] Ivan Damgård and Alessandra Scafuro. Unconditionally secure and universally composable commitments from physical assumptions. Cryptology ePrint Archive, Report 2013/108, 2013. http://eprint.iacr.org/.

[12] Ivan Damgård and Alessandra Scafuro. Unconditionally secure and universally composable commitments from physical assumptions. In *In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, ASIACRYPT '13*, pages 100–119, 2013.

[13] Sezer Goren, H. Fatih Ugurdag, Abdullah Yildiz, and Ozgur Ozkurt. FPGA design security with time division multiplexed PUFs. In *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS)*, volume 28, pages 608–614, July 2010.

[14] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding Cryptography on Tamper-Proof Hardware Tokens. In *Theory of Cryptography*, pages 308–326, 2010.

[15] J. Katz. Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In *Theory and Application of Cryptographic Techniques*, pages 115–128, 2007.

[16] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for sram pufs. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES 2009, pages 332–347, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer, 2010.

[18] P. Mateus and S. Vaudenay. On Tamper-Resistance from a Theoretical Viewpoint. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems(CHES)*, volume 5747 of *Lecture Notes in Computer Science*, pages 411–428. Springer, 2009.

[19] T. Moran and M. Naor. Basing Cryptographic Protocols on Tamper-Evident Seals. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, Jul 2005.

[20] T. Moran and M. Naor. Polling with Physical Envelopes: A Rigorous Analysis of a Human-Centric Protocol. In *Advances in Cryptology, Proceedings of the 25th Annual International Conference on Theory and Application of Cryptographic Techniques – EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 88–108. Springer Berlin / Heidelberg, May 2006.

[21] T. Moran and M. Naor. Basing Cryptographic Protocols on Tamper-Evident Seals. *Theoretical Computer Science*, 411:1283–1310, March 2010.

[22] T. Moran and G. Segev. David and Goliath Commitments: UC Computation for Asymmetric Parties Using Tamper-Proof Hardware. In *Theory and Application of Cryptographic Techniques*, pages 527–544, 2008.

[23] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. Universally composable secure computation with (malicious) physically uncloneable functions. In *Advances in Cryptology, Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT*, pages 702–718, 2013.

[24] Ravikanth S. Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, March 2001.

[25] Masoud Rostami, Mehrdad Majzoobi, Farinaz Koushanfar, Dan S. Wallach, and Srinivas Devadas. Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching. *IEEE Transactions on Emerging Topics in Computing*, 2014. to appear.

[26] Ulrich Rührmair and Marten van Dijk. On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols. *J. Cryptographic Engineering*, 3(1):17–28, 2013.

[27] Ulrich Rührmair and Marten van Dijk. PUFs in security protocols: Attack models and security evaluations. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 286–300. IEEE Computer Society, 2013.

[28] Marten van Dijk and Ulrich Rührmair. Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results. Cryptology ePrint Archive, Report 2012/228, 2012. `http://eprint.iacr.org/`.

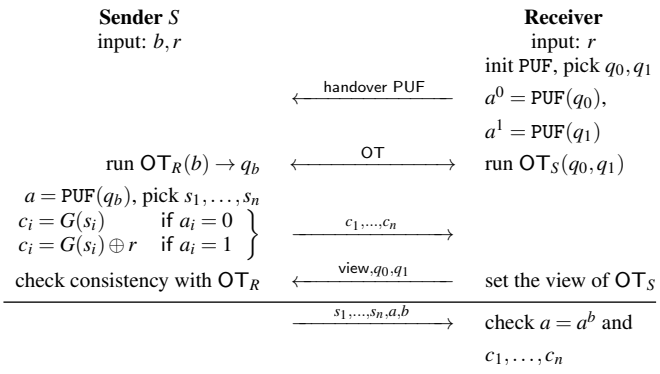

Figure 8: The Simplified Com_{equiv} Protocol

APPENDIX

A. AUTHENTICATED CHANNELS & UC-SECURITY WITH PUFS

In [23], there are two commitment protocols using PUFs. One is inherited from the protocol of Brzuska *et al.* and, like it, it requires authentication of communication as an extra assumption on top of the insecure-channel UC model; this assumption is stated clearly in [4]. The other, completely novel protocol in [23] also requires authenticated channels, even if this is not stated in the paper.

To see that this is the case, we sketch a man-in-the-middle attack against the Com_{equiv}, if the UC channels therein were not authenticated. The protocol is recalled on Fig. 8. Note that the input r is the result of a previous coin flipping protocol in the full commitment protocol by [23]. It is known by the adversary. The step where S sends some c_i's based on r to R is known as the Naor commitment of a_b. It is opened by giving $s_1, \ldots, s_n$ with a. The protocol uses a subprotocol OT where R plays the role of the sender. This protocol runs based on a view to be disclosed later on, so that S – playing the role of the receiver– can check for consistencies.

Thus, imagine a honest receiver R following the protocol, in –let us say– session 1 of Com_{equiv}. I.e., R chooses q_0 and q_1, queries $\mathtt{PUF}_R$ on one of them, obtains a_i, with $i \in \{0,1\}$, and then hands over the PUF to the honest sender S. He then runs the inner OT protocol on q_0, q_1. The man-in-the-middle (MiM) adversary $\mathcal{A}$ is playing two different protocols with S and R. To R in session 1, he runs the OT protocol with a random bit $\overline{b}$, As expected, this man-in-the-middle gets out of the OT protocol a state $q_{\overline{b}}$. In the OT session which is run with S, let us call it session 2, $\mathcal{A}$ selects q'_0 and q'_1 such that $q'_{1-\overline{b}} = q_{\overline{b}}$ and $q'_{\overline{b}}$ is random. Supposing that S inputted a bit b into the OT protocol, then this sender S gets q'_b as the OT output. So, $q'_b = q_{1-b}$ with probability $\frac{1}{2}$. Then, the Naor commitment c from S to R is left unchanged. The view (with the OT inputs) that $\mathcal{A}$ sends to S is set so that it is consistent with q'_0 and q'_1. In the opening, the Naor commitment is left unchanged but the bit b is replaced by $\overline{b}$.

In this MiM-infected real world execution, R receives $\overline{b} = 1 - b$ with probability $\frac{1}{2}$. Of course, this will never be the case in the ideal-world execution. This is due to S and R being honest and the commitment functionality making sure that only the bit b set by S can be opened to R. So, authentication is a requirement in order for the Com_{equiv} protocol in [23] to enjoy UC-security.