Intercepting Tokens: The Empire Strikes Back in the Clone Wars

Özgür Dagdelen Marc Fischlin

Darmstadt University of Technology, Germany
oezguer.dagdelen@cased.de, marc.fischlin@gmail.com

Abstract We discuss interception attacks on cryptographic protocols which rely on trustworthy hardware like one-time memory tokens (Goldwasser et al., Crypto 2008). In such attacks the adversary can mount man-in-the-middle attacks and access, or even substitute, transmitted tokens. We show that many of the existing token-based protocols are vulnerable against this kind of attack, which typically lies outside of the previously considered security models.

We also give a positive result for protocols remaining secure against such attacks. We present a very efficient protocol for password-based authenticated key exchange based on the weak model of one-time memory tokens. Our protocol only requires four moves, very basic operations, and the sender to send ℓ tokens in the first step for passwords of length ℓ. At the same time we achieve information-theoretic security in Canetti's universal composition framework (FOCS 2001) against adaptive adversaries (assuming reliable erasure), even if the tokens are not guaranteed to be transferred securely, i.e., even if the adversary can read or substitute transmitted tokens.

1 Introduction

Recently, the area of designing cryptographic protocols in the presence of hardware tokens has gained significant attention [39, 40, 38, 32, 49, 16, 20, 47, 42, 34, 33, 28, 25, 9, 23, 18, 26, 8]. Nonetheless, the idea of relying on tamper-proof hardware devices to relax assumptions dates back to the 80's [62, 50]. Indeed, security tokens such as smart cards have already undergone a comprehensive treatment from a hardware's perspective, e.g., [3, 37, 17, 22, 60, 53, 6, 61, 45], and have been commercially available even in simple forms such as for one-time passwords [27], before their usefulness for the design of more complex cryptographic protocols has begun to be fully explored. The pleasant aspect of using hardware tokens is that they facilitate the design of efficient protocols while still achieving strong security guarantees. Notably, most of the aforementioned works design protocols in Canetti's universal composition (UC) framework [12]. There are two reasons for this: (a) the derived protocols provide strong security even in "hostile" environments (e.g., in presence of concurrent executions of other protocols), and (b) vice versa, the UC framework allows an easy modeling of such tokens.

The aforementioned works use different types of hardware tokens, ranging from very powerful smartcards as in [40] to very minimalistic one-time memory (OTM) tokens as in [32]. The latter are tamper-proof tokens which allow a sender to store two values and to transfer the token to a receiver, and when one of the two values is read according to a receiver's choice, the other value collapses and is irrevocably lost. Although resembling a hardware-based implementation of an oblivious transfer protocol, Goldwasser et al. [32] point out that these tokens implement a weaker functionality than oblivious transfer. The main difference is that the (single) measurement on the receiver's side is carried out locally such that an adversarial receiver may postpone the measurement instead of performing the read-out when supposed to do. This is in contrast to an execution of an oblivious transfer protocol where the sender knows when the transfer takes place (and can therefore enforce a transfer at a certain point in a protocol run).

There are essentially two types of approaches using tokens: some devised protocols focus on concrete problems [38, 42, 28, 10] like secure computation of set intersections and aim at providing really efficient protocols. Others provide general feasibility results to compute arbitrary functions [40, 32, 33, 34, 25] and are, therefore, more general but at the same time often less efficient than desired. Another important research direction, which emerged recently, is to investigate how much trust in the tokens is required. That is, the works by Kolesnikov [42], Fischlin et al. [28], Ostrovsky et al. [52], Damgård and Scafuro [21], and van Dijk and Rührmair [64, 59] consider the problem of untrustworthy tokens. These works investigate which security properties can be achieved if an adversary is able to introduce malicious tokens with diverging behavior in the protocol.

1.1 Interception Attacks on Token-Based Protocols

Here we discuss a somewhat dual problem to malicious tokens, namely, the adversary's ability to intercept trustworthy tokens. Previous efforts often assumed that tokens are eventually faithfully delivered to the intended receiver, and that the adversary could not intercept and replace tokens *on behalf of the sender*. This is testified by the common definition of token functionalities (e.g., [40, 32, 16, 34, 8]) in which the receiver is informed by the functionality about a token creation from the sender. In particular, the adversary thus cannot send a different token on behalf of the sender. This is even the case for the tokens considered in [16] where the adversary can basically place tokens inside other tokens, but creation of such tokens again reveals the sender's identity to the receiver. We show that a vast number of previous proposals (e.g., [32, 42, 34, 24]) becomes insecure when the adversary is able to mount such substitution attacks; this is independent of the question whether tokens can be malicious or not.

Remarkably, the above problem refers to the authenticity of the tokens in the sense that one can reliably identify the sender of the token. This thwarts attacks in which the adversary tries to inject tokens. The adversary, however, may not even need to be able to replace tokens to mount successful attacks. It sometimes suffices to be able to access the honestly generated token, instead of the intended receiver. Attacks against this "confidentiality" property have been partially described in the area of physical unclonable functions (PUFs) in UC protocols for oblivious transfer, key exchange, or bit commitments, where the adversary can communicate with the PUF before it eventually reaches the designated receiver [10, 52, 21]. These papers show that security can still be achieved in this case, albeit further attacks exist if the adversary can access PUFs *after* the protocol has ended [64, 59]. However, the positive results [10, 52, 21] crucially rely on the authenticated transmission of tokens, preventing substitution attacks.

Here we show that one cannot realize any functionalities with an embedded oblivious transfer (or similar properties) if transmitted tokens can be intercepted by the adversary. This also presumes that there is no other authenticated (digital) channel, which could otherwise be used to run an authenticated OT protocol without tokens. It follows that the known OT-like protocol of [34], which merely transmit tokens, cannot be secure against token interceptions attacks. The result extends to other protocols like [32, 24] if the additional digital transmissions in these protocols are not authenticated either. We note that for these protocols the results merely rely on the ability to *read out* intercepted tokens, and do not even need to take advantage of substitution attacks.

1.2 Positive Results: Password-Based Key Exchange

Since we do not touch the issue of malicious tokens here —which, as explained above, is orthogonal to the question of interceptions— we investigate a solution which presumes that one can identify well-formed tokens. This is similar to bank notes for which recipients can verify the integrity of notes, but for which the reliable authentication of the payer by purchases over the counter, is usually not guaranteed. Nonetheless, this could be ensured in principle by means of certification through, say, identification documents. As in this example we assume that tokens contain an imprinted and tamper-proof serial number. In order to provide authentic transmissions, i.e., allowing for trustworthy identification of the source, we can assume that the sender simply also sends the signed serial number or, more generally, re-sends the serial number through an authenticated channel between the two parties.

The above solution, however, does not work in the case that there is no pre-established authenticated channel. A concrete example for such a setting is authenticated key exchange. Roughly, in an authenticated key exchange protocol two parties aim to establish a shared key with a designated partner, usually with the goal to form a confidential and authenticated channel between them. In the password-based case the parties initially share a human-memorizable password of limited entropy, and still try to establish a strong cryptographic key. In this case it is often assumed that the key establishment is not carried out over an authenticated channel, such that the aforementioned solution of authenticated transmission of the token's serial number does not work offhandedly. Instead, the authenticity of tokens must originate from within the password-based protocol itself.

A widely-deployed security notion for such password-based authenticated key exchange protocols has been given by Bellare et al. [7]. Later, Canetti et al. [14] (and subsequently [1]) gave stronger formalizations for password-based key exchange in the UC framework. So far, only a few password-based key exchange protocols meeting the UC notion have been proposed, usually relying on encryption schemes and hash proof systems [14, 1, 31, 36, 41]. An exception is the recently proposed solution by Canetti et al. [13]. Their efficient UC-secure password-based key exchange protocol is based on oblivious transfers. However, the security of their construction relies on computational assumptions and the common reference string model. Note that Barak et al. [4] pointed out that UC-secure password-based key exchange should be infeasible in the unauthenticated case, unless some auxiliary mean is provided (like a common reference string in [4]). Let us also stress again that general feasibility results like the one in [34], showing that one can realize any functionality with a non-interactive, statistically UC-secure protocol based on one-time (bit) memory tokens, is not applicable as it requires authenticated channels, as our interception attacks show.

Here we design a new protocol for authenticated password-based key exchange, resistant against interception attacks. In line with previous efforts for strong security based on tokens, our protocol achieves UC security. Remarkably, we use only the weak one-time memory tokens (for strings) and, in contrast to previous efforts, our protocols are information-theoretically secure against adaptive corruptions (if the sender can reliably erase randomness). That is, we formally show UC security for unbounded adaptive adversaries and unbounded environments in the hybrid model where we model one-time memory tokens through ideal functionalities. In particular, we do not rely on any unproven cryptographic assumption. According to a recent result by Unruh [63] our protocols thus also resist quantum adversaries. In addition, from a result by Nielsen [51] about non-committing encryption, it follows there cannot exist key agreement protocols secure against adaptive corruptions in the UC model, in the standard model and assuming a full state reveal. Our protocol overcomes this impossibility result by assuming reliable erasures.

As for efficiency, our protocol only requires very basic operations, and for ℓ-bit passwords requires only ℓ tokens, sent in the first step from the sender to the receiver. The overall protocol requires four moves and $\mathcal{O}(\ell\lambda)$ bits of communication for security parameter λ, determining the level of statistical security.

1.3 Discussion

At a superficial glance, it may seem easy to design key exchange protocols when one-time memory tokens are available. For instance, simple approaches like having the sender encode random pairs in ℓ tokens, asking the receiver to read the i-th token according to the i-th password bit and to send parts of the value back to authenticate and use the other part to compute the key, are imaginable. This construction, however, does not withstand very basic attacks. Namely, a malicious sender could prepare such tokens and compute the password from the receiver's answer. In fact, the main obstacle in designing UC secure protocols is to solve the "chicken-and-egg" problem of authentication, i.e., one party needs to authenticate first without revealing anything useful to the adversary about the password. Our solution shows that this is still possible with the approach of having the receiver read out the i-th of the ℓ tokens according to the i-th password bit. But we let the sender first authenticate by having the receiver send a random challenge vector after having received the tokens, with the guarantee that the sender's subsequent authentication reply looks random if the receiver reads out a single token at the "wrong" position. In the fourth round of out protocol the receiver then authenticates.

One-time memory tokens are advantageous due to their simple functionality. However, once the token has been queried, the token is useless. Ideally, one would like to have such tokens which implement a reusable version, but then usually information-theoretic security cannot be achieved anymore. If one settles for computational security, then the token can either be stateful, i.e., change its state after queries, or stateless. Clearly, the latter tokens are again preferred as they are, for example, resettably secure [16, 42, 34]. Our protocols can be easily implemented in a stateful version, as the values stored in the memory are purely random. Hence, one can easily use a pseudorandom function to let the token return (pseudo)random answers and update the key of the pseudorandom function afterwards. It remains open if one can implement a version using a stateless token, without involving additional assumptions like an oblivious transfer protocol. Another alternative, to reduce the number of tokens required per execution, is to use the (statistically-secure) sequential-OTM functionality derived from a single wrapper token [25]. While the solution requires additional interaction it is quite efficient; one would need to check, though, that the construction remains secure for unauthenticated transmissions.

To support the argument of the difficulty to design secure key-exchange protocols based on one-time memory tokens note that, as mentioned before, such tokens provide a weaker version of oblivious transfer in terms of functionality. The important difference, as pointed out in [32], is that a malicious receiver may decide to postpone the read-out of the token, whereas the sender in an oblivious transfer protocol can be assured about the point in time when the receiver obtained one of the two values. In analogy to such "forced read-outs" we note that such delayed measurements (in case of quantum states) are the reason that the common security definition for the BB84 quantum key exchange protocol turned out to be inadequate when composed with the one-time pad encryption [57].

Additionally, referring to the case of authentic vs. unauthentic transmissions, most oblivious transfer protocols today are assumed to be run over an authenticated channel and merely provide computational security [29, 35, 19, 55, 58, 13].[1] Furthermore, in order to be UC secure they also need some kind of set-up assumption like a common reference string [12]. It is currently unclear if one can implement such set-up assumptions if the tokens are not sent in an authenticated way.

From a high-level perspective our protocol bears some similarities with the solutions based on oblivious transfer, proposed by Canetti et al. [13]. A closer look, however, reveals the differences in the designs of the protocols. Basically, [13] presents two UC-secure protocols, one using oblivious transfer in both directions and achieving adaptive security, the other one using OT only from the initiator to the responder but withstanding static corruptions only. Looking at the underlying assumptions, both their solutions work in the common reference string model and provide computational security, whereas our token-based protocol does not require any set-up assumption, and achieves statistical security against adaptive corruptions, but only if assuming reliable erasure.

But the main difference between the protocols in [13] and here lies in the way the oblivious transfer is used and how authenticated channels are implemented. One solution in [13] uses OT in both directions which, if implemented with hardware tokens, would require exchange of tokens and constitute a major disadvantage to other token-based protocols. Hence, we are interested in protocols in which one party sends tokens at the outset only. The other solution in [13] indeed uses OT in one direction only, but requires chosen-ciphertext secure encryption instead. Secondly, [13] first design their protocols in the presence of authenticated channels, and then use known techniques [4, 30] to implement the OT functionality securely without authenticated channels. In our solution, the authentication step is "built-in" into the protocol, without relying on further techniques, yielding more efficient solutions and showing that one-time memory tokens only are sufficient.

Finally, we remark that there seems to be an obvious relation of one-time memory protocols to quantum cryptography. Like a one-time memory token a quantum state collapses when measured. There are, nonetheless, important differences. First note that one-time memory tokens allow "secure local" measurements, whereas a quantum adversary could entangle quantum states and any "local" measurement, e.g., based on the password, would immediately become available to the adversary. Indeed, the impossibility of unconditionally secure 1-out-of-2 oblivious transfer in the quantum setting is well established today [46, 48]. For more discussions about the relationship of quantum power and one-time memory tokens see [8].

2 Security Model

2.1 Universal Composition Model

We analyze our password-based key exchange protocol in the Universal Composition Framework by Canetti [12]. In the UC framework one compares *real-world* executions of a protocol π among (possibly corrupt) parties $P_1, \dots, P_n$ in the presence of an adversary $\mathcal{A}$ to *ideal-world* executions

[1]It must be said though that, except for [29, 13], none of the papers is explicit about authenticated channels; we draw this conclusion by consultation of the proofs.

in the presence of an ideal-model adversary $\textsc{Sim}$, usually called simulator. In the ideal-world, the parties have secure access to an ideal functionality $\mathcal{F}$ which captures the desired functional and security properties in an abstract way; in this case, the parties essentially only forward any input to the functionality and are thus called dummy parties. As usual, both adversaries may corrupt parties at the beginning only (static corruptions) or during protocol executions (adaptive corruptions). In the latter case, one often assumes that honest parties are able to erase unnecessary internal data reliably such that these data are not available to the adversary upon corruption. For technical reasons each execution is accompanied by a globally unique session identifier sid available to all parties.

In both worlds a special party, called the environment $\mathcal{Z}$, provides the input to the parties, reads their outputs, and communicates with the adversary. Roughly, the environment models an active distinguisher which determines any setting in which the protocol may be executed. Ideally, the environment should not be able to tell apart real-world executions from ideal-model executions. That is, let $\textsc{Real}_{\mathcal{A},\mathcal{Z},\pi}(\lambda)$ be the output of the environment when interacting with a real-world execution of protocol π in the presence of $\mathcal{A}$ for security parameter λ, and $\textsc{Ideal}_{\textsc{Sim},\mathcal{Z},\mathcal{F}}(\lambda)$ denote $\mathcal{Z}$'s output for the ideal world. Then, a protocol π securely realizes $\mathcal{F}$ (information-theoretically) if for any efficient (resp. unbounded) adversary $\mathcal{A}$ there should be an efficient (resp. unbounded) simulator $\textsc{Sim}$ such that for any efficient (resp. unbounded) environment $\mathcal{Z}$ we have

$$\textsc{Real}_{\mathcal{A},\mathcal{Z},\pi}(\lambda) \approx \textsc{Ideal}_{\textsc{Sim},\mathcal{Z},\mathcal{F}}(\lambda),$$

where $\approx$ denotes computationally indistinguishability (resp. statistical indistinguishability).[2]

To model hardware tokens we follow previous approaches and specify tokens through ideal functionalities as well, and work in the *hybrid world*. In this real-world surrogate all parties $P_1, \ldots, P_n$ and $\mathcal{A}$ now also have access to an ideal functionality $\mathcal{G}$, and we write $\textsc{Hybrid}_{\mathcal{A},\mathcal{Z},\pi^{\mathcal{G}}}(\lambda)$ for the environment's output. We say that π *securely realizes* $\mathcal{F}$ *in the* $\mathcal{G}$*-hybrid world (information-theoretically)* if for any efficient (resp. unbounded) adversary $\mathcal{A}$ there should be an efficient (resp. unbounded) simulator $\textsc{Sim}$ such that for any efficient (resp. unbounded) environment $\mathcal{Z}$ we have

$$\textsc{Hybrid}_{\mathcal{A},\mathcal{Z},\pi^{\mathcal{G}}}(\lambda) \approx \textsc{Ideal}_{\textsc{Sim},\mathcal{Z},\mathcal{F}}(\lambda).$$

One may again assume here that the simulator is at most polynomial time in the adversary's running time; we do not impose this restriction.

2.2 One-Time Memory Tokens

As explained above we model the hardware token as an ideal functionality and work in the corresponding hybrid model. We model sending of tokens in the protocol by having a party create the token and transmitting the unique token identifier tid (which corresponds to the session identifier of the token hybrid in the UC model) to the partner. We assume that each tid is of the form $\mathrm{tid} = (\mathrm{sid}, \mathrm{tid}')$ for the session identifier sid for which the token is created.[3] Hence, when considering authenticated transmissions of tokens, i.e., sending the tid over an authenticated channel, the

[2]Here, one may further assume that the simulator is at most polynomial time in the adversary's running time; as remarked in the introduction we stick to the more liberal definition and revisit this issue when stating the theorem.

[3]In a stronger model one could demand that tokens are not even session-specific. But then the benefits of using the UC model, analyzing a single protocol execution and concluding composition immediately, would be lost. Instead one would need to manually deal with multiple dependent executions, similar to the joint-state UC framework [15]. For the same reasons we do not allow the adversary to put names of other parties into tokens, because the parties are bound to the session.

adversary cannot change the identifier tid sent from an honest sender. Nonetheless, the adversary can still access the token via calling the ideal functionality for the tid. In previous definitions authentic token transfers have been ensured by letting the functionality inform the receiver about a newly created token by the sender. Interestingly, most definitions (e.g., [16, 34]) neither allow calls of the adversary to the token created for the honest receiver and thus, formally, even exclude temporary access through the adversary when the token is in transmission.

To consider unauthenticated transmissions of tokens, we let the adversary intercept tokens and read them, even when the designated receiver is honest. This requires the adversary also to create tokens (within the session sid) and to replace the depleted token. We note that we do not need to make any assumption about who currently is in possession of the token (i.e., that the adversary cannot read the token if it has already been delivered). The reason is that we can assume that, if the adversary forgoes reading the token when sent and instead delivers it to the receiver who then reads the token and ultimately makes it inaccessible for the adversary.

Figure 1 illustrates the ideal functionalities of One-Time-Memories in both versions. Besides the secure and insecure versions $\mathcal{F}_{\mathrm{OTM}}^{sec}$ and $\mathcal{F}_{\mathrm{OTM}}^{insec}$ where the adversary can neither access nor create tokens, or do both, we occasionally denote by $\mathcal{F}_{\mathrm{OTM}}^{auth}$ the functionality where the adversary cannot call `create` on behalf of another party, but can still access sent tokens. In other words, the transmissions are authentic but not confidential.

In the sequel, we often merely say that a party creates a token. It is understood that the party calls the functionality in the hybrid model with the corresponding input command `create`. The party sends the token to another party if it later transmits the token identifier to this party. The receiver is assumed to check that the identifier matches the session identifier. We then say that the receiver queries or accesses the token if it sends a `choice` command.

We note that the token identifier tid is immediately available to the adversary, once the token has been created by an honest sender. She could thus access the token before it is even delivered, i.e., before tid is sent to the intended receiver. However, since we can assume that the honest sender never accesses its own token (because it already knows the encapsulated values) and that tid may be sent by the protocol, this does not violate generality. Analogously, the protocol can easily enforce that an intended honest receiver P_j does not access the token before actually receiving tid.

2.3 Stronger Hardware Tokens

While one-time-memory tokens are simple but still powerful, other tamper-proof token models exist, implementing more complex operations and consequently are based on stronger (physical) assumptions. Closely related to OTMs are parallel OTMs (pOTMs) and external OTMs (ExtOTM). The former essentially allows a receiver to query multiple OTM tokens simultaneously, and return an output once the receiver committed to all inputs. ExtOTM tokens are OTM tokens augmented with a confirmation string r chosen by the sender, i.e., the receiver upon input bit b obtains (s_b, r) where the r-part is identical for both queries. The string r can be used to convince the sender that the token has been queried already. In other words, if the receiver sends back the string r to the sender after accessing the token, this essentially implements an OT protocol. Our changes in the definition for plain OTMs can be applied to all these tokens accordingly.

An even stronger token, called One-Time Programs (OTP), implements one-sided secure function evaluation non-interactively. The sender programs an OTP with its input and sends it over to the receiver, which learns the output of the function once the token is queried with its part of the input. Several works (e.g., [34, 24, 32]) show that pOTM, ExtOTM and OTPs can be instantiated

Functionalities $\mathcal{F}^{sec}_{\text{OTM}}$ and $\underline{\mathcal{F}^{insec}_{\text{OTM}}}$
(secure/<u>insecure</u> transmission)

The functionalities are parameterized by the security parameter λ. The interaction with an adversary and a set of parties is enabled via the following queries:

- On input $(\texttt{create}, \text{tid}, P_i, P_j, (s_0, s_1))$ from party P_i <u>or the adversary</u>, check if an entry $(\text{tid}, *, *)$ exists. If so, return $\perp$; else store the tuple (tid, s_0, s_1). Send the publicly delayed output $(\textbf{ready}, \text{tid}, P_i, P_j)$ to party P_j. (Note that the public delay means that the adversary is informed about the message, and can for example delay delivery infinitely long and thus basically substitute the token by another one.)
- On input $(\texttt{choice}, \text{tid}, P_i, P_j, x)$ from party P_j <u>or the adversary</u>, where $x \in \{0, 1\}$, check if an entry (tid, s_0, s_1) exists. If not, return $\perp$; else, send $(\textbf{out}, \text{tid}, P_i, P_j, s_x)$ to P_j resp. the adversary.

Figure 1: Ideal functionalities $\mathcal{F}^{sec}_{\text{OTM}}$ and $\mathcal{F}^{insec}_{\text{OTM}}$ for One-Time-Memory (OTM). The underlined text is valid only when considering insecure transmission. The first change refers to authenticated vs. unauthenticated transmissions, the second difference to confidential vs. accessible tokens. It is sometimes convenient to denote by $\mathcal{F}^{auth}_{\text{OTM}}$ the functionality as above, but where the adversary cannot call `create` on behalf of another party. In this sense the transmissions are authentic but not confidential.

by a polynomial number of OTMs. However, they typically rely on the fact that the OTMs are transmitted securely. We show in the next section that all those constructions are insecure in the UC model if considering insecure (even merely unauthenticated) transmission of hardware tokens.

3 Interception Attacks on Tamper-Proof Tokens

We first show that one cannot even obtain bit-OT in the UC model if one assumes only accessible tokens. For a formal definition of the OT functionality we refer to [12]. Basically, the sender inputs two bits x_0, x_1 and the receiver, upon inputting a bit b, receives x_b. For simplicity we state the result for our insecurely transmitted OTM token functionality $\mathcal{F}^{insec}_{\text{OTM}}$, but it extends straightforwardly to any tokens which are accessible by the adversary. The proposition presumes unauthenticated channels otherwise (such that one cannot simply run a token-free OT protocol over this channel), and that no set-up is used, especially no shared secrets between the parties.[4]

Proposition 3.1 *There is no UC-secure protocol for oblivious bit-transfer in the $\mathcal{F}^{insec}_{OTM}$-hybrid model (assuming unauthenticated channels).*

In case that only the sender creates tokens in the protocol the proposition holds even if the adversary can merely access the transmitted tokens; it does not even need to be able to substitute them. Put differently, the claim then still holds in the $\mathcal{F}^{auth}_{\text{OTM}}$-hybrid model.

[4]The fact that any other transmissions are unauthenticated, too, avoids conflicts with the PUF-based UC-secure OT-constructions in [10, 52] where the adversary can indeed access the PUF; these protocols send additional digital messages over authenticated channels.

Proof. Assume that we have a protocol π in the one-time token hybrid world which allegedly implements the oblivious transfer functionality UC-securely in which the sender P_i inserts a pair of bits (x_0, x_1) and the receiver P_j a bit b to receive x_b. Consider the following environment $\mathcal{Z}$ and the following adversary $\mathcal{A}$ against this protocol.

The environment $\mathcal{Z}$ picks $x_0, x_1 \leftarrow \{0,1\}$ randomly and writes these two values on the input tape of the honest sender P_i, which is supposed to start an oblivious transfer with P_j. It also writes $b = 1$ on the input tape of the honest receiver P_j. It also internally invokes a copy of the receiver's program P'_j of π but for input $b = 0$. Next, it instructs the adversary $\mathcal{A}$ to relay the communication between the original sender P_i and the internal copy P'_j of the receiver, making P_i believe that it is executing the OT with P_j. In particular, the fact that the adversary can intercept and read-out tokens sent from P_i to P_j resp. create tokens on behalf of the receiver, and that the channels are unauthenticated, enables $\mathcal{A}$ to perform all steps on behalf of P_j. The environment finally outputs 1 if and only if the local copy P'_j recovers x_0 correctly.

Clearly, if $\mathcal{Z}$ communicates with $\mathcal{A}$ in the hybrid world with the actual protocol π it outputs 1 with probability negligibly close to 1; the small error may be due to the fact that the protocol does not implement the functionality perfectly (but security demands that it cannot be more than negligible loss). On the other hand, in the ideal world the simulator $\textsc{Sim}$ does not get to see any information about x_0, because both the sender and the receiver remain honest. (Interestingly, this would even remain true if $\textsc{Sim}$ could corrupt P_j and learn x_1). Hence, $\textsc{Sim}$ can make the environment output 1 with probability at most $\frac{1}{2}$, allowing $\mathcal{Z}$ to distinguish the two worlds with a constant difference in the probabilities. □

With the above approach one can conclude, for example, that the token-only protocols by Goyal et al. [34] for securely realizing pOTM from OTMs, Goyal et al. [34] for securely realizing ExtOTM from OTMs, and Goldwasser et al. [32] for securely realizing OTP from OTMs cannot be secure when considering interceptable tokens. The same is true for the protocols in [42, 24] if we assume that the additional digital transmissions are also unauthenticated. We note that neither of these protocols claims to provide security in this case, since interception attacks are not considered.

4 UC-Secure Password-Based Key Exchange

We first recall the ideal functionality for password-based key exchange, and then present a "light-weight" protocol which realizes the functionalities in the $\mathcal{F}^{auth}_{\mathrm{OTM}}$-hybrid world (and assuming authenticated channels). Recall that this means that the adversary can at least read intercepted tokens, but it cannot inject new ones. We also discuss that this version of the protocol falls prey to interception attacks (showing that also protocols in which the parties potentially share some secret at the outset are vulnerable). We then discuss how to modify our protocol to achieve security in the $\mathcal{F}^{insec}_{\mathrm{OTM}}$-hybrid setting.

4.1 Modeling Password-Based Key Exchange

We next revisit the definition of an ideal functionality $\mathcal{F}_{\mathrm{pwKE}}$ of Canetti et al. [14] (See Fig. 2). The ideal functionality captures the abstract properties of a key exchange protocol in the sense that two parties agree upon the same cryptographic key only if they hold the same passwords. Furthermore, by definition, the adversary can test only a single (explicit) password for correctness in each execution; for a wrong guess, the execution is interrupted. Note that the definition does not make any claims about the additional property of contributiveness, in the sense that both parties contribute to the key (as defined in [2]).

For more motivational discussion about the definition see [14], where it is shown that any protocol realizing this definition also provides security according to the common Bellare-Pointcheval-Rogaway notion for password-based authenticated key exchange [7] (except for technical issues regarding session identifiers). As pointed out in [1] any protocol meeting the definition immediately provides forward security as well, since the environment chooses passwords. The latter also makes the deployment of the joint-state UC framework [15] obsolete.

Functionality $\mathcal{F}_{\text{pwKE}}$

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by the security parameter λ. The interaction with an adversary SIM and a set of parties is enabled via the following queries:

- On input $(\texttt{NewSession}, \text{sid}, P_i, P_j, \text{pw}, \text{role})$ from party P_i, send $(\texttt{NewSession}, \text{sid}, P_i, P_j, \text{role})$ to SIM. Furthermore, in case this is the first `NewSession` query, or the second query while there exists a record $(\text{sid}, P_j, P_i, \text{pw}')$, then record $(\text{sid}, P_i, P_j, \text{pw})$ and mark this record *fresh*.
- On input $(\texttt{TestPwd}, \text{sid}, P_i, \text{pw}')$ from the adversary SIM, check for a fresh record of the form $(\text{sid}, P_i, P_j, \text{pw})$. If found and
  - $\text{pw} = \text{pw}'$, mark the record as *compromised* and send "correct guess" back to SIM.
  - $\text{pw} \neq \text{pw}'$, mark the record as *interrupted* and send "wrong guess" back to SIM.
- On input $(\texttt{NewKey}, \text{sid}, P_i, sk)$ from SIM and sk is an element in the session key space, check for a record of the form $(\text{sid}, P_i, P_j, \text{pw})$ and that this is the first `NewKey` query for P_i. If so, then
  - If this record is *compromised*, or either P_i or P_j is corrupted, then output $(\texttt{NewKey}, \text{sid}, sk)$ to party P_i.
  - If this record is *fresh*, and there is a record $(\text{sid}, P_j, P_i, \text{pw}')$ with $\text{pw}' = \text{pw}$, and a key sk' was sent to P_j where $(\text{sid}, P_j, P_i, \text{pw}')$ was *fresh* at the time, then output $(\texttt{NewKey}, \text{sid}, sk')$ to P_i.
  - Otherwise, pick a new random key sk' in the key space and send $(\texttt{NewKey}, \text{sid}, sk')$ to P_i.

  In any case, this record is marked as *completed*.

Figure 2: Ideal functionality $\mathcal{F}_{\text{pwKE}}$ for password-based key exchange

4.2 Key Exchange in the Authenticated Setting

In this section, we propose an information-theoretically secure password-based authenticated key-exchange protocol using one-time memory tokens. As explained we first assume that all transmissions are authenticated, including the token handover. In other words, in each execution the adversary may eavesdrop on the communication of honest parties, or impersonate either side if the corresponding party has already been corrupted. Since the adversary cannot modify tokens nor the communication, we thus explicitly exclude man-in-the-middle attacks, i.e., the adversary can only send a chosen token in an execution if it impersonates the actual sender in that execution. We do, nonetheless, allow the adversary to access transmitted tokens in all cases.

Sender :		**Receiver** :
$s^A_{b,i}, s^B_{b,i}, s^K_{b,i} \overset{\$}{\leftarrow} \{0,1\}^s$ for $b \in \{0,1\}$, $i = 1, \dots, \ell = \lvert\text{pw}\rvert$		abort $\leftarrow$ false
For $i = 1 \dots \ell$ generate token $\mathcal{T}_i$ with tid_i: $\mathcal{T}_i$ outputs $(s^A_{b_i,i}, s^B_{b_i,i}, s^K_{b_i,i})$ on input b_i		
Erase all $(s^A_{b_i,i}, s^B_{b_i,i}, s^K_{b_i,i})$ for $b_i \neq \text{pw}_i$	$\xrightarrow[(1)]{tid_1 \dots tid_\ell}$	$(s^A_i, s^B_i, s^K_i) \leftarrow \mathcal{T}_i(\text{pw}_i)\ \forall i$
		If sid in tid is invalid, then abort $\leftarrow$ true
	$\xleftarrow[(2)]{t_{b,i}}$	$t_{b,i} \overset{\$}{\leftarrow} \{0,1\}^s$ for $b \in \{0,1\}$ and $\forall i$
$y \leftarrow \bigoplus_{i=1}^{\ell} (s^A_{\text{pw}_i,i} \oplus t_{\text{pw}_i,i})$	$\xrightarrow[(3)]{y}$	If $y \neq \bigoplus_{i=1}^{\ell} (s^A_i \oplus t_{\text{pw}_i,i})$, then abort $\leftarrow$ true
		If abort = false, then $z \leftarrow \bigoplus_{i=1}^{\ell} s^B_i$
	$\xleftarrow[(4)]{z}$	else $z \overset{\$}{\leftarrow} \{0,1\}^s$
If $z = \bigoplus_{i=1}^{\ell} s^B_{\text{pw}_i,i}$, then $sk \leftarrow \bigoplus_{i=1}^{\ell} s^K_{\text{pw}_i,i}$ else $sk \overset{\$}{\leftarrow} \{0,1\}^s$		If abort = false, then $sk \leftarrow \bigoplus_{i=1}^{\ell} s^K_i$ else $sk \overset{\$}{\leftarrow} \{0,1\}^s$
session key := sk		session key := sk

Figure 3: Password-based authenticated key-exchange protocol $\mathsf{KE}_{\text{auth}}$ using OTM tokens

4.2.1 The Protocol $\mathsf{KE}_{\text{auth}}$

The idea of our protocol $\mathsf{KE}_{\text{auth}}$ is as follows. For each bit pw_i of the password pw the sender A hands over an OTM token to the receiver B with two tuples $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$ for $b \in \{0,1\}$ of random s-bit strings. The receiver fetches the tuple corresponding to bit pw_i and the other tuple is then erased by the definition of the token's functionality. Next, both parties authenticate themselves using the password by having A compute $\bigoplus s^A_{\text{pw}_i,i}$ resp. having B send $\bigoplus s^B_{\text{pw}_i,i}$. The computation of A actually requires B to first send pairs $(t_{0,i}, t_{1,i})$ of random s-bit strings and have A compute $\bigoplus(s^A_{\text{pw}_i,i} \oplus t_{\text{pw}_i,i})$ instead. Otherwise a malicious A could choose the $s^A_{b,i}$'s such that multiple passwords map to the same string, e.g., all $s^A_{b,i} = 0$, could then easily pass the authentication step and eventually find out valuable information about pw through well chosen $s^B_{b,i}$'s from B's authentication step.

Figure 3 illustrates our protocol $\mathsf{KE}_{\text{auth}}$ in more detail. For technical reasons, we let a party send a random answer in case of an inconsistent incoming message in the third step or fourth step, and set the session key to a random string of length s. This is necessary because, in case of premature aborts, the UC simulator simulating an execution between honest parties would need to know if the passwords of the two parties match. This information, however, is not available through the ideal functionality $\mathcal{F}_{\text{pwKE}}$.

In fact, the possibility that two honest parties may have distinct passwords, e.g., by mistyping them, also requires the reliable erasure on the sender's side for adaptive corruptions. In case either of the parties gets corrupted, we learn the password of this party, but can only test for equality with the partner's password. That is, we cannot determine the actual password of the partner. However, then the other parties' sent data must still be independent of its password in the simulation, and this is ensured by noting that the sender's internal data lacks the random token inputs for false password bits which the receiver (with a distinct password) has used. For the receiver, this holds by construction, because reading the token at a different position has somewhat erased the other data anyway.

4.2.2 Security

Theorem 4.1 *Protocol KE_{auth} securely realizes $\mathcal{F}_{pwKE}$ in the $\mathcal{F}^{auth}_{OTM}$-hybrid world information-theoretically with adaptive corruptions, assuming reliable erasures and authenticated channels. This holds as long as $s-2\ell=\Omega(\lambda)$ for security parameter λ. The simulator's running time is polynomial in the adversary's running time, the security parameter, and 2^{ℓ}.*

Proof. Fix an arbitrary (unbounded) adversary $\mathcal{A}$ for the real-world protocol $\mathsf{KE}_{\mathrm{auth}}$. We describe an (unbounded) ideal-model simulator Sim running essentially a black-box simulation of $\mathcal{A}$ as a sub routine by feigning an execution in the real-world, using the data it gets in the ideal world. The simulator relays any communication between the adversarial sub routine $\mathcal{A}$ and the environment $\mathcal{Z}$, and since we consider adaptive corruptions, the black-box simulation therefore boils down to provide replies on behalf of (simulated) honest parties and to be able to provide a valid-looking internal state in case of a corruption. In what follows, when we say that an honest party or the simulator "rejects", we mean that the honest party continues running the protocol but sends random values instead of honest responses, ultimately setting the session key to an s-bit random value.

We note that in the simulated hybrid world the simulator has full control over the token functionality and simulates it internally. In particular, it can read any data which the adversary submits to this functionality and can prepare replies as required. We especially make use of the fact that the simulator does not need to explicitly query the token on behalf of an honest receiver (but the simulator then still needs to ignore further adversarial read requests because ideal token would do so as well).

Description of the Simulator. Next, we describe the simulator in detail. At any point in the simulation we consider the point in time in which a corrupt receiver in an execution or the adversary makes the query to the tokens for the last remaining bit position. We then say that the tokens are *depleted*, otherwise the tokens are called *unexhausted.* In case of depletion the simulator possibly has to adapt the (last) token's answer according to previous steps. Note that simulated honest receivers in the hybrid model can be assumed not to query the tokens at all.

Tokens are depleted. If at any point during the simulation the tokens are depleted by a query about bit position i, then the simulator adapts the token's final answer as follows. We note that the simulator already holds candidate values $(s^{A}_{b,i}, s^{B}_{b,i}, s^{K}_{b,i})$ for this token, either because a corrupt sender has put them into the token (and which Sim can simply read off in the hybrid model), or because the simulator has chosen them on behalf of the honest sender (and possibly adapted them before, when the sender or receiver got corrupted).

The simulator first determines the password candidate as $\overline{\mathrm{pw}} := \mathrm{pw}'_1||\ldots||\mathrm{pw}'_\ell$ for the adversary's queries to the token, including the final one pw'_i, and then —if the sender is still honest— tests it by calling $\mathcal{F}_{\mathrm{pwKE}}$ with input $(\texttt{TestPwd}, \mathrm{sid}, P_i, \overline{\mathrm{pw}})$ where P_i is the sender's identity.

- If the test returns "correct guess", then Sim stores this password $\mathrm{pw} = \overline{\mathrm{pw}}$ and $sk = \bigoplus_{i=1}^{\ell} s^K_{\mathrm{pw}_i,i}$ for the values in the token for further use. In particular, if the simulated sender later receives a correct value z then set the session key of the sender to sk via a `NewKey` query about (sid, P_i, sk).

  If the simulator, at that point, has already sent y in this simulated execution on behalf of an honest sender, then Sim resets the entry $s^A_{\mathrm{pw}_i,i}$ to

  $$s^A_{\mathrm{pw}_i,i} = y \oplus \bigoplus_{j=1,j\neq i}^{\ell} (s^A_{\mathrm{pw}_j,j} \oplus t_{\mathrm{pw}_j,j})$$

  such that y matches the sender's answer it would give for the password.

- Suppose that the test returns "wrong guess". Then the simulator not change anything at this point. If not done already, it will later send an independent random value y, and reject any answer z of the receiver.

In either case return the (possibly updated) values $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$.

Sender Corruption. Suppose that the adversary corrupts the sender in a simulated execution. Then the simulator corrupts the sender in the ideal world and learns the input password pw. For any execution with unexhausted tokens in which Sim has already sent y, adapt the value $s^A_{\mathrm{pw}_i,i}$ for unqueried position i as in the case of token depletion. For an honest receiver test the password under the receiver's identity and possibly adapt the value $s^B_{\mathrm{pw}_i,i}$ accordingly, if z has already been sent.

If y has not been sent yet, or the tokens are already depleted, the values remain unchanged. Hand over the password and the internal data (excluding the token inputs for the inverse bit values of pw_i, which have been erased before).

Receiver Corruption. As in the case of sender corruption Sim then learns the actual input password. Test this password also for the sender (if it is still honest). If y has already been sent in an execution with a sender who is still honest, and the password test with the sender returned "correct guess", then adapt the values $s^A_{b,i}$ in $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$ allegedly in the token accordingly, else the values remain unchanged. If, in addition, the simulated receiver has sent z already, then also adapt the values $s^B_{b,i}$. Besides the password, present the adversary the $t_{b,i}$'s (if chosen already) and the entries $(s^A_{\mathrm{pw}_i,i}, s^B_{\mathrm{pw}_i,i}, s^K_{\mathrm{pw}_i,i})$.

Simulating Steps of Honest Parties. We show next how to simulate the other steps on behalf of honest parties.

Simulating Step (1): For an honest sender the simulator picks values $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$ at random and puts them into a token.

Simulating Step (2): If the receiver is honest, then the simulator picks the values $t_{b,i}$ at random but does not query the token. It sends the $t_{b,i}$'s on behalf of the receiver in the simulation.

Simulating Step (3): If the tokens are already depleted and the test query returned "correct guess", then the simulator computes y genuinely according to the (possibly adapted) values $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$ and the $t_{b,i}$'s. If the test returned "wrong guess" or the tokens are unexhausted at this point, then send a random y instead.

Simulating Step (4): In this step the receiver is supposed to verify the sender's value y and to provide its own authentication value z.

Assume that the sender is controlled by the adversary when sending y. Then Sim already knows all values $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$ stored in the token (either because it put them there on behalf of an honest sender before being corrupted, or since the adversary put them into the token right away). The simulator now tries all possible passwords pw, in polynomial time $2^\ell = \text{poly}(\lambda)$, and for each password computes y_{pw} as the honest sender would (given the $s_{b,i}$'s and $t_{b,i}$'s). If none value matches y, then Sim lets the receiver abort; if more than two values match, then Sim aborts the simulation with an error message. Else, the simulator makes a `TestPwd` query about the (only) candidate password pw for sid and the receiver's identity P_j. If this query returns "correct guess", then compute the final message z and the key sk according to this password and make a call $(\texttt{NewKey}, \text{sid}, P_j, sk)$. For the reply "wrong guess" let the receiver abort the execution, i.e., use random data from this point on.

For an honest sender the receiver accepts and sends a random answer z. The simulator calls the functionality about $(\texttt{NewKey}, \text{sid}, P_j, sk_0)$ for an arbitrary key sk_0 in the key space.

Simulating the Sender's Final Step: When the honest sender P_i obtains the final message z from an honest receiver, it accepts, and the simulator calls the functionality about $(\texttt{NewKey}, \text{sid}, P_i, sk_0)$ for an arbitrary key sk_0 in the key space. If the receiver is corrupt at this point and the tokens are unexhausted, then Sim lets the sender reject. If the tokens of the corrupt receiver are depleted at this point, then the simulator has already tested a password. If this test returned "correct guess", then the simulated sender uses the guessed password to compute the final answer and creates the output via $(\texttt{NewKey}, \text{sid}, P_i, sk)$ for the previously derived key. Else, for "wrong guess", the simulator lets the sender reject.

Analysis. We show next that the above simulation through Sim is indistinguishable from a real-world execution, i.e., any (computationally unbounded) environment is unable to differentiate between the protocol execution built up by Sim or a real execution influenced by the adversary $\mathcal{A}$.

We first observe that unexhausted tokens leave the simulator enough freedom to adapt values perfectly, i.e., the distributions of computing the correct value y from random $s^A_{b,i}$ and $t_{b,i}$, and from picking y at random and later setting the "undetermined" value $s^A_{b,j}$ accordingly, are identical. The same is true for the receiver's values. This holds, in particular, in case of corruption of a party where testing the password on the side of the honest partner either yields a correct guess (in which case the values are adapted correctly), or a wrong guess in which case the partner has a distinct password and derives independent random strings. The latter is perfectly indistinguishable if the receiver gets corrupted, because the receiver reads out at least one token at a different position such that the other random string originally stored in the token is not available to the receiver (and eventually to the adversary) anymore. It also holds in case of a sender corruption because then the sender has already erased reliably the other data.

It remains to argue that the simulator's verifications of the values y and z are sound. For two honest parties, since we assume immutable transmissions, and because the ideal functionality takes care of deciding equality of the passwords of the two parties, calling `NewKey` as described provides an almost perfect simulation.

For a malicious partner in the simulation, the difference to the real-world protocol is y could match two possible passwords such that the simulator aborts, but the genuine receiver would accept (in the case that y does not match any of the y_{pw} of possible passwords the receiver in the real-world protocol would also reject). We next bound the probability of two passwords $\mathrm{pw} \neq \mathrm{pw}'$ yielding the same $y_{\mathrm{pw}} = y_{\mathrm{pw}'}$. Note that the values $s^A_{b,i}$ are chosen and put into the token before the $t_{b,i}$'s are sent (either by the simulator or the adversary). Furthermore, the receiver is still honest, such that the $t_{b,i}$'s are random, mutually independent, and independent of the $s^A_{b,i}$. We conclude that the probability (over the $t_{b,i}$'s) that

$$y_{\mathrm{pw}} = \bigoplus_{i=1}^{\ell} (s_{\mathrm{pw}_i,i} \oplus t_{\mathrm{pw}_i,i}) = \bigoplus_{i=1}^{\ell} (s_{\mathrm{pw}'_i,i} \oplus t_{\mathrm{pw}'_i,i}) = y_{\mathrm{pw}'}$$

for fixed $\mathrm{pw} \neq \mathrm{pw}'$, is at most 2^{-s}. Hence, with probability at most $2^{2\ell-s-1}$ there exist some $\mathrm{pw} \neq \mathrm{pw}'$ for which a collision occurs. By the choice of parameters this is exponentially small.

Finally, we note that in the case the tokens are unexhausted, or are depleted but the simulator obtained the reply "wrong guess" for the tested password, the probability that z sent by the adversary would pass the verification step in the real-world protocol, is at most $2^{\ell-s}$ (implying that the simulator's strategy of rejecting is valid except for an exponentially small error). This can be seen as follows. In case the tokens are unexhausted the adversary has not obtained the independent random value $s^B_{b,j}$ for some b, j, such that, for a fixed password, the value z can be correct with probability at most 2^{-s} only. This is also true in case the tokens are depleted but the password was wrong, because then one of the correct values $s^B_{b,i}$ is irrevocably lost for the adversary. Summing over all 2^ℓ passwords yields the claim. $\square$

We note again that the simulator above is polynomial-time in the running time of the adversary and of the number of possible passwords. For passwords of logarithmic length this number is polynomial in the security parameter.

On Efficient Simulators. Both in our basic as well in our advanced protocol the simulator for showing UC security runs in polynomial time in the running time of the adversary and the *size of the password space*. Hence, because human-memorizable passwords are typically short and can be considered to be of logarithmic length, this provides a simulator whose complexity is polynomially related to the adversary's running time and the security parameter. Recall that the goal of the paper is indeed to derive *password-based* protocols.

Still, if one, nonetheless, considers passwords of super-logarithmic length, such as full-fledged cryptographic keys, the running time of our simulator becomes exponential. Whether this now provides the desired level of security or not depends on the definition of statistical (UC) security: If one considers statistical security with respect to the class unbounded algorithms, in the sense that for any potentially unbounded adversary there is a potentially unbounded simulator, then our simulator achieves this goal. If, on the other hand, one requires that the simulator's complexity is also polynomially related to the adversary's running time, then we no longer attain this security level. See the discussions in [11, 12] for more details on the two approaches. Here, we use the more liberal definition and state the runtime of the simulator in terms of the running time of the adversary and the security parameter in the theorems, allowing to impose either definition. We note that the idea of using strong, super-polynomial time simulators has been applied in the UC framework before, usually to overcome set-up assumptions [54, 56, 5, 44, 43]. We are not aware if our protocol can be modified to yield one, which is also secure according to the stronger notion.

4.3 Key Exchange in the $\mathcal{F}_{\mathrm{OTM}}^{insec}$-hybrid Setting

We have shown so far that information-theoretically secure authenticated key exchange is possible using tamper-proof hardware tokens. However, the security relies in the assumptions that the adversary does not tamper with the tokens as man-in-the-middle. Here we show that the previous protocol succumbs to such attacks and then present a method to secure it in the unauthenticated setting.

4.3.1 Attacking our Protocol in the Unauthenticated Setting

We have shown that $\mathsf{KE}_{\mathrm{auth}}$ securely realizes $\mathcal{F}_{\mathrm{pwKE}}$ in the $\mathcal{F}_{\mathrm{OTM}}^{auth}$-hybrid setting information-theoretically. In this section, we provide a successful attack scenario against $\mathsf{KE}_{\mathrm{auth}}$ when the adversary is allowed to act as man-in-the-middle accessing the transmitted tokens and creating new hardware tokens for the same session. Thereby, we conclude that $\mathsf{KE}_{\mathrm{auth}}$ does not realize $\mathcal{F}_{\mathrm{pwKE}}$ in the (unauthenticated) $\mathcal{F}_{\mathrm{OTM}}^{insec}$-hybrid world. Roughly, the problem is that the simulation in the proof of Theorem 4.1 implicitly assumes that, whenever an honest party sends y or z to the honest partner, the simulated honest receiver should accept. This may not be a viable strategy if, as below, the adversary may modify transmissions.

We first describe our attack for simplicity not in the UC framework, but as a protocol attack to recover the password. It is easy to turn this into a successful attack in the UC setting (as described below). The attack works as follows. Let A and B be an honest sender and receiver, respectively. When A sends the prepared tokens to B, the adversary $\mathcal{A}$ intercepts these tokens and queries the i-th token for some $i \in \{1, \ldots, |\mathrm{pw}|\}$ with input $b = 0$. Adversary $\mathcal{A}$ obtains the values $(\mathrm{sid}, s_{0,i}^A, s_{0,i}^B, s_{0,i}^K)$. This token is now exhausted and needs to be replaced by a new token such that B does not notice the modification. This is done by creating a new token $\mathcal{T}_i$ by $\mathcal{A}$ which outputs $(\mathrm{sid}, s_{0,i}^A, s_{0,i}^B, s_{0,i}^K)$ for input bit 0, and some independent random values for the other value. All the tokens are now handed over to B. Henceforth, $\mathcal{A}$ eavesdrops on the acceptance or rejection reaction of the receiver when the value y is sent. This information is usually conveyed to the adversary for free in the BPR model [7] resp. eventually available through the environment in the UC setting, even though in our protocol both parties first continue the execution. If y is accepted, then $\mathcal{A}$ knows that the i-th bit of the password is 0 (with overwhelming probability). Otherwise, the i-th password bit is 1. Hence, the adversary learns one bit of the password and by repeating this attack the adversary can recover the entire password after $|\mathrm{pw}|$ executions between A and B.

Note that this attack is even possible without any modifications on the tokens. For this $\mathcal{A}$ simply does not touch the tokens but instead modifies the value $t_i = (t_{0,i}, t_{1,i})$. More precisely, $\mathcal{A}$ waits until the receiver sends $t_{b,i}$ values and resets $t_{0,i}$ for some $i \in \{1, \ldots, |\mathrm{pw}|\}$ to an independent random value before delivering all $t_{b,i}$ to A. Now, if y sent by A is accepted by B, then $\mathcal{A}$ knows that the i-bit of the password is 1 with overwhelming probability. Otherwise, rejection implies that the i-bit is 0.

We note that (either one of) the above attacks can be easily turned into a successfully distinguishing UC environment. Namely, $\mathcal{Z}$ has the two honest parties use the same password whose first bit is 0 or 1, with probability $\frac{1}{2}$ each. Now it asks the adversary to do the modification as above, and finally checks if both parties agree upon the same key. For an actual protocol execution this will only happen iff the first bit of the password was really 0 and thus with probability $\frac{1}{2}$. In the ideal model, however, both parties would always derive the same key because they hold the same passwords.

4.3.2 The Protocol $\mathsf{KE}_{\mathrm{insec}}$

The idea to make our protocol immune against the above modification attacks is to thwart the adversary from learning individual password bits through one's party reaction. For this we add an authentication step which provides integrity of the tokens and of the $t_{b,i}$ values via the password. It is only necessary to authenticate the transmission from the sender to the receiver, as in case of an invalid authentication code the (honest) receiver will reply with random and thus most likely invalid z anyway.

To authenticate the tokens and the $t_{b,i}$ values we assume an unconditionally secure authentication code like the one of Carter and Wegman [65] which takes an m-bit message M (we assume that m is at least the security parameter, else one may pad messages first) and a key $a = (\alpha, \beta) \in \{0,1\}^{2m}$ and (deterministically) computes $\tau = \mathsf{MAC}(a, M) = \alpha M + \beta$ over $\mathrm{GF}(2^m)$. To verify a MAC one recomputes the MAC and compares it to the given τ. Note that, besides providing a (one-time) unforgeability security level of 2^{-m} against unbounded adversaries, this message authentication code has another useful property: given a random m-bit string τ and an m-bit message M one can easily solve an equation to find a matching key $a = (\alpha, \beta) \in \{0,1\}^{2m}$ with $\tau = \mathsf{MAC}(a, M)$.

We now describe the full protocol $\mathsf{KE}_{\mathrm{insec}}$. We assume the total length of messages $\mathrm{tid}_1, \ldots, \mathrm{tid}_\ell$ and of the $t_{b,i}$ values is at most m. Note that $m = \Omega(s) = \Omega(\lambda)$. As before, for each bit pw_i of the password pw the sender A hands over an OTM token to the receiver B with two tuples of random s-bit strings $(s^A_{b,i}, s^B_{b,i}, s^K_{b,i})$, but this time each tuple is augmented by random $2m$-bit strings $a_{b,i}$ for the MAC. The receiver again reads the tuple corresponding to bit pw_i such that the other tuple is irrevocably lost. It sends strings $(t_{0,i}, t_{1,i})$. Next, both parties as before authenticate themselves using the password by having A compute $\bigoplus(s^A_{\mathrm{pw}_i,i} \oplus t_{\mathrm{pw}_i,i})$ resp. having B send $\bigoplus s^B_{\mathrm{pw}_i,i}$. Only this time A also computes $a_{\mathrm{pw}} = \bigoplus a_{\mathrm{pw}_i,i}$ and sends $\tau = \mathsf{MAC}(a_{\mathrm{pw}}, \mathrm{tid}_1, \ldots, \mathrm{tid}_\ell, t_{0,1}, \ldots, t_{1,\ell})$ in addition to y. The receiver answers with z as before (or a random value z in case the value y or the value τ does not verify).

Figure 4 illustrates our protocol $\mathsf{KE}_{\mathrm{insec}}$ in more detail. Before giving the proof we first discuss why the attack described in the previous section does not work anymore against this protocol. There, the adversary randomly changed a single token tid or value $t_{b,i}$ in order to test for single bit of the password: if the guess was right both parties would continue the execution, else they would reject. Here, any such change would make the MAC over all tid and $t_{i,b}$ invalid and thus make both parties reject any modification. This only option for the adversary is to compute a new valid MAC. But since the key for the MAC is distributed over all tokens, the adversary would either need to forge a MAC without the key —which is improbable— or need to read *all* tokens to potentially get the key. In the latter case the adversary commits to a password guess which can be used by the simulator for a test and for a corresponding reaction.

Sender :		**Receiver** :
$s^A_{b,i}, s^B_{b,i}, s^K_{b,i} \xleftarrow{\$} \{0,1\}^s,\ a_{b,i} \xleftarrow{\$} \{0,1\}^{2m}$ for $b \in \{0,1\}$, $i = 1, \dots, \ell = \lvert\mathrm{pw}\rvert$ For $i = 1 \dots \ell$ generate token $\mathcal{T}_i$ with tid_i: $\mathcal{T}_i$ outputs $(s^A_{b_i,i}, s^B_{b_i,i}, s^K_{b_i,i}, a_{b_i,i})$ on input b_i		abort $\leftarrow$ false
Erase all $(s^A_{b_i,i}, s^B_{b_i,i}, s^K_{b_i,i}, a_{b_i,i})$ for $b_i \neq \mathrm{pw}_i$	$\xrightarrow[(1)]{tid_1, \dots, tid_\ell}$	$(s^A_i, s^B_i, s^K_i, a_i) \leftarrow \mathcal{T}_i(\mathrm{pw}_i)\ \forall i$ If *sid* in *tid* is invalid, then abort $\leftarrow$ true
$a_{\mathrm{pw}} \leftarrow \bigoplus_{i=1}^{\ell} a_{\mathrm{pw}_i,i}$	$\xleftarrow[(2)]{t_{b,i}}$	$t_{b,i} \xleftarrow{\$} \{0,1\}^s$ for $b \in \{0,1\}$ and $\forall i$
$y \leftarrow \bigoplus_{i=1}^{\ell} (s^A_{\mathrm{pw}_i,i} \oplus t_{\mathrm{pw}_i,i})$ $\tau \leftarrow \mathsf{MAC}(a_{\mathrm{pw}}, \mathrm{tid}_1 \dots, \mathrm{tid}_\ell, t_{0,1}, \dots, t_{1,\ell})$	$\xrightarrow[(3)]{y, \tau}$	$a_{\mathrm{pw}} \leftarrow \bigoplus_{i=1}^{\ell} a_i$ If $\tau \neq \mathsf{MAC}(a_{\mathrm{pw}}, \mathrm{tid}_1 \dots, \mathrm{tid}_\ell, t_{0,1}, \dots, t_{1,\ell})$ or $y \neq \bigoplus_{i=1}^{\ell} (s^A_i \oplus t_{\mathrm{pw}_i,i})$ then abort $\leftarrow$ true If abort = false, then $z \leftarrow \bigoplus_{i=1}^{\ell} s^B_i$
	$\xleftarrow[(4)]{z}$	else $z \xleftarrow{\$} \{0,1\}^s$
If $z = \bigoplus_{i=1}^{\ell} s^B_{\mathrm{pw}_i,i}$, then $sk \leftarrow \bigoplus_{i=1}^{\ell} s^K_{\mathrm{pw}_i,i}$ else $sk \xleftarrow{\$} \{0,1\}^s$		If abort = false, then $sk \leftarrow \bigoplus_{i=1}^{\ell} s^K_i$ else $sk \xleftarrow{\$} \{0,1\}^s$
session key:= *sk*		session key:= *sk*

Figure 4: Password-based authenticated key-exchange protocol $\mathsf{KE}_{\mathrm{insec}}$ using OTM tokens

4.3.3 Security

Theorem 4.2 *Protocol KE_{insec} securely realizes $\mathcal{F}_{pwKE}$ in the $\mathcal{F}_{OTM}^{insec}$-hybrid world information-theoretically with adaptive corruptions, assuming reliable erasure. This holds as long as $s - 2\ell = \Omega(\lambda)$. The simulator's running time is polynomial in the adversary's running time, the security parameter, and 2^ℓ.*

Proof. We again assume an adaptive (unbounded) adversary $\mathcal{A}$ and construct a corresponding (unbounded) ideal-model simulator Sɪᴍ via black-box simulation. We note that we can assume that, once the adversary has taken control over a party, that the adversary does not change transmissions from or to that party (over the channel); the adversary could easily make these modifications internally. This, in particular, means that the adversary only modifies transmissions or injects messages in executions between (currently) honest parties.

Description of the Simulator. We use the same notion of depleted and unexhausted tokens as in the proof of Theorem 4.1. It is also easy to see that, analogously to there, the simulator can easily adapt the values in case of a corruption. Here, we use the fact that the authentication code allows to adapt the key accordingly.

Simulation of Honest Parties. It remains to discuss how to simulate honest parties. Since this requires additional steps, compared to the proof of Theorem 4.1, we discuss this in more detail:

Simulating Step (1): For an honest sender the simulator picks values $(s_{b,i}^A, s_{b,i}^B, s_{b,i}^K, a_{b,i})$ at random and puts them into a token.

Simulating Step (2): If the receiver is honest, then the simulator picks the values $t_{b,i}$ at random but does not query the token. It sends the $t_{b,i}$'s on behalf of the receiver in the simulation.

Simulating Step (3): If the tokens are already depleted and the test query returned "correct guess", then the simulator computes y and τ genuinely according to the (possibly adapted) values $(s_{b,i}^A, s_{b,i}^B, s_{b,i}^K, a_{b,i})$ and the $t_{b,i}$'s. If the test returned "wrong guess" or the tokens are unexhausted at this point, then send random y and τ instead.

Simulating Step (4): In this step the receiver is supposed to verify the sender's values y, τ and to provide its own authentication value z. There are four cases:

1. Assume that the sender is controlled by the adversary when sending y, τ (in which case the adversary also delivers these values to the honest receiver by assumption). Then Sim already knows all values $(s_{b,i}^A, s_{b,i}^B, s_{b,i}^K, a_{b,i})$ stored in the token (either because it put them there on behalf of an honest sender before being corrupted, or since the adversary put them into the token right away). The simulator now tries all possible passwords pw and for each password computes y_{pw} as the honest sender would (given the $s_{b,i}$'s and $t_{b,i}$'s). If none value matches y, then Sim lets the receiver abort; if more than two values match, then Sim aborts the simulation with an error message. Else, the simulator makes a `TestPwd` query about the (only) candidate password pw for sid and the receiver's identity P_j. If this query returns "correct guess", then compute a_{pw} according to the protocol and verify τ with this key. If this test also succeeds then compute the final message z and the key *sk* according to this password and make a call $(\texttt{NewKey}, \text{sid}, P_j, sk)$. If the verification of τ fails or the reply is "wrong guess" in the password test, let the receiver send random z and abort the execution.
2. For an honest sender, and if the adversary has not tampered the tokens, i.e., delivered the same tid's as the sender has sent in this session, we have the receiver send a random answer z. If, in addition, the adversary has delivered the same $t_{b,i}$ values sent by the receiver to the sender in this session, and the same reply y, τ from the sender to the receiver, then the simulator calls the functionality about $(\texttt{NewKey}, \text{sid}, P_j, sk_0)$ for an arbitrary key sk_0 in the key space; else, if the adversary has tampered with the $t_{b,i}$ values or the sender's reply, then the simulator does nothing beyond sending a random value, i.e., lets the receiver abort.
3. If the sender is still honest, but the adversary has sent only new tokens $\text{tid}'_1, \ldots, \text{tid}'_\ell \notin \{\text{tid}_1, \ldots, \text{tid}_\ell\}$, then the simulator determines a password candidate from y as in the case of a corrupt sender and proceeds according to this case (as described above).
4. If the sender is honest, but the adversary has forwarded one of the tokens created by the honest sender but also at least one fresh token (in which case the sender's tokens must be unexhausted), then we let the simulated receiver answer with a random z and internally abort.

Simulating the Sender's Final Step: When the honest sender P_i obtains the final message z from a receiver, it does the following:

1. If the receiver is corrupt at this point and the token is unexhausted, then SIM lets the sender simply abort.
2. If the tokens are depleted at this point, independently of the question whether the receiver is honest or malicious, then the simulator has already tested a password. If this test returned "correct guess", then the simulated sender uses the guessed password to verify the value z. If this succeeds, then it creates the output via $(\texttt{NewKey}, \mathrm{sid}, P_i, sk)$ for the previously derived key. Else, if the verification tests fail or the password test has returned "wrong guess", the simulator lets the sender simply abort.
3. If the receiver is still honest and the adversary has simply forwarded all tokens of the sender (in the right order) to the receiver and forwarded also the $t_{b,i}$ values without modifications, and the receiver has not aborted, then the simulator lets the sender accept and calls the functionality about $(\texttt{NewKey}, \mathrm{sid}, P_i, sk_0)$ for an arbitrary key sk_0 in the key space, if and only if the adversary faithfully delivers the receiver's final message z.
4. In any other case the simulator lets the sender abort.

Analysis. We note that the analysis in case of a malicious sender or a malicious receiver is as before. Moreover, if the simulator extracts a password from the adversary and tests it, correctness follows as before. It thus suffices to look at the case of the adversary mounting a man-in-the-middle attack on an execution between honest parties.

First note that, if the adversary simply relays all the data, in particular, without depleting the tokens, then the simulation (basically sending and accepting random values) is perfectly indistinguishable from an actual protocol execution in the real world. This is still true for executions in which the adversary does not tamper with the tokens nor the $t_{b,i}$-values but possibly modifies any of the final two messages. By construction, the corresponding simulated party then aborts, which is what would also happen in the real world (as the messages y, τ, z are deterministically determined from the first two messages). It thus remains to analyze the case that the adversary somehow modifies any of the two first messages.

Assume that the adversary changes one of the tokens (or the order) but still passes one of sender's tokens to the receiver, or faithfully relays the tokens but alters the $t_{b,i}$ values. Then the adversary is completely oblivious about the secret(s) on the sender's and on the receiver's side. (Note that if both parties use distinct passwords, then they also hold distinct secrets). It follows that the adversary cannot make the receiver accept the subsequent message y, τ, except with exponentially small probability, because it would need to forge a MAC for the distinct series of token identifiers or $t_{b,i}$ values (in case of identical secrets on both sides the adversary would need to create another MAC for a new message, in case of distinct passwords on the sender's and on the receiver's side, the adversary would need to forge a MAC from scratch). Hence, our simulation in which the receiver then aborts and sends a random answer, is statistically close to an actual protocol execution.

Note that letting the receiver abort and send random values implies that the honest sender then, too, most likely reacts correctly. That is, the only case in which the sender in a protocol execution should *not* reject is when either the adversary by chance supplies correct values to the sender without having corrupted the receiver (which then happens with negligible probability only by the unforgeability of the MAC), or if the adversary corrupts the receiver after the receiver has sent its final random values, and then replaces the random values by correct values with the help of the now known password. In this case, however, the simulator tests the password candidate and rejects if the guess is correct, but the verification fails (as the actual sender would), or if the guess is incorrect and verification would, then also fail with overwhelming probability in the real world.

Hence, assume now that the adversary creates new tokens $\mathrm{tid}'_1, \ldots, \mathrm{tid}'_\ell$, all distinct from the tokens created by the sender. Then we can think of this as essentially two executions where the adversary in one plays the malicious receiver, and in the other one the malicious sender (but where, depending on the adversary's actions, the data may not be independent). In the latter execution, the adversary-receiver execution, the incoming message y, τ to the receiver allows to determine a password candidate and then proceed as in the real-world case, i.e., the simulation is statistically close. Analogously, in the sender-adversary execution we then act like the real sender would, except that we let the sender immediately reject if the tokens are unexhausted (whereas in the real-world there is a negligible probability that the data z still verifies).

In summary, for any possible adversarial strategy the simulation is statistically close to real-world executions. This proves security. $\square$

5 Conclusion

We view the contribution of this paper as three-fold. First, we raise the issue that, previously, token-based protocols neglected man-in-the-middle adversaries which can intercept and substitute tokens while in transmission. Secondly, we show that one can still design secure protocols based on simple tokens, even when allowing such strong adversaries, and what techniques may be useful. Thirdly, complementing previous feasibility results, we exemplified for password-based key exchange that such protocols can be very efficient and still provide high security standards.

One question our solutions leave open is, if there are key exchange protocols with comparable security, but efficient simulators even for "passwords" of super-logarithmic length. It is tempting to believe that splitting the longer passwords into smaller chunks and then executing our protocol for passwords of logarithmic size would be a viable strategy. The main problem with this approach is that our simulator would need to know where the password chunks of the two parties coincide, and where they differ; in our proof the simulator acts differently in both cases. However, the simulator's password testing for the composed protocol through the ideal functionality only reveals if the long passwords match or do not, but does not give out the required information about the individual chunks. It seems as if using error correction or other types of tokens is not helpful in this regard. The other main question imposed by our paper is if previous results about token-based cryptography are affected by our stronger token model with insecure transmissions.

References

[1] Michel Abdalla, Dario Catalano, Celine Chevalier, and David Pointcheval. Efficient two-party password-based key exchange protocols in the UC framework. In *Topics in Cryptology — CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 335–351. Springer-Verlag, 2008.

[2] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Password-authenticated group key agreement with adaptive security and contributiveness. In Bart Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, vol-

ume 5580 of *Lecture Notes in Computer Science*, pages 254–271, Gammarth, Tunisia, June 21–25, 2009. Springer, Berlin, Germany.

[3] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1997.

[4] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.

[5] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552. IEEE Computer Society, 2005.

[6] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 608–625, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[7] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 2000.

[8] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In Ran Canetti and Juan Garay, editors, *Advances in Cryptology – Proc. CRYPTO 2013*, LNCS. Springer, 2013.

[9] Christina Brzuska, Marc Fischlin, Stefan Katzenbeisser, and Heike Schröder. Physically uncloneable functions in the universal composition framework. In *Advances in Cryptology — Crypto 2011*, Lecture Notes in Computer Science. Springer-Verlag, 2011.

[10] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 51–70, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.

[11] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[12] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 2001.* IEEE Computer Society Press, for an updated version see `eprint.iacr.org`, 2001.

[13] Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In *Public-Key Cryptography (PKC)'12*, Lecture Notes in Computer Science. Springer-Verlag, 2012.

[14] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

[15] Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO*, volume 2729 of *LNCS*, pages 265–281. Springer, 2003.

[16] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for uc secure computation using tamper-proof hardware. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 545–562. Springer-Verlag, 2008.

[17] Benoît Chevallier-Mames, David Naccache, Pascal Paillier, and David Pointcheval. How to disembed a program? In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 441–454, Cambridge, Massachusetts, USA, August 11–13, 2004. Springer, Berlin, Germany.

[18] Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable two-party computation using a minimal number of stateless tokens. Cryptology eprint archive, 2011.

[19] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. Essentially optimal universally composable oblivious transfer. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC 08: 11th International Conference on Information Security and Cryptology*, volume 5461 of *Lecture Notes in Computer Science*, pages 318–335, Seoul, Korea, December 3–5, 2008. Springer, Berlin, Germany.

[20] Ivan Damgård, Jesper Buus Nielsen, and Daniel Wichs. Isolated proofs of knowledge and isolated zero knowledge. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 509–526. Springer-Verlag, 2008.

[21] Ivan Damgård and Alessandra Scafuro. Unconditionally secure and universally composable commitments from physical assumptions. Cryptology ePrint Archive, Report 2013/108, 2013.

[22] Jerome Di-Battista, Jean-Christophe Courrège, Bruno Rouzeyre, Lionel Torres, and Philippe Perdu. When failure analysis meets side-channel attacks. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 188–202, Santa Barbara, California, USA, August 17–20, 2010. Springer, Berlin, Germany.

[23] Ning Ding and Dawu Gu. A general and efficient obfuscation for programs with tamper-proof hardware. In *Information Security Practice and Experience*, volume 6672 of *Lecture Notes in Computer Science*, pages 401–416. 2011.

[24] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Efficient reductions for non-signaling cryptographic primitives. In Serge Fehr, editor, *Information Theoretic Security*, volume 6673 of *Lecture Notes in Computer Science*, pages 120–137. Springer Berlin Heidelberg, 2011.

[25] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 164–181. Springer, 2011.

[26] Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. Implementing resettable uc-functionalities with untrusted tamper-proof hardware-tokens. In *TCC*, LNCS, pages 642–661. Springer, 2013.

[27] EMC^2. RSA SecureID (commercial website). http://www.emc.com/security/rsa-securid.htm.

[28] Marc Fischlin, Benny Pinkas, Ahmad-Reza Sadeghi, Thomas Schneider, and Ivan Visconti. Secure set intersection with untrusted hardware tokens. In *CT-RSA*, Lecture Notes in Computer Science, pages 1–16. Springer-Verlag, 2011.

[29] Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 297–316, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.

[30] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 505–523, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

[31] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 142–159, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.

[32] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, volume 5157 of *LNCS*, pages 39–56. Springer, 2008.

[33] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge pcps, and unconditional cryptography. In *Advances in Cryptology — Crypto 2010*, Lecture Notes in Computer Science, pages 173–190. Springer-Verlag, 2010.

[34] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, Lecture Notes in Computer Science, pages 308–326. Springer-Verlag, 2010.

[35] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 179–197, Melbourne, Australia, December 7–11, 2008. Springer, Berlin, Germany.

[36] Adam Groce and Jonathan Katz. A new framework for efficient password-based authenticated key exchange. In *ACM Conference on Computer and Communications Security*, pages 516–525. ACM, 2010.

[37] Helena Handschuh, Pascal Paillier, and Jacques Stern. Probing attacks on tamper-resistant devices. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 303–315, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Berlin, Germany.

[38] Carmit Hazay and Yehuda Lindell. Constructions of truly practical secure protocols using standard smartcards. In *ACM CCS*, pages 491–500. ACM, 2008.

[39] Dennis Hofheinz, Dominique Unruh, and Jörn Müller-Quade. Universally composable zero-knowledge arguments and commitments from signature cards. *Tatra Mt. Math. Pub.*, pages 93–103, 2007.

[40] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, volume 4515 of *LNCS*, pages 115–128. Springer, 2007.

[41] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310, Providence, RI, USA, March 28–30, 2011. Springer, Berlin, Germany.

[42] Vladimir Kolesnikov. Truly efficient string oblivious transfer using resettable tamper-proof tokens. In *TCC*, Lecture Notes in Computer Science, pages 327–342. Springer-Verlag, 2010.

[43] Huijia Lin and Rafael Pass. Black-box constructions of composable protocols without set-up. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 461–478. Springer, 2012.

[44] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *STOC*, pages 179–188. ACM, 2009.

[45] Yi-Kai Liu. Building one-time memories from isolated qubits. *CoRR*, abs/1304.5007, 2013.

[46] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Phys. Rev. Lett.*, 78(17):3410–3413, Apr 1997.

[47] Paulo Mateus and Serge Vaudenay. On tamper-resistance from a theoretical viewpoint. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 411–428, Lausanne, Switzerland, September 6–9, 2009. Springer, Berlin, Germany.

[48] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, 78(17):3414–3417, Apr 1997.

[49] Tal Moran and Gil Segev. David and goliath commitments: Uc computation for asymmetric parties using tamper-proof hardware. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 527–544. Springer-Verlag, 2008.

[50] Rolf T. Moulton. A practical approach to system security devices. *Computers & Security*, 3:9399, 1984.

[51] Jesper Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology — Crypto 2002*, volume 1442 of *Lecture Notes in Computer Science*, pages 111–126. Springer-Verlag, 2002.

[52] Rafail Ostrovsky, Alessandra Scafuroa, Ivan Visconti, and Akshay Wadia. Universally composable secure computation with (malicious) physically uncloneable functions. In *Advances in Cryptology — Eurocrypt 2013*, Lecture Notes in Computer Science. Springer-Verlag, 2013.

[53] David Oswald and Christof Paar. Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 207–222, Nara, Japan, September 28 – October 1, 2011. Springer, Berlin, Germany.

[54] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany.

[55] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[56] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251. ACM, 2004.

[57] Renato Renner. *Security of Quantum Key Distribution.* PhD thesis, ETH Zurich, 2005. http://arxiv.org/abs/quant-ph/0512258.

[58] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009: 3rd International Conference on Pairing-based Cryptography*, volume 5671 of *Lecture Notes in Computer Science*, pages 231–247, Palo Alto, CA, USA, August 12–14, 2009. Springer, Berlin, Germany.

[59] Ulrich Rührmair and Marten van Dijk. PUFs in security protocols: Attack models and practical security evaluations. In *IEEE Symposium on Security and Privacy*, 2013.

[60] Sergei Skorobogatov. Flash memory 'bumping' attacks. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 158–172, Santa Barbara, California, USA, August 17–20, 2010. Springer, Berlin, Germany.

[61] Sergei Skorobogatov and Christopher Woods. Breakthrough silicon scanning discovers backdoor in military chip. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 23–40, Leuven, Belgium, September 9–12, 2012. Springer, Berlin, Germany.

[62] M. Smid. Integrating the data encryption standard into computer networks. *Communications, IEEE Transactions on*, 29(6):762 – 772, jun 1981.

[63] Dominique Unruh. Universally composable quantum multi-party computation. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 486–505. Springer, May 2010. Preprint on arXiv:0910.2912 [quant-ph].

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

[65] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.