

General Statistically Secure Computation with Bounded-Resetable Hardware Tokens

Nico Döttling^{*,**1}, Daniel Kraschewski^{***2}, Jörn Müller-Quade³, and Tobias Nilges³

¹ Aarhus University, Denmark

² TNG Technology Consulting GmbH, Munich, Germany

³ Karlsruhe Institute of Technology, Germany

Abstract. Universally composable secure computation was assumed to require trusted setups, until it was realized that parties exchanging (untrusted) tamper-proof hardware tokens allow an alternative approach (Katz; EUROCRYPT 2007). This discovery initialized a line of research dealing with two different types of tokens. Using only a single *stateful* token, one can implement general statistically secure two-party computation (Döttling, Kraschewski, Müller-Quade; TCC 2011); though all security is lost if an adversarial token receiver manages to physically reset and rerun the token. *Stateless* tokens, which are secure by definition against any such resetting-attacks, however, do provably not suffice for statistically secure computation in general (Goyal, Ishai, Mahmoody, Sahai; CRYPTO 2010).

We investigate the natural question of what is possible if an adversary can reset a token at most a bounded number of times (e.g., because each resetting attempt imposes a significant risk to trigger a self-destruction mechanism of the token). Somewhat surprisingly, our results come close to the known positive results with respect to non-resettable stateful tokens. In particular, we construct polynomially many instances of statistically secure and universally composable oblivious transfer, using only a constant number of tokens. Our techniques have some abstract similarities to previous solutions, which we grasp by defining a new security property for protocols that use oracle access. Additionally, we apply our techniques to zero-knowledge proofs and obtain a protocol that achieves the same properties as bounded-query zero-knowledge PCPs (Kilian, Petrank, Tardos; STOC 1997), even if a malicious prover may issue *stateful* PCP oracles.

* Supported by European Research Commission Starting Grant no. 279447.

** The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

*** Work done while at Technion, Israel. Supported by the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 – ERC Cryptography and Complexity.

1 Introduction

The model of untrusted tamper-proof hardware was introduced by Katz [38] to circumvent trusted setup assumptions and has proven to be a strong tool for creating cryptographic protocols, especially in the context of universally composable (UC-secure) [6] multi-party computation. In the tamper-proof hardware model, (possibly malicious) parties can create tokens and send them to other parties, who then can interact with the tokens but not access any internal secrets.

In this line of research, there are two different types of tokens considered: *stateful* and *stateless/resettable* tokens. Studies of the latter are usually motivated by so-called *resetting-attacks*, meaning that an adversarial receiver could physically reset a token's internal state (e.g., by cutting off the power supply). Döttling et al. [21] implemented multiple instances of statistically UC-secure oblivious transfer (OT), using only a single stateful token. On the downside, Goyal et al. [28] showed that with any number of stateless tokens statistical OT is impossible, even if one only goes for stand-alone security. In fact, only a few statistically secure protocols based on stateless tokens have been proposed, namely stand-alone secure commitments [28] and a UC-secure variant [16], the latter using bidirectional exchange of polynomially many tokens for each commitment. These positive results are complemented again by [28], showing that unconditional non-interactive commitments cannot be performed by using only stateless tokens. Since all known approaches based on stateful tokens completely break down if only a single resetting attempt is successful, and strong impossibility results hold with respect to arbitrarily resettable tokens, it seems a natural question to ask what is still possible if an a priori bound for successful resettings is known. Therefore, similar in nature to the well-studied problems of bounded leakage [39,24], bounded-resettable zero-knowledge [40,42,47,3], and bounded-query zero-knowledge PCPs [41,35], we propose a bounded-resettable hardware model.

The new model can also be seen as a variant of the PCP model [25,1] or interactive PCP model [37], depending on whether a considered protocol contains direct interaction between the token issuer and the token receiver. The difference to the (interactive) PCP model is that maliciously issued tokens/oracles can be stateful. This seems reasonable, since it is hard to verify that a malicious token is stateless. We show that this weakened version of the PCP model still allows non-interactive zero-knowledge with $O(1)$ rounds of oracle queries and even general (interactive) secure computation. [28] also considered stateful malicious PCP oracles, though without an a priori query bound. They constructed in that model interactive zero-knowledge proofs with non-constant round complexity and showed impossibility of general statistically secure computation. Moreover, the result on ZK-PCPs by Kilian et al. [41] can be made robust against malicious stateful oracles straightforwardly at the cost of issuing polynomially many oracles and having the verifier query each oracle only once. It is not clear, however, if the same result can be obtained by using only a constant amount of oracles.

Our Results. We define a *bounded-resettable hardware model* and achieve in this model to a large extent the known positive results for stateful tokens. This is surprising, because the corresponding stateful-token protocols from the literature are all susceptible to resetting-attacks. We construct

- multiple commitments, based on a single token issued by the commitment sender,
- a single string-commitment, based on one token issued by the commitment receiver,
- multiple OT instances, based on $O(1)$ tokens issued by the OT-sender, and
- a bounded-resettable zero-knowledge proof of knowledge, based on $O(1)$ tokens issued by the prover.

All protocols are statistically UC-secure and efficient. The first commitment protocol can be made non-interactive, sacrificing UC-security against a corrupted sender, remaining statistically binding. The zero-knowledge protocol can be implemented such that the verifier does not communicate with the prover but only with tokens sent by the prover. Moreover, if we assume that even malicious provers can only issue stateless tokens, then all token functionalities in the zero-knowledge protocol can be combined on a single token and we end up with the same result as Kilian et al. [41].

Our Techniques. The main technical difficulty we have to deal with is that a malicious token issuer can store an arbitrarily complicated function as a token. We enforce (to some extent) honest programming by a simple challenge-response protocol. The domain of allowed token functionalities is chosen such that it is a linear space. The token receiver announces a random linear projection and the token issuer has to reveal the token functionality under this projection. The receiver can then check if the token reacts consistently, while learning only part of the function parameters. We put forward an abstract notion of this technique, which we call *oracle validation*. It has previously been used in a more ad-hoc manner by [21,11], though their space of token functions is quite different from ours: They use affine functions that map length- n bit-vectors to $(n \times n)$ -matrices, whereas we use higher-degree univariate polynomials that operate on a large finite field.

The composability proof for one of our commitment schemes also requires some constructive algebraic geometry, namely efficient uniform sampling from large finite varieties [9].

Related Work. The notion of resettable zero-knowledge was introduced by [7]. In this model, a malicious verifier is allowed to reset the prover arbitrarily and rerun the protocol. Constant-round black-box zero-knowledge protocols with resettable provers were only achieved in various public key models where the verifier has to register a public key, like Bare Public Key Model [7,14,46,19], Upperbounded Public Key (UPK) Model [47], Weak Public Key (WPK) Model [42] and Counter Public Key Model [15]. UPK and WPK assume that the amount of resets is a

priori bounded, similar to our model. Barak et al. [2] provided the first construction of a (non-black-box) resettably-sound zero-knowledge argument system, where soundness for a resettable verifier is achieved. This work was later improved [5,13] and generalized to simultaneously resettable zero-knowledge protocols [18,17,10,5,12]. Since then resettability has found its way into general multi-party computation [31,30].

Early works concerning tamper-proof hardware made computational assumptions and assumed stateful tokens [38,27]. This was later relaxed to resettable or stateless tokens [8,23,11] and/or unconditional security [43,29,28,21,16].

2 Preliminaries

2.1 The UC-Framework

We state and prove the security of our protocols in the Universal Composability (UC) framework of Canetti [6]. In this framework security is defined by comparison of a *real model* and an *ideal model*. The protocol of interest Π is running in the former, where an adversary \mathcal{A} coordinates the behavior of all corrupted parties. We assume static corruption, i.e., the adversary \mathcal{A} cannot adaptively change corruption during a protocol run. In the ideal model, which is secure by definition, an ideal functionality \mathcal{F} implements the desired protocol task and a simulator \mathcal{S} tries to mimic the actions of \mathcal{A} . An environment \mathcal{Z} is plugged either to the ideal or the real model and has to guess which model it is actually plugged to. Denote the random variable representing the output of \mathcal{Z} when interacting with the real model by $\text{Real}_{\mathcal{A}}^{\Pi}(\mathcal{Z})$ and when interacting with the ideal model by $\text{Ideal}_{\mathcal{S}}^{\mathcal{F}}(\mathcal{Z})$. Protocol Π is said to UC-implement \mathcal{F} , if for every adversary \mathcal{A} there exists a simulator \mathcal{S} , such that for all environments \mathcal{Z} the distributions of $\text{Real}_{\mathcal{A}}^{\Pi}(\mathcal{Z})$ and $\text{Ideal}_{\mathcal{S}}^{\mathcal{F}}(\mathcal{Z})$ are indistinguishable. Since we aim at statistical security, all entities are computationally unbounded. However, the (expected) runtime complexity of the ideal model has to be polynomial in the runtime complexity of the real model.

2.2 Definitions and Notations

We write $\Delta(x, y)$ for the statistical distance between x and y . The inner product of x, y is denoted as $\langle x | y \rangle$ and their concatenation as $x || y$. By \mathbb{F}_q we denote the finite field with q elements.

We canonically extend the notion of polynomials over a field \mathbb{F} as follows. By $\mathbb{F}^n[X]$ we denote the set of all n -tuples of polynomials $p_1, \dots, p_n \in \mathbb{F}[X]$. Each polynomial $p := (p_1, \dots, p_n) \in \mathbb{F}^n[X]$ can be seen as a function $\mathbb{F} \rightarrow \mathbb{F}^n$, $x \mapsto (p_1(x), \dots, p_n(x))$, whose degree is $\deg(p) := \max_{i=1}^n (\deg(p_i))$. We treat $\mathbb{F}^n[X]$ as an \mathbb{F} -linear vector space in the natural way.

All (close to) standard ideal functionalities for the UC-framework can be found in Appendix A.

Protocol $\Pi_{k\text{-ind}}^{\text{val}}$

Implicitly parametrized by a finite field \mathbb{F} and a dimension $n \in \mathbb{N}$. The sender's input domain consists of all polynomials $p \in \mathbb{F}^n[X]$ of degree at most $k - 1$. The security parameter is $\ell := \log |\mathbb{F}|$.

1. Sender: Let $p \in \mathbb{F}^n[X]$ be the sender's input. Pick $p' \in \mathbb{F}^n[X]$ of degree at most $k - 1$ uniformly at random. Program the oracle such that on input $x \in \mathbb{F}$ it outputs $(p(x), p'(x))$.
2. (a) Receiver: Pick $\lambda \in \mathbb{F}$ uniformly at random and send it to the sender.
(b) Sender: Compute $\tilde{p} := \lambda \cdot p + p'$ and send it to the receiver.
3. Receiver: Let $x \in \mathbb{F}$ be the receiver's input. Input x into the oracle; let (y, y') denote the response.
4. Receiver: Verify that $\deg(\tilde{p}) \leq k - 1$ and $\lambda \cdot y + y' = \tilde{p}(x)$. If so, output y ; otherwise reject.

Fig. 1. Construction of a query-once validation scheme for a k -wise independent oracle.

3 Query-Once Oracle Validation

We introduce now our abstract notion of enforcing honest token programming. Consider a scenario consisting of an honest receiver party, a (possibly) malicious sender party, and an oracle which is arbitrarily programmable by the sender in a setup phase. All entities are computationally unbounded. The security feature we aim at is that the sender has to choose the oracle functionality from some predefined class and otherwise is caught cheating, even though the receiver queries the oracle only once. If this is achieved, we speak of a *query-once oracle validation scheme*. More particularly, such a scheme consists of four stages (for a concrete example protocol, where the domain of allowed functions consists of bounded degree polynomials over a finite field, see Figure 1):

1. The sender programs the oracle.
2. Sender and receiver run an interactive protocol which is independent of the receiver's input.
3. The receiver chooses his input and queries the oracle.
4. The receiver either rejects or produces some output.

Let g denote the sender input and x the receiver input. We require the following properties.

Efficiency: All computations by honest entities have polynomial complexity.

Correctness: If the sender is honest, then the receiver always outputs $g(x)$ and never rejects.

Privacy: The receiver does not learn anything else about g than $g(x)$.

Extractability: Even if the sender is corrupted, an extractor Ext with access to the oracle program \mathbb{T}^* and the message transcript τ of Stage 2 can compute a valid sender input g such that a receiver R with uniformly random input

x with overwhelming probability (taken over the randomness of x and all of R 's and Ext 's random choices) either rejects or outputs $g(x)$. The extractor has to be efficient in the sense that its expected runtime on any input (τ, T^*) is asymptotically bounded by $(\ell \cdot |T^*|)^{O(1)} \cdot \rho^{-1}$, where ℓ is a security parameter, $|T^*|$ is the size of the oracle program, and ρ is R 's accept probability conditioned on τ (still with random x).

Note that $g(x')$ can be information-theoretically reconstructed from the receiver's view for any input x' that matches his oracle query. It follows by the privacy property that his input x must be uniquely determined by his message to the oracle. Thus, w.l.o.g. he just sends x to the oracle.

Next, we show that the oracle validation scheme $\Pi_{k\text{-ind}}^{\text{val}}$ is indeed extractable—efficiency, correctness, and privacy are straightforward to see. The extractor construction is the main ingredient for our upcoming UC proofs.

Lemma 1. *Figure 1 describes an oracle validation scheme. In particular, there exists an extractor Ext , such for every pair (S^*, T^*) of a corrupted sender S^* and a corrupted oracle T^* it holds:*

- Provided arbitrarily rewindable access to T^* and given a transcript $\tau = (\lambda, \tilde{p})$ of the messages between S^* and an honest receiver R (i.e., with uniformly random $\lambda \in \mathbb{F}$), Ext computes a polynomial $p \in \mathbb{F}^n[X]$ of degree at most $k - 1$.
- If R 's input x is uniformly random, then with some overwhelming probability $1 - \rho'$ (taken over the randomness of λ , x , and Ext 's random tape), R either rejects or outputs $p(x)$. In particular, we have a failure probability $\rho' \leq |\mathbb{F}|^{-\Omega(1)}$.
- For every possible transcript τ , the expected number of queries from Ext to T^* is $k \cdot \rho^{-1}$, where ρ is R 's accept probability conditioned on τ and averaged over all inputs $x \in \mathbb{F}$. The rest of Ext 's calculations have an overall time complexity which is polynomial in $n, k, \log |\mathbb{F}|$.

Proof (Sketch). The extractor Ext runs a simple trial-and-error approach. It repeatedly samples a uniformly random oracle input $x \in \mathbb{F}$, until it has found inputs x_1, \dots, x_k such that the corresponding oracle outputs $(y_i, y'_i) := T^*(x_i)$ pass the consistency checks $\lambda \cdot y_i + y'_i \stackrel{?}{=} \tilde{p}(x_i)$. Then, Ext computes and outputs the minimal-degree interpolation polynomial $p \in \mathbb{F}^n[X]$ with $p(x_i) = y_i$ for $i = 1, \dots, k$. Note that we do not enforce pairwise distinctness of x_1, \dots, x_k .

There are two things to show. Firstly, we have to show that R on random input x basically either rejects or produces output $p(x)$. Secondly, we have to estimate the expected number of queries from Ext to T^* . We start with the latter. The sampling of each x_i is a stochastic process with geometric distribution of the number of oracle queries: Given that ρ is R 's accept probability conditioned on some transcript τ , the expected number of queries for sampling one x_i is $\sum_{j=1}^{\infty} j \cdot (1 - \rho)^{j-1} \cdot \rho = \rho^{-1}$. The sampling of x_1, \dots, x_k hence requires $k \cdot \rho^{-1}$ queries to T^* on average.

Next, we turn to the question of how well the extracted polynomial p approximates the functionality of a real protocol run. W.l.o.g., S^* follows a deterministic worst-case strategy and we can consider it as a function that maps each possible challenge $\lambda \in \mathbb{F}$ to a polynomial $\tilde{p}_\lambda \in \mathbb{F}^n[X]$ with $\deg(\tilde{p}_\lambda) \leq k - 1$. Analogously, T^* implements a deterministic function by assumption. Thus, for each combination of λ and x it is fixed whether R finally rejects or not. This defines a relation between challenges λ and oracle inputs x . It can be represented as a bipartite graph, where a left-hand vertex λ is adjacent to a right-hand vertex x if R does not reject the corresponding protocol run. Our proof now boils down to showing that there exists a subset of “bad” edges E' such that

1. uniformly random λ and x are adjacent via a “bad” edge only with negligible probability, namely $|E'|/|\mathbb{F}|^2 \leq |\mathbb{F}|^{-\Omega(1)}$, and
2. after removal of all “bad” edges from the graph, T^* implements on each neighborhood of a possible challenge λ a polynomial function of degree at most $k - 1$.

For the existence proof of E' see the full version [22]. The key observations used there are

- that T^* implements a polynomial function of low degree on the common neighborhood $\mathcal{N}(\lambda) \cap \mathcal{N}(\lambda')$ of any distinct challenges λ, λ' and
- that after removal of only a few edges, our graph decomposes into a disjoint collection of complete bipartite subgraphs.

Once E' is shown to exist, we finally need to argue that the following event has probability $|\mathbb{F}|^{-\Omega(1)}$:

- The receiver does not reject and
- one of the oracle inputs x_1, \dots, x_k sampled by Ext is adjacent via a “bad” edge to the challenge λ given by τ , or $x_i = x_j$ for some $i \neq j$.

This implies that $\rho' \leq |\mathbb{F}|^{-\Omega(1)}$. However, since $|E'|/|\mathbb{F}|^2 \leq |\mathbb{F}|^{-\Omega(1)}$, we already have with probability $1 - |\mathbb{F}|^{-\Omega(1)}$ (taken over the randomness of λ) that the given challenge λ is only adjacent to an $|\mathbb{F}|^{-\Omega(1)}$ -fraction of all inputs $x \in \mathbb{F}$ or λ is adjacent to $|\mathbb{F}|^{\Omega(1)}$ edges of which only an $|\mathbb{F}|^{-\Omega(1)}$ -fraction is “bad”. It follows that the event above has the claimed negligible probability. \square

4 Bounded-Resetable Tamper-Proof Hardware

In this section we define and discuss the ideal functionality for bounded-resetable tamper-proof hardware (q.v. Figure 2). It is a slightly modified version of the $\mathcal{F}_{\text{wrap}}$ -functionality introduced by [38]. The token sender provides a (w.l.o.g., deterministic) Turing machine and the receiver can then run it once on an input word of his choice, staying oblivious of any internal secrets. A malicious receiver can reset the token and query it repeatedly, until some bound q is reached and the functionality does not respond any more. The query bound q models

Functionality $\mathcal{F}_{\text{wrap}}^{\text{b-r}}$

Implicitly parametrized by a query bound q . The variable resets_left is initialized by $\text{resets_left} \leftarrow q - 1$.

Creation:

1. Await an input (**create**, \mathcal{M}, b) from the token issuer, where \mathcal{M} is a deterministic Turing program and $b \in \mathbb{N}$. Then, store (\mathcal{M}, b) and send (**created**) to the adversary.
2. Await a message (**delivery**) from the adversary. Then send (**ready**) to the token receiver.

Execution:

3. Await an input (**run**, w) from the receiver. Run \mathcal{M} on input w . When \mathcal{M} halts without generating output or b steps have passed, send a special symbol \perp to the receiver; else send the output of \mathcal{M} .

Reset (adversarial receiver only):

4. Upon receiving a message (**reset**) from a corrupted token receiver, verify that $\text{resets_left} > 0$. If so, decrease resets_left by 1 and go back to Step 3; otherwise ignore that message.

Fig. 2. The wrapper functionality by which we model bounded-resettable tamper-proof hardware. The runtime bound b is merely needed to prevent malicious token senders from providing a perpetually running program code \mathcal{M} ; it will be omitted throughout the rest of the paper.

an estimation for how often an adversary could reset a token that is meant to shut down for good after the first query. All our protocols rely on q being polynomially bounded in the security parameter and a smaller bound q implies better efficiency.

We stress that tokens are not actually required to contain a state that counts the number of queries. Our definition of $\mathcal{F}_{\text{wrap}}^{\text{b-r}}$ is just the most general way to model *any* kind of token for which an upper bound of resets can be derived. E.g., it suffices that $(1 - \rho)^q$ is negligible, where ρ is an upper bound for the probability that the token successfully self-destructs after a query. As well, the token could try to delete its program \mathcal{M} or make it inaccessible but an adversarial receiver could slow down that process or interrupt the deletion before it is complete, so that several queries are possible before \mathcal{M} becomes finally out of reach for him. One can also imagine that security is only needed for some limited time (which is usually the case for the binding property of commitments) and hence it suffices to estimate the number of queries within this time. The latter seems particularly feasible, because it relies on the minimum possible response time of an honestly generated token.

Further note that our definition can be canonically extended to tokens that can be queried more than once also by honest users. However, our approach has the advantage to be trivially secure against tokens that maliciously change their functionality depending on the input history.

Our model is weaker than the stateful-token model in the sense that no previously known protocol with stateful tokens can tolerate even a single reset. They would be all completely broken. Therefore, none of the known positive results for stateful tokens does carry over to our model (unless $q = 1$). In turn, bounded-resettable tokens can be trivially implemented from unresettable stateful tokens. So, our results are strictly stronger than the corresponding results for stateful tokens. On the other hand, bounded-resettable tokens are strictly more powerful than arbitrarily resettable (i.e., standard stateless) tokens, since non-interactive commitments and statistically secure OT are possible with the former but impossible with the latter.

4.1 Commitments from the Token Sender to the Token Receiver

The basic idea how the token issuer can commit himself to some secret s is quite simple. He stores a random degree- q polynomial p on the token and sends the token together with $r := s + p(0)$ to the receiver. The token lets the receiver evaluate p on arbitrary challenges x , except for $x = 0$. To unveil s , the sender sends a description of p . The scheme is perfectly hiding, because even a corrupted receiver can query the token on at most q inputs, receiving only randomness that is statistically independent of $p(0)$. The scheme is statistically binding, because for any two distinct unveil messages p, p' and a uniformly random token input x it holds with overwhelming probability (namely at least $1 - \frac{q}{|\mathbb{F}|-1}$, where \mathbb{F} is the finite field in which all computations take place) that $p(x) \neq p'(x)$ and thus at least one unveil message will be inconsistent with the receiver's view.

Unfortunately, the scheme as stated above is not UC-secure against a corrupted sender. The reason for this is that the sender simulator must be able to extract the secret s from the token program and the commit message r . If the token is issued honestly and thus implements a degree- q polynomial p , the simulator can evaluate the token code on $q + 1$ different inputs, then reconstruct p , and compute $s = r - p(0)$. However, a maliciously issued token can implement an arbitrarily complicated function, which behaves like a degree- q polynomial only on a vanishing but still non-negligible fraction of inputs. It is at the very least unclear if one can extract the correct polynomial from such a token efficiently. Therefore, we employ our oracle validation scheme from Section 3 to make the token extractable. See Figure 3 for the resulting commitment protocol, which even implements many commitments using only one token.

Lemma 2. *The protocol $\Pi_{\text{COM}}^{\text{s-o}}$ (q.v. Figure 3) UC-implements $\mathcal{F}_{\text{COM}}^{\text{s-o}}$ (q.v. Appendix A.2).*

Proof (Sketch). We start with the case of a corrupted receiver. The main issue in this case is that the simulator has to equivocate commitments in the unveil phase. He can do so by picking polynomials $\hat{p}, \hat{p}' \in \mathbb{F}_{2^t}^n[X]$ such that

Protocol $\Pi_{\text{COM}}^{\text{S-O}}$

Implicitly parametrized by a token query bound q , a commitment number n , and a commitment length ℓ . The security parameter is ℓ . For any vector $v = (v_1, \dots, v_n)$ and $I \subseteq \{1, \dots, n\}$ let $v_I := (v_i)_{i \in I}$.

Setup phase:

1. Sender: Pick two uniformly random polynomials $p, p' \in \mathbb{F}_{2^\ell}^n[X]$ of degree at most q . Program a token \mathbb{T} which on input $x \in \mathbb{F}_{2^\ell} \setminus \{0\}$ outputs $(p(x), p'(x))$ and ignores input $x = 0$. Send \mathbb{T} to the receiver.
2. Receiver: Pick $\lambda \in \mathbb{F}_{2^\ell}$ uniformly at random and send it to the sender.
3. Sender: Compute $\tilde{p} := \lambda \cdot p + p'$ and send it to the receiver.

Commit phase:

4. Sender: Let $s := (s_1, \dots, s_n) \in \mathbb{F}_{2^\ell}^n$ be the sender's input. Send $r := s + p(0)$ to the receiver.
5. Receiver: Input a uniformly random $x \in \mathbb{F}_{2^\ell} \setminus \{0\}$ into \mathbb{T} ; let (y, y') denote the response.

Unveil phase:

6. Sender: Let $I \subseteq \{1, \dots, n\}$ indicate the commitments to be opened. Send (I, p_I) to the receiver.
7. Receiver: If $\deg(\tilde{p}) \leq q$ and $\lambda \cdot y + y' = \tilde{p}(x)$ and $p_I(x) = y_I$, output $\hat{s}_I := r_I - p_I(0)$; else reject.

Fig. 3. Statistically UC-secure commitments where the sender is the token issuer.

- $(\hat{p}(x), \hat{p}'(x)) = (p(x), p'(x))$ for all inputs x on which the simulated token was queried so far,
- $\lambda \cdot \hat{p} + \hat{p}' = \tilde{p}$, $\deg(\hat{p}, \hat{p}') \leq q$, and
- $\hat{p}_I(0) = \hat{s}_I$, where \hat{s}_I is the desired result of the equivocation,

and reprogramming the token such that on input x it now outputs $(\hat{p}(x), \hat{p}'(x))$. The unveil message for equivocating the commitment to \hat{s}_I is just (I, \hat{p}_I) . Since the corrupted receiver can query the token at most q times, this is in his view perfectly indistinguishable from a proper commitment.

Now we show security against a corrupted sender. The simulator has to extract commitments in the unveil phase. He can do so by running the extractor Ext from Lemma 1 on the transcript of the setup phase and with rewindable access to the token code \mathbb{T}^* . The extracted polynomial p allows the simulator to reconstruct the committed secret s from the corrupted sender's commit message r as $s = r - p(0)$. Note that Ext may have exponential runtime, but only needs to be run if by the end of the commit phase it is not already clear that the receiver will reject anyway. Therefore, the simulator must first check that $\lambda \cdot y + y' = \tilde{p}(x)$ and then run Ext only if the check is passed. Since Ext has

complexity $(\ell \cdot |\mathbb{T}^*|)^{O(1)} \cdot \rho^{-1}$, where ρ is just the probability that this check is passed, we end up with an expected simulation complexity of $(\ell \cdot |\mathbb{T}^*|)^{O(1)}$. \square

Remark 1. The commitment scheme $\Pi_{\text{COM}}^{\text{s-o}}$ is statistically binding, even if λ is fixed and known to the sender. This yields a statistically secure non-interactive commitment scheme in the bounded-resetable hardware model, which was proven impossible in the stateless-token model [28].

4.2 Commitments from the Token Receiver to the Token Sender

For a commitment from the token receiver to the token sender we need a slightly more sophisticated approach. As in our previous commitment scheme, the token implements a random degree- q polynomial p . The token receiver can then commit to some secret s by inputting a random x into the token, thus learning $p(x)$, and announcing a commit message that consists of

- a fraction of bits of $p(x)$, say the first quarter of its bit-string representation,
- a 2-universal hash function h , and
- $m := s + h(x)$.

To unveil s , he just needs to announce the used token input x . We briefly sketch now why this scheme is hiding and binding. We start with the latter. Due to the query-bound q , the token acts just like a perfectly random function. Thus, a corrupted commitment sender may only with negligible probability find two distinct unveil messages x, x' such that $p(x)$ and $p(x')$ agree on the first quarter of their bit-string representation. This establishes the binding property. The token issuer, however, learns only several bits of information about x during the commit phase, so that from his view x has still linear entropy afterwards. Since h is a 2-universal hash function, this means that he cannot predict $h(x)$ and thus the commitment is hiding. Still, we need to employ our oracle validation scheme from Section 3 again to make the token extractable, as otherwise we have no UC-security against a corrupted commitment receiver. See Figure 4 for the resulting protocol.

Lemma 3. *The protocol $\Pi_{\text{COM}}^{\text{rev}}$ (q.v. Figure 4) implements \mathcal{F}_{COM} (q.v. Appendix A.1) UC-secure against a corrupted commitment sender.*

Proof (Sketch). We just have to exploit that the simulator sees all token inputs. As the number of token queries by the commitment sender is upper bounded by q , the token acts from his views like a perfectly random function. Hence, with overwhelming probability his announcement of \tilde{y} in the commit phase either corresponds to a unique input x already sent to the token or he is caught cheating in the unveil phase. In the former case, the simulator can find x just by scanning through the token’s input history, compute the correct secret $s = m - \langle h | \sigma(x) \rangle$ and send it to the ideal commitment functionality \mathcal{F}_{COM} . In the other case, the simulator can just send anything to the ideal functionality, because only with negligible probability he might need to unveil it later. \square

Protocol $\Pi_{\text{COM}}^{\text{rev}}$

Implicitly parametrized by a token query bound q and a commitment length ℓ . The security parameter is ℓ . Let $\sigma : \mathbb{F}_{2^{4\ell}} \rightarrow \mathbb{F}_{2^\ell}^4$, $x \mapsto (\sigma_1(x), \dots, \sigma_4(x))$ be the canonical \mathbb{F}_{2^ℓ} -vector space isomorphism.

Setup phase:

1. Receiver: Pick two uniformly random polynomials $p, p' \in \mathbb{F}_{2^{4\ell}}[X]$ of degree at most q and program a token T which on input $x \in \mathbb{F}_{2^{4\ell}}$ outputs $(p(x), p'(x))$. Send T to the sender.
2. Sender: Pick $\lambda \in \mathbb{F}_{2^{4\ell}}$ uniformly at random and send it to the receiver.
3. Receiver: Compute $\tilde{p} := \lambda \cdot p + p'$ and send it to the sender.

Commit phase:

4. Sender: Let $s \in \mathbb{F}_{2^\ell}$ be the sender's input. Input a uniformly random $x \in \mathbb{F}_{2^{4\ell}}$ into the token T ; let (y, y') denote the response. If $\lambda \cdot y + y' = \tilde{p}(x)$ and $\deg(\tilde{p}) \leq q$, pick a uniformly random $h \in \mathbb{F}_{2^\ell}^4$ and compute $m := s + \langle h | \sigma(x) \rangle$ and $\tilde{y} := \sigma_1(y)$ and send (m, h, \tilde{y}) to the receiver; otherwise abort.

Unveil phase:

5. Sender: Send x to the receiver.
6. Receiver: Verify that $\tilde{y} = \sigma_1(p(x))$. If so, output $\hat{s} := m - \langle h | \sigma(x) \rangle$; otherwise reject.

Fig. 4. Statistically UC-secure commitment where the receiver is the token issuer.

Proving UC-security against a corrupted commitment receiver, i.e. providing a simulator that equivocates commitments, is more challenging. Note that even after extracting a polynomial p that approximates the token functionality, it is still nontrivial to find a token input \hat{x} such that the first quarter of bits of $p(\hat{x})$ matches the given commit message (m, h, \tilde{y}) while $m - h(x) = \hat{s}$ for a new secret \hat{s} . This problem can be expressed as a polynomial equation system. Here the efficient algorithm of [9] for sampling random solutions comes into play. (See Appendix B for a brief explanation that all preconditions of [9, Theorem 1.1] are met.) In addition, the simulator has to make sure that the sampled solution \hat{x} is actually possible in the real model: He has to (re)sample \hat{x} until $p(\hat{x})$ agrees with the token functionality and the consistency check in Step 4 of the commit phase of $\Pi_{\text{COM}}^{\text{rev}}$ is passed. See Figure 5 for the detailed simulator description. The resampling of \hat{x} imposes some extra difficulty for the runtime estimation, but we refer to the full version [22] for the technical calculation. Next, we show that our scheme is statistically hiding. This is needed for the UC proof and has further application later in our construction of resettable zero-knowledge.

Lemma 4. *The protocol $\Pi_{\text{COM}}^{\text{rev}}$ is statistically hiding, even if λ is fixed.*

Simulator for a corrupted token issuer that receives commitments

Setup phase: Simulated straightforwardly, using a simulated version of the complete real model where the simulated adversary is wired to the ideal model's environment in the canonical way. Store (λ, \tilde{p}) and the token program T^* sent by the corrupted commitment receiver to the simulated functionality $\mathcal{F}_{\text{wrap}}^{\text{b-r}}$.

Commit phase: Simulated straightforwardly, with random sender input s . Store (m, h, \tilde{y}) .

Unveil phase: If the simulated commitment sender has already aborted, do nothing. Otherwise, upon receiving (opened, \hat{s}) from \mathcal{F}_{COM} replace the stored unveil information x in the simulated sender's memory with \hat{x} , computed by the following equivocation program, and let him then proceed with the protocol.

1. Setup the extractor Ext from Lemma 1 with parameters $\mathbb{F} := \mathbb{F}_{2^{4\ell}}$, $n := 1$, and $k := q + 1$. Provide Ext with the transcript $\tau := (\lambda, \tilde{p})$ and rewindable access to the token code T^* .
2. Start Ext . If Ext queries T^* more than 2^ℓ times, give up; otherwise let p denote Ext 's output.
3. Compute the unique polynomial $p_1 \in \mathbb{F}_{2^\ell}[X_1, \dots, X_4]$ such that $\deg(p_1) \leq \deg(p)$ and $\sigma_1 \circ p = p_1 \circ \sigma$, where “ \circ ” denotes the function composition operator. Then pick a uniformly random solution $\hat{x} \in \mathbb{F}_{2^{4\ell}}$ of the following polynomial equation system, using the efficient algorithm of [9]:

$$\begin{aligned} p_1(\sigma(\hat{x})) &= \tilde{y} \\ \langle h \mid \sigma(\hat{x}) \rangle &= m - \hat{s} \end{aligned}$$

- Resample \hat{x} until $p(\hat{x}) = t(\hat{x})$ and $\tilde{p}(\hat{x}) = \lambda \cdot t(\hat{x}) + t'(\hat{x})$, where $t, t' : \mathbb{F}_{2^{4\ell}} \rightarrow \mathbb{F}_{2^{4\ell}}$ such that $\mathsf{T}^*(\hat{x}) = (t(\hat{x}), t'(\hat{x}))$. Give up, if more than $2^{\sqrt{\ell}}$ iterations are required.
4. Replace x in the simulated sender's memory by \hat{x} .

Fig. 5. Simulator for a corrupted token issuer in the protocol $\Pi_{\text{COM}}^{\text{rev}}$ (q.v. Figure 4).

Proof. Let λ and \tilde{p} be arbitrary but fixed. Let $t, t' : \mathbb{F}_{2^{4\ell}} \rightarrow \mathbb{F}_{2^{4\ell}}$ represent the (possibly) corrupted token functionality in the sense that the token maps $x \mapsto (t(x), t'(x))$. Moreover, let $Z := \mathbb{F}_{2^\ell} \cup \{\perp\}$ and for each $z \in Z$ let M_z denote the set of all token inputs x that lead to a commit message (m, h, \tilde{y}) with $\tilde{y} = z$. I.e., $M_z = \{x \in \mathbb{F}_{2^{4\ell}} \mid \lambda \cdot t(x) + t'(x) = \tilde{p}(x) \wedge \sigma_1(t(x)) = z\}$ for $z \in \mathbb{F}_{2^\ell}$ and $M_\perp = \{x \in \mathbb{F}_{2^{4\ell}} \mid \lambda \cdot t(x) + t'(x) \neq \tilde{p}(x)\}$. For uniformly random $x \in \mathbb{F}_{2^{4\ell}}$ and the corresponding \tilde{y} (meaning that $\tilde{y} = \sigma_1(t(x))$ if $\lambda \cdot t(x) + t'(x) = \tilde{p}(x)$ and else $\tilde{y} = \perp$) it holds:

$$\begin{aligned} \max_{z \in Z} \Pr[x = e(\tilde{y})] &= \mathbb{E}(|M_{\tilde{y}}|^{-1}) = \sum_{z \in Z} \Pr[x \in M_z] \cdot |M_z|^{-1} \\ &= \sum_{z \in Z} \frac{1}{|\mathbb{F}_{2^{4\ell}}|} = 2^{-3\ell} + 2^{-4\ell} \end{aligned}$$

Hence, for uniformly random $u \in \mathbb{F}_{2^\ell}$ we can conclude by the Generalized left-over hash lemma [20, Lemma 2.4]:

$$\begin{aligned} \Delta((\langle h | \sigma(x) \rangle, h, \tilde{y}), (u, h, \tilde{y})) &\leq \frac{1}{2} \sqrt{\max_{e: Z \rightarrow \mathbb{F}_{2^{4\ell}}} \Pr[x = e(\tilde{y})] \cdot |\mathbb{F}_{2^\ell}|} \\ &= \frac{1}{2} \sqrt{2^{-2\ell} + 2^{-3\ell}} < 2^{-\ell} \end{aligned}$$

It directly follows now that the statistical distance between a commitment on any secret s and a commitment on uniform randomness is also upper bounded by $2^{-\ell}$. \square

Corollary 1. *The protocol $\Pi_{\text{COM}}^{\text{rev}}$ implements \mathcal{F}_{COM} (q.v. Appendix A.1) UC-secure against a corrupted receiver. The simulation depicted in Figure 5 is indistinguishable from the real model.*

Proof. Consider the following sequence of experiments.

Experiment 1: This is the real model.

Experiment 2: The same as Experiment 1, except that the commitment sender commits to pure randomness in the commit phase and runs in the unveil phase a complete search over all token inputs to equivocate the commitment to his real input (which requires to reset the token exponentially many times).

Experiment 3: The same as Experiment 2, except that the complete search in the equivocation step is only over token inputs x on which the token functionality $x \mapsto (t(x), t'(x))$ coincides with the mapping $x \mapsto (p(x), t'(x))$, where p denotes the polynomial computed by Ext from the token program and the transcript of the setup phase.

Experiment 4: The ideal model, conditioned on the event that the simulator does not give up.

Experiment 5: This is the ideal model.

Experiment 1 and Experiment 2 are indistinguishable, because the commitment is statistically hiding (Lemma 4). Indistinguishability between Experiment 2 and Experiment 3 follows from the negligibility of Ext’s failure probability ρ' (Lemma 1). Experiment 3 and Experiment 4 are indistinguishable by construction of the simulator—here we need that by [9] one finds solutions for a polynomial equation system that are statistically close to random solutions (cf. Appendix B). Experiment 4 and Experiment 5 are indistinguishable, since the simulator has polynomial expected runtime complexity (see full version [22]) and thus gives up only with negligible probability. \square

5 Multiple OT from $O(1)$ Tokens

5.1 Multiple OT with Combined Abort

We adapt and enhance a protocol idea by [28] for a single OT instance in the stateless-token model. It works as follows. The OT-receiver first commits to his

Protocol $\Pi_{\text{MOT}}^{c\text{-ab}}$

Implicitly parametrized by the number n of single OTs to be implemented. Based upon our commitment schemes $\Pi_{\text{COM}}^{s\text{-o}}$ and $\Pi_{\text{COM}}^{\text{rev}}$ and a statistically secure message authentication scheme MAC, e.g. from [44].

1. Sender: Let $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)})$ be the sender's n OT-inputs. Sample a key k for the message authentication scheme MAC. Commit to the $2n$ values $s_0^{(i)}, s_1^{(i)}$ via $\Pi_{\text{COM}}^{s\text{-o}}$ and prepare a hardware token T_{OT} with the following functionality and send it to the receiver:
 - On input (c, w, τ, σ) , verify that $c \in \{0, 1\}^n$, $\sigma = \text{MAC}_k(\tau)$, and w is a correct $\Pi_{\text{COM}}^{\text{rev}}$ -unveil of c with commit phase transcript τ . If so, return the $\Pi_{\text{COM}}^{s\text{-o}}$ -unveil messages for $s_{c_1}^{(1)}, \dots, s_{c_n}^{(n)}$.
2. Receiver: Let $c = (c_1, \dots, c_n)$ be the receiver's choice bits. Commit to c via $\Pi_{\text{COM}}^{\text{rev}}$.
3. Sender: Take the message transcript τ of Step 2, compute $\sigma = \text{MAC}_k(\tau)$, and send σ to the receiver.
4. Receiver: Let w be the $\Pi_{\text{COM}}^{\text{rev}}$ -unveil message for c . Input (c, w, τ, σ) into T_{OT} ; let (r_1, \dots, r_n) denote the response. Verify that r_1, \dots, r_n are correct unveil messages for the corresponding $\Pi_{\text{COM}}^{s\text{-o}}$ -commitments from Step 1 indexed by c . If so, output the unveiled values; otherwise abort.

Fig. 6. Reduction of multiple OT with combined abort to our commitment protocols.

choice bit. The OT-sender then programs a token T_{OT} and provides it with all his random coins and the message transcript of the commitment protocol. The token implements the following functionality. Upon receiving an unveil message for a bit c , the token checks if the unveil is correct; if so, it will provide an OT output s_c . The token T_{OT} is transferred to the receiver, he unveils to it his choice bit and learns the corresponding OT output.

Since the commitments of [28] in the stateless-token model require the commitment receiver to access some token in the unveil phase, they need the OT-sender to encapsulate tokens into each other. We can circumvent this by our commitment scheme $\Pi_{\text{COM}}^{\text{rev}}$, where the commitment receiver does not access any tokens at all. So far, we can implement one OT instance with two tokens. Now, if we implement many OT instances in parallel the straightforward way, i.e. letting the receiver unveil all his choice bits to the token T_{OT} , we run into trouble: Each of the many OT outputs by T_{OT} can arbitrarily depend on all choice bits. Therefore, we let the sender first commit to the OT outputs via our construction $\Pi_{\text{COM}}^{s\text{-o}}$. The token T_{OT} then merely unveils the requested OT outputs. Still, T_{OT} can abort depending on all the choice bits, but we are fine with this for the moment and deal with it in the next section. Thus, our OT construction implements a flawed version of the ideal multiple-OT functionality, where a corrupted sender can additionally upload a predicate that decides whether the receiver's choice bits are accepted (cf. Appendix A.3). A similar level of security was achieved by [33] in the context of non-interactive secure computation.

There is one further refinement of the protocol, by which we achieve that all tokens can be issued independently of the parties' OT inputs. So far, the program code of T_{OT} depends on the message sent by the OT-receiver for the Π_{COM}^{rev} -commitment on his choice bits. Instead, the token sender can give the receiver an information-theoretic MAC for this message, the receiver can input it together with the unveil message into T_{OT} , and the code of T_{OT} thus needs to depend only on the MAC-key—note that by construction of Π_{COM}^{s-o} , the unveil messages that T_{OT} outputs are independent of the committed secrets. The complete OT protocol is given in Figure 6.

Lemma 5. *The protocol Π_{MOT}^{c-ab} (q.v. Figure 6) UC-implements \mathcal{F}_{MOT}^{c-ab} (q.v. Appendix A.3).*

Proof (Proof-sketch). We first show UC-security against a corrupted OT-receiver. In this case, the simulator can fake a real protocol run, exploiting extractability of Π_{COM}^{rev} -commitments and equivocality of Π_{COM}^{s-o} -commitments. The simulation basically works as follows. Step 1 of Π_{MOT}^{c-ab} is simulated straightforwardly with random input for the simulated sender. In Step 2, the corrupted receiver's choice bits (c_1, \dots, c_n) can be extracted (using the sender simulator for Π_{COM}^{rev}) and sent to the ideal functionality \mathcal{F}_{MOT}^{c-ab} . Then, Step 3 again is simulated straightforwardly. Finally, in Step 4, the unveil messages output by the simulated token T_{OT} are replaced (using the receiver simulator for Π_{COM}^{s-o}) such that the commitments from Step 1 are equivocated to the OT-outputs $\hat{s}_{c_1}^{(1)}, \dots, \hat{s}_{c_n}^{(n)}$ received from \mathcal{F}_{MOT}^{c-ab} . Indistinguishability of the simulation from the real model follows from the UC-security of Π_{COM}^{s-o} and Π_{COM}^{rev} and the unforgeability of the message authentication scheme MAC.

Next, we show UC-security against a corrupted OT-sender. The simulator works as follows. In Step 1 of Π_{MOT}^{c-ab} , the corrupted sender's OT inputs $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)})$ can be extracted (using the sender simulator for Π_{COM}^{s-o}). Step 2 and Step 3 of Π_{MOT}^{c-ab} are simulated straightforwardly with random input for the simulated receiver. Then the simulator has to send the extracted OT inputs $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)})$ together with an abort predicate Q to the ideal functionality \mathcal{F}_{MOT}^{c-ab} . The predicate Q is defined by the following program, parametrized with the Π_{COM}^{s-o} -commitments and the token code T_{OT}^* obtained in Step 1 of Π_{MOT}^{c-ab} , the transcript τ of Step 2, and σ from Step 3:

1. Upon input $c \in \{0, 1\}^n$, use the receiver simulator for Π_{COM}^{rev} to obtain an unveil message \hat{w} that equivocates τ to c .
2. Run T_{OT}^* on input $(c, \hat{w}, \tau, \sigma)$; let (r_1, \dots, r_n) denote the response.
3. Simulate the check in Step 4 of Π_{MOT}^{c-ab} , i.e., verify that r_1, \dots, r_n are correct unveil messages for the corresponding Π_{COM}^{s-o} -commitments indexed by c . If so, accept; otherwise reject.

Indistinguishability of the simulation from the real model just follows from the UC-security of Π_{COM}^{s-o} and Π_{COM}^{rev} . \square

Remark 2. Though stated as a three-token construction, our protocol Π_{MOT}^{c-ab} can as well be implemented with two tokens, if one allows a token to be queried

twice. In particular, the token T_{OT} gets with w a complete transcript of the messages sent to the token used in the subprotocol $\Pi_{\text{COM}}^{\text{rev}}$ anyway. Hence, even if maliciously issued tokens can keep a complex state, it does not compromise security if these two tokens are combined into one query-twice token.

5.2 How to Get Rid of the Combined-Abort Flaw

The question of how to implement ideal oblivious transfer from the flawed version $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ is closely related to the research field of OT combiners. However, an OT combiner needs access to *independent* OT instances, some of which may be corrupted. In contrast, $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ leaks a predicate over the receiver’s *joint* inputs for the multiple OT instances. Therefore we need an OT extractor, as defined in [34], rather than an OT combiner. However, the scope of [34] is skew to ours. They consider semi-honest parties, which follow the protocol, and only the leakage function is chosen maliciously. In this setting they aim at a constant extraction rate. In contrast, we consider malicious parties that may try to cheat in the extraction protocol, but we do not care much about the rate. For our purpose it suffices to implement n ideal OT instances from $n^{O(1)}$ flawed instances.

Our solution follows the basic idea of [36] to take an *outer protocol* with many parties and emulate some of the parties by an *inner protocol*, such that the security features of both protocols complement each other. However, before we describe our solution, we briefly sketch why a more classic OT combiner based on 2-universal hashing would be insecure in our case. Such combiners are usually built such that the receiver’s ideal-OT choice bits are basically 2-universal hash values of his flawed-OT inputs, which are uniformly random, and similarly for the outputs. In the $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ -hybrid model, such an approach is susceptible to the following generic attack. The sender just follows the protocol, except that he randomly chooses two of his $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ -input tuples, say $(\tilde{s}_0^{(i)}, \tilde{s}_1^{(i)})$ and $(\tilde{s}_0^{(j)}, \tilde{s}_1^{(j)})$, and flips the bits of $\tilde{s}_1^{(i)}$ and $\tilde{s}_1^{(j)}$. Furthermore, he defines the abort predicate Q such that it rejects the receiver’s choice bits $\tilde{c} := (\tilde{c}_1, \tilde{c}_2, \dots)$ if and only if $\tilde{c}_i = \tilde{c}_j = 1$. This attack has the following effect. With non-negligible probability, the 2-universal hash functions are chosen such that

- the receiver’s i -th flawed-OT input-output tuple $(\tilde{c}_i, \tilde{r}_i)$ influences the calculation of an ideal-OT input-output tuple (c_k, r_k) , but not (c_l, r_l) , where l is an index such that
- the receiver’s j -th flawed-OT input-output tuple $(\tilde{c}_j, \tilde{r}_j)$ influences (c_l, r_l) , but not (c_k, r_k) .

In such a case, it happens with probability $\frac{1}{2}$ that (c_k, r_k) is affected by the bit-flip of $\tilde{s}_1^{(i)}$, namely if $\tilde{c}_i = 1$. Likewise, (c_l, r_l) is affected by the bit-flip of $\tilde{s}_1^{(j)}$ if $\tilde{c}_j = 1$. Both events are statistically independent of each other, but by definition of the abort predicate Q it will never happen that the receiver produces regular output while (c_k, r_k) and (c_l, r_l) are both affected by the attack. This correlation between the joint distribution of the receiver’s ideal-OT inputs and outputs and the event of an abort is not simulatable with an ideal OT.

Our construction is at an abstract level very similar to the OT combiner of [32]. We believe that the constructions of [32] can also be proven secure when based on $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$, but we prefer to present a simple combination of results from the literature as opposed to tampering with the proof details. Our final OT construction consists of three ingredients:

1. Our $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ implementation,
2. a construction for general, statistically UC-secure two-party computation in the OT-hybrid model, e.g. from [36], and
3. a statistically UC-secure protocol for multiple OT based on a single untrusted stateful tamper-proof hardware token, which we take from [21].

We take the token functionality from [21] and implement it by secure two-party computation from [36], based on $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ instead of ideal OT. Note that OT can be stored and reversed [4,45] and therefore it suffices to query $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ just once in the beginning with the token receiver being also the OT-receiver. The “emulated token” then replaces all token queries in the otherwise unchanged protocol of [21]. Now, any specification of the abort predicate in $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$ directly corresponds to a maliciously programmed token that stops functioning depending on its inputs. Since the construction of [21] is UC-secure against any malicious token behavior, we finally obtain UC-secure OT.

Remark 3. Notice that this directly provides an impossibility result for commitments in the stateless-token model where the unveil phase consists only of a single message from the sender to the receiver and local computations (without accessing any tokens) by the receiver. Otherwise our OT construction could be implemented in the stateless-token model (without encapsulation), contradicting the impossibility result for OT given in [28].

6 Bounded-Resetable Zero-Knowledge Proofs of Knowledge

We modify the constant-round zero-knowledge protocol of [26] for 3-COLOR such that the prover becomes resettable and only two tokens have to be sent to the verifier. In the protocol of [26], the verifier first commits to his challenge (the edges determining the vertices that are to be revealed), then the prover commits to permutations of the colored vertices. The verifier then reveals the challenge and the prover opens the specified commitments. The main problem imposed by a *resettable* prover is that a malicious verifier could try to run the same protocol several times, each time with different challenges, and hence step by step learn the prover’s witness. The standard technique to deal with this is to let the prover’s color permutations depend on the verifier’s commitment in a pseudorandom way. Since we aim for *statistical* zero-knowledge, we cannot use a pseudorandom function, but need to replace it by a random polynomial of sufficient degree.

For our construction, we replace the computational commitments in [26] with the statistical commitments presented in the previous sections. Though, our

Protocol Π_{SZK}^{b-r}

Implicitly parametrized by a simple 3-colorable graph $G = (V, E)$ and a query bound q in the sense that a malicious verifier can reset the prover at most $q - 1$ times. Let $n := |V|$, $t := n \cdot |E|$, and $V := \{1, \dots, n\}$.

Auxiliary input for prover: A 3-coloring of G , denoted $\varphi : V \rightarrow \{1, 2, 3\}$.

Setup phase:

- Prover: Select a random degree- q polynomial $f \in \mathbb{F}_{2^l}[X]$, where l is the number of random bits needed for token generation in Π_{COM}^{s-o} for $n \cdot t$ commitments. Further, select a random degree- q polynomial $g \in \mathbb{F}_{2^k}[X]$, where k is the number of random bits needed to generate t random permutations over $\{1, 2, 3\}$. W.l.o.g., l and k are larger than the commit message length in $\tilde{\Pi}_{COM}^{rev}$. Create two tokens T^{rev} and T^{s-o} with the following functionalities and send them to the receiver:
 - T^{rev} : Just implement the token functionality of $\tilde{\Pi}_{COM}^{rev}$.
 - T^{s-o} : Upon input (x, c^{rev}) , simulate the token generation procedure of Π_{COM}^{s-o} with randomness $f(c^{rev}||0\dots 0)$, evaluate the generated token program on input x , and output the result.

Proof phase:

1. Verifier: Uniformly and independently select a random value λ^{s-o} according to the setup phase of Π_{COM}^{s-o} and a t -tuple of edges $\bar{E} = (\{u_1, v_1\}, \dots, \{u_t, v_t\})$ as a challenge for the zero-knowledge proof. Use $\tilde{\Pi}_{COM}^{rev}$ to commit to (\bar{E}, λ^{s-o}) and send the corresponding commit message c^{rev} to the prover.
2. Prover: Compute $r = f(c^{rev}||0\dots 0)$ and $r' = g(c^{rev}||0\dots 0)$. Use r' to select t random permutations π_1, \dots, π_t over $\{1, 2, 3\}$ and set $\phi_i(v) = \pi_i(\varphi(v))$ for each $v \in V$ and $i \in \{1, \dots, t\}$. Use r to simulate the token generation of Π_{COM}^{s-o} and compute the corresponding Π_{COM}^{s-o} -commit message c^{s-o} to commit to $\phi_i(v)$ for all $v \in V$ and $i \in \{1, \dots, t\}$. Send c^{s-o} to the verifier.
3. Verifier: Send c^{rev} and the corresponding $\tilde{\Pi}_{COM}^{rev}$ -unveil message to the prover, thus unveiling (\bar{E}, λ^{s-o}) .
4. Prover: If the unveil was not correct, abort. Else, compute $r = f(c^{rev}||0\dots 0)$ and simulate the token generation of Π_{COM}^{s-o} as in Step 2. Compute the response \tilde{p}^{s-o} for λ^{s-o} according to the setup phase of Π_{COM}^{s-o} . Let w^{s-o} be the Π_{COM}^{s-o} -unveil message for the commitments indexed by \bar{E} . Send $(\tilde{p}^{s-o}, w^{s-o})$ to the verifier.
5. Verifier: Check the unveiled commitments according to the unveil phase of Π_{COM}^{s-o} . Also verify for each edge $\{u_i, v_i\} \in \bar{E}$ that $\phi_i(u_i) \neq \phi_i(v_i)$. If all checks are passed, accept the proof; if not, reject.

Fig. 7. Construction of a bounded-resetable statistical zero-knowledge proof of knowledge.

commitment schemes have an interactive setup phase and become insecure if the token issuer is resetable. However, by fixing λ in the setup phase of Π_{COM}^{rev} , the resulting commitment scheme $\tilde{\Pi}_{COM}^{rev}$ becomes resetable and remains statis-

tically hiding (cf. Lemma 4). Making all the prover’s random choices dependent on the verifier’s first $\tilde{\Pi}_{\text{COM}}^{\text{rev}}$ -commit message c^{rev} is the lever we use to obtain resettability. This particularly has to include the randomness used in $\Pi_{\text{COM}}^{\text{s-o}}$ for token generation. Therefore, we need to use a modified token in $\Pi_{\text{COM}}^{\text{s-o}}$ with input domain $X \times C$, where C is the set of all possible commit messages c^{rev} in $\tilde{\Pi}_{\text{COM}}^{\text{rev}}$ and X is the input space for the original token program in $\Pi_{\text{COM}}^{\text{s-o}}$. On input (x, c^{rev}) , the modified $\Pi_{\text{COM}}^{\text{s-o}}$ -token first simulates the token generation of $\Pi_{\text{COM}}^{\text{s-o}}$ with randomness c^{rev} and then runs the generated token program on input x . See Figure 7 for all further details.

Lemma 6. *The protocol $\Pi_{\text{SZK}}^{\text{b-r}}$ UC-implements \mathcal{F}_{ZK} (q.v. Appendix A.4).*

Proof (Sketch). We first show UC-security against a corrupted verifier, i.e., the simulator must fake a protocol run without knowing a witness. In Step 1 of $\Pi_{\text{SZK}}^{\text{b-r}}$, we exploit that $\tilde{\Pi}_{\text{COM}}^{\text{rev}}$ is still UC-secure against a corrupted commitment sender and thus the challenge \bar{E} can be extracted. Then, in Step 2, the simulated prover can commit to different colorings for each challenged vertex pair $\{u_i, v_i\} \in \bar{E}$ and to arbitrary colorings otherwise. The remaining protocol is just simulated straightforwardly. Indistinguishability from a real protocol run follows, because $\Pi_{\text{COM}}^{\text{s-o}}$ is statistically hiding.

We move on to show UC-security against a corrupted prover, i.e., the simulator has to extract a witness. The complete simulation just follows the real protocol. If in the end the simulated verifier accepts, the sender simulator for $\Pi_{\text{COM}}^{\text{s-o}}$ (provided with the corresponding message transcript and the token code $\mathbb{T}^{\text{s-o}}$) is used to extract the commitments from Step 2 of $\Pi_{\text{SZK}}^{\text{b-r}}$, which yields t colorings for the graph G . If none of them is a valid 3-coloring, the simulator gives up; otherwise he sends a valid one to the ideal functionality \mathcal{F}_{ZK} . It remains to show that the simulator gives up only with negligible probability. However, if none of the committed colorings is a valid 3-coloring, then the proof is accepted by the simulated verifier at most with the following probability (abstracting from the negligible case that some commitment is successfully broken by the corrupted prover):

$$\begin{aligned} \left(1 - \frac{1}{|\bar{E}|}\right)^t &= \left(1 - \frac{1}{|\bar{E}|}\right)^{n \cdot |\bar{E}|} = \exp\left(n \cdot |\bar{E}| \cdot \log\left(1 - \frac{1}{|\bar{E}|}\right)\right) \\ &\leq \exp\left(n \cdot |\bar{E}| \cdot \left(-\frac{1}{|\bar{E}|}\right)\right) = \exp(-n) \quad \square \end{aligned}$$

Remark 4. Furthermore, $\Pi_{\text{SZK}}^{\text{b-r}}$ is bounded-resettably zero-knowledge. The resettability of the prover follows from two facts. Firstly, the prover’s randomness (r, r') depends deterministically but otherwise unpredictable by the verifier on his first message c^{rev} . Secondly, by the binding property of $\tilde{\Pi}_{\text{COM}}^{\text{rev}}$, a corrupted verifier cannot cheat in Step 3 of $\Pi_{\text{SZK}}^{\text{b-r}}$ other than switch to another instance of the zero-knowledge protocol with unrelated prover randomness (r, r') .

Remark 5. Our construction $\Pi_{\text{SZK}}^{\text{b-r}}$ can directly be used to obtain a *non-interactive* zero-knowledge proof of knowledge scheme in the bounded-resettable hardware model by storing the prover functionality in another token (or two other tokens, if each token should be queried only once).

References

1. Arora, S., Safra, S.: Probabilistic checking of proofs; a new characterization of NP. In: Foundations of Computer Science - Proceedings of FOCS 1992. pp. 2–13. IEEE Computer Society (1992)
2. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resetably-sound zero-knowledge and its applications. In: Foundations of Computer Science - Proceedings of FOCS 2001. pp. 116–125. IEEE Computer Society (2001)
3. Barak, B., Lindell, Y., Vadhan, S.P.: Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.* 72(2), 321–391 (2006)
4. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) *Advances in Cryptology - Proceedings of CRYPTO '95*. Lecture Notes in Computer Science, vol. 963, pp. 97–109. Springer (1995)
5. Bitansky, N., Paneth, O.: On the impossibility of approximate obfuscation and applications to resettable cryptography. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) *Symposium on Theory of Computing - Proceedings of STOC 2013*. pp. 241–250. ACM (2013)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Foundations of Computer Science - Proceedings of FOCS 2001. pp. 136–145. IEEE Computer Society (2001), revised full version online available at <http://eprint.iacr.org/2000/067>
7. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: Yao, F.F., Luks, E.M. (eds.) *Symposium on Theory of Computing - Proceedings of STOC 2000*. pp. 235–244. ACM (2000)
8. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2008*. Lecture Notes in Computer Science, vol. 4965, pp. 545–562. Springer (2008)
9. Cheraghchi, M., Shokrollahi, A.: Almost-uniform sampling of points on high-dimensional algebraic varieties. In: Albers, S., Marion, J.Y. (eds.) *Symposium on Theoretical Aspects of Computer Science - Proceedings of STACS 2009*. LIPIcs, vol. 3, pp. 277–288. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany (2009)
10. Cho, C., Ostrovsky, R., Scafuro, A., Visconti, I.: Simultaneously resettable arguments of knowledge. In: Cramer, R. (ed.) *Theory of Cryptography - Proceedings of TCC 2012*. Lecture Notes in Computer Science, vol. 7194, pp. 530–547. Springer (2012)
11. Choi, S.G., Katz, J., Schröder, D., Yerukhimovich, A., Zhou, H.S.: (Efficient) universally composable oblivious transfer using a minimal number of stateless tokens. In: Lindell, Y. (ed.) *Theory of Cryptography - Proceedings of TCC 2014*. Lecture Notes in Computer Science, vol. 8349, pp. 638–662. Springer (2014)
12. Chung, K.M., Ostrovsky, R., Pass, R., Visconti, I.: Simultaneous resettable security from one-way functions. In: Foundations of Computer Science - Proceedings of FOCS 2013. pp. 60–69. IEEE Computer Society (2013)
13. Chung, K.M., Pass, R., Seth, K.: Non-black-box simulation from one-way functions and applications to resettable security. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) *Symposium on Theory of Computing - Proceedings of STOC 2013*. pp. 231–240. ACM (2013)
14. Crescenzo, G.D., Persiano, G., Visconti, I.: Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In: Franklin, M.K.

- (ed.) *Advances in Cryptology - Proceedings of CRYPTO 2004*. Lecture Notes in Computer Science, vol. 3152, pp. 237–253. Springer (2004)
15. Crescenzo, G.D., Persiano, G., Visconti, I.: Improved setup assumptions for 3-round resetttable zero knowledge. In: Lee, P.J. (ed.) *Advances in Cryptology - Proceedings of ASIACRYPT 2004*. Lecture Notes in Computer Science, vol. 3329, pp. 530–544. Springer (2004)
 16. Damgård, I., Scauro, A.: Unconditionally secure and universally composable commitments from physical assumptions. In: Sako, K., Sarkar, P. (eds.) *Advances in Cryptology - Proceedings of ASIACRYPT 2013*. Lecture Notes in Computer Science, vol. 8270, pp. 100–119. Springer (2013)
 17. Deng, Y., Feng, D., Goyal, V., Lin, D., Sahai, A., Yung, M.: Resetttable cryptography in constant rounds - the case of zero knowledge. In: Lee, D.H., Wang, X. (eds.) *Advances in Cryptology - Proceedings of ASIACRYPT 2011*. Lecture Notes in Computer Science, vol. 7073, pp. 390–406. Springer (2011)
 18. Deng, Y., Goyal, V., Sahai, A.: Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In: *Foundations of Computer Science - Proceedings of FOCS 2009*. pp. 251–260. IEEE Computer Society (2009)
 19. Deng, Y., Lin, D.: Resetttable zero knowledge with concurrent soundness in the bare public-key model under standard assumption. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) *Information Security and Cryptology - Proceedings of Inscrypt 2007*. Lecture Notes in Computer Science, vol. 4990, pp. 123–137. Springer (2008)
 20. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* 38(1), 97–139 (2008)
 21. Döttling, N., Kraschewski, D., Müller-Quade, J.: Unconditional and composable security using a single stateful tamper-proof hardware token. In: Ishai, Y. (ed.) *Theory of Cryptography - Proceedings of TCC 2011*. Lecture Notes in Computer Science, vol. 6597. Springer (2011), extended full version online available at <http://eprint.iacr.org/2012/135>
 22. Döttling, N., Kraschewski, D., Müller-Quade, J., Nilges, T.: General statistically secure computation with bounded-resetttable hardware tokens. *IACR Cryptology ePrint Archive 2014*, Report 555 (2014), <http://eprint.iacr.org/2014/555>
 23. Döttling, N., Mie, T., Müller-Quade, J., Nilges, T.: Implementing resetttable UC-functionalities with untrusted tamper-proof hardware-tokens. In: *Theory of Cryptography - Proceedings of TCC 2013* (2013)
 24. Dziembowski, S., Kazana, T., Wichs, D.: Key-evolution schemes resilient to space-bounded leakage. In: Rogaway, P. (ed.) *Advances in Cryptology - Proceedings of CRYPTO 2001*. Lecture Notes in Computer Science, vol. 6841, pp. 335–353. Springer (2011)
 25. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Approximating clique is almost NP-complete (preliminary version). In: *Foundations of Computer Science - Proceedings of FOCS 1991*. pp. 2–12. IEEE Computer Society (1991)
 26. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology* 9(3), 167–190 (1996)
 27. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) *Advances in Cryptology - Proceedings of CRYPTO 2008*. Lecture Notes in Computer Science, vol. 5157, pp. 39–56. Springer (2008)
 28. Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In: Rabin, T. (ed.) *Advances in Cryptology - Proceedings of CRYPTO 2010*. Lecture Notes in Computer Science, vol. 6223, pp. 173–190 (2010)

29. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) *Theory of Cryptography - Proceedings of TCC 2010*. Lecture Notes in Computer Science, vol. 5978, pp. 308–326. Springer (2010)
30. Goyal, V., Maji, H.K.: Stateless cryptographic protocols. In: Ostrovsky, R. (ed.) *Foundations of Computer Science - Proceedings of FOCS 2011*. pp. 678–687. IEEE (2011)
31. Goyal, V., Sahai, A.: Resetably secure computation. In: Joux, A. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2009*. Lecture Notes in Computer Science, vol. 5479, pp. 54–71. Springer (2009)
32. Harnik, D., Ishai, Y., Kushilevitz, E., Nielsen, J.B.: OT-combiners via secure computation. In: Canetti, R. (ed.) *Theory of Cryptography - Proceedings of TCC 2008*. Lecture Notes in Computer Science, vol. 4948, pp. 393–411. Springer (2008)
33. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2011*. Lecture Notes in Computer Science, vol. 6632, pp. 406–425. Springer (2011)
34. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Extracting correlations. In: *Foundations of Computer Science - Proceedings of FOCS 2009*. pp. 261–270. IEEE Computer Society (2009)
35. Ishai, Y., Mahmoody, M., Sahai, A.: On efficient zero-knowledge PCPs. In: Cramer, R. (ed.) *Theory of Cryptography - Proceedings of TCC 2012*. Lecture Notes in Computer Science, vol. 7194, pp. 151–168. Springer (2012)
36. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) *Advances in Cryptology - Proceedings of CRYPTO 2008*. pp. 572–591. Lecture Notes in Computer Science, Springer (2008)
37. Kalai, Y.T., Raz, R.: Interactive PCP. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *Automata, Languages, and Programming - Proceedings of ICALP 2008, Part II*. Lecture Notes in Computer Science, vol. 5126, pp. 536–547. Springer (2008)
38. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2007*. Lecture Notes in Computer Science, vol. 4515. Springer (2007)
39. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) *Advances in Cryptology - Proceedings of ASIACRYPT 2009*, Lecture Notes in Computer Science, vol. 5912, pp. 703–720. Springer (2009)
40. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In: Vitter, J.S., Spirakis, P.G., Yannakakis, M. (eds.) *Symposium on Theory of Computing - Proceedings of STOC 2001*. pp. 560–569. ACM (2001)
41. Kilian, J., Petrank, E., Tardos, G.: Probabilistically checkable proofs with zero knowledge. In: Leighton, F.T., Shor, P.W. (eds.) *Symposium on Theory of Computing - Proceedings of STOC 1997*. pp. 496–505. ACM (1997)
42. Micali, S., Reyzin, L.: Min-round resettable zero-knowledge in the public-key model. In: Pfitzmann, B. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2001*. Lecture Notes in Computer Science, vol. 2045, pp. 373–393. Springer (2001)
43. Moran, T., Segev, G.: David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2008*, Lecture Notes in Computer Science, vol. 4965, pp. 527–544. Springer (2008)

44. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.* 22(3), 265–279 (1981)
45. Wolf, S., Wullschleger, J.: Oblivious transfer is symmetric. In: Vaudenay, S. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2006*. Lecture Notes in Computer Science, vol. 4004. Springer (2006)
46. Yung, M., Zhao, Y.: Generic and practical resettable zero-knowledge in the bare public-key model. In: Naor, M. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2007*. Lecture Notes in Computer Science, vol. 4515, pp. 129–147. Springer (2007)
47. Zhao, Y., Deng, X., Lee, C.H., Zhu, H.: Resettable zero-knowledge in the weak public-key model. In: Biham, E. (ed.) *Advances in Cryptology - Proceedings of EUROCRYPT 2003*. Lecture Notes in Computer Science, vol. 2656, pp. 123–139. Springer (2003)

A Ideal Functionalities

In this section we provide the ideal functionalities for our security proofs in the UC-framework. For better readability, we omit session identifiers and cover only the two-party case for each protocol.

A.1 Ideal Functionality for a Single Commitment

Functionality \mathcal{F}_{COM}

Implicitly parametrized by a domain of secrets S .

Commit phase:

1. Await an input (**commit**, s) with $s \in S$ from the sender. Then, store s and send (**committed**) to the adversary.
2. Await a message (**notify**) from the adversary. Then send (**committed**) to the receiver.

Unveil phase:

3. Await an input (**unveil**, \hat{s}) with $\hat{s} \in S$ from the sender. Then, store \hat{s} and send (**opened**) to the adversary.
4. Await a message (**output**) from the adversary. Then, if $\hat{s} = s$, send \hat{s} to the receiver; otherwise, send a special reject message \perp .

A.2 Ideal Functionality for Commitments with Selective Opening

Functionality $\mathcal{F}_{\text{COM}}^{\hat{s},0}$

Implicitly parametrized by a domain of secrets S and the number n of commitments to be implemented.

Commit phase:

1. Await an input (**commit**, s) with $s = (s_1, \dots, s_n) \in S^n$ from the sender. Then, store s and send (**committed**) to the adversary.
2. Await a message (**notify**) from the adversary. Then send (**committed**) to the receiver.

Unveil phase:

3. Await an input (**unveil**, I, \hat{s}) with $I \subseteq \{1, \dots, n\}$ and $\hat{s} = (\hat{s}_i)_{i \in I} \in S^{|I|}$ from the sender. Then, store (I, \hat{s}) and send (**opened**) to the adversary.
4. Await a message (**output**) from the adversary. Then, if $\hat{s} = (s_i)_{i \in I}$, send (I, \hat{s}) to the receiver; otherwise, send a special reject message \perp .

A.3 Ideal Functionality for Multiple Oblivious Transfer with Combined Abort

Functionality $\mathcal{F}_{\text{MOT}}^{\text{c-ab}}$

Implicitly parametrized by a sender input domain S and the number n of single OTs to be implemented.

- Upon input (**create**, $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)})$) with $(s_0^{(i)}, s_1^{(i)}) \in S \times S$ from the sender, verify that the sender is uncorrupted; otherwise ignore that input. Next, store $(s_0^{(0)}, s_1^{(0)}), \dots, (s_0^{(n)}, s_1^{(n)})$, send (**sent**) to the adversary, and henceforth ignore any further input from the sender.
- Upon input (**mal.create**, $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)}), Q$) with $(s_0^{(i)}, s_1^{(i)}) \in S \times S$ and a predicate $Q: \{0, 1\}^n \rightarrow \{\text{accept}, \text{reject}\}$ from the sender, verify that the sender is corrupted; otherwise ignore that input. Next, store $(s_0^{(0)}, s_1^{(0)}), \dots, (s_0^{(n)}, s_1^{(n)})$ and Q , send (**sent**) to the adversary, and henceforth ignore any further input from the sender.
- Upon input (**choice**, c) with $c = (c_1, \dots, c_n) \in \{0, 1\}^n$ from the receiver, store c , send (**chosen**) to the adversary, and henceforth ignore any further input from the receiver.
- Upon receiving a message (**output**) from the adversary, check that there are stored inputs $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)})$ from the sender and c from the receiver; else ignore this message. If the sender is corrupted, compute $Q(c)$ and abort if $Q(c) = \text{reject}$. Next, send $(s_{c_1}^{(1)}, \dots, s_{c_n}^{(n)})$ to the receiver and ignore any further (**output**)-messages from the adversary.
- Upon receiving a message (**notify**) from the adversary, check that there are stored inputs $(s_0^{(1)}, s_1^{(1)}), \dots, (s_0^{(n)}, s_1^{(n)})$ from the sender and c from the receiver; else ignore this message. Next, send an empty output to the sender and ignore any further (**notify**)-messages from the adversary.

A.4 Ideal Functionality for Zero-Knowledge

Functionality \mathcal{F}_{ZK}

Implicitly parametrized with an NP -language L and a corresponding NP -problem instance x .

1. Await an input w from the sender. Then, store w and send (**sent**) to the adversary.
2. Await a message (**verify**) from the adversary. Then, if w is a witness for $x \in L$, send (**accept**) to the verifier; else send (**reject**).

B Sampling Uniformly from Varieties of Constant Codimension

We briefly state the main theorem of [9], which is used in the proof of Corollary 1.

Theorem 1 ([9, Theorem 1.1]). *Let $k > 0$ be a constant integer, $n > k$ and $d > 0$ be integers, let p^ℓ be a sufficiently large prime power and $\epsilon > 0$ be an arbitrarily small constant. Suppose that $f_1, \dots, f_k \in \mathbb{F}_{p^\ell}[x_1, \dots, x_n]$ are polynomials, each of total degree at most d , and let*

$$V = V(f_1, \dots, f_k) = \{\xi \in \mathbb{F}_{p^\ell}^n \mid f_1(\xi) = \dots = f_k(\xi) = 0\}$$

be the variety defined by f_1, \dots, f_k . There exists a randomized algorithm that, given the description of f_1, \dots, f_k as a list of their nonzero monomials, outputs a random point $v \in \mathbb{F}_{p^\ell}^n$ such that the distribution of v is $\frac{6}{p^{\epsilon(1-\epsilon)}}$ -close to the uniform distribution on V . The worst-case runtime complexity of this algorithm is polynomial in $n, d, \ell \log(p)$ and the description of f_1, \dots, f_k .

Concretely, in Corollary 1 the field is \mathbb{F}_{2^ℓ} and $n = 4$, as elements of $\mathbb{F}_{2^{4\ell}}$ are interpreted as 4-dimensional vectors over \mathbb{F}_{2^ℓ} . The variety V is given by the polynomials (in $x = (x_1, \dots, x_4)$)

$$\begin{aligned} p_1(x) - \tilde{y} &= 0 \\ \langle h \mid x \rangle - m + \hat{s} &= 0 \end{aligned}$$

where $p_1(x) - \tilde{y} \in \mathbb{F}_{2^\ell}[x_1, \dots, x_4]$ is a polynomial of degree q and $\langle h \mid x \rangle - m + \hat{s} \in \mathbb{F}_{2^\ell}[x_1, \dots, x_4]$ is trivially a polynomial of degree 1. Thus the parameters are $k = 2$, $n = 4$, $p = 2$, and $d = q$. We can set $\epsilon = \frac{1}{2}$ and Theorem 1 yields an efficient algorithm that samples $\frac{6}{2^{\ell/2}}$ -close to uniform from V .