

Cryptography in the Multi-string Model*

Jens Groth[†]
University College London
j.groth@ucl.ac.uk

Rafail Ostrovsky[‡]
University of California, Los Angeles
rafail@cs.ucla.edu

Preliminary version appeared in CRYPTO 2007: 323-341

Abstract

The common random string model introduced by Blum, Feldman and Micali permits the construction of cryptographic protocols that are provably impossible to realize in the standard model. We can think of this model as a trusted party generating a random string and giving it to all parties in the protocol. However, the introduction of such a third party should set alarm bells going off: Who is this trusted party? Why should we trust that the string is random? Even if the string is uniformly random, how do we know it does not leak private information to the trusted party? The very point of doing cryptography in the first place is to prevent us from trusting the wrong people with our secrets.

In this paper, we propose the more realistic multi-string model. Instead of having one trusted authority, we have several authorities that generate random strings. We do not trust any single authority; we only assume a majority of them generate random strings honestly. This security model is reasonable, yet at the same time it is very easy to implement. We could for instance imagine random strings being provided on the Internet, and any set of parties that want to execute a protocol just need to agree on which authorities' strings they want to use.

We demonstrate the use of the multi-string model in several fundamental cryptographic tasks. We define multi-string non-interactive zero-knowledge proofs and prove that they exist under general cryptographic assumptions. Our multi-string NIZK proofs have very strong security properties such as simulation-extractability and extraction zero-knowledge, which makes it possible to compose them with arbitrary other protocols and to reuse the random strings. We also build efficient simulation-sound multi-string NIZK proofs for circuit satisfiability based on groups with a bilinear map. The sizes of these proofs match the best constructions in the single common random string model.

We suggest a universally composable commitment scheme in the multi-string model. It has been proven that UC commitment does not exist in the plain model without setup assumptions. Prior to this work, constructions were only known in the common reference string model and the registered public key model. One of the applications of the UC commitment scheme is a coin-flipping protocol in the multi-string model. Armed with the coin-flipping protocol, we show that any multi-party computation protocol can be securely realized in the multi-string model.

Keywords: Common random string model, multi-string model, non-interactive zero-knowledge, universally composable commitment, multi-party computation.

*An extended abstract appeared in *Advances in Cryptology - CRYPTO 2007, Lecture Notes in Computer Science*, vol. 4622, pages 323–341.

[†]Work partially done while at UCLA Computer Science Department and while visiting IPAM and supported in part by NSF ITR/Cybertrust grant No. 0456717 and Cybertrust grant No. 0430254.

[‡]Work partially done while visiting IPAM, and supported in part by IBM Faculty Award, Xerox Innovation Group Award, NSF Cybertrust grant no. 0430254, and U.C. MICRO grant.

1 Introduction

THE PROBLEM. In the common random string model, the parties executing a protocol have access to a uniformly random bit-string. A generalization of this model is the common reference string (CRS) model, where the string may have a non-uniform distribution. Blum, Feldman and Micali [BFM88] introduced the CRS model to construct non-interactive zero-knowledge (NIZK) proofs. Some setup assumption was needed, since only languages in BPP can have non-interactive or two-round zero-knowledge proofs in the plain model [GO94]. There are other examples of protocols that cannot be realized in the standard model but are possible in the CRS model, for instance universally composable (UC) commitment [CF01]. The CRS-model has therefore found wide-spread use in the field of cryptology.

Using the CRS-model creates a problem: Where does the CRS come from? One option is to have a trusted third party that generates the CRS, but this raises a trust issue. It is very possible that the parties cannot find a party that they all trust. Would Apple trust a CRS generated by Microsoft? Would US government agencies be willing to use a CRS generated by their Russian counterparts?

Alternatively, the parties could generate the CRS themselves at the beginning of the protocol. If a majority is honest, they could for instance use multi-party computation to generate a CRS. However, this makes the whole protocol more complicated and requires them to have an initial round of interaction. They could also trust a group of parties to jointly generate a CRS; however, this leaves them with the task of finding a volunteer group of authorities to run a multi-party computation protocol whenever a CRS is needed. There is also no guarantee that different sets of parties can agree on trusting the same group of authorities, so potentially this method will require authorities to participate in many generations of CRS's.

Barak, Canetti, Nielsen and Pass [BCNP04] suggest the registered public key model as a relaxed setup that makes multi-party computation possible. In the registered public key model, parties can only register correctly generated keys. While there is no longer a common reference string in the registered public key model, the underlying problem still persists: who is the trusted party that will check that the parties only register correctly generated public keys?

THE MULTI-STRING MODEL. We propose the multi-string model as a solution to the above mentioned problem. In this model, we have a number of authorities that assist the protocol execution by providing random strings. If a majority of these authorities are honest the protocol will be secure.

There are two reasons that the multi-string model is attractive. First, the authorities play a minimal role in the protocol. They simply publish random strings, they do not need to perform any computation, be aware of each other or any other parties, or have any knowledge about the specifics of the protocol to be executed. This permits easy implementation, the parties wishing to execute a protocol can for instance simply download a set of random strings from agreed upon authorities on the internet. Second, the security of the protocols only needs to rely on a majority of the authorities being honest at the time they created the strings. Even if they are later corrupted, the random strings can still be used. Also, no matter how untrustworthy the other parties in your protocol are, you can trust the protocol if a majority of the authorities is honest. In other words, the honesty of a small group of parties can be magnified and used by any set of parties.

The multi-string model is a very reasonable setup assumption. The next question is whether there are interesting protocols that can be securely realized in the multi-string model. We will answer this question affirmatively by constructing non-interactive zero-knowledge proofs, UC commitment and general UC-secure multi-party computation in the multi-string model in the presence of adaptive adversaries.

1.1 Non-interactive Zero-Knowledge

A zero-knowledge proof [GMR89, GMW87] is a two-party protocol, where a prover tries to convince a verifier about the truth of some statement, typically membership of an NP-language. The proof should have the following three properties: completeness, soundness and zero-knowledge. Completeness means that a

prover who has an NP-witness for the truth of the statement can convince the verifier. Soundness means that if the statement is false, then it is impossible to convince the verifier. Zero-knowledge means that the verifier does not learn anything else from the proof than the fact that the statement is true. Interactive zero-knowledge proofs are known to exist in the plain model without a CRS, however, non-interactive and 2-round zero-knowledge proofs only exist for trivial languages [GO94]. Instead, much research has gone into constructing non-interactive zero-knowledge proofs in the CRS-model [BFM88, BDMP91, FLS99, Dam92, DP92, DDP99, DDP02, KP98, Sah01, DDO⁺02, GOS06b, GOS06a].

MULTI-STRING NIZK. We define the notion of multi-string NIZK proofs in Section 2. In the definitions, we let the adversary see many honestly generated strings and pick the ones it likes. We also allow the adversary to generate some of the strings itself, possibly in a malicious and adaptive manner. Our definition of multi-string NIZK proofs calls for completeness, soundness and zero-knowledge to hold in a threshold manner. If t_c out of n common reference strings are honest, then the prover holding an NP-witness for the truth of the statement should be able to create a convincing proof. If t_s out of n common reference strings are honest, then it should be infeasible to convince the verifier about a false statement. If t_z out of n common reference strings are honestly generated, then it should be possible to simulate the proof without knowing the witness.

It is desirable to minimize t_c, t_s, t_z . As we shall see, $t_c = 0$ is achievable, however, multi-string soundness and multi-string zero-knowledge are complementary in the sense that there is a lower bound $t_s + t_z > n$ for non-trivial languages, see Section 2.

A natural question is under which assumptions we can obtain multi-string NIZK proofs. We prove that if hard on average languages exist in NP then single-string NIZK implies the existence of multi-string NIZK and vice versa.

BEYOND VANILLA MULTI-STRING NIZK. It is undesirable to require a group of authorities to produce random strings for each proof we want to make. We prefer it to be possible to use the same strings over and over again, so each authority has to produce only one single random string. We must therefore consider a setting, where multiple protocols may be running concurrently and of which some of them may require the use of multi-string NIZK proofs. When the protocol designer has to prove security in such a setting, it may very well be that some of the proofs are simulated, while we still need other proofs to be sound. Moreover, in some cases we may want to extract the witness from a proof. To deal with this realistic setting, where we have both simulations of some proofs and witness extraction of other proofs going on at the same time, we introduce the notions of simulation-extractable multi-string NIZK and extraction zero-knowledge multi-string NIZK.

In simulation-extractable multi-string NIZK, we require that it be possible to extract a witness from the proof if t_s strings are honestly generated, even if the adversary sees simulated proofs for arbitrary other statements. In extraction zero-knowledge, we require that if there are t_z honest strings, then even if the adversary sees extractions of witnesses in some proofs, the other proofs remain zero-knowledge and reveal nothing. We offer a multi-string NIZK proof based on general assumptions, which is both simulation-extractable and extraction zero-knowledge.

MULTI-STRING NIZK PROOFS FROM BILINEAR GROUPS. Groth, Ostrovsky and Sahai [GOS06b, GOS06a] constructed NIZK proofs from groups with a bilinear map. Their CRS contains a description of a bilinear group and a set of group elements. The group elements can be chosen such that the CRS gives either perfect soundness or perfect zero-knowledge. Soundness strings and simulation strings are computationally indistinguishable, so this gives a NIZK proof in the CRS model.

There is a major technical hurdle to overcome when trying to apply their techniques in the multi-string model: the single-string NIZK proofs rely on the common reference string to contain a description of a bilinear group. In the multi-string model, the authorities generate their random strings completely oblivious of the other authorities. There is therefore no agreement on which bilinear group to use. One might try to let the prover pick the bilinear group, however, this too causes problems since now we need to set up the random

strings such that they will work for many choices of bilinear groups.

We resolve these problems by inventing a novel technique to “translate” common reference strings in one group to common reference strings in another group. Each authority picks its own bilinear group and the prover also picks a bilinear group. Using our translation technique, we can translate simulation reference strings chosen by the authorities to simulation reference strings in the prover’s bilinear group. Similarly, we can translate soundness reference strings chosen by the authorities to soundness reference strings in the prover’s bilinear group.

The resulting multi-string NIZK proofs for circuit satisfiability have size $\mathcal{O}(n + |C|)k$, where n is the number of random strings, $|C|$ is the size of the circuit, and k is a security parameter specifying the size of a group element. Typically n will be much smaller than $|C|$, so this matches the best single-string NIZK proofs of [GOS06b, GOS06a] that have complexity $\mathcal{O}(|C|k)$.

1.2 Multi-party Computation

Canetti’s UC framework [Can01] defines secure execution of a protocol under concurrent execution of arbitrary protocols. Informally a protocol is UC secure if its execution is equivalent to handing protocol input to an honest trusted party that computes everything securely and returns the resulting outputs. We refer the reader to Section 6 for a sketch of the UC framework and to Canetti’s paper for details.

UC COMMITMENT. It is known that in the plain model, any (well-formed) ideal functionality can be securely realized if a majority of the parties are honest. On the other hand, if a majority may be corrupt, there are certain functionalities that are provably impossible to realize. An example of an unrealizable functionality is UC commitment [CF01]. We demonstrate that in the multi-string model UC commitment can be securely realized. The key idea in this construction is to treat each common random string as the key for a commitment scheme. By applying threshold secret-sharing techniques, we can spread the message out on the n commitment scheme in a way such that we can tolerate a minority of fake common reference strings.

GENERAL MULTI-PARTY COMPUTATION. Canetti, Lindell, Ostrovsky and Sahai [CLOS02] showed that any (well-formed) ideal functionality can be securely realized in the CRS-model, even against adversaries that can adaptively corrupt arbitrary parties and where parties are not assumed to be able to securely erase any of their data. However, it was an open question where this CRS should come from, since the parties provably could not compute it themselves.

Armed with our UC commitment it is straightforward to solve this problem. We simply run a coin-flipping protocol to create a CRS. This result points out a nice feature of the multi-string model; it scales extremely well. We just require a majority of the authorities to be honest. Then no matter which group of parties, even if it is a large group of mostly untrustworthy parties, we can magnify the authorities’ honesty to enable this entire group to do secure computation.

REMARK. The multi-string model is described in the UC framework as an ideal functionality that provides random strings and allows the adversary to inject a minority of malicious strings as well. This functionality is of course easy to implement with a set of authorities that just provide random strings. It is important though that these strings are local to the protocol, we do not guarantee security of other protocols that use the same strings. Canetti, Dodis, Pass and Walfish [CDPW07] have demonstrated that it is not possible to have one fixed global common random string that is used for multiple arbitrary protocol executions and this result extends to the multi-string model. Orthogonally to our work, they instead suggest the augmented common reference string model where general UC secure multi-party computation is possible.

REMARK. Building on our multi-string NIZK, an alternative proof of our multiparty computation result was shown by [PPS06].

2 Definitions

We model algorithms and adversaries as Turing machines. They get a security parameter k as input written in unary, we will often not write this explicitly. The adversary may be an interactive Turing machine that keeps state between different invocation and may have or not have bounded running time.

We say a function $\nu : \mathbb{N} \rightarrow [0; 1]$ is negligible if for all constants $c > 0$ there exists a K_c so for all $k > K_c$ we have $\nu(k) < k^{-c}$. For two functions f, g we write $f(k) \approx g(k)$ if $|f(k) - g(k)|$ is negligible.

Let R be an efficiently computable binary relation. For pairs $(x, w) \in R$ we call x the statement and w the witness. Let L be the NP-language consisting of statements in R .

A multi-string proof system for a relation R consists of probabilistic polynomial time algorithms K, P, V , which we will refer to as respectively the key generator, the prover and the verifier. The key generation algorithm can be used to produce common reference strings σ . In the present paper, we can implement our protocols with a key generator that outputs a uniformly random string of polynomial length $\ell(k)$, however, for the sake of generality, we include a key generator in our definitions.

The prover takes as input $(\vec{\sigma}, x, w)$, where $\vec{\sigma}$ is a set of n different common reference strings and $(x, w) \in R$, and produces a proof π . The verifier takes as input $(\vec{\sigma}, x, \pi)$ and outputs 1 if the proof is acceptable and 0 if rejecting the proof. We call (K, P, V) a (t_c, t_s, t_z, n) -NIZK proof system for R if it has the completeness, soundness and zero-knowledge properties described below. We remark that $(1, 1, 1, 1)$ -NIZK proofs correspond to the standard NIZK proofs in the CRS-model.

(t_c, t_s, t_z, n) -COMPLETENESS. We will say that (K, P, V) is (t_c, t_s, t_z, n) -complete if the prover can convince the verifier of a true statement, when at least t_c strings have been generated honestly.

Definition 1 (K, P, V) is (computationally) (t_c, t_s, t_z, n) -complete if for all non-uniform polynomial time adversaries \mathcal{A} we have

$$\Pr \left[(\vec{\sigma}, x, w) \leftarrow \mathcal{A}^K(1^k); \pi \leftarrow P(\vec{\sigma}, x, w) : V(\vec{\sigma}, x, \pi) = 1 \right] \approx 1,$$

where K on query i outputs $\sigma_i \leftarrow K(1^k)$ such that at least t_c of the σ_i 's generated by K are included and \mathcal{A} outputs $(x, w) \in R$.

As we will see later, our protocols have *perfect* (t_c, t_s, t_z, n) -completeness for all $0 \leq t_c \leq n$. In other words, even if the adversary chooses all common reference strings itself, we have probability exactly 1 of outputting an acceptable proof when $(x, w) \in R$.

(t_c, t_s, t_z, n) -SOUNDNESS. Soundness says that an adversary cannot forge a proof when at least t_s of the common reference strings are honestly generated. The adversary gets to see possible choices of correctly generated common reference strings and can adaptively choose n of them, it may also in these n common reference strings include up to $n - t_s$ fake common reference strings chosen by itself.

Definition 2 We say (K, P, V) is (t_c, t_s, t_z, n) -sound if for all adversaries \mathcal{A} we have

$$\Pr \left[(\vec{\sigma}, x, \pi) \leftarrow \mathcal{A}^K(1^k) : V(\vec{\sigma}, x, \pi) = 1 \text{ and } x \notin L \right] \approx 0,$$

where K is an oracle that on query i outputs $\sigma_i \leftarrow K(1^k)$ and the adversary outputs $\vec{\sigma}$ such that at least t_s of the σ_i 's generated by K are included.

The definition above refers to *statistical* soundness, where the adversary has unbounded time. We call it *perfect* soundness, when the probability is exactly 0.

(t_c, t_s, t_z, n) -ZERO-KNOWLEDGE. Zero-knowledge informally means that if t_z common reference strings are correctly generated, then the adversary learns nothing from the proof. As is standard in the zero-knowledge literature, we will say this is the case, when we can simulate the proof given only the statement x . Let therefore S_1 be a probabilistic polynomial time algorithm that outputs (σ, τ) , respectively a simulation reference string and a simulation trapdoor. Let furthermore, S_2 be a probabilistic polynomial time algorithm that takes input $(\vec{\sigma}, \vec{\tau}, x, w)$ and simulates a proof π if $\vec{\tau}$ contains t_z simulation trapdoors for common reference strings in $\vec{\sigma}$.

We will strengthen the standard definition of zero-knowledge, by splitting the definition of zero-knowledge into two parts. The first part simply says that the adversary cannot distinguish real common reference strings from simulation reference strings. The second part, says that *even with access to the simulation trapdoors* the adversary cannot distinguish the prover from the simulator on a set of simulated reference strings.

Definition 3 We say (K, P, V, S_1, S_2) is (t_c, t_s, t_z, n) -zero-knowledge if we have reference string indistinguishability and simulation indistinguishability as described below.

REFERENCE STRING INDISTINGUISHABILITY. For all non-uniform polynomial time adversaries \mathcal{A} we have

$$\Pr \left[\sigma \leftarrow K(1^k) : \mathcal{A}(\sigma) = 1 \right] \approx \Pr \left[(\sigma, \tau) \leftarrow S_1(1^k) : \mathcal{A}(\sigma) = 1 \right].$$

(t_c, t_s, t_z, n) -SIMULATION INDISTINGUISHABILITY. For all non-uniform interactive polynomial time adversaries \mathcal{A} we have

$$\begin{aligned} & \Pr \left[(\vec{\sigma}, \vec{\tau}, x, w) \leftarrow \mathcal{A}^{S_1}(1^k); \pi \leftarrow P(\vec{\sigma}, x, w) : \mathcal{A}(\pi) = 1 \right] \\ & \approx \Pr \left[(\vec{\sigma}, \vec{\tau}, x, w) \leftarrow \mathcal{A}^{S_1}(1^k); \pi \leftarrow S_2(\vec{\sigma}, \vec{\tau}, x) : \mathcal{A}(\pi) = 1 \right], \end{aligned}$$

where S_1 on query i outputs $(\sigma_i, \tau_i) \leftarrow S_1(1^k)$, the adversary outputs $(x, w) \in R$ and $\vec{\sigma}, \vec{\tau}$ so at least t_z of the σ_i 's generated by S_1 are included and $\vec{\tau}$ contains t_z simulation trapdoors τ_i corresponding to σ_i 's that have been generated by the oracle S_1 .

LOWER BOUNDS FOR MULTI-STRING NIZK PROOFS. Soundness and zero-knowledge are complementary. The intuition is that if an adversary controls enough strings to simulate a proof, then she can prove anything and we can no longer have soundness. We capture this formally in the following theorem.

Theorem 4 If L is a language with a proof system (K, P, V) that has (t_c, t_s, t_z, n) -completeness, soundness and zero-knowledge then $L \in \text{P/poly}$ or $t_s + t_z > n$.

Proof. Assume we have a (t_c, t_s, t_z, n) -NIZK proof system for R defining L and $t_s + t_z \leq n$. Given an element x , we wish to decide whether $x \in L$ or not. We simulate t_z common reference strings $(\sigma_i, \tau_i) \leftarrow S_1(1^k)$ and generate $n - t_z$ common reference strings $\sigma_j \leftarrow K(1^k)$ setting $\tau_j = \perp$. We then simulate the proof $\pi \leftarrow S_2(\vec{\sigma}, \vec{\tau}, x)$. Output $V(\vec{\sigma}, x, \pi)$.

Let us analyze this algorithm. If $x \in L$, then by (t_c, t_s, t_z, n) -completeness a prover with access to a witness w would output a proof that the verifier accepts if all common reference strings are generated correctly. By reference string indistinguishability, we will also accept the proof when some of the common reference strings are simulated. By (t_c, t_s, t_z, n) -simulation indistinguishability, where we give (x, w) as non-uniform advice to \mathcal{A} , the proof will also be accepted when we simulate it instead of proving it using w . We will therefore output 1 with overwhelming probability on $x \in L$.

On the other hand, if $x \notin L$, then by the (t_c, t_s, t_z, n) -soundness we output 0 with overwhelming probability, since $n - t_z \geq t_s$ common reference strings have been generated correctly. This shows that $L \in \text{BPP/poly}$. Adleman [Adl78] has shown $\text{P/poly} = \text{BPP/poly}$, which concludes the proof. \square

In general, the verifier wishes to minimize t_s to make it more probable that the protocol is sound, and at the same time the prover wishes to minimize t_z to make it more probable that the protocol is zero-knowledge. In many cases, choosing n odd, and setting $t_s = t_z = \frac{n+1}{2}$ will be a reasonable compromise. However, there are also cases where it is relevant to have an unbalanced setting. Consider for instance the case, where Alice wants to e-mail a NIZK proof to Bob, but does not know Bob's preferences with respect to common reference strings. She may pick a set of common reference strings and make a multi-string proof. Bob did not participate in deciding which common reference strings to use, however, if they belong to trustworthy authorities he may be willing to believe that one of them is honest. On the other hand, Alice gets to choose the authorities, so she may be willing to believe that all of them are honest. The appropriate choice in this situation, is a multi-string proof with $t_s = 1, t_z = n$.

(t_c, t_s, t_z, n) -KNOWLEDGE. Extending the definition of soundness, we say (K, P, V) is a (t_c, t_s, t_z, n) proof of knowledge for R if there are probabilistic polynomial time algorithms E_1, E_2 that can extract a witness from a valid proof.

Definition 5 We say (K, P, V, E_1, E_2) has (t_c, t_s, t_z, n) -knowledge if for all non-uniform polynomial time adversaries \mathcal{A} we have

$$\Pr \left[\sigma \leftarrow K(1^k) : \mathcal{A}(\sigma) = 1 \right] \approx \Pr \left[(\sigma, \xi) \leftarrow E_1(1^k) : \mathcal{A}(\sigma) = 1 \right],$$

and for all non-uniform polynomial time adversaries \mathcal{A} we have

$$\Pr \left[(\vec{\sigma}, x, \pi) \leftarrow \mathcal{A}^{E_1}(1^k); w \leftarrow E_2(\vec{\sigma}, \vec{\xi}, x, \pi) : V(\vec{\sigma}, x, \pi) = 1 \text{ and } (x, w) \notin R \right] \approx 0,$$

where E_1 is an oracle that returns $(\sigma_i, \xi_i) \leftarrow E_1(1^k)$, and $\vec{\xi}$ contains at least t_s ξ_i 's corresponding to the σ_i 's generated by E_1 .

(t_c, t_s, t_z, n) -SIMULATION-SOUNDNESS. In security proofs, it is often useful to simulate a proof for a false statement. However, seeing a simulated proof for a false statement might enable an adversary to generate more proofs for false statements. We say an NIZK proof is (t_c, t_s, t_z, n) -simulation-sound if an adversary cannot prove any false statement even after seeing simulated proofs of arbitrary statements.

Definition 6 A (t_c, t_s, t_z, n) -NIZK proof system (K, P, V, S_1, S_2) is (t_c, t_s, t_z, n) -simulation-sound if for all non-uniform polynomial time adversaries we have

$$\Pr \left[(\vec{\sigma}, x, \pi) \leftarrow \mathcal{A}^{S_1, S_2(\cdot, \cdot)}(1^k) : (\vec{\sigma}, x, \pi) \notin Q \text{ and } x \notin L \text{ and } V(\vec{\sigma}, x, \pi) = 1 \right] \approx 0,$$

where S_1 on query i returns $(\sigma_i, \tau_i) \leftarrow S_1(1^k)$, and S_2 on input $(\vec{\sigma}_j, x_j)$ returns $\pi \leftarrow S_2(\vec{\sigma}_j, \vec{\tau}_j, x_j)$ with $\vec{\tau}_j$ having simulation trapdoors for the σ_i 's generated by S_1 , and the adversary produces $\vec{\sigma}_j$ containing at least t_s σ_i 's generated by S_1 , and Q is the list of statements and corresponding proofs $(\vec{\sigma}_j, x_j, \pi_j)$ in the queries to S_2 .

(t_c, t_s, t_z, n) -SIMULATION-EXTRACTABILITY. Since we are working in the multi-string model, we assume strings can be used by anybody who comes along. Knowledge extraction and zero-knowledge may both be very desirable properties, however, we may also imagine security proofs where we at the same time need to extract witnesses from some proofs and simulate other proofs. This joint simulation/extraction is for instance often seen in security proofs in the UC framework [Can01].

Combining simulation-soundness and knowledge extraction, we may therefore require that even after seeing many simulated proofs, whenever the adversary makes a new proof we are able to extract a witness. We call this property simulation-extractability. Simulation-extractability implies simulation-soundness, because if we can extract a witness from the adversary's proof, then obviously the statement must belong to the language in question.

Definition 7 We say $(K, P, V, S_1, S_2, E_1, E_2, SE_1)$ is (t_c, t_s, t_z, n) -simulation-extractable if (K, P, V, E_1, E_2) has (t_c, t_s, t_z, n) -knowledge, (K, P, V, S_1, S_2) is a (t_c, t_s, t_z, n) -NIZK proof, SE_1 is a probabilistic polynomial time algorithm that outputs (σ, τ, ξ) such that it is identical to S_1 when restricted to the first two parts (σ, τ) , and for all non-uniform polynomial time adversaries we have

$$\Pr \left[(\vec{\sigma}, x, \pi) \leftarrow \mathcal{A}^{SE_1', S_2(\cdot, \cdot)}(1^k); w \leftarrow E_2(\vec{\sigma}, \vec{\xi}, x, \pi) : \right. \\ \left. (\vec{\sigma}, x, \pi) \notin Q \text{ and } (x, w) \notin R \text{ and } V(\vec{\sigma}, x, \pi) = 1 \right] \approx 0,$$

where SE_1' on query i outputs (σ_i, ξ_i) from $(\sigma_i, \tau_i, \xi_i) \leftarrow SE_1(1^k)$, S_2 outputs $\pi_j \leftarrow S_2(\vec{\sigma}_j, \vec{\tau}_j, x_j)$, where $\vec{\tau}_j$ contains t_z τ_i 's corresponding to σ_i 's generated by SE_1 , Q is a list of statements and corresponding proofs $(\vec{\sigma}_j, x_j, \pi_j)$ made by S_2 , and $\vec{\xi}$ contains the first t_s ξ_i 's generated by SE_1 corresponding to σ_i 's in $\vec{\sigma}$.

(t_c, t_s, t_z, n) -EXTRACTION ZERO-KNOWLEDGE. Combining simulation soundness and knowledge extraction, we may also require that even after seeing many extractions, it should still be hard to distinguish real proofs and simulated proofs from one another. This definition resembles the definition of chosen ciphertext attack secure public key encryption.

Definition 8 Consider a (t_c, t_s, t_z, n) -NIZK proof of knowledge $(K, P, V, S_1, S_2, E_1, E_2)$. Let SE_1 be a probabilistic polynomial time algorithm that outputs (σ, τ, ξ) such that it is identical to S_1 when restricted to the first two parts (σ, τ) . We say $(K, P, V, S_1, S_2, E_1, E_2, SE_1)$ is (t_c, t_s, t_z, n) -extraction zero-knowledge if for all non-uniform interactive polynomial time adversaries we have

$$\Pr \left[(\vec{\sigma}, x, w) \leftarrow \mathcal{A}^{SE_1', E_2(\cdot, \cdot)}(1^k); \pi \leftarrow P(\vec{\sigma}, x, w) : \mathcal{A}^{E_2(\cdot, \cdot)}(\pi) = 1 \text{ and } (x, w) \in R \right] \\ \approx \Pr \left[(\vec{\sigma}, x, w) \leftarrow \mathcal{A}^{SE_1', E_2(\cdot, \cdot)}(1^k); \pi \leftarrow S_2(\vec{\sigma}, \vec{\tau}, x) : \mathcal{A}^{E_2(\cdot, \cdot)}(\pi) = 1 \text{ and } (x, w) \in R \right],$$

where SE_1' on query i outputs (σ_i, τ_i) from $(\sigma_i, \tau_i, \xi_i) \leftarrow SE_1(1^k)$, E_2 outputs $w \leftarrow E_2(\vec{\sigma}_j, \vec{\xi}_j, x_j)$, when the query contains t_s σ_i 's generated by SE_1 and π is not the challenge proof.

3 Multi-string NIZK Proofs based on General Assumptions

MULTI-STRING NIZK PROOFS. As a warm-up, we will start out with a simple construction of a multi-string NIZK proof that works for $t_c = 0$ and all choices of t_s, t_z, n where $t_s + t_z > n$. We use two tools in the construction, a pseudorandom generator PRG and a zap $(\ell_{\text{zap}}, P_{\text{zap}}, V_{\text{zap}})$. Zaps, introduced by Dwork and Naor [DN02], are two-round public coin witness-indistinguishable proofs, where the verifier's first message is a random string that can be fixed once and for all and be reused in subsequent zaps.

A common random string in our multi-string NIZK proof will consist of a random value r and an initial message σ for the zap. Given a statement $x \in L$, the prover makes n zaps using respectively initial messages $\sigma_1, \dots, \sigma_n$ for the statement

$$x \in L \quad \text{or} \quad \text{there are } t_z \text{ common reference strings where } r_i \text{ is a pseudorandom value.}$$

In the simulation, we create simulation reference strings as $r := \text{PRG}(\tau)$ enabling the simulator to make zaps without knowing a witness w for $x \in L$ if instead the simulator knows the seeds of t_z pseudorandom values r_i .

Common reference string: Generate $r \leftarrow \{0, 1\}^{2k}$; $\sigma \leftarrow \{0, 1\}^{\ell_{\text{zap}}(k)}$. Output $\Sigma := (r, \sigma)$.

Proof: Given input $(\Sigma_1, \dots, \Sigma_n)$, a statement x and a witness w so $(x, w) \in R$, we wish to prove $x \in L$. Using NP-reductions, we create a polynomial size circuit C that is satisfiable if and only if

$$x \in L \quad \text{or} \quad \left| \{r_i \mid \exists \tau_i : r_i = \text{PRG}(\tau_i)\} \right| \geq t_z.$$

Chosen appropriately, NP-reductions are witness preserving, so we also reduce w to a witness W for C being satisfiable. For all n common reference strings, generate $\pi_i \leftarrow P_{\text{zap}}(\sigma_i, C, W)$. Return the proof $\Pi := (\pi_1, \dots, \pi_n)$.

Verification: Given n common reference strings $(\Sigma_1, \dots, \Sigma_n)$, a statement x and a proof $\Pi = (\pi_1, \dots, \pi_n)$ return 1 if and only if all of them satisfy $V_{\text{zap}}(\sigma_i, C, \pi_i) = 1$, where C is generated as in the proof.

Simulated reference string: Select $\tau \leftarrow \{0, 1\}^k$; $r := \text{PRG}(\tau)$ and $\sigma \leftarrow \{0, 1\}^{\ell_{\text{zap}}(k)}$. Output $((r, \sigma), \tau)$.

Simulated proof: Given input $t_z, (\Sigma_1, \dots, \Sigma_n), (\tau_1, \dots, \tau_n), x$ such that for t_z reference strings $r_i = \text{PRG}(\tau_i)$ we wish to simulate a proof Π . As in a proof, use NP-reductions to get a circuit C that is satisfiable if and only if $x \in L$ or $\left| \{r_i \mid \exists \tau_i : r_i = \text{PRG}(\tau_i)\} \right| \geq t_z$. Pick the first t_z common reference string Σ_i , where $r_i = \text{PRG}(\tau_i)$, and reduce this to a witness W for the satisfiability of C . For all n common reference strings, generate $\pi_i \leftarrow P_{\text{zap}}(\sigma_i, C, W)$. Return the simulated proof $\Pi := (\pi_1, \dots, \pi_n)$.

This construction will be used to prove the following theorem connecting single-string NIZK proofs and multi-string NIZK proofs.

Theorem 9 *Assuming hard on average languages exist in NP, the existence of NIZK proofs for all NP-languages in the common random string model is equivalent to the existence of multi-string NIZK proofs for all NP-languages in the common random strings model. The equivalence preserves perfect completeness.*

Proof. We first show that (n, t_s, t_z, n) -NIZK proofs implies the existence of standard NIZK proofs. We simply generate n common reference strings and concatenate them to get a single common reference string. To make an NIZK proof, we run the multi-string prover on this reference string. Completeness, soundness and zero-knowledge follow directly. If the multi-string NIZK proof uses random strings, then obviously we get a random string NIZK proof. If the multi-string NIZK proof has perfect completeness, then we get perfect completeness.

To go the other way, we use the fact that the existence of hard on average languages in NP and NIZK proofs imply the existence of one-way functions [Ost91, OW93]. One-way functions imply pseudorandom generators [HILL99]. NIZK proofs in the common random string model also imply the existence of zaps [DN02]. If we use an NIZK proof with perfect completeness, then we also get a zap with perfect completeness. There is one detail worth mentioning. At the time of generating the common reference string, we do not know the size of the circuit we will be proving. We therefore need an NIZK proof that works for arbitrarily large circuits, so we can build zaps that work for large circuits. Using the one-way function and the non-interactive version of Naor's statistically binding commitment scheme based on one-way functions [Nao91, DIO98] we can chop the circuit up into manageable pieces. We can commit to each wire-value in the circuit and make NIZK proofs for each wire that it contains 0 or 1 and make NIZK proofs for each gate

that the committed values respect the gate. It is therefore straightforward to get NIZK proofs where the size of the circuit can be arbitrarily large.

Direct verification of our construction reveals that we have completeness, even for $t_c = 0$. If the NIZK proof has perfect completeness, then we get perfect completeness. Let us prove that we have $(0, t_s, t_z, n)$ -soundness. Any honestly generated common reference string has negligible probability of containing a pseudorandom value r . With t_s honestly generated strings and $t_z > n - t_s$, there is negligible probability that $(\Sigma_1, \dots, \Sigma_n)$ have t_z or more pseudorandom values. If $x \notin L$, the resulting circuit C is unsatisfiable. Also, at least one of the common reference strings has a correctly generated initial message for the zap. By the statistical soundness of this zap there is negligible probability that there exists a valid zap for C being satisfiable.

We now turn to the question of $(0, t_s, t_z, n)$ -zero-knowledge. Computational reference string indistinguishability follows from the pseudorandomness of PRG. With at least t_z simulated reference strings the only difference between proofs using the witness of $x \in L$ and simulated proofs using the simulation trapdoors is the witnesses we are using in the zaps. Computational simulation indistinguishability therefore follows from a standard hybrid argument using the witness indistinguishability of the zaps. \square

4 Multi-string Simulation-Extractable NIZK Proofs

We will now construct more advanced multi-string NIZK proofs of knowledge that are $(0, t_s, t_z, n)$ -simulation-extractable and $(0, t_s, t_z, n)$ -extraction zero-knowledge.

To permit the extraction of witnesses, we include a public key for a cryptosystem secure against adaptive chosen ciphertext attacks in each common reference string. In a proof, the prover will make a (t_s, n) -threshold secret sharing of the witness and encrypt the shares under the n public keys. To extract the witness, we will decrypt t_s of these ciphertexts and combine the shares to get the witness.

To avoid tampering with the proof, we will use a strong one-time signature. The prover generates a key $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(1^k)$ that she will use to sign the proof. The implication is that the adversary, who sees simulated proofs, must use a different vk_{sots} in her forged proof, because she cannot forge the strong one-time signature.

The common reference string will contain a value, which in a simulation string will be a pseudorandom $2k$ -bit value. The prover will prove that she encrypted a (t_s, n) -threshold secret sharing of the witness, or that she knows how to evaluate t_z pseudorandom functions on vk_{sots} using the seeds of the respective common reference strings. On a real common reference string, this seed is not known and therefore she cannot make such a proof. On the other hand, in the simulation the simulator does know these seeds and can therefore simulate without knowing the witness. Simulation soundness follows from the adversary's inability to guess the pseudorandom functions' evaluations on vk_{sots} , even if she knew the evaluations on many other verification keys.

Zero-knowledge under extraction attack follows from the CCA2-security of the cryptosystem. Even after having seen many extractions, the ciphertexts reveal nothing about the witness, or even whether the trapdoor has been used to simulate a proof.

Common reference string/simulation string: Generate $(pk_1, dk_1), (pk_2, dk_2) \leftarrow K_{\text{CCA2}}(1^k); r \leftarrow \{0, 1\}^{2k}; \sigma \leftarrow \{0, 1\}^{\ell_{\text{zap}}(k)}$. Return $\Sigma := (pk_1, pk_2, r, \sigma)$.

The simulators and extractors S_1, E_1, SE_1 will generate the simulated reference strings in the same way, except for choosing $\tau \leftarrow \{0, 1\}^k$ and $r := \text{PRF}_\tau(0)$. We use the simulation trapdoor τ and the extraction key $\xi := dk_1$.

Proof: $P((\Sigma_1, \dots, \Sigma_n), x, w)$ where $(x, w) \in R$ runs as follows: First, generate a key pair for a strong one-time signature scheme $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(1^k)$. Use (t_s, n) -threshold secret sharing to get

shares w_1, \dots, w_n of w . Encrypt the shares as $c_{1i} := E_{pk_{1i}}(w_i, vk_{\text{sots}}; r_{1i})$. Also encrypt dummy values $c_{2i} \leftarrow E_{pk_{2i}}(0)$. Consider the statement:

“All c_{1i} encrypt (w_i, vk_{sots}) , where w_1, \dots, w_n is a (t_s, n) -secret sharing of a witness w so $(x, w) \in R$ or there exist at least t_z seeds τ_i so $r_i = \text{PRF}_{\tau_i}(0)$ and c_{2i} encrypts $\text{PRF}_{\tau_i}(vk_{\text{sots}})$.”

We can reduce this statement to a polynomial size circuit C and a satisfiability witness W . For all i 's we create a zap $\pi_i \leftarrow P_{\text{zap}}(\sigma_i, C, W)$ for C being satisfiable. Finally, we sign everything using the strong one-time signature $sig \leftarrow \text{Sign}_{sk_{\text{sots}}}(vk_{\text{sots}}, x, \Sigma_1, c_{11}, c_{21}, \pi_1, \dots, \Sigma_n, c_{1n}, c_{2n}, \pi_n)$.

The proof is $\Pi := (vk_{\text{sots}}, c_{11}, c_{21}, \pi_1, \dots, c_{1n}, c_{2n}, \pi_n, sig)$.

Verification: To verify Π on the form described above, verify the strong one-time signature and verify the n zaps π_1, \dots, π_n .

Extraction: To extract a witness check that the proof is valid. Next, use the first t_s extraction keys in $\vec{\xi}$ to decrypt the corresponding t_s ciphertexts. We combine the t_s secret shares to recover the witness w .

Simulated proof: To simulate a proof, pick the first t_z simulation trapdoors in $\vec{\tau}$. These are τ_i so $r_i = \text{PRF}_{\tau_i}(0)$. As in the proof generate $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(1^k)$. Create t_z pseudorandom values $v_i := \text{PRF}_{\tau_i}(vk_{\text{sots}})$. Encrypt the values as $c_{2i} \leftarrow E_{pk_{2i}}(v_i)$. For the other reference strings, just let $c_{2i} \leftarrow E_{pk_{2i}}(0)$. Let w_1, \dots, w_n be a (t_s, n) -threshold secret sharing of 0. We encrypt also these values as $c_{1i} \leftarrow E_{pk_{1i}}(w_i, vk_{\text{sots}})$. Let again C be the circuit corresponding to the statement

“All c_{1i} encrypt (w_i, vk_{sots}) , where w_1, \dots, w_n is a (t_s, n) -secret sharing of a witness w or there exist at least t_z seeds τ_i so $r_i = \text{PRF}_{\tau_i}(0)$ and c_{2i} encrypts $\text{PRF}_{\tau_i}(vk_{\text{sots}})$.”

From the creation of the ciphertexts c_{2i} we have a witness W for C being satisfiable. Create zaps $\pi_i \leftarrow P_{\text{zap}}(\sigma_i, C, W)$ for C being satisfiable. Finally, make a strong one-time signature on everything $sig \leftarrow \text{Sign}_{sk_{\text{sots}}}(vk_{\text{sots}}, x, \Sigma_1, c_{11}, c_{21}, \pi_1, \dots, \Sigma_n, c_{1n}, c_{2n}, \pi_n)$. The simulated proof is $\Pi := (vk_{\text{sots}}, c_{11}, c_{21}, \pi_1, \dots, c_{1n}, c_{2n}, \pi_n, sig)$.

Theorem 10 *The construction given above is a $(0, t_s, t_z, n)$ -NIZK proof for all choices of $t_s + t_z > n$. It has $(0, t_s, t_z, n)$ -simulation-soundness, $(0, t_s, t_z, n)$ -extraction zero-knowledge and statistical $(0, t_s, t_z, n)$ -knowledge. It can be securely implemented if enhanced trapdoor permutations¹ exist, and it can be implemented with random strings if dense cryptosystems [DP92] and enhanced trapdoor permutations exist.*

Proof. Let us start with the latter part. Enhanced trapdoor permutations, imply the existence of NIZK proofs with perfect completeness in the random string model, which in turn imply the existence of zaps with perfect completeness. Enhanced trapdoor permutations also imply the existence of pseudorandom functions, strong one-time signatures and CCA2-secure public key encryption with errorless decryption. In case dense public key cryptosystems and enhanced trapdoor permutations exist, CCA2-secure encryption with random strings as public keys exists.

Perfect completeness follows by direct verification. Common reference strings and simulated reference strings are indistinguishable by the pseudorandomness of the pseudorandom function PRF.

Let us consider $(0, t_s, t_z, n)$ -extraction zero-knowledge. The adversary knows the simulation trapdoors τ_i , and has access to an extraction oracle. She selects a statement x and a witness w and has to distinguish a proof on a simulated reference string using respectively the witness or the simulator. We consider a series of hybrid experiments.

Hybrid 1: This is the experiment, where we run the adversary on a simulated reference string and make proofs using the witness w .

¹Enhanced trapdoor permutations are trapdoor permutations that are hard to invert on a random element even when the random coins used to select that element are known to the adversary

Hybrid 2: We modify hybrid 1 by encrypting t_z pseudorandom values in c_{21}, \dots, c_{2n} . We know t_z seeds τ_i such that $r_i = \text{PRF}_{\tau_i}(0)$. Instead of setting $c_{2i} \leftarrow E_{pk_2}(0)$, we encrypt $c_{2i} \leftarrow E_{pk_2}(\text{PRF}_{\tau_i}(vk_{\text{sots}}))$.

By the semantic security of the cryptosystem, hybrid 1 and hybrid 2 are computationally indistinguishable.

Hybrid 3: We modify hybrid 2, by reducing the pseudorandom values and the randomness used in forming the ciphertexts c_{21}, \dots, c_{2n} to form a witness W for C being satisfiable. We use this witness in the zaps, instead of the witness w .

By the witness-indistinguishability of the zaps, hybrid experiments 2 and 3 are indistinguishable.

Hybrid 4: We modify hybrid 3 such that if the adversary ever recycles one of the ciphertext c_{1i} from the challenge proof in one of the encryption queries and this is a valid proof, then we abort.

There is negligible probability of aborting. To make a valid proof, the adversary has to sign the proof using her chosen verification key vk_{sots} . By the existential unforgeability of the strong one-time signature scheme, this verification key has to differ from the verification key vk'_{sots} used in the challenge. In other words, c_{1i} contains the wrong verification key. However, in the zaps, of which at least one is made using a correctly generated initial message, the adversary proves that c_{1i} does contain vk'_{sots} or alternatively c_{2i} contain t_z pseudorandom evaluations of vk_{sots} . By decrypting we could therefore get out such pseudorandom function evaluations on vk_{sots} . However, since vk_{sots} is different from vk'_{sots} used in the challenge, this implies a breach of the pseudorandomness of the pseudorandom function.

Hybrid 5: We modify hybrid 4 by making a (t_s, n) -threshold secret sharing w_1, \dots, w_n of 0 instead of secret sharing w . We encrypt these shares in $c_{1i} \leftarrow E_{pk_{1i}}(w_i, vk_{\text{sots}})$. This hybrid is identical to the simulation process.

Hybrid 4 and hybrid 5 are indistinguishable. We have ruled out that the adversary ever makes an extraction query, recycling a c_{1i} from the challenge. Using a hybrid argument on the chosen ciphertext attack security of the cryptosystems, the adversary cannot distinguish encryptions of shares of a threshold secret sharing of w from shares of a threshold secret sharing of 0. The remaining $n - t_z < t_s$ shares do not reveal anything.

Next, let us consider simulation-sound extractability. Here the adversary sees extraction keys, but not the simulation trapdoors of the common reference strings generated by SE_1 . It has access to a simulation oracle, and in the end it outputs a statement and a proof. By the unforgeability of the strong one-time signature scheme, she cannot reuse a strong verification key vk_{sots} used in a simulated proof. Let us look at a simulated reference string generated by SE_1 . Since the adversary does not know the seed for the pseudorandom function, she cannot encrypt a pseudorandom function evaluation of vk_{sots} . The zaps, of which at least one uses a correctly generated initial message, then tells us that c_{11}, \dots, c_{1n} contain a (t_s, n) -threshold secret sharing of w . Decrypting t_s of these ciphertexts, permits us to reconstruct the witness w .

A similar proof, shows that we have *statistical* $(0, t_s, t_z, n)$ -knowledge extraction. The point in this proof is that with overwhelming probability a random string does not contain a pseudorandom value r , so therefore c_{11}, \dots, c_{1n} must encrypt a (t_s, n) -threshold secret sharing of a witness for $x \in L$. \square

5 Multi-string NIZK Proofs from Bilinear Groups

We will use bilinear groups to construct a $(0, t_s, t_z, n)$ -simulation-sound NIZK proof for circuit satisfiability consisting of $\mathcal{O}((n+|C|)k)$ bits, where $|C|$ is the number of gates in the circuit and k is the security parameter specifying the size of the bilinear group elements. Typically, n is much smaller than $|C|$, so the complexity

matches the best known NIZK proofs for circuit satisfiability in the single common reference string model [GOS06b, GOS06a] that have proofs of size $\mathcal{O}(|C|k)$.

SETUP. We will use bilinear groups generated by $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^k)$ such that:

- p is a k -bit prime.
- \mathbb{G}, \mathbb{G}_T are cyclic groups of order p .
- g is a generator of \mathbb{G} .
- $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map such that $e(g, g)$ generates \mathbb{G}_T and for all $a, b \in \mathbb{Z}_p$ we have: $e(g^a, g^b) = e(g, g)^{ab}$.
- Group operations, group membership, and the bilinear map are efficiently computable.
- Given a description $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ it is verifiable that indeed it is a bilinear group and that g generates \mathbb{G} .
- There is a decoding algorithm that given a random string of $(n + 1)k$ bits interprets it as n random group elements. The decoding algorithm is reversible, such that given n group elements we can pick at random one of the $(n + 1)k$ -bit strings that decode to the n group elements.
- The length of the description of $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is at most $4k$ bits.²
- When working in the random multi-string model, we will assume \mathcal{G} simply outputs a uniformly random $4k$ -bit string, from which $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ can be sampled.

We use the decisional linear assumption introduced by Boneh, Boyen and Shacham [BBS04], which says that given group elements (f, g, h, f^r, g^s, h^t) it is hard to tell whether $t = r + s$ or t is random. Throughout the paper, we use bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^k)$ generated such that the DLIN assumption holds for \mathcal{G} .

Example. We will offer a class of candidates for DLIN groups as described above. Consider the elliptic curve $y^2 = x^3 + 1 \pmod q$, where $q = 2 \pmod 3$ is a prime. It is straightforward to check that a point (x, y) is on the curve. Furthermore, picking $y \in \mathbb{Z}_q$ at random and computing $x = (y^2 - 1)^{\frac{q+1}{3}} \pmod q$ gives us a random point on the curve. The curve has a total of $q + 1$ points, where we include also the point at infinity. When generating bilinear groups, we will pick p as a k -bit prime. We then let q be the smallest prime³ so $p|q + 1$ and define \mathbb{G} to be the order p subgroup of the curve. The target group is the order p subgroup of $\mathbb{F}_{q^2}^*$ and the bilinear map is the modified Weyl-pairing [BF03]. Verification of $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ being a group with bilinear maps is straightforward, since it corresponds to checking that p, q are primes so $p|q + 1$ and $q = 2 \pmod 3$ and g is an order p element on the curve. A random point in the group \mathbb{G} can be sampled by picking a random point (x, y) on the curve and raising it to $\frac{q+1}{p}$. Reverse sampling is possible, since multiplying a group element with a random point of order $\frac{q+1}{p}$ gives a random (x, y) on the curve that would generate the group element.

PSEUDORANDOM GENERATORS IN DLIN GROUPS Before proceeding, let us demonstrate that the DLIN assumption permits the construction of a pseudorandom number generator. Consider a DLIN group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Choose $x, y \leftarrow \mathbb{Z}_p^*$ at random and set $f = g^x, h = g^y$. Given random elements $u, v \leftarrow \mathbb{G}$,

²It is easy to modify the protocol to work whenever the description of the bilinear group is $\mathcal{O}(k)$ bits.

³In other words, q is the smallest prime in the arithmetic progression $3p - 1, 6p - 1, 9p - 1, \dots$. Granville and Pomerance [GP90] conjectured that it requires $\mathcal{O}(k^2)$ steps in this progression to encounter a prime q .

we can compute $w = u^{1/x}v^{1/y}$. The DLIN assumption says that (f, h, u, v, w) is indistinguishable from (f, h, u, v, r) , where r is a random group element from \mathbb{G} . In other words, we can create a pseudorandom generator $(x, y, u, v) \mapsto (g^x, g^y, u, v, u^{1/x}v^{1/y})$ that stretches our randomness with an extra group element.

We will need a bigger stretch, so let us generalize the construction above using the idea of synthesizers from Naor and Reingold [NR99]. We pick m pairs $(x_i, y_i) \leftarrow \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ and create corresponding $f_i = g^{x_i}, h_i = g^{y_i}$. We can now stretch $2n$ group elements $u_1, v_1, \dots, u_n, v_n$ with mn extra group elements by computing $w_{ij} := u_j^{1/x_i}v_j^{1/y_i}$.

If the n pairs of group elements (u_j, v_j) are chosen at random, then $(f_1, h_1, \dots, f_m, h_m, u_1, v_1, \dots, u_n, v_n, w_{11}, \dots, w_{mn})$ looks like a random $2m + 2n + mn$ -tuple of group elements. To see this, consider the following hybrid experiment $E_{I,J}$, where we pick w_{ij} at random for pairs (i, j) where $i < I \vee (i = I \wedge j < J)$ and compute the rest of the w_{ij} 's according to the method described above. We need to prove that the w_{ij} 's generated in respectively E_{11} and $E_{m,n+1}$ are indistinguishable.

Consider first experiments $E_{I,J}, E_{I,J+1}$ for $1 \leq I \leq m, 1 \leq J \leq n$. In case there is a non-uniform polynomial time adversary \mathcal{A} that can distinguish these two experiments, then we can break the DLIN assumption as follows. We have a challenge (f, h, u, v, w) and wish to know whether $w = u^{1/x}v^{1/y}$ or w is random. We let $f_I := f, h_I := h$ and generate all the other f_i, h_i 's according to the protocol. We set $u_J := u, v_J := v$ and $w_{IJ} := w$. For $i < I$ we pick w_{ij} at random. Also, for $i = I, j < J$ we pick w_{ij} at random. For $i = I, j > J$ we pick r_j, s_j at random and set $(u_j, v_j, w_{IJ}) = (f^{r_j}, h^{s_j}, g^{r_j+s_j})$. For $j < J$ we select (u_j, v_j) at random. Finally, for $i > I$ we compute all w_{ij} according to the protocol. If (u, v, w) is a linear tuple, we have the distribution from experiment $E_{I,J}$, whereas if (u, v, w) is a random tuple we have the distribution from experiment $E_{I,J+1}$. An adversary distinguishing these two experiments, therefore permits us to distinguish linear tuples from random tuples. We conclude the proof by observing $E_{I+1,1} = E_{I,n+1}$.

Observe, it is straightforward to provide a witness for (u, v, w) being a linear tuple. The witness consists of $\pi = u^{y/x}$. (u, v, w) is a linear tuple if and only if $e(u, h) = e(f, \pi)$ and $e(g, \pi v) = e(w, h)$. In other words, we can provide n^2 proofs π_{ij} for w_{ij} being correct. Furthermore, all these proofs consist of group elements and can be verified by checking a set of pairing product equations. It follows from Groth [Gro06] that there exists a simulation-sound NIZK proof of size $\mathcal{O}(mn)$ group elements for the w_{ij} 's having been computed correctly.

MULTI-STRING NIZK PROOFS FROM DLIN GROUPS. One could hope that the construction from Section 3 could be implemented efficiently using groups with a bilinear map. This strategy does not work because each common reference string is generated at random and independently of the others. This means that even if the common reference strings contain descriptions of groups with bilinear maps, most likely they are different and incompatible groups.

In our construction, we instead let all the common reference strings describe different groups and we also let the prover pick a group with a bilinear map. Our solution to the problem described above, is to translate simulation reference strings created by the authorities into simulation reference strings in the prover's group. This translation will require the use of a pseudorandom generator that we constructed earlier. As mentioned earlier this pseudorandom generator is constructed in such a way that there exist linear size simulation-sound NIZK proofs for a value being pseudorandom [Gro06].

Consider a common reference string with group \mathbb{G}_i and the prover's group \mathbb{G} . We will let the common reference string contain a random string r_i . The prover will choose a string s_i . Consider the pair of strings $(r_i \oplus s_i, s_i)$. Since strings can be interpreted as group elements, we have corresponding sets of group elements in respectively \mathbb{G}_i and \mathbb{G} . However, since r_i is chosen at random it is unlikely that both $r_i \oplus s_i$ corresponds to a pseudorandom value in \mathbb{G}_i and at the same time s_i corresponds to a pseudorandom value in \mathbb{G} . Of course, the prover has some degree of freedom in choosing the group \mathbb{G} , but if one is careful and chooses a pseudorandom generator that stretches the input sufficiently then one can use an entropy argument for it being unlikely that both strings are pseudorandom values.

Now we use non-interactive zaps and NIZK proofs to bridge the two groups. The prover will select s_i so

$r_i \oplus s_i$ is a pseudorandom value in \mathbb{G}_i specified by the common reference string and give an NIZK proof for this using that common reference string. In her own group, she gets n values s_1, \dots, s_n and proves that t_z of those are pseudorandom or C is satisfiable. In the simulation, she knows the simulation trapdoors for t_z reference strings and she can therefore simulate NIZK proofs of $r_i \oplus s_i$ being pseudorandom. This means, she can select the corresponding s_i 's as pseudorandom values and use this to prove that there are at least t_z pseudorandom values in her own group, so she does not need to know the satisfiability witness w for C being satisfiable to carry out the proof in her own bilinear group.

There is another technical detail to consider. We want the construction to be efficient in n . Therefore, instead of proving directly that there are t_z pseudorandom values or C is satisfiable, we use a homomorphically encrypted counter. In the simulation, we set the counter to 1 for each pseudorandom value and to 0 for the rest of the values in the prover's group. The homomorphic property enables us to multiply these ciphertexts and get an encrypted count of t_z . It is straightforward to prove that the count is t_z or C is satisfiable.

These ideas describe how to get soundness. We can set up the common reference strings such that they enable us to make simulation-sound NIZK proofs in their bilinear groups. With a few extra ideas, we then get a $(0, t_s, t_z, n)$ -simulation-sound NIZK proof for circuit satisfiability when $t_s + t_z > n$.

Common reference string/simulation reference string: Generate a DLIN group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^k)$. Generate a common reference string for a simulation-sound NIZK proof on basis of this group $\Sigma \leftarrow K_{\text{sim-sound}}(p, \mathbb{G}, \mathbb{G}_T, e, g)$ as in [Gro06]. Also, pick a random string $r \leftarrow \{0, 1\}^{61k}$. Output $\Sigma := (p, \mathbb{G}, \mathbb{G}_T, e, g, \sigma, r)$.

Provided one can sample groups from random strings, this can all be set up in the random multi-string model.

When generating a simulation reference string, use the simulator for the simulation-sound NIZK proof to generate $(\sigma, \tau) \leftarrow S_{\text{sim-sound}}(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Output Σ as described above and simulation trapdoor τ .

Proof: Given $(\Sigma_1, \dots, \Sigma_n), C, w$ so $C(w) = 1$ do the following. Pick a group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^k)$. Pick also keys for a strong one-time signature scheme $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(1^k)$. Encode vk_{sots} as a tuple of $\mathcal{O}(1)$ group elements from \mathbb{G} .

For each common reference string Σ_i do the following. Pick a pseudorandom value with 6 key pairs, 6 input pairs and 36 structured elements. This gives us a total of 60 group elements from \mathbb{G}_i . Concatenate the tuple of 60 group elements with vk_{sots} to get $\mathcal{O}(1)$ group elements from \mathbb{G}_i . Make a simulation-sound NIZK proof, using σ_i , for these $\mathcal{O}(1)$ group elements being of a form such that the first 60 of them constitute a pseudorandom value. From [Gro06] we know that the size of this proof is $\mathcal{O}(1)$ group elements from \mathbb{G}_i . Define $s_i \in \{0, 1\}^{61k}$ to be a random string such that $r_i \oplus s_i$ parses to the 60 elements from the pseudorandom value.

From now on we will work in the group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ chosen by the prover. Pick $pk := (f, h)$ as two random group elements. This gives us a CPA-secure cryptosystem [BBS04], encrypting a message $m \in \mathbb{G}$ with randomness $r, s \in \mathbb{Z}_p$ as $E_{pk}(m; r, s) := (f^r, h^s, g^{r+s}m)$. For each $i = 1, \dots, n$ we encrypt $1 = g^0$ as $c_i \leftarrow E_{pk}(1)$. Also, we take s_i and parse it as 60 group elements. Call this tuple z_i .

Make a non-interactive zap π using the group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ and combining techniques of [GOS06a] and [Gro06] for the following statement:

$$C \text{ satisfiable} \quad \vee \quad \left(\prod_{i=1}^n c_i \text{ encrypts } g^{t_z} \right. \\ \left. \wedge \forall i : c_i \text{ encrypts } g^0 \text{ or } g^1 \wedge (z_i \text{ is a pseudorandom value} \vee c_i \text{ encrypts } g^0) \right).$$

The zap consists of $\mathcal{O}(n + |C|)$ group elements and has perfect soundness.

Sign everything $sig \leftarrow \text{Sign}_{sk_{\text{sots}}}(vk_{\text{sots}}, C, \Sigma_1, s_1, \pi_1, c_1, \dots, \Sigma_n, s_n, \pi_n, c_n, p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, \pi)$.

The proof is $\Pi := (vk_{\text{sots}}, s_1, \pi_1, c_1, \dots, s_n, \pi_n, c_n, p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, \pi, sig)$.

Verification: Given common reference strings $\Sigma_1, \dots, \Sigma_n$, a circuit C and a proof as described above, do the following. For all i check the simulation-sound NIZK proofs π_i for $r_i \oplus s_i$ encoding a pseudorandom value in \mathbb{G}_i using common reference string σ_i . Verify $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is a group with a bilinear map. Verify the zap π . Verify the strong one-time signature on everything. Output 1 if all checks are ok.

Simulated proof: We are given reference strings $\Sigma_1, \dots, \Sigma_n$. t_z of them are simulation strings, where we know the simulation trapdoors τ_i for the simulation-sound NIZK proofs. We wish to simulate a proof for a circuit C being satisfiable.

We start by choosing a group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^k)$ and public key $f, h \leftarrow \mathbb{G}$. We create ciphertexts $c_i \leftarrow E_{pk}(g^1)$ for the t_z simulation reference strings, where we know the trapdoor τ_i , and set $c_i \leftarrow E_{pk}(g^0)$ for the rest. We also choose a strong one-time signature key pair $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(1^k)$.

For t_z of the common reference strings, we know the simulation key τ_i . This permits us to choose an arbitrary string s_i and simulate a proof π_i that $r_i \oplus s_i$ encodes a 60 element pseudorandom value. This means, we are free to choose s_i so it encodes a pseudorandom value z_i in \mathbb{G}^{60} . For the remaining $n - t_z < t_s$ reference strings, we select s_i so $r_i \oplus s_i$ does encode a pseudorandom value in \mathbb{G}_i and carry out a real simulation-sound NIZK proof π_i for it being a pseudorandom value concatenated with vk_{sots} .

For all i we have c_i encrypting g^b , where $b \in \{0, 1\}$. We have $\prod_{i=1}^n c_i$ encrypting g^{t_z} . We also have for the t_z simulation strings, where we know τ_i that s_i encodes a pseudorandom value, whereas for the other common reference strings we have c_i encrypts g^0 . This means we can create the non-interactive zap π without knowing C 's satisfiability witness.

Sign everything $sig \leftarrow \text{Sign}_{sk_{\text{sots}}}(vk_{\text{sots}}, C, \Sigma_1, s_1, \pi_1, c_1, \dots, \Sigma_n, s_n, \pi_n, c_n, p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, \pi)$.

The simulated proof is $\Pi := (vk_{\text{sots}}, s_1, \pi_1, c_1, \dots, s_n, \pi_n, c_n, p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, \pi, sig)$.

Theorem 11 *Assuming we have a DLIN group as described above, then the construction above gives us a $(0, t_s, t_z, n)$ -simulation-sound NIZK proof for circuit satisfiability, where the proofs have size $\mathcal{O}((n + |C|)k)$ bits. The proof has statistical $(0, t_s, t_z, n)$ -soundness. The scheme can be set up in the random multi-string model if we can sample groups with bilinear maps from random strings.*

Proof. We have already argued in the construction that if we can sample groups and group elements from random strings and vice versa given groups and group elements sample random strings that yield these group elements, then the common reference strings can be set up in the random strings model. Perfect completeness follows by straightforward verification.

Let us prove that we have statistical $(0, t_s, t_z, n)$ -soundness. Consider first an arbitrary group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ chosen by the prover. By assumption, it can be verified that this describes a group with a bilinear map.

We will now bound the probability of both $r_i \oplus s_i$ and s_i specifying pseudorandom values in their respective groups for a random choice of r_i . Consider first the probability that a random string s_i specifies a pseudorandom value in \mathbb{G}^{60} . There are at most 2^{24k} pseudorandom strings, since the 12 pairs (f_i, h_i) and the 12 pairs (u_j, v_j) fully define the pseudorandom value. 60 random group elements have at least $59k$ bits of entropy, so we get a probability of at most $2^{24k-59k} = 2^{-35k}$ of s_i specifying a pseudorandom value in \mathbb{G}^{60} . Similarly, for a random choice of r_i we have at most probability 2^{-35k} that $r_i \oplus s_i$ is a pseudorandom value in the group specified by the common reference string. With r_i, s_i both chosen at random, we have a

combined maximal probability of 2^{-70k} of both $r_i \oplus s_i$ and s_i specifying pseudorandom values. The prover can choose the group freely, giving her at most 2^{4k} different choices for describing the group \mathbb{G} and g . She can also choose s_i freely, giving her 2^{61k} possibilities. Since r_i is chosen at random, there is at most probability $2^{4k+61k-70k} = 2^{-5k}$ of it being possible to choose s_i and the group \mathbb{G} so both $r_i \oplus s_i$ and s_i specify pseudorandom values. With overwhelming probability, we can therefore assume that no honestly generated common reference string exists such that both $r_i \oplus s_i$ and s_i specify pseudorandom values in respectively \mathbb{G}_i and \mathbb{G} .

Any common reference string Σ_i that is honestly generated has overwhelming probability of having a common reference string σ_i for the simulation-sound NIZK with perfect soundness. Whenever the prover makes a proof using this string, she must therefore pick s_i so $r_i \oplus s_i$ is pseudorandom. Consequently, s_i does not specify a pseudorandom value in the group \mathbb{G} . The zap has perfect soundness, so it shows that C is satisfiable or c_i contains g^0 . Similarly, for any string Σ_i that is not honestly generated, the zap demonstrates that C is satisfiable or c_i contains g^0 or g^1 . Since at least $t_s > n - t_z$ strings are honestly generated, we see that if C is unsatisfiable, then $\prod_{i=1}^n c_i$ contains one of the values g^0, \dots, g^{t_z-1} . The zap therefore shows us that C must be satisfiable.

To argue computational $(0, t_s, t_z, n)$ -simulation-soundness, observe that simulated proofs are signed with a strong one-time signature. Since the signature scheme has existential unforgeability, the adversary must choose a different vk_{sots} that it has not seen in a simulation. Recall, whenever we make a simulation-sound NIZK using a particular common reference string Σ_i , we concatenate vk_{sots} to $r_i \oplus s_i$ to get the statement we wish to prove. By the simulation-soundness of the NIZK proofs on honestly generated strings, we can not forge such a proof even though we have already seen simulated proofs. Therefore, $r_i \oplus s_i$ must be a pseudorandom string. We can now argue $(0, t_s, t_z, n)$ -simulation-soundness just as we argued $(0, t_s, t_z, n)$ -soundness.

It remains to prove computational $(0, t_s, t_z, n)$ -zero-knowledge. Reference string indistinguishability follows from the reference string indistinguishability of the simulation-sound NIZK proofs. We will now consider simulation indistinguishability, so consider a case where the adversary sees simulated reference strings and gets the simulation trapdoors that allow the simulation of proofs for the reference strings. The adversary, chooses a set of common reference strings and receives a proof generated with the satisfiability witness for C or alternatively a simulated proof and wants to distinguish between the two possibilities.

Let us start with a simulated proof and compare it with a hybrid experiment, where we use the satisfiability witness for C in the non-interactive zap. By the computational witness-indistinguishability of the zap, the adversary cannot tell these two experiments apart. Next, let us choose all c_i 's as encryptions of g^0 . By the semantic security of the cryptosystem, the adversary cannot detect this change. We already select s_i so $r_i \oplus s_i$ specifies a pseudorandom value for the reference strings not generated by S_1 . Let us switch to also selecting s_i so $r_i \oplus s_i$ specify a pseudorandom value in the common reference strings where we do know the simulation trapdoor. By the pseudorandomness of the strings, the adversary cannot detect this change either. Finally, instead of simulating the proofs for $r_i \oplus s_i$ specifying a pseudorandom value in \mathbb{G}_i , let us make a real proof. By the zero-knowledge property of the simulated reference strings for the simulation-sound NIZK proofs, the adversary cannot distinguish here either. With this last modification, we have actually ended up constructing proofs exactly as a real prover with access to a satisfiability witness does, so we have $(0, t_s, t_z, n)$ -zero-knowledge. \square

6 The UC Framework

In the rest of the paper, we will work in Canetti's UC framework. The universal composability (UC) framework, see [Can01] for a detailed description, is a strong security model capturing security of a protocol under concurrent execution of arbitrary protocols. We model everything not directly related to the protocol through an environment \mathcal{Z} . The environment can at its own choosing give inputs to the parties running the protocol,

and according to the protocol specification, the parties can give outputs to the environment. In addition, there is an adversary \mathcal{A} that attacks the protocol. \mathcal{A} can communicate freely with the environment. It can adaptively corrupt parties, in which case it learns the entire history of that party and gains complete control over the actions of this party. The environment learns whenever a party is corrupted.

To model security we use a simulation paradigm. We specify the functionality \mathcal{F} that the protocol should realize. The functionality \mathcal{F} can be seen as a trusted party that handles the entire protocol execution and tells the parties what they would output if they executed the protocol correctly. In the ideal process, the parties simply pass on inputs from the environment to \mathcal{F} and whenever receiving a message from \mathcal{F} they output it to the environment. In the ideal process, we have an ideal process adversary \mathcal{S} . \mathcal{S} does not learn the content of messages sent from \mathcal{F} to the parties, but is in control of when, if ever, a message from \mathcal{F} is delivered to the designated party. \mathcal{S} can corrupt parties, at the time of corruption it will learn all inputs the party has received and all outputs it has sent to the environment. As the real world adversary, \mathcal{S} can freely communicate with the environment.

We now compare these two models and say that the protocol securely realizes \mathcal{F} if no environment can distinguish between the two worlds. This means, the protocol is secure, if for any polynomial time \mathcal{A} running in the real world, there exists a polynomial time \mathcal{S} running in the ideal process with \mathcal{F} , so no non-uniform polynomial time environment can distinguish between the two worlds.

One of our goals is to show that any well-formed functionality can be securely realized in the multi-string model. By well-formed functionality, we mean a functionality that is oblivious of corruptions of parties, runs in polynomial time, and in case all parties are corrupted it reveals the internal randomness used by the functionality to the ideal process adversary. This class contains all functionalities that we can reasonably expect to implement with multi-party computation, because an adversary can always corrupt a party and just have it follow the protocol, in which case the other parties in the protocol would never learn that it was corrupted.

IDEAL FUNCTIONALITIES Let us formalize the multi-string model in the UC framework. Figure 1 gives an ideal multi-string functionality $\mathcal{F}_{\text{MCRS}}$.

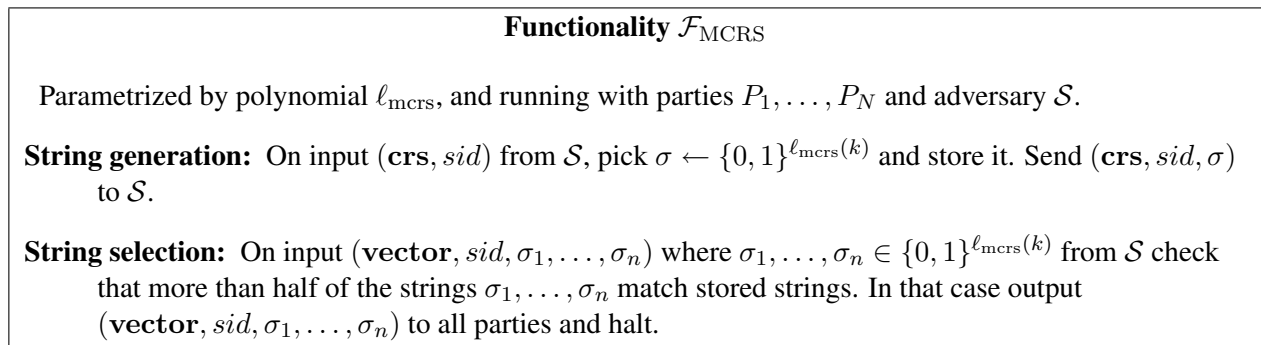


Figure 1: The ideal multi-string generator.

We will assume parties can broadcast their messages and we make this assumption explicit by giving them access to an ideal broadcast functionality. Ideal broadcast permits each party to broadcast messages to other parties. We remark that broadcast can be securely realized in a constant number of rounds if authenticated communication is available [GL05]. Furthermore, authenticated communication can be securely realized using digital signatures, so one possible setup is that the parties somehow have exchanged verification keys for a digital signature scheme.

Functionality \mathcal{F}_{BC}

Running with parties P_1, \dots, P_n and adversary \mathcal{S} .

Broadcast: On input $(\text{broadcast}, sid, ssid, m)$ from P_i , send $(\text{broadcast}, sid, ssid, P_i, m)$ to all parties and \mathcal{S} . Ignore future $(\text{broadcast}, sid, ssid, \cdot)$ inputs from P_i .

Figure 2: The ideal authenticated broadcast functionality.

7 Tools

This section will present a number of tools we will need in our constructions.

PSEUDORANDOM CRYPTOSYSTEM WITH PSEUDORANDOM KEYS. A cryptosystem $(K_{\text{pseudo}}, E, D)$ has pseudorandom ciphertexts of length $\ell_E(k)$ if for all non-uniform polynomial time adversaries \mathcal{A} we have

$$\begin{aligned} & \Pr \left[(pk, dk) \leftarrow K_{\text{pseudo}}(1^k) : \mathcal{A}^{E_{pk}(\cdot)}(pk) = 1 \right] \\ & \approx \Pr \left[(pk, dk) \leftarrow K_{\text{pseudo}}(1^k) : \mathcal{A}^{R_{pk}(\cdot)}(pk) = 1 \right], \end{aligned}$$

where $R_{pk}(m)$ runs $c \leftarrow \{0, 1\}^{\ell_E(k)}$ and returns c . We require that the cryptosystem have errorless decryption.

Trapdoor permutations imply pseudorandom cryptosystems, since we can use the Goldreich-Levin hardcore bit [GL89] of a trapdoor permutation to make a one-time pad. For setting up our scheme in the common random string model, we will require that the cryptosystem has a pseudorandom public key as well. Pseudorandom cryptosystems with pseudorandom keys can be built from various assumptions such as RSA, DDH and DLIN.

TAG-BASED SIMULATION-SOUND TRAPDOOR COMMITMENT. A tag-based commitment scheme has four algorithms. The key generation algorithm $K_{\text{tag-com}}$ produces a commitment key ck as well as a trapdoor key tk . There is a commitment algorithm that takes as input the commitment key ck , a message m and any tag tag and outputs a commitment $c := \text{Com}_{ck}(tag; m; r)$. To open a commitment c with tag tag we reveal m and the randomness r . Anybody can now verify $c = \text{Com}_{ck}(tag; m; r)$. As usual, the commitment scheme must be both hiding and binding.

In addition, to these two algorithms there are also a couple of trapdoor algorithms $T_{\text{com}}, T_{\text{open}}$ that allow us to create an equivocal commitment and later open this commitment to any value we prefer. We create an equivocal commitment and an equivocation key as $(c, ek) \leftarrow T_{\text{com}_{tk}}(tag)$. Later we can open it to any message m as $r \leftarrow T_{\text{open}_{ek}}(tag; m)$, such that $c = \text{Com}_{ck}(tag; m; r)$.

We require that equivocal commitments and openings are indistinguishable from real openings. For all non-uniform polynomial time adversaries \mathcal{A} we have

$$\begin{aligned} & \Pr \left[(ck, tk) \leftarrow K_{\text{tag-com}}(1^k) : \mathcal{A}^{\mathcal{R}(\cdot, \cdot)}(ck) = 1 \right] \\ & \approx \Pr \left[(ck, tk) \leftarrow K_{\text{tag-com}}(1^k) : \mathcal{A}^{\mathcal{O}(\cdot, \cdot)}(ck) = 1 \right], \end{aligned}$$

where $\mathcal{R}(m, tag)$ returns a randomly selected randomizer and $\mathcal{O}(m, tag)$ computes $(c, ek) \leftarrow T_{\text{com}_{tk}}(tag, m); r \leftarrow T_{\text{open}_{ek}}(tag, m)$ and returns r . Both oracles ignore tags that have already been submitted once.

The tag-based simulation-soundness property means that a commitment using tag remains binding even if we have made equivocations for commitments using different tags. For all non-uniform polynomial time

adversaries \mathcal{A} we have

$$\Pr \left[(ck, tk) \leftarrow K_{\text{tag-com}}(1^k); (tag, c, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(ck) : tag \notin Q \text{ and } c = \text{Com}_{ck}(tag; m_0; r_0) = \text{Com}_{ck}(tag; m_1; r_1) \text{ and } m_0 \neq m_1 \right] \approx 0,$$

where $\mathcal{O}(\text{Com}, tag)$ computes $(c, ek) \leftarrow \text{Tcom}_{tk}(tag)$, returns c and stores (c, tag, ek) , and $\mathcal{O}(\text{Open}, c, m, tag)$ returns $r \leftarrow \text{Topen}_{ek}(tag, m)$ if (c, tag, ek) has been stored, and where Q is the list of tags for which equivocal commitments have been made by \mathcal{O} .

Tag-based simulation-sound trapdoor commitment were implicitly used in Di Crescenzo, Ishai and Ostrovsky [DIO98]. The explicit term was coined by Garay, MacKenzie and Yang [GMY06], while the definition presented here is from MacKenzie and Yang [MY04]. In addition, since we are working over random strings, we want $K_{\text{tag-com}}$ to output public keys that are random or pseudorandom, i.e., we can use $pk \leftarrow \{0, 1\}^{\ell_{\text{tag-com}}}$ instead of a public key.

TAG-BASED SIMULATION-EXTRACTABLE COMMITMENT SCHEME. We will need something that is stronger than tag-based simulation-sound commitments, namely a tag-based simulation-extractable commitment. This is a tag-based simulation-sound trapdoor commitment scheme with an additional algorithm `Extract` that given an extraction key is able to extract the message inside the commitment. More precisely, with the trapdoor we can make trapdoor commitments, however, for all other tags, the adversary will end up making unconditionally binding commitments.

A tag-based simulation-extractable commitment scheme consists of five polynomial time algorithms $(K_{\text{se-com}}, \text{Com}, \text{Tcom}, \text{Topen}, \text{Extract})$, such that the first 4 constitute a tag-based trapdoor commitment scheme, and such that $(K_{\text{se-com}}, \text{Com}, \text{Extract})$ is a semantically secure cryptosystem. It will have the property that a non-uniform adversary with access to trapdoor openings of commitments and the extraction key, still cannot create a new commitment and opening thereof, such that the message it opens to differs from the extracted message.

For all non-uniform polynomial time adversaries \mathcal{A} we have

$$\Pr \left[Q := \emptyset; (\sigma, \tau, \xi) \leftarrow K_{\text{se-com}}(1^k); (tag, m, r) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\sigma, \xi); c := \text{Com}_{\sigma}(tag; m; r) : \text{Extract}_{\xi}(tag, c) \neq m \text{ and } tag \notin Q \right] \approx 0,$$

where $\mathcal{O}(\text{Com}, tag)$ computes $(c, ek) \leftarrow \text{Tcom}_{\tau}(tag)$, returns c and stores (c, tag, ek) , and $\mathcal{O}(\text{Open}, c, m, tag)$ returns $r \leftarrow \text{Topen}_{ek}(tag, m)$ if (c, tag, ek) has been stored, and where Q is the list of tags for which equivocal commitments have been made by \mathcal{O} .

We will construct a tag-based simulation-extractable commitment scheme from the tools in this section. We use a tag-based simulation-sound trapdoor commitment scheme to commit to each bit of m . If m has length ℓ this gives us commitments c_1, \dots, c_{ℓ} . When making trapdoor commitments, we can use the trapdoor key tk to create equivocal commitments c_1, \dots, c_{ℓ} that can be opened to any bit we like.

We still have an extraction problem, we may be unable to extract a message from tag-based commitments created by the adversary. To solve this problem we choose to encrypt the openings of the commitments. Now we can extract messages, but we have reintroduced the problem of equivocation. In a trapdoor commitment we may know two different openings of a commitment c_i to respectively 0 and 1, however, if we encrypt the opening then we are stuck with one possible opening. This is where the pseudorandomness property of the cryptosystem comes in handy. We can simply make two encryptions, one of an opening to 0 and one of an opening to 1. Since the ciphertexts are pseudorandom, we can open the ciphertext containing the opening we want and claim that the other ciphertext was chosen as a random string. To recap, the idea so far to commit to a bit b is to make a tag-based simulation-sound trapdoor commitment c_i to this bit, and create a ciphertext $c_{i,b}$ containing an opening of c_i to b , while choosing $c_{i,1-b}$ as a random string.

These are the main ideas, we now present the protocol in Figure 3.

Random key: Return $\sigma := (ck, pk) \leftarrow \{0, 1\}^{\ell_{\text{tag-com}}(k)} \times \{0, 1\}^{\ell_{\text{pseudo}}(k)}$

Simulation-extraction key:

1. $(ck, tk) \leftarrow K_{\text{tag-com}}(1^k)$
2. $(pk, dk) \leftarrow K_{\text{pseudo}}(1^k)$
3. Return $\sigma = (ck, pk)$, $\tau = (\sigma, tk)$, $\xi = (\sigma, xk)$

Commitment: On input (σ, tag, m) and randomizers as described below do

1. For $i = 1$ to ℓ select r_i at random and let $c_i := \text{Com}_{ck}(tag, i; m_i; r_i)$
2. For $i = 1$ to ℓ select R_{i, m_i} at random and set $c_{i, m_i} = E_{pk}(r_i; R_{i, m_i})$ and choose $c_{i, 1-m_i}$ as a random string.
3. Return $c := (c_1, c_{10}, c_{11}, \dots, c_\ell, c_{\ell 0}, c_{\ell 1})$

Opening: On input $(tag, c, m, r_1, R_{1, m_1}, \dots, r_\ell, R_{\ell, m_\ell})$ do

1. Verify that for all i we have $c_i = \text{Com}_{ck}(tag, i; m_i; r_i)$
2. Verify that for all i we have $c_{i, m_i} = E_{pk}(r_i; R_{i, m_i})$
3. Return 1 if all checks work out, else return 0

Trapdoor commitment: On input $\tau = (\sigma, tk)$ do

1. For $i = 1$ to ℓ let $(c_i, ek_i) \leftarrow \text{Tcom}_{tk}(tag, i)$ and let r_{i0}, r_{i1} be equivocations so $c_i = \text{Com}_{ck}(tag, i; 0; r_{i0}) = \text{Com}_{ck}(tag, i; 1; r_{i1})$.
2. For $i = 1$ to ℓ select randomness $R_{i, b}$ and set $c_{ib} := E_{pk}(r_{ib}; R_{i, b})$.
3. Return $c := (c_1, c_{10}, c_{11}, \dots, c_\ell, c_{\ell 0}, c_{\ell 1})$ and $ek := (\sigma, r_1, R_{1,0}, R_{1,1}, \dots, r_\ell, R_{\ell,0}, R_{\ell,1})$.

Trapdoor opening: On input (tag, ek, c, m) return $(r_{1, m_1}, R_{1, m_1}, \dots, r_{\ell, m_\ell}, R_{\ell, m_\ell})$.

Extraction: On input $(tag, (\sigma, dk), c)$ use the decryption key to decrypt the ciphertexts c_{ib} . In case, we for i have exactly one ciphertext c_{ib} that decrypts to r_{ib} so $c_i = \text{Com}_\sigma(tag, i; b; r_{ib})$, we set $m_i := b$. In case all these processes succeed, we return the concatenation m , else we return \perp .

Figure 3: Tag based simulation-extractable commitment.

Theorem 12 *Tag-based simulation-extractable commitment schemes exist with pseudorandom keys if pseudorandom cryptosystems with pseudorandom keys exist.*

Proof. Tag-based simulation-sound trapdoor commitments with pseudorandom keys can be built from one-way functions, so we have the tools needed in the construction. This also shows that we have pseudorandom keys for the tag-based simulation-extractable commitment scheme.

We now need to prove that even after seeing trapdoor commitments and openings, it is hard to come up with a commitment with a different tag and open this commitment to a message that differs from the extraction. Consider first the case, where the adversary for some index i creates c_i, c_{i0}, c_{i1} so both c_{i0} and c_{i1} decrypt to valid openings of c_i to respectively 0 and 1. Since tag has not been used before, we have not used tag, i in any commitment we have trapdoor opened before, so we have broken the simulation-sound binding property of the tag-based simulation-sound trapdoor commitment. The errorless decryption property of the pseudorandom cryptosystem now tells us that if the adversary opens all triples c_i, c_{i0}, c_{i1} successfully

to either 0 or 1, then we get the opening when decrypting.

We also need to prove that we have the trapdoor property. We will modify the trapdoor oracle in several steps and show that \mathcal{A} cannot tell the difference. Let us start with the oracle that on input (tag, m) returns a randomly chosen randomizer $r_1, R_{1,m_1}, c_{1,1-m_1}, \dots, r_\ell, R_{\ell,m_\ell}, c_{1,1-m_\ell}$. Instead of making commitments $c_i := \text{Com}_{ck}(tag, i; m_i; r_i)$, we may instead run $(c_i, ek_i) \leftarrow \text{Tcom}_{tk}(tag, i); r_i \leftarrow \text{Topen}_{ek_i}(m_i)$ and use r_i as the randomizer. By the trapdoor property of the tag-based simulation-sound commitment the two oracles are indistinguishable to \mathcal{A} .

Next, consider the trapdoor oracle, where we make trapdoor openings to both r_{i0} and r_{i1} so $c_i = \text{Com}_{ck}(tag, i; b; r_{i,b})$ for both $b = 0$ and $b = 1$. We encrypt $r_{i,b}$ with randomness $R_{i,b}$. We then return $r_i, R_{i,m_i}, c_{i,1-m_i}$. By the pseudorandomness of the ciphertexts, this is indistinguishable from the previous oracle. \square

8 UC Commitment in the Multi-String Model

We will now show how to securely realize the following ideal UC commitment functionality $\mathcal{F}_{\text{COM}}^{1:N}$ in the multi-string model.

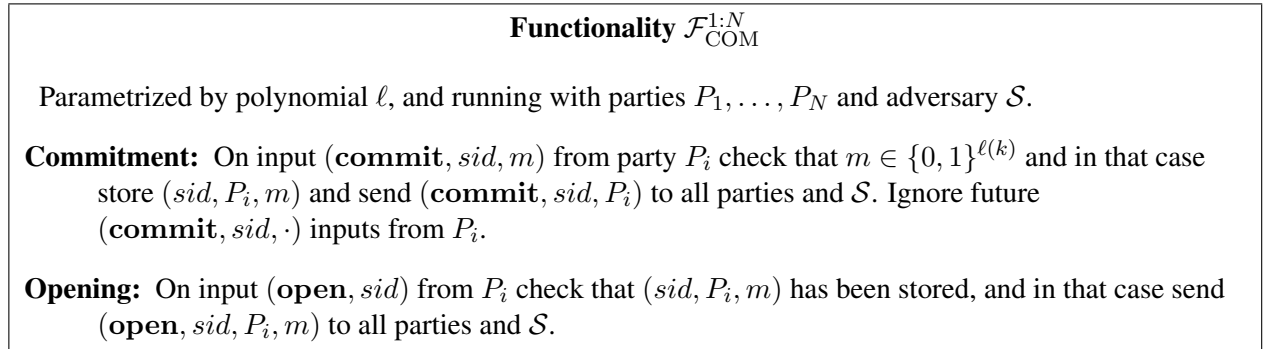


Figure 4: The ideal commitment functionality.

Before describing our UC commitment protocol, let us offer some intuition. To prove that our UC commitment is secure, we will describe an ideal process adversary \mathcal{S} that interacts with $\mathcal{F}_{\text{COM}}^{1:N}$ and makes a black-box simulation of \mathcal{A} running with $\mathcal{F}_{\text{MCRS}}$ and P_1, \dots, P_N . There are two general types of issues that can come up in the ideal process simulation. First, when $\mathcal{F}_{\text{COM}}^{1:N}$ tells \mathcal{S} that a party has committed to some message, \mathcal{S} does not know which message it is, however, \mathcal{S} has to simulate to \mathcal{A} that this party makes a UC commitment. Therefore, we want to be able to make trapdoor commitments and later open them to any value. Second, when a corrupt party controlled by \mathcal{A} sends a UC commitment, then \mathcal{S} needs to input some message to $\mathcal{F}_{\text{COM}}^{1:N}$. In this case, we therefore need to extract the message from the UC commitment.

As a tool to get both the trapdoor/simulation property and at the same time the extractability property, we will use a tag-based simulation-extractable commitment. Our idea in constructing a UC commitment is to use each of the n common random strings output by $\mathcal{F}_{\text{MCRS}}$ as a public key for a tag-based simulation-extractable commitment scheme. This gives us a set of n commitment schemes, of which at least $t = \lceil \frac{n+1}{2} \rceil$ are secure. Without loss of generality, we will from now on assume we have exactly t secure commitment schemes. In the ideal process, the ideal process adversary simulates $\mathcal{F}_{\text{MCRS}}$ and can therefore pick the strings as simulation-extractable public keys where it knows both the simulation trapdoors and the extraction keys.

To commit to a message m , a party makes a (t, n) -threshold secret sharing of it and commits to the n secret share using the n public keys specified by the random strings. When making a trapdoor commitment, \mathcal{S} makes honest commitments to $n - t$ random shares for the adversarial keys, and trapdoor commitments with the t simulation-extractable keys. Since the adversary knows at most $n - t < t$ shares, we can later

open the commitment to any message we want by making suitable trapdoor openings of the latter t shares. To extract a message m from a UC commitment made by the adversary, we extract t shares from the simulation-extractable commitments. We can now combine the shares to get the adversarial message.

One remaining issue is when the adversary recycles a commitment or parts of it. This way, we may risk that it uses a trapdoor commitment made by an honest party, in which case we are unable to extract a message. To guard against this problem, we will let the tag for the simulation-extractable commitment scheme contain the identity of the sender P_i , forcing the adversary to use a different tag, which in turn enables us to extract.

Another problem arises when the adversary corrupts a party, which enables it to send messages on behalf of this party. At this point, however, we learn the message so we just need to force it to reuse the same message if it reuses parts of the trapdoor commitment. We therefore introduce a second commitment scheme, which will be a standard trapdoor commitment scheme, and use this trapdoor commitment scheme to commit to the shares of the message. The tag for the simulation-extractable commitment will include this trapdoor commitment. Therefore, if reusing a tag, the adversary must also reuse the same trapdoor commitment given by this tag, which in turn computationally binds her to use the same share as the one the party committed to before being corrupted.

These ideas give us a UC commitment scheme in the multi-string model. As an additional bonus, the protocol is non-interactive except for a little coordination to ensure that everybody received the same commitment.

Commitment: On input (**vector**, sid , $(ck_1, \sigma_1), \dots, (ck_n, \sigma_n)$) from $\mathcal{F}_{\text{MCRS}}$ and (**commit**, sid , m) from \mathcal{Z} , the party P_i does the following. She makes a (t, n) -threshold secret sharing s_1, \dots, s_n of m . She picks randomizers r_j and makes commitments $c_j := \text{Com}_{ck_j}(s_j; r_j)$. She also picks randomizers R_j and makes tag-based commitments $C_j := \text{Com}_{\sigma_j}((P_i, c_j); s_j; R_j)$. The commitment is $c := (c_1, C_1, \dots, c_n, C_n)$. She broadcasts (**broadcast**, sid , c).

Receiving commitment: A party on input (**vector**, sid , $(ck_1, \sigma_1), \dots, (ck_n, \sigma_n)$) from $\mathcal{F}_{\text{MCRS}}$ and (**broadcast**, sid , P_i , c) from \mathcal{F}_{BC} broadcasts (**broadcast**, sid , P_i , c).

Once he receives similar broadcasts from all parties, all containing the same P_i, c , he outputs (**commit**, sid , P_i) to the environment.

Opening commitment: Party P_i wishing to open the commitment broadcasts (**open**, sid , $s_1, r_1, R_1, \dots, s_n, r_n, R_n$).

Receiving opening: A party receiving an opening (**open**, sid , $P_i, s_1, r_1, R_1, \dots, s_n, r_n, R_n$) from \mathcal{F}_{BC} to a commitment he received earlier, checks that all commitments are correctly formed $c_j = \text{Com}_{ck_j}(s_j; r_j)$ and $C_j = \text{Com}_{\sigma_j}((P_i, c_j); s_j; r_j)$. He also checks that s_1, \dots, s_n all are valid shares of a (t, n) -threshold secret sharing of some message m . In that case he outputs (**open**, sid , P_i, m).

Theorem 13 *The protocol securely realizes $\mathcal{F}_{\text{COM}}^{1:N}$ in the $(\mathcal{F}_{\text{BC}}, \mathcal{F}_{\text{MCRS}})$ -hybrid model, assuming tag-based simulation-extractable commitment schemes with pseudorandom keys exist in the common random string model.*

Proof. We describe the ideal-process adversary \mathcal{S} and sketch why it is secure along the way. It will run a black-box simulation of \mathcal{A} . In particular, it will simulate the parties P_1, \dots, P_N and the ideal functionalities $\mathcal{F}_{\text{MCRS}}$ and \mathcal{F}_{BC} . The dummy parties that are actually involved in the protocol and communicate with \mathcal{Z} are written as $\tilde{P}_1, \dots, \tilde{P}_N$.

Communication: Forward all communication between \mathcal{A} and \mathcal{Z} . Also, whenever \mathcal{A} delivers a message to a party P_i , simulate this delivery.

Common random strings: Whenever \mathcal{A} asks $\mathcal{F}_{\text{MCRS}}$ for a common random string, select $(ck, tk) \leftarrow K_{\text{trapdoor}}(1^k)$ and $(\sigma, \tau, \xi) \leftarrow K_{\text{se-com}}(1^k)$ and return $(\text{crs}, \text{sid}, (ck, \sigma))$, while storing $(ck, tk, \sigma, \tau, \xi)$.

When \mathcal{A} inputs $(\text{vector}, \text{sid}, (ck_1, \sigma_1), \dots, (ck_n, \sigma_n))$ to $\mathcal{F}_{\text{MCRS}}$ check that more than half the pairs $(ck_1, \sigma_1), \dots, (ck_n, \sigma_n)$ match the stored public keys. In that case, send $(\text{vector}, \text{sid}, (ck_1, \sigma_1), \dots, (ck_n, \sigma_n))$ to all parties and halt the simulation of $\mathcal{F}_{\text{MCRS}}$. Note, we only need t stored keys, so if there are more than t honest key pairs, we just act as if we only knew t of the trapdoors.

Commitment by honest party: On receiving $(\text{commit}, \text{sid}, P_i)$ from $\mathcal{F}_{\text{COM}}^{1:N}$ we learn that P_i has made a commitment, albeit we do not know the message. We wait until \mathcal{A} has submitted reference strings to $\mathcal{F}_{\text{MCRS}}$ and delivers them to P_i .

We select a (t, n) -threshold secret sharing s_1, \dots, s_n of 0. For the $n - t$ reference strings where we do not know the trapdoor keys, we commit to s_j as $c_j := \text{Com}_{tk}(s_j; r_j)$ and $C_j := \text{Com}_{\sigma_j}(P_i, c_j; s_j; R_j)$. For the t reference strings where we do know the trapdoor keys, we make trapdoor commitments $(c_j, ek_j) \leftarrow \text{Tcom}(tk)$ and $(C_j, EK_j) \leftarrow \text{Tcom}_{\tau_j}(P_i, c_j)$. We simulate broadcasting $(\text{broadcast}, \text{sid}, c_1, C_1, \dots, c_n, C_n)$.

The process for receiving a commitment is exactly the same as in the protocol, when simulated parties see the commitments they broadcast it. When everybody has broadcast, they are supposed to output $(\text{commit}, \text{sid}, P_i)$ to the environment. \mathcal{S} therefore delivers the corresponding commitment message from $\mathcal{F}_{\text{COM}}^{1:N}$ to the dummy party.

Opening: When \mathcal{S} receives $(\text{open}, \text{sid}, P_i, m)$ from $\mathcal{F}_{\text{COM}}^{1:N}$ it means that \tilde{P}_i has been instructed to open the commitment, and it was a commitment to m . We recall the $n - t$ shares that we committed to honestly, and fit them into a (t, n) -threshold secret sharing s_1, \dots, s_n of m . We open the $n - t$ commitments c_j, C_j correctly. We then trapdoor open the t commitments c_j, C_j where we know the corresponding equivocation keys as $r_j \leftarrow \text{Topen}_{ek_j}(s_j)$ and $R_j \leftarrow \text{Topen}_{EK_j}((P_i, c_j), s_j)$. We broadcast $(\text{broadcast}, \text{sid}, s_1, r_1, R_1, \dots, s_n, r_n, R_n)$.

Receiving an opening: On receiving an opening of an earlier received commitment, we check that the commitments contains a consistent (t, n) -threshold secret sharing of s_1, \dots, s_n of a message m and for all j we have $c_j = \text{Com}_{ck_j}(s_j; r_j)$ and $C_j = \text{Com}_{\sigma_j}(P_i, c_j; s_j; R_j)$. In that case, we deliver $(\text{open}, \text{sid}, P_i, m)$ from $\mathcal{F}_{\text{COM}}^{1:N}$ to our dummy party that outputs the opening to \mathcal{Z} .

Corruption: In case a party P_i is corrupted, we corrupt the corresponding dummy party \tilde{P}_i . We need to simulate the history of this party. If the party has not yet made a commitment, this is easy since there is no history to simulate. If the party has already opened the commitment, we just need to reveal the randomness used in generating the one-time signature.

If the party has made a commitment but not yet opened it, we must simulate an opening of it. On corrupting \tilde{P}_i , we learn the message it committed to, so we can use the opening simulation for honest parties described earlier.

Commitment by corrupt party: When a corrupt party makes a commitment $(c_1, C_1, \dots, c_n, C_n)$ so our simulated party would output $(\text{commit}, \text{sid}, P_i)$, we need to input some message to $\mathcal{F}_{\text{COM}}^{1:N}$ so we can make the corresponding dummy party output this in the ideal process.

We use the extraction keys, to extract t committed values $s_j \leftarrow \text{Extract}_{\xi_j}((P_i, c_j), C_i)$. The only case, where we cannot do this is when the tag (P_i, c_j) has been used before by P_i , because then it may be a trapdoor commitment we are looking at. However, this can only happen if P_i used (P_i, c_j) as a tag

when it was honest, and then upon corruption we have made a trapdoor opening of c_j to some s_j and therefore do not need to do any extraction.

We then reconstruct m from these shares and input $(\text{commit}, \text{sid}, m)$ to $\mathcal{F}_{\text{COM}}^{1:N}$ on behalf of the dummy party. In case we did not manage to extract a message, we input $m := 0$ to $\mathcal{F}_{\text{COM}}^{1:N}$, which is ok as long as we do not end up in a situation, where we need to ask $\mathcal{F}_{\text{COM}}^{1:N}$ to open the commitment. This causes $\mathcal{F}_{\text{COM}}^{1:N}$ to send out $(\text{commit}, \text{sid}, P_i)$ messages to all dummy parties that we can deliver when needed in the simulation.

Opening by corrupt party: When a corrupt party wants to open a commitment, we check the opening and if acceptable we input $(\text{open}, \text{sid})$ to $\mathcal{F}_{\text{COM}}^{1:N}$. If any honest party receives the opening, we deliver the message $(\text{open}, \text{sid}, P_i, m)$ to the corresponding dummy party \tilde{P}_j that outputs it to the environment.

To see that this gives us a good simulation, consider the following hybrid experiments for adversary \mathcal{A} and environment \mathcal{Z} .

Hybrid 1: This is the protocol executed with \mathcal{A} and environment \mathcal{Z} .

Hybrid 2: This is the protocol, where we store $(ck, tk, \sigma, \tau, \xi)$ and return (ck, σ) , whenever \mathcal{A} queries $\mathcal{F}_{\text{MCRS}}$ for a common reference string.

Since both commitment schemes have pseudorandom keys, hybrid 1 and 2 cannot be distinguished.

Hybrid 3: This is hybrid 2 modified such that honest party P_i for t commitments where it knows the key, creates equivocal commitments using the trapdoor keys, instead of making real commitments. To produce the openings, it then uses the equivocation keys to generate randomizers so the commitments open to the relevant shares.

Hybrid 2 and hybrid 3 are indistinguishable due to the trapdoor properties of the commitment schemes.

Hybrid 4: We modify hybrid 3 such that when an honest party P_i makes a commitment, it uses a (t, n) -threshold secret sharing of 0 instead of a threshold secret sharing of m . In the opening phase, it opens the $n - t$ pairs (c_j, C_j) where it does not know the trapdoors honestly to the s_j it committed to. It reconstructs shares s_j for the t equivocal commitments so s_1, \dots, s_n is a (t, n) -threshold secret sharing of m . It then opens the equivocal commitments to these values.

Hybrid 3 and hybrid 4 are perfectly indistinguishable, since $n - t < t$ shares in a (t, n) -threshold secret sharing scheme do not reveal anything about m .

Hybrid 5: We now turn to modify the way we handle corrupt parties. Whenever a corrupt party P_i submits a commitment $(c_1, C_1, \dots, c_n, C_n)$ to \mathcal{F}_{BC} , we want to extract a message.

For any of the t C_j 's where we know the key, there are two cases to consider. One case is where (P_i, c_j) has been used as a tag when P_i was still honest. In this case, we learned an opening s_j, r_j of c_j upon corruption, and will therefore consider s_j the share. The second case is when (P_i, c_j) has not been used as a tag in a simulation-extractable commitment. In that case, we can extract a share s_j .

We now have t shares, so we can recombine them to get a possible message m . We input $(\text{commit}, \text{sid}, m)$ on behalf of \tilde{P}_i . In case anything fails, we input $m := 0$ on behalf of \tilde{P}_i .

Hybrid 4 and hybrid 5 are indistinguishable. The problem arises if the extracted m does not match the opening. There are two ways this could happen. One possibility is that c_j created by an honest party that is later corrupted is opened to a different share than in the simulation. However, this would imply a breach of the binding property of the trapdoor commitment scheme. Another possibility is that the extraction fails. However, this would imply breaking the simulation-extractability of the commitment scheme.

We conclude the proof by observing that hybrid 5 is identical to the simulation.

9 Multi-party Computation

We will now show how to generate a common random string on the fly using UC commitment. More precisely, we will securely realize an ideal functionality \mathcal{F}_{CRS} that produces a random bit-string.

Functionality \mathcal{F}_{CRS}
Parameterized with polynomial ℓ and running with parties P_1, \dots, P_n and adversary \mathcal{S} .
CRS generation: Generate random $\sigma \leftarrow \{0, 1\}^{\ell(k)}$ and output $(\text{crs}, \text{sid}, \sigma)$ to all parties and \mathcal{S} . Halt.

Figure 5: The ideal common random string generator.

COIN-FLIPPING. The parties will use the following natural coin-flipping protocol, where all parties first commit to a string of random bits and subsequently open all the commitments and use the exclusive-or of the random strings as the output.

Commitment: P_i chooses at random $r_i \leftarrow \{0, 1\}^{\ell(k)}$. It submits $(\text{commit}, \text{sid}, r_i)$ to $\mathcal{F}_{\text{COM}}^{1:N}$. $\mathcal{F}_{\text{COM}}^{1:N}$ on this input sends $(\text{commit}, \text{sid}, P_i)$ to all parties.

Opening: Once P_i sees $(\text{commit}, \text{sid}, P_j)$ for all j , it sends $(\text{open}, \text{sid}, r_i)$ to $\mathcal{F}_{\text{COM}}^{1:N}$. $\mathcal{F}_{\text{COM}}^{1:N}$ on this input sends $(\text{open}, \text{sid}, P_i, r_i)$ to all parties.

Output: Once P_i sees $(\text{commit}, \text{sid}, P_j, r_j)$ for all j , it outputs $(\text{crs}, \text{sid}, \bigoplus_{j=1}^N r_j)$ and halts.

Theorem 14 *The protocol securely realizes (perfectly) the ideal common reference string generator \mathcal{F}_{CRS} in the $\mathcal{F}_{\text{COM}}^{1:N}$ -hybrid model.*

Proof. Consider the following ideal process adversary \mathcal{S} working in the \mathcal{F}_{CRS} -hybrid model, giving it a common reference string σ . It runs a black-box simulation of \mathcal{A} , a simulated copy of $\mathcal{F}_{\text{COM}}^{1:N}$ and simulated parties P_1, \dots, P_N , not to be confused with the dummy parties $\tilde{P}_1, \dots, \tilde{P}_N$ that interact with \mathcal{Z} and \mathcal{F}_{CRS} . Whenever the simulated \mathcal{A} communicates with the environment \mathcal{Z} it simply forwards those messages. We now list the events that can happen in the protocol.

On activation of P_i , it simulates $\mathcal{F}_{\text{COM}}^{1:N}$ receiving a commitment from P_i by outputting $(\text{commit}, \text{sid}, P_i)$ to all parties and \mathcal{A} .

On delivery of commitments from all parties to an honest party P_i , it selects r_i at random, subject to the continued satisfiability of condition $\sigma = \bigoplus_{j=1}^N r_j$ and stores it. It then simulates $\mathcal{F}_{\text{COM}}^{1:N}$ receiving an opening of P_i 's commitment to r_i .

In case \mathcal{A} corrupts a party P_i , we corrupt the corresponding dummy party \tilde{P}_i . If P_i has made a commitment but it has not yet been opened, we select r_i at random, subject to the continued satisfiability of the condition $\sigma = \bigoplus_{j=1}^N r_j$, and simulate that this was the commitment P_i made. In all other cases of corruption, either r_i has not yet been selected, or the commitment has already been opened and \mathcal{A} already knows r_i .

The two experiments, \mathcal{A} running with parties P_1, \dots, P_N in the $\mathcal{F}_{\text{COM}}^{1:N}$ -hybrid model, and \mathcal{S} running with dummy parties $\tilde{P}_1, \dots, \tilde{P}_N$ in the \mathcal{F}_{CRS} -hybrid model are perfectly indistinguishable to \mathcal{Z} . To see this, consider a hybrid experiment, where we run the simulation and choose all r_i 's at random and then set $\sigma := \bigoplus_{i=1}^N r_i$. Inspection shows that this gives a perfect simulation of \mathcal{Z} 's view of the protocol in the $\mathcal{F}_{\text{COM}}^{1:N}$ -hybrid model. At the same time, also here we get a uniform random distribution on σ and the r_j 's subject to the condition $\sigma = \bigoplus_{j=1}^N r_j$. \square

MULTI-PARTY COMPUTATION. Armed with a coin-flipping protocol, we can generate random strings. Canetti, Lindell, Ostrovsky and Sahai [CLOS02] demonstrated that with access to a common random string, it is possible to do any kind of multi-party computation, even if only a minority of the parties is honest. We therefore get the following corollary to Theorems 13 and 14.

Theorem 15 *For any well-formed functionality \mathcal{F} there is a non-trivial protocol that securely realizes it in the $(\mathcal{F}_{BC}, \mathcal{F}_{MCRS})$ -hybrid model, provided enhanced trapdoor permutations with dense public keys and augmented non-committing encryption exists.*

Proof. Canetti, Lindell, Ostrovsky and Sahai [CLOS02] show that assuming enhanced trapdoor permutations with dense public keys⁴ and augmented non-committing encryption, there is a non-trivial protocol that securely realizes \mathcal{F} in the $(\mathcal{F}_{BC}, \mathcal{F}_{CRS})$ -hybrid model. To clarify matters; they actually claim that enhanced trapdoor permutation, dense cryptosystems and augmented non-committing encryption suffice for realizing general multi-party computation in the common random string model, however, a careful reading of their paper reveals that in the common random string model they need a pseudorandom cryptosystem with pseudorandom keys in the construction of their UC commitment scheme. Enhanced trapdoor permutations with dense public keys, imply the existence of enhanced trapdoor permutations, dense public key cryptosystems and pseudorandom cryptosystems with pseudorandom public keys.

Theorem 14 shows that we can securely realize \mathcal{F}_{CRS} in the $\mathcal{F}_{COM}^{1:N}$ -hybrid model. Therefore, by the universal composability theorem [Can01], we can securely realize \mathcal{F} in the $(\mathcal{F}_{BC}, \mathcal{F}_{COM}^{1:N})$ -hybrid model.

Theorem 13 shows that we can securely realize $\mathcal{F}_{COM}^{1:N}$ in the $(\mathcal{F}_{BC}, \mathcal{F}_{MCRS})$ -hybrid model assuming the existence of extractable trapdoor commitments. Recall from Theorem 12 that pseudorandom cryptosystems imply the existence of extractable trapdoor commitments. By the universal composability theorem we get that \mathcal{F} can be securely realized in the $(\mathcal{F}_{BC}, \mathcal{F}_{MCRS})$ -hybrid model under the stated cryptographic assumptions. \square

Acknowledgments

We thank Silvio Micali and Eyal Kushilevitz for an inspiring discussion in February of 2004 that motivated us to explore this setting.

References

- [Adl78] Leonard M. Adleman. Two theorems on random polynomial time. In *proceedings of FOCS '78*, pages 75–83, 1978.
- [BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 41–55, 2004.
- [BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *proceedings of FOCS '04*, pages 186–195, 2004.
- [BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal of Computation*, 20(6):1084–1118, 1991.
- [BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

⁴Enhanced trapdoor permutations with dense public keys, are enhanced trapdoor permutations, where the public keys can be sampled from a random string obliviously of the trapdoor, and at the same time we can sample the public key with the trapdoor and create convincing randomness that would make the public key be sampled.

- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *proceedings of STOC '88*, pages 103–112, 1988.
- [Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *proceedings of FOCS '01*, pages 136–145, 2001. Full paper available at <http://eprint.iacr.org/2000/067>.
- [CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with pre-existing setup. In *TCC '07, LNCS series*, pages 61–85, 2007. Full paper available at <http://eprint.iacr.org/2006/432>.
- [CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 19–40, 2001. Full paper available at <http://eprint.iacr.org/2001/055>.
- [CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *proceedings of STOC '02*, pages 494–503, 2002. Full paper available at <http://eprint.iacr.org/2002/140>.
- [Dam92] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *proceedings of EUROCRYPT '92, LNCS series, volume 658*, pages 341–355, 1992.
- [DDO⁺02] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 566–598, 2002.
- [DDP99] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Non-interactive zero-knowledge: A low-randomness characterization of np. In *proceedings of ICALP '99, LNCS series, volume 1644*, pages 271–280, 1999.
- [DDP02] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-optimal characterization of two np proof systems. In *proceedings of RANDOM '02, LNCS series, volume 2483*, pages 179–193, 2002.
- [DIO98] Giovanni Di Crescenzo, Yvail Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *proceedings of STOC '98*, pages 141–150, 1998.
- [DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 581–596, 2002. Full paper available at <http://www.brics.dk/RS/01/41/index.html>.
- [DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *proceedings of FOCS '92*, pages 427–436, 1992.
- [FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.
- [GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *proceedings of STOC '89*, pages 25–32, 1989.
- [GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, 2005.

- [GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. *SIAM Journal of Computing*, 18(1):186–208, 1989. First published at STOC 1985.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game, or A completeness theorem for protocols with honest majority. In *proceedings of STOC '87*, pages 218–229, 1987.
- [GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.
- [GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.
- [GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *proceedings of CRYPTO '06, LNCS series, volume 4117*, pages 97–111, 2006.
- [GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero-knowledge for np. In *proceedings of EUROCRYPT '06, LNCS series, volume 4004*, pages 339–358, 2006.
- [GP90] Andrew Granville and Carl Pomerance. On the Least Prime in Certain Arithmetic Progressions. *Journal of the London Mathematical Society*, s2-41(2):193–200, 1990.
- [Gro06] Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In *proceedings of ASIACRYPT '06, LNCS series*, 2006. Full paper available at <http://www.brics.dk/~jg/NIZKGroupSignFull.pdf>.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computation*, 28(4):1364–1396, 1999.
- [KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.
- [MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 382–400, 2004. Full paper available at <http://eprint.iacr.org/2003/252>.
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
- [NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and Systems Sciences*, 58(2):336–375, 1999.
- [Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of Structure in Complexity Theory Conference*, pages 133–138, 1991.
- [OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *proceedings of ISTCS '93*, pages 3–17, 1993.
- [PPS06] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Personal communication, November, 2006.
- [Sah01] Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '01*, pages 543–553, 2001.