Cryptography in the Multi-string Model

Jens Groth[1⋆] and Rafail Ostrovsky[1⋆⋆]

University of California, Los Angeles, CA 90095
{jg,rafail}@cs.ucla.edu

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 the random string 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 can securely realize any multi-party computation protocol.

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

⋆ Computer Science Department. Work partially done while visiting IPAM and supported in part by NSF ITR/Cybertrust grant No. 0456717 and Cybertrust grant No. 0430254.

⋆⋆ Computer Science Department and Department of Mathematics. Research 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

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 NIZK 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 is therefore widely used in cryptographic protocols.

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. The honesty of a small group of parties that are minimally involved can be magnified and used by a larger 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 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 standard model, 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, see for instance [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 may be requiring 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. Recently Groth, Ostrovsky and Sahai [GOS06b,GOS06a] have shown how to construct 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 the security parameter, i.e., the size of a group element. We will typically have n much smaller than $|C|$, so this matches the best single-string NIZK proofs [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.

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. One such example 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.

MULTI-PARTY COMPUTATION. Canetti, Lindell, Ostrovsky and Sahai [CLOS02] show 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 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 [RCW07] have demonstrated that it is not possible to have a fixed global common random string that is used for multiple and arbitrary different protocol executions and this result extends to the multi-string model.

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

2 Definitions

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 $(t_c, t_s, t_z, \boldsymbol{\sigma}, x, w)$, where $\boldsymbol{\sigma}$ is a set of n common reference strings and $(x, w) \in R$, and produces a proof π. The verifier takes as input $(t_c, t_s, t_z, \boldsymbol{\sigma}, x, \pi)$ 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 proof systems correspond closely to the standard notion of 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 string have been generated honestly. As we shall see later, our protocols will have perfect (t_c, t_s, t_z, n)-completeness for all $0 \le t_c \le n$. In other words, even if the adversary chooses all common reference strings itself, we still have probability 1 of outputting an acceptable proof.

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

$$\Pr\Big[S := \emptyset; (\boldsymbol{\sigma}, x, w) \leftarrow \mathcal{A}^K; \pi \leftarrow P(t_c, t_s, t_z, \boldsymbol{\sigma}, x, w) : \\ V(t_c, t_s, t_z, \boldsymbol{\sigma}, x, \pi) = 0 \text{ and } (x, w) \in R \text{ and } |\boldsymbol{\sigma} \setminus S| \ge t_c\Big] \approx 0,$$

where K on query i outputs $\sigma_i \leftarrow K(1^k)$ and sets $S := S \cup \{\sigma_i\}$.

(t_c, t_s, t_z, n)-SOUNDNESS. The goal of the adversary in the soundness definition is to forge a proof using n common reference strings, even if t_s of them 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 non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr\Big[S := \emptyset; (\boldsymbol{\sigma}, x, \pi) \leftarrow \mathcal{A}^K : V(t_c, t_s, t_z, \boldsymbol{\sigma}, x, \pi) = 1 \text{ and } x \notin L \text{ and } |\boldsymbol{\sigma} \backslash S| \ge t_s\Big] \approx 0,$$

where K is an oracle that on query i outputs $\sigma_i \leftarrow K(1^k)$ and sets $S := S \cup \{\sigma_i\}$.

(t_c, t_s, t_z, n)-ZERO-KNOWLEDGE. We wish to formulate 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 an algorithm that outputs (σ, τ), respectively a simulation reference string and a simulation trapdoor. Let furthermore, S_2 be an algorithm that takes input $(t_c, t_s, t_z, \boldsymbol{\sigma}, \boldsymbol{\tau}, x, w)$ and simulates a proof π if $\boldsymbol{\tau}$ contains t_z simulation trapdoors for common reference strings in $\boldsymbol{\sigma}$.

We will strengthen the standard definition of zero-knowledge, by splitting the definition 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*

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

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

$$\Pr\Big[S := \emptyset; (\boldsymbol{\sigma}, \boldsymbol{\tau}, x, w) \leftarrow \mathcal{A}^{S_1}(1^k); \pi \leftarrow P(t_c, t_s, t_z, \boldsymbol{\sigma}, x, w) : \\ \mathcal{A}(\pi) = 1 \text{ and } (x, w) \in R \text{ and } |\boldsymbol{\sigma} \setminus S| \geq t_z\Big]$$
$$\approx \Pr\Big[S := \emptyset; (\boldsymbol{\sigma}, \boldsymbol{\tau}, x, w) \leftarrow \mathcal{A}^{S_1}(1^k); \pi \leftarrow S_2(t_c, t_s, t_z, \boldsymbol{\sigma}, \boldsymbol{\tau}, x) : \\ \mathcal{A}(\pi) = 1 \text{ and } (x, w) \in R \text{ and } |\boldsymbol{\sigma} \setminus S| \geq t_z\Big],$$

where S_1 on query i outputs $(\sigma_i, \tau_i) \leftarrow S_1(1^k)$ and sets $S := S \cup \{\sigma_i\}$, and $\boldsymbol{\tau}$ contains t_z simulation trapdoors corresponding to σ_i's in $\boldsymbol{\sigma}$ generated by 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 he can prove anything and we can no longer have soundness. We capture this formally in the following theorem.

Theorem 1. *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 $L \in \mathrm{P/poly}$ or $t_s + t_z > n$.*

Proof. Assume we have an (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 $\pi \leftarrow S_2(\boldsymbol{\sigma}, \boldsymbol{\tau}, x)$. Output $V(\boldsymbol{\sigma}, x, \pi)$.

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 therefore 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 $\mathcal{A}$, we will 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 $L \in \mathrm{BPP/poly}$. By [Adl78] we have $\mathrm{P/poly} = \mathrm{BPP/poly}$, 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 might be relevant to have an unbalanced setting. Consider 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 came from trustworthy authorities he may be willing to believe that one of the authorities is honest. On the other hand, Alice gets to choose the authorities, so she may be wiling to believe that all of them are honest. The appropriate choice in this situation, is a multi-string proof with $t_s = 1, t_z = n$.

ADVANCED ZERO-KNOWLEDGE PROOFS. Multi-string NIZK proofs can have more advanced properties. In a multi-string proof of knowledge, the reference strings can be generated with an extraction trapdoor. If you hold at least t_s extraction keys, it is possible to extract a witness from the multi-string NIZK proof. We say a multi-string NIZK proof has extraction zero-knowledge, if it is zero-knowledge even to an adversary that can ask for arbitrary extractions of witnesses. This latter notion is similar in nature to CCA-secure encryption.

Another way of strengthening soundness is to say that even after seeing arbitrary simulated proofs, even on false statements, it should not be able to prove another false statement. We call this simulation soundness. This notion can be extended and combined with proofs of knowledge to simulation-extractability, which means that even if we give the adversary access to see simulated multi-string proofs, it cannot produce another proof without us being able to extract a witness from it.

We refer to the full paper [GO07] for formal definitions of multi-string proofs of knowledge, simulation soundness, simulation-sound extractability and extraction zero-knowledge.

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 so $t_s + t_z > n$. This construction is used in the full paper [GO07] to prove the following theorem connecting single-string NIZK proofs and multi-string NIZK proofs.

Theorem 2. *Assuming hard on average languages exist in NP, the existence of NIZK proofs for NP in the common random string model is equivalent to the existence of multi-string NIZK proofs for NP in the common random strings model. The equivalence preserves perfect completeness.*

We use two tools in the construction: a zap $(\ell_{\rm zap}, P_{\rm zap}, V_{\rm zap})$ and a pseudorandom generator PRG. 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 $\sigma_1, \ldots, \sigma_n$ for

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

In the simulation, we create simulation reference strings as $r := \mathrm{PRG}(\tau)$ enabling the simulator to make zaps without knowing a witness w for $x \in L$.

MULTI-STRING SIMULATION-EXTRACTABLE NIZK PROOF. 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 CCA2-secure cryptosystem 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 $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow K_{\mathrm{sots}}(1^k)$ that he will use to sign the proof. The implication is that the adversary, who sees simulated proofs, must use a different vk_{sots} in his forged proof, because he 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 he encrypted a (t_s, n)-threshold secret sharing of the witness, or that he 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 he 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 he 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: $(pk_1, dk_1), (pk_2, dk_2) \leftarrow K_{\mathrm{CCA2}}(1^k)$; $r \leftarrow \{0,1\}^{2k}; \sigma \leftarrow \{0,1\}^{\ell_{\mathrm{zap}}(k)}$. 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 $r := \mathrm{PRF}_\tau(0)$. We use the simulation trapdoor τ and the extraction key $\xi := dk_1$.

Proof: $P(0, t_s, t_z, (\Sigma_1, \ldots, \Sigma_n), x, w)$ where $(x, w) \in R$ runs as follows: First, generate a key pair for a strong one-time signature scheme $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow K_{\mathrm{sots}}(1^k)$. Use (t_s, n)-threshold secret sharing to get shares $w_1, \ldots, w_n$ of w. Encrypt the shares as $c_{1i} := E_{pk_{1i}}(w_i, vk_{\mathrm{sots}}; r_{1i})$. Also encrypt dummy values $c_{2i} \leftarrow E_{pk_{2i}}(0)$. Consider the statement: "All c_{1i} encrypt $(w_i, vk_{\mathrm{sots}})$, where $w_1, \ldots, w_n$ is a (t_s, n)-secret sharing of a witness w so $(x, w) \in R \quad or \quad$ there exist at least t_z seeds τ_i so $r_i = \mathrm{PRF}_{\tau_i}(0)$ and c_{2i} encrypts $\mathrm{PRF}_{\tau_i}(vk_{\mathrm{sots}})$." We can reduce this statement to a polynomial size circuit C and a satisfiability witness W. For all i's we create a zap $\pi_i \leftarrow P_{\mathrm{zap}}(\sigma_i, C, W)$ for C being satisfiable. We sign everything using the one-time signature $sig \leftarrow \mathrm{Sign}_{sk_{\mathrm{sots}}}(vk_{\mathrm{sots}}, x, \Sigma_1, c_{11}, c_{21}, \pi_1, \ldots, \Sigma_n, c_{1n}, c_{2n}, \pi_n)$. The proof is $\Pi := (vk_{\mathrm{sots}}, c_{11}, c_{21}, \pi_1, \ldots, c_{1n}, c_{2n}, \pi_n, sig)$.

Verification: To verify Π on the form described above, verify the strong one-time signature and verify the n zaps $\pi_1, \ldots, \pi_n$.

Extraction: To extract a witness check that the proof is valid. Next, use the first t_s extraction keys in $\boldsymbol{\xi}$ 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 $\boldsymbol{\tau}$. These are τ_i so $r_i = \mathrm{PRF}_{\tau_i}(0)$. As in the proof generate $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow K_{\mathrm{sots}}(1^k)$. Create t_z pseudorandom values $v_i := \mathrm{PRF}_{\tau_i}(vk_{\mathrm{sots}})$. 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 $w_1, \ldots, w_n$ 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 $w_1, \ldots, w_n$ 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 $sig \leftarrow \text{Sign}_{sk_{\text{sots}}}(vk_{\text{sots}}, x, \Sigma_1, c_{11}, c_{21}, \pi_1, \ldots, \Sigma_n, c_{1n}, c_{2n}, \pi_n)$. The simulated proof is $\Pi := (vk_{\text{sots}}, c_{11}, c_{21}, \pi_1, \ldots, c_{1n}, c_{2n}, \pi_n, sig)$.

Theorem 3. *The above protocol 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 permutations exist, and it can be implemented with random strings if dense cryptosystems [DP92] and enhanced trapdoor permutations exist.*

We refer to the full paper [GO07] for the proof.

4 Multi-string NIZK Proofs from Groups with a Bilinear Map

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 $\mathbb{G}$.
- $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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.[1]
- 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 $\mathcal{G}$.

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

Example. We will offer a class of candidates for DLIN groups as described above. Consider the elliptic curve $y^2 = x^3 + 1 \bmod q$, where $q = 2 \bmod 3$ 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 $x = (y^2 - 1)^{\frac{q+1}{3}} \bmod q$ 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 prime[2] 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 $q = 2 \bmod 3$ 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.

MULTI-STRING NIZK PROOFS FROM DLIN GROUPS. We will 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. 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 $\mathcal{O}(|C|k)$.

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 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, which we construct from the DLIN assumption in the full paper [GO07]. This pseudorandom generator is constructed in such a way that there exist efficient 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

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

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 his own group, he gets n values $s_1, \ldots, s_n$ and proves that t_z of those are pseudorandom or C is satisfiable. In the simulation, he knows the simulation trapdoors for t_z reference strings and he can therefore simulate NIZK proofs of $r_i \oplus s_i$ being pseudorandom. This means, he 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 his own group, so he does not need to know the satisfiability witness w for C being satisfiable to carry out the proof in his 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 $\Sigma \leftarrow K_{\mathrm{sim-sound}}(p, \mathbb{G}, \mathbb{G}_T, e, g)$ as in [Gro06]. 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 $(\sigma, \tau) \leftarrow S_{\mathrm{sim-sound}}(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Output Σ as described above and simulation trapdoor τ.

Proof: Given $t_z, (\Sigma_1, \ldots, \Sigma_n), C, w$ 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 $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow K_{\mathrm{sots}}(1^k)$. 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, as described in the full paper [GO07]. 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 $\mathbb{G}_i$. Choose $s_i \in \{0,1\}^{61k}$ 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, 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 $i = 1, \ldots, n$ 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:

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

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

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

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

Verification: Given common reference strings $\Sigma_1, \ldots, \Sigma_n$, 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 structure 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 $\Sigma_1, \ldots, \Sigma_n$. 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 $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow K_{\mathrm{sots}}(1^k)$.

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 structure. This means, we are free to choose s_i so it encodes a pseudorandom structure 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 structure, 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 $sig \leftarrow \mathrm{Sign}_{sk_{\mathrm{sots}}}(vk_{\mathrm{sots}}, C, \Sigma_1, s_1, \pi_1, c_1, \ldots, \Sigma_n, s_n, \pi_n, c_n, p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, \pi)$. The simulated proof is $\Pi := (vk_{\mathrm{sots}}, s_1, \pi_1, c_1, \ldots, s_n, \pi_n, c_n, p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, \pi, sig)$.

Theorem 4. *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.*

The proof can be found in the full paper [GO07].

5 UC Commitment in the Multi-String Model

In the rest of the paper, we will work in Canetti's UC framework. We refer to Canetti [Can01] for a detailed description. Very briefly, the UC framework compares a real world execution of a protocol with an ideal process where the parties have access to an ideal functionality that handles all protocol execution honestly and securely.

IDEAL FUNCTIONALITIES. Let us first formalize the multi-string model in the UC framework. Figure 1 gives an ideal multi-string functionality $\mathcal{F}_{\mathrm{MCRS}}$. We will construct universally composable commitments, see Figure 2, in the multi-string model.

Functionality $\mathcal{F}_{\mathrm{MCRS}}$

Parameterized by polynomial ℓ_{mcrs}, and running with parties $P_1, \dots, P_N$ and adversary $\mathcal{S}$.

String generation: On input $(\mathbf{crs}, sid)$ from $\mathcal{S}$, pick $\sigma \leftarrow \{0,1\}^{\ell_{\mathrm{mcrs}}(k)}$ and store it. Send $(\mathbf{crs}, sid, \sigma)$ to $\mathcal{S}$.

String selection: On input $(\mathbf{vector}, sid, \sigma_1, \dots, \sigma_n)$ where $\sigma_1, \dots, \sigma_n \in \{0,1\}^{\ell_{\mathrm{mcrs}}(k)}$ from $\mathcal{S}$ check that more than half of the strings $\sigma_1, \dots, \sigma_n$ match stored strings. In that case output $(\mathbf{vector}, sid, \sigma_1, \dots, \sigma_n)$ to all parties and halt.

Fig. 1. The ideal multi-string generator.

Functionality $\mathcal{F}_{\mathrm{COM}}^{1:N}$

Parameterized by polynomial ℓ, and running with parties $P_1, \dots, P_N$ and adversary $\mathcal{S}$.

Commitment: On input $(\mathbf{commit}, sid, m)$ from party P_i check that $m \in \{0,1\}^{\ell(k)}$ and in that case store (sid, P_i, m) and send $(\mathbf{commit}, sid, P_i)$ to all parties and $\mathcal{S}$. Ignore future $(\mathbf{commit}, sid, \cdot)$ inputs from P_i.

Opening: On input $(\mathbf{open}, sid)$ from P_i check that (sid, P_i, m) has been stored, and in that case send $(\mathbf{open}, sid, P_i, m)$ to all parties and $\mathcal{S}$.

Fig. 2. The ideal commitment functionality.

We will assume the parties can broadcast messages, i.e., have access to an ideal broadcast functionality $\mathcal{F}_{\mathrm{BC}}$.

UC COMMITMENT IN THE MULTI-STRING MODEL. We will describe our UC commitment protocol later but first 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 $\mathcal{F}_{\mathrm{COM}}^{1:N}$ and makes a black-box simulation of $\mathcal{A}$ running with $\mathcal{F}_{\mathrm{MCRS}}$ and $P_1, \ldots, P_N$. There are two general types of issues that can come up in the ideal process simulation. First, when $\mathcal{F}_{\mathrm{COM}}^{1:N}$ 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 $\mathcal{F}_{\mathrm{COM}}^{1:N}$. 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. Informally, a tag-based simulation-extractable commitment scheme, is a non-interactive commitment scheme that takes as input an arbitrary tag, a message and a randomizer (tag, m, r) and outputs a commitment c. The commitment can be opened for tag simply by revealing m, r. The commitment must be a trapdoor commitment: given a simulation trapdoor we can construct commitments for an arbitrary tag that can be opened to any value we desire. At the same time it must be extractable: given an extraction key we can extract the message from any commitment that uses a tag tag that has not been used in a trapdoor commitment. In addition, we will need that the public key for the tag-based simulation-extractable commitment scheme is pseudorandom such that we can set it up in the common random strings model. Tag-based simulation-extractable commitments are formally defined in the full paper [GO07] where we also give a construction.

Our idea in constructing a UC commitment is to use each of the n common random strings output by $\mathcal{F}_{\mathrm{MCRS}}$ 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 $\mathcal{F}_{\mathrm{MCRS}}$ 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 him 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 $(\mathbf{vector}, sid, (ck_1, \sigma_1), \ldots, (ck_n, \sigma_n))$ from $\mathcal{F}_{\mathrm{MCRS}}$ and $(\mathbf{commit}, sid, m)$ from $\mathcal{Z}$, the party P_i does the following. He makes a (t, n)-threshold secret sharing $s_1, \ldots, s_n$ of m. He picks randomizers r_j and makes commitments $c_j := \mathrm{Com}_{ck_j}(s_j; r_j)$. He also picks randomizers R_j and makes tag-based commitments $C_j := \mathrm{Com}_{\sigma_j}((P_i, c_j); s_j; R_j)$. The commitment is $c := (c_1, C_1, \ldots, c_n, C_n)$. He broadcasts $(\mathbf{broadcast}, sid, c)$.

Receiving commitment: A party on input $(\mathbf{vector}, sid, (ck_1, \sigma_1), \ldots, (ck_n, \sigma_n))$ from $\mathcal{F}_{\mathrm{MCRS}}$ and $(\mathbf{broadcast}, sid, P_i, c)$ from $\mathcal{F}_{\mathrm{BC}}$ broadcasts $(\mathbf{broadcast}, sid, P_i, c)$. Once it receives similar broadcasts from all parties, all containing the same P_i, c, it outputs $(\mathbf{commit}, sid, P_i)$ to the environment.

Opening commitment: Party P_i wishing to open the commitment broadcasts $(\mathbf{open}, sid, s_1, r_1, R_1, \ldots, s_n, r_n, R_n)$.

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

Theorem 5. *The protocol securely realizes $\mathcal{F}_{\mathrm{COM}}^{1:N}$ in the $(\mathcal{F}_{\mathrm{BC}}, \mathcal{F}_{\mathrm{MCRS}})$-hybrid model, assuming tag-based simulation-extractable commitment schemes with pseudorandom keys exist in the common random string model.*

See the proof in the full paper [GO07].

6 Multi-party Computation

COIN-FLIPPING. A nice application of UC commitment is coin-flipping. In a coin-flipping protocol the parties generate a series of uniformly random bits. In other words, all the protocols we have in the CRS-model can be securely realized if we can do coin-flipping.

We will now show how to generate a common random string on the fly. The parties will use the following natural coin-flipping protocol.

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

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

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

Theorem 6. *The protocol securely realizes (perfectly) the ideal common reference string generator $\mathcal{F}_{\mathrm{CRS}}$ in the $\mathcal{F}_{\mathrm{COM}}^{1:N}$-hybrid model.*

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 5 and 6, which we prove in the full paper [GO07].

Theorem 7. *For any well-formed functionality $\mathcal{F}$ there is a non-trivial protocol that securely realizes it in the $(\mathcal{F}_{\mathrm{BC}}, \mathcal{F}_{\mathrm{MCRS}})$-hybrid model, provided enhanced trapdoor permutations with dense public keys and augmented non-committing encryption exists.*

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 Jornal 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.

[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`.

[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 proprocessing. 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.

[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.

[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.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *proceedings of CRYPTO '07, LNCS series*, 2007. Full paper available at `http://www.cs.ucla.edu/~rafail/PUBLIC/index.html`.

[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`.

[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.

[PPS06] Omkant Padney, Manoj Prabhakaran, and Amit Sahai. personal communication, November, 2006.

[RCW07] Rafael Pass Ran Canetti, Yevgeniy Dodis 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`.

[Sah01] Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '01*, pages 543–553, 2001.