J. Cryptol. (2014) 27: 506–543
DOI: 10.1007/s00145-013-9152-y

Journal of
CRYPTOLOGY

Cryptography in the Multi-string Model*

Jens Groth†
Computer Science Department, University College London, London, UK
j.groth@ucl.ac.uk

Rafail Ostrovsky‡
Department of Computer Science and Department of Mathematics, University of California, Los Angeles, USA
rafail@cs.ucla.edu

Communicated by Yevgeniy Dodis

Received 11 October 2008
Online publication 29 May 2013

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. Our results also hold even if different subsets of these strings are

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

† The research of J. Groth leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937 and the Engineering and Physical Sciences Research Council grants EP/G013829/1 and EP/J009520/1. The work was partially done while at UCLA Department of Computer Science and while visiting IPAM and supported in part by NSF ITR/Cybertrust grant No. 0456717 and Cybertrust grant No. 0430254.

‡ Work of R. Ostrovsky partially done while visiting IPAM. Supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Research Award. This material is also based upon work supported in part by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

used in different instances, as long as a majority of the strings used at any particular invocation is honestly generated. This security model is reasonable and at the same time 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 also 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. The UC commitment scheme can be used in a simple coin-flipping protocol to create a uniform random string, which in turn enables the secure realization of any multi-party computation protocol.

Key words. Common random string model, Multi-string model, Non-interactive zero-knowledge, Universally composable commitment, Multi-party computation.

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 [7] introduced the CRS model (with a uniform random string) 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 [30]. 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 [12]. The CRS-model has therefore found widespread use in the field of cryptology.

Using the CRS-model creates a problem: where should 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 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 are 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 some initial rounds 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.

The Multi-string Model We propose the multi-string model as a solution to the above mentioned problem. In the multi-string model a number of authorities assist the protocol execution by providing random strings. If a majority of the 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 protocol 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. This is in contrast with multi-party computation protocols such as [6, 16,32] where the actual players must be aware of each other at all times with a majority of players remaining uncorrupted at all times. Also, no matter how untrustworthy the other parties in your protocol are, the protocol is secure 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.

A natural generalization of the uniform random multi-string model described above is the common reference multi-string model where the strings are sampled with a non-uniform distribution. Since it is easier to generate uniform random strings and they can be sampled without learning any secret trapdoor information about them we are most interested in the uniform random multi-string model. However, for some of our results the common reference multi-string model permits a relaxation of the underlying cryptographic assumptions.

Related Work In the context of non-interactive proofs witness-indistinguishablity can be obtained without a CRS [3,39] or zero-knowledge may be obtained using super-polynomial simulation techniques [1]. But it is impossible to construct non-interactive zero-knowledge proofs under standard assumptions without some sort of setup [30]. To the best of our knowledge the multi-string model is the simplest trustworthy setup.

Multi-party computation is possible under computational assumptions when an honest majority is available [32] and information theoretically when more than 2/3 of the parties are honest [6,16]. However, if more parties are corrupt general multi-party computation is impossible without some setup. Canetti, Lindell, Ostrovsky, and Sahai [14] used the CRS-model as a setup to overcome this problem. As an alternative, Barak, Canetti, Nielsen, and Pass [2] suggested 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?

In the information-theoretic setting Beaver suggested commodity-based [4] and server-assisted [5] cryptography where multi-party computation is made possible with the assistance of third parties that are oblivious to the actual protocol executed as a way to reduce the involvement of multiple parties. In his protocols the servers give distinct

correlated values to the parties in the protocol. We on the other hand rely on computational assumptions, but get a simpler setup model. In the multi-string model each server provides the same input to all parties and in the uniform random multi-string model this input is just a random bit-string.

Results As argued above, the multi-string model is a reasonable and simple 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 in two separate directions: Our first set of results give constructions of NIZK proofs in the multi-string model. Our second set of results give UC commitment and general UC-secure multi-party computation in the multi-string model in the presence of adaptive adversaries. We stress that different parties may have different beliefs about which common strings were generated by honest parties and which were generated maliciously and that our results hold despite these different beliefs (even if only a subset of strings are used) as long as the number of honestly generated strings used in any application satisfies the threshold.

1.1. *Non-interactive Zero-Knowledge*

A zero-knowledge proof [32,34] is a two-party protocol, where a prover tries to convince a verifier of 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 [30]. Instead, much research has gone into constructing non-interactive zero-knowledge proofs in the CRS-model [7,8,18,21–24,26,39,41].

Multi-string NIZK We define the notion of multi-string NIZK proofs in Sect. 2. In the definitions, the adversary sees honestly generated strings and pick the ones she likes. The adversary may also 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 of 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 Sect. 2.

A natural question is under which assumptions we can obtain multi-string NIZK proofs. We prove that if one-way functions exist then the existence of single-string NIZK proofs imply the existence of multi-string NIZK proofs.

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 where the adversary simultaneously acts as prover in some multi-string NIZK proofs and as verifier in other multi-string NIZK proofs. When the protocol designer has to prove security in such a setting, 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 enable security proofs, 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 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 [39] 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 an NIZK proof in the CRS model.

There is a technical hurdle to overcome when trying to apply their techniques in the multi-string model: single-string NIZK proofs rely on the common reference string to contain a description of a bilinear group. In the multi-string model, authorities generate their random strings obliviously of 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 [39] that have complexity $\mathcal{O}(|C|k)$.

1.2. *Multi-party Computation*

Canetti's UC framework [11] defines secure execution of a protocol under concurrent execution of arbitrary protocols. Informally a protocol is UC secure if its execution is equivalent to the parties handing their protocol inputs to an honest trusted third party that computes everything securely and returns the resulting outputs to the involved parties. We refer the reader to Sect. 6 for a sketch of the UC framework.

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, there are certain functionalities that are provably impossible to realize in the plain model if half or more of the parties may be corrupted. An example of an unrealizable two-party functionality in the plain model is UC commitment [12].

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 over several commitments and tolerate a minority of fake common reference strings.

General Multi-party Computation Canetti, Lindell, Ostrovsky, and Sahai [14] 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 the CRS should come from, since the parties provably could not compute it themselves.

Armed with our multi-string UC commitment it is straightforward to solve the problem. We run a coin-flipping protocol using the UC commitment given above to create a CRS. We can then use the CRS to securely realize any ideal functionality. 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.

UC Multi-string Model We formalize the multi-string model in the UC framework as an ideal functionality that provides random strings and allows the adversary to inject a minority of malicious strings before a protocol execution. This functionality is easy to implement with a set of authorities that provide random strings on request.

We note that each string should only be used in one protocol; we do not guarantee security if many protocols use the same strings. Canetti, Dodis, Pass, and Walfish [15] 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.

Follow-up Works The conference version of this article [38] initiated the study of constructing UC secure protocols without relying on a single trusted external entity. In other words, one of the important contributions of our work is to initiate research where the beliefs in which cryptographic objects (e.g., reference strings) have the correct properties need not be agreed upon by all players, and different players may have different beliefs. There were a number of follow-up works (that consider not just random strings but also other cryptographic gadgets) where different participants may have different beliefs [28,35].

2. Definitions

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

We say a function $\nu : \mathbb{N} \to [0; 1]$ 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. We say f is overwhelming if $f(k) \approx 1$.

Let R be a polynomial time 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. We are most interested in the case where the key generator outputs a uniformly random string of polynomial length $\ell(k)$ but for the sake of generality we permit other types of key generators in our definitions as well.

The prover takes as input $(\vec{\Sigma}, x, w)$, where $\vec{\Sigma}$ is a tuple of n 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 (t_c, n)-completeness, (t_s, n)-soundness and (t_z, n)-zero-knowledge properties defined below. We remark that $(1, 1, 1, 1)$-NIZK proofs correspond to standard NIZK proofs in the common reference string model (with composable zero-knowledge [37], which is stronger than the standard definition of zero-knowledge).

(t_c, n)-COMPLETENESS Completeness means that the prover can create acceptable proofs for true statements when at least t_c out of n common reference strings have been generated honestly.

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

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

where K is an oracle that on the ith query outputs $\Sigma_i \leftarrow K(1^k)$ and the tuple $\vec{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$ output by $\mathcal{A}$ includes at least t_c common reference strings (possibly with repetition) generated by K.

The protocols we construct in this paper are (t_c, n)-complete for all $0 \leq t_c \leq n$. This means that the prover always outputs an acceptable proof whenever $(x, w) \in R$; even if all the common reference strings are chosen adversarially. We remark that with $t_c = 0$ the adversary could in principle choose all $\Sigma_i = \perp$. Typically we handle this by using a default common reference string such as $\Sigma = K(1^k; 0)$ in place of bad common reference strings.

The definition could be generalized slightly to computational (t_c, n)-completeness where the equality only needs to hold approximately. If the verifier is deterministic the multi-string NIZK proof can easily be made perfectly complete though by letting the prover use the witness itself as a replacement proof in the negligibly few occasions where the normal NIZK proof is not accepting. The same holds for single-string NIZK proofs and indeed all known constructions of single-string NIZK proofs can easily be made perfectly complete, so we will with little loss of generality assume throughout the paper that both multi-string and single-string NIZK proofs are perfectly complete.

(t_s, n)-SOUNDNESS Soundness says that an adversary cannot forge a proof when at least t_s out of n common reference strings have been honestly generated.

Definition 2. (K, P, V) is (statistically) (t_s, n)-*sound* if for all adversaries $\mathcal{A}$ we have

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

where K on the ith query outputs $\Sigma_i \leftarrow K(1^k)$ and the tuple $\vec{\Sigma} = (\Sigma_1, \ldots, \Sigma_n)$ output by $\mathcal{A}$ includes at least t_s common reference strings generated by K.

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

(t_s, n)-PROOF OF KNOWLEDGE Soundness prohibits giving valid proofs for false statements, but does not imply the ability to compute a witness for the statement. Strengthening the notion of soundness, we define a proof of knowledge as a proof system where it is possible to extract a witness from a valid proof.

Definition 3. We say (K, P, V) is a (statistical) (t_s, n)-*proof of knowledge* for R with extractor (E_1, E_2) if E_1, E_2 are probabilistic polynomial time algorithms such that for all adversaries $\mathcal{A}$ we have

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

and for all adversaries $\mathcal{A}$ we have

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

where the oracle E_1 on query i returns $(\Sigma_i, \xi_i) \leftarrow E_1(1^k)$ and is queried at most a polynomial number of times by the adversary, and the adversary outputs $\vec{\Sigma}, \vec{\xi}$ with t_s pairs (Σ_i, ξ_i) having been generated by E_1 and the remaining $n - t_s$ pairs being of the form $(\Sigma_i, \perp)$.

As in the definition of soundness, we can define a *perfect* (t_s, n)-proof of knowledge by requiring the equalities to be exact instead of allowing a negligible difference. A statistical (t_s, n)-proof of knowledge is statistically (t_s, n)-sound and a perfect (t_s, n)-proof of knowledge is perfectly (t_s, n)-sound.

(t_z, n)-ZERO-KNOWLEDGE Zero-knowledge means that the adversary learns nothing from the proof (besides the truth of the statement) if at least t_z out n common reference strings have been honestly generated. We capture zero-knowledge by requiring that the proof can be simulated without knowing the witness. A simulator for (K, P, V) consists of two probabilistic polynomial time algorithms (S_1, S_2). S_1 takes 1^k as input and outputs (Σ, τ), respectively a simulation reference string and a simulation trapdoor. S_2 takes as input $(\vec{\Sigma}, \vec{\tau}, x, w)$ and simulates a proof π when $\vec{\tau} = (\tau_1, \ldots, \tau_n)$ is an n-tuple with exactly t_z values $\tau_i \neq \perp$.

We will strengthen the standard definition of zero-knowledge by splitting the definition into two parts. The first part 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 real proofs from simulated proofs on a set of simulation reference strings.

Definition 4. We say (K, P, V) is (computationally) (t_z, n)-*zero-knowledge* if there is a simulator (S_1, S_2) with reference string indistinguishability and simulation indistinguishability as described below.

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

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

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

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

where S_1 on the ith query outputs $(\Sigma_i, \tau_i) \leftarrow S_1(1^k)$, and the adversary outputs $(x, w) \in R$ and $\vec{\Sigma}, \vec{\tau}$ with t_z pairs (Σ_i, τ_i) generated by S_1 and $n - t_z$ pairs of the form $(\Sigma_i, \perp)$.

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

Theorem 5. *If L is an NP-language with a (t_c, t_s, t_z, n)-NIZK proof system (K, P, V) (for any $t_c \geq 0$) then $L \in$ BPP or $t_s + t_z > n$.*

Proof. Assume without loss of generality $t_c = n$ and that we have an (n, t_s, t_z, n)-NIZK proof system for the relation R defining L with simulator (S_1, S_2) and $t_s + t_z \leq n$. We will build a probabilistic polynomial time algorithm that has more than $2/3$ chance of deciding whether $x \in L$ or $x \notin L$.

We first construct a decision algorithm that works well for large statements. Our algorithm gets x as input and sets $k = |x|$. It simulates t_z common reference strings $(\Sigma_i, \tau_i) \leftarrow S_1(1^k)$ and generates $n - t_z$ common reference strings $\Sigma_j \leftarrow K(1^k)$ setting $\tau_j = \perp$. It then simulates the proof $\pi \leftarrow S_2(\vec{\Sigma}, \vec{\tau}, x)$ and outputs $V(\vec{\Sigma}, x, \pi)$. It is clear that this is a probabilistic polynomial time algorithm.

Let us analyze the probability of the algorithm deciding membership correctly on a family of worst-case choices of statements $\{x_k\}_{k=1}^{\infty}$ with $|x_k| = k$. For $x_k \notin L$ the (t_s, n)-soundness gives us overwhelming (in k) probability of outputting 0 since $n - t_z \geq t_s$ common reference strings have been generated correctly.

For $x_k \in L$ the (n, n)-completeness means that a prover with access to a witness w_k with overwhelming probability outputs an acceptable proof if all common reference strings are generated correctly. The reference string indistinguishability property gives overwhelming probability of accepting the proof even when some of the common reference strings are simulated. The (t_c, n)-simulation indistinguishability, where we give (x_k, w_k) as non-uniform advice to $\mathcal{A}$, shows that a simulated proof also has overwhelming probability of being accepted. We therefore have overwhelming probability of the algorithm outputting 1 on $x_k \in L$.

We now have an algorithm that decides membership of L correctly with overwhelming probability as the size of the statements grows. This implies that there is at most a constant number of statements for which the algorithm has less than $2/3$ probability of giving the right decision. We get a BPP decision algorithm for L by hardcoding these statements and the corresponding membership decision into our algorithm. □

It is in the verifier's interest to minimize t_s to make it more probable that the protocol is sound and it is in the prover's interest 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 appropriate to have an imbalance between t_s and t_z. Consider for instance the case, where Alice wants to e-mail an 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 NIZK proof with $t_s = 1, t_z = n$.

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 a multi-string NIZK

proof is 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) with simulator (S_1, S_2) is *simulation-sound* if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

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

where S_1' on query i runs $(\Sigma_i, \tau_i) \leftarrow S_1(1^k)$ and returns Σ_i, and S_2' on input $(\vec{\Sigma}_j, I_j, x_j)$ where I_j contain t_z indices corresponding to common reference strings in $\vec{\Sigma}_j$ that have been generated by S_1, returns $\pi \leftarrow S_2(\vec{\Sigma}_j, \vec{\tau}_j, x_j)$ where τ_j containing the t_z simulation trapdoors generated by S_1 corresponding to the indices in I_j in the same positions and the remaining entries are $\perp$, and Q is a list of statements and corresponding proofs $(\vec{\Sigma}_j, x_j, \pi_j)$ in the queries to S_2'.

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 [11].

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 the (t_c, t_s, t_z, n)-NIZK proof of knowledge (K, P, V) with simulator (S_1, S_2) and extractor (E_1, E_2) is *simulation-extractable* if there is a probabilistic polynomial time algorithm SE_1 that outputs (Σ, τ, ξ) where (Σ, τ) is distributed identically to the output of S_1, and for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

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

where SE_1' on query i outputs (Σ_i, ξ_i) from $(\Sigma_i, \tau_i, \xi_i) \leftarrow SE_1(1^k)$, and S_2' on input $(\vec{\Sigma}_j, I_j, x_j)$ outputs $\pi_j \leftarrow S_2(\vec{\Sigma}_j, \vec{\tau}_j, x_j)$ where $\vec{\tau}_j$ as in the definition of simulation-soundness is such that the t_z pairs (Σ_i, τ_i) in $(\vec{\Sigma}_j, \vec{\tau}_j)$ corresponding to the indices in I_j have been generated by SE_1' and the remaining pairs are of the form $(\Sigma_i, \perp)$, and Q is a list of statements and corresponding proofs $(\vec{\Sigma}_j, x_j, \pi_j)$ made by S_2', and $\vec{\Sigma}, \vec{\xi}$ contains exactly t_s pairs (Σ_i, ξ_i) generated by SE_1' and the remaining pairs are of the form $(\Sigma_i, \perp)$.

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 public-key encryption secure against chosen ciphertext attack.

Definition 8. We say the (t_c, t_s, t_z, n)-NIZK proof of knowledge (K, P, V) with simulator (S_1, S_2) and extractor (E_1, E_2) is *extraction zero-knowledge* if there is a probabilistic polynomial time algorithm SE_1 that outputs (Σ, τ, ξ) with an identical distribution to S_1 when restricted to (Σ, τ) and for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\begin{aligned}
&\Pr\big[(\vec{\Sigma}, x, w) \leftarrow \mathcal{A}^{SE_1'', E_2'(\cdot,\cdot,\cdot,\cdot)}\big(1^k\big); \pi \leftarrow P(\vec{\Sigma}, x, w) : \\
&\qquad \mathcal{A}^{E_2'(\cdot,\cdot,\cdot,\cdot)}(\pi) = 1 \text{ and } (x, w) \in R\big] \\
&\approx \Pr\big[(\vec{\Sigma}, x, w) \leftarrow \mathcal{A}^{SE_1'', E_2'(\cdot,\cdot,\cdot,\cdot)}\big(1^k\big); \pi \leftarrow S_2(\vec{\Sigma}, \vec{\tau}, x) : \\
&\qquad \mathcal{A}^{E_2'(\cdot,\cdot,\cdot,\cdot)}(\pi) = 1 \text{ and } (x, w) \in R\big],
\end{aligned}$$

where SE_1'' on query i outputs (Σ_i, τ_i) from $(\Sigma_i, \tau_i, \xi_i) \leftarrow SE_1(1^k)$, and E_2' on input $(\vec{\Sigma}_j, I_j, x_j, \pi_j)$ outputs $w \leftarrow E_2(\vec{\Sigma}_j, \vec{\xi}_j, x_j, \pi_j)$ where I_j contains exactly t_s indices corresponding to strings Σ_i in $\vec{\Sigma}_j$ generated by SE_1'' and $\vec{\xi}_j$ in the same positions has the corresponding ξ_i extraction keys and $\perp$ in all other positions, and after seeing π the adversary does not make a query of the form $(\vec{\Sigma}, *, x, \pi)$.

3. Multi-string NIZK Proofs Based on General Assumptions

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 length-doubling pseudorandom generator PRG and a zap $(\ell_{\text{zap}}, P_{\text{zap}}, V_{\text{zap}})$.

ZAPS Zaps, introduced by Dwork and Naor [20], 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. It follows from Dwork and Naor's construction that zaps exist if NIZK proofs exist in the common random string model.

A zap for the NP-relation R is a triple $(\ell_{\text{zap}}, P_{\text{zap}}, V_{\text{zap}})$, consisting of an input-increasing polynomial ℓ_{zap}, a probabilistic polynomial time prover P_{zap} and a probabilistic polynomial time verifier V_{zap}. Given an $\ell_{\text{zap}}(k)$-bit random string σ, a statement x and a witness w such that $(x, w) \in R$, the prover outputs a proof π. The verifier given σ, x, π outputs 1 if accepting and 0 if rejecting the proof. The zap is complete, sound, and witness-indistinguishable as defined below.

Definition 9 (Completeness of zap). We say $(\ell_{\text{zap}}, P_{\text{zap}}, V_{\text{zap}})$ is (perfectly) complete if for all $(x, w) \in R$ we have

$$\Pr\left[\sigma \leftarrow \{0,1\}^{\ell_{\text{zap}}(k)}; \pi \leftarrow P_{\text{zap}}(\sigma, x, w) : V_{\text{zap}}(\sigma, x, \pi) = 1\right] = 1.$$

Definition 10 (Soundness of zap). We say $(\ell_{\text{zap}}, P_{\text{zap}}, V_{\text{zap}})$ is (statistically) sound if for all adversaries $\mathcal{A}$ we have

$$\Pr\left[\sigma \leftarrow \{0,1\}^{\ell_{\text{zap}}(k)}; (x, \pi) \leftarrow \mathcal{A}(\sigma) : x \notin L \ \wedge \ V_{\text{zap}}(\sigma, x, \pi) = 1\right] \approx 0.$$

Definition 11 (Witness-indistinguishability of zap). We say $(\ell_{\text{zap}}, P_{\text{zap}}, V_{\text{zap}})$ is (computationally) witness-indistinguishable if for all non-uniform polynomial time interactive adversaries $\mathcal{A}$ we have

$$\Pr\left[\sigma \leftarrow \{0,1\}^{\ell_{\text{zap}}(k)}; (x, w_0, w_1) \leftarrow \mathcal{A}(\sigma); b \leftarrow \{0,1\}; \pi \leftarrow P_{\text{zap}}(\sigma, x, w_b) : \mathcal{A}(\pi) = b\right] \approx \frac{1}{2},$$

where we require $\mathcal{A}$ outputs (x, w_0, w_1) such that $(x, w_0) \in R$ and $(x, w_1) \in R$.

We will use zaps for circuit satisfiability. The verifier may not know the size of the circuit when generating the common reference string so we need zaps that work for arbitrarily large circuits. Following Dwork and Naor's construction such zaps exist if we have NIZK proofs that work for arbitrarily large circuits. Using the non-interactive version of Naor's statistically binding commitment scheme with security based on the existence of one-way functions [25,43] such an NIZK proof can be constructed by committing to each wire-value in the circuit and making NIZK proofs for each wire commitment that it contains 0 or 1 and making NIZK proofs for each gate that the committed values respect the gate. We conclude that one-way functions can be used to stretch NIZK proofs to work for arbitrary circuit sizes and hence to get zaps for arbitrary circuit sizes.

MULTI-STRING NIZK PROOFS A common reference 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 initial messages $\sigma_1, \ldots, \sigma_n$ for the statement

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

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 $(\Sigma_1, \ldots, \Sigma_n)$, a statement x and a witness w such that $(x, w) \in R$, replace any malformed Σ_i with the default CRS $\Sigma_i = (0^{2k}, 0^{\ell_{\text{zap}}(k)})$ and construct a polynomial size circuit C that is satisfiable if and only if

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

The prover does this using a witness-preserving NP-reduction so it can use w to compute 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 $\Pi := (\pi_1, \ldots, \pi_n)$.

Verification: Given n common reference strings $(\Sigma_1, \ldots, \Sigma_n)$ (again replacing any malformed Σ_i with the default CRS $(0^{2k}, 0^{\ell_{\text{zap}}(k)})$), a statement x and a proof $\Pi = (\pi_1, \ldots, \pi_n)$ 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 $(\Sigma_1, \ldots, \Sigma_n)$, $(\tau_1, \ldots, \tau_n)$, x such that there are exactly t_z non-trivial τ_i where $r_i = \text{PRG}(\tau_i)$ we wish to simulate a proof Π. As in a proof, use the witness-preserving NP-reduction to get a circuit C that is satisfiable if and only if $x \in L$ or $|\{r_i \mid \exists \tau_i : r_i = \text{PRG}(\tau_i)\}| \geq t_z$. Use the t_z values $\tau_i \neq \perp$ to get $r_i = \text{PRG}(\tau_i)$ and compute 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 $\Pi := (\pi_1, \ldots, \pi_n)$.

Lemma 12. *The construction given above is a $(0, t_s, t_z, n)$-NIZK proof with uniformly random common reference strings for Circuit Satisfiability assuming PRG is a length-doubling pseudorandom number generator and $(\ell, P_{\text{zap}}, V_{\text{zap}})$ is a zap.*

Proof. Direct verification of our construction reveals that we have perfect completeness even for $t_c = 0$.

Let us now prove that we have statistical $(0, t_s, t_z, n)$-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 $\Sigma_1, \ldots, \Sigma_n$ 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 the zap there is negligible probability that there exists a valid zap on this initial message for C being satisfiable.

We now turn to the question of computational $(0, t_s, t_z, n)$-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. □

Theorem 13. *Assuming one-way functions exist,*[1] *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 uniformly random multi-string model.*

[1] Assuming the existence of NIZK proofs the existence of one-way functions is equivalent to the existence of hard on average languages in NP [45,46].

Proof. If one-way functions exist then pseudorandom generators exist [40]. If NIZK proofs exist in the common random string model then zaps exist in the common random string model [20]. Lemma 12 now shows that if one-way functions and NIZK proofs with common random strings exist then multi-string NIZK proofs exist with uniformly random strings.

Next, we will show that the existence of (n, t_s, t_z, n)-NIZK proofs implies the existence of standard NIZK proofs. The key generator picks n common reference strings for the NIZK proof and concatenates them to get a single common reference string. The prover interprets the common reference string as n common reference strings and runs the multi-string prover. The verifier interprets the common reference string as n common reference strings and runs the multi-string verifier. Completeness, soundness, and zero-knowledge follow directly from the multi-string completeness, soundness, and zero-knowledge. If the multi-string NIZK proof uses random strings then we get a random string NIZK proof. □

4. Multi-string Simulation-Extractable NIZK Proofs

We will now construct advanced multi-string NIZK proofs of knowledge that are both simulation-extractable and extraction zero-knowledge.

To permit the extraction of witnesses, we include a public key for an encryption scheme secure against adaptive chosen ciphertext attacks $(K_{\mathrm{CCA2}}, E, D)$ in each common reference string. The encryption scheme should have perfect decryption. To deal with bad public keys injected by the adversary we require without loss of generality that the encryption algorithm on an invalid public key outputs some ciphertext although with no guarantees of security or decryptability. 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, the extractor decrypts t_s of these ciphertexts and combines the shares to get the witness.

We will use a strong one-time signature scheme $(K_{\mathrm{sots}}, \mathrm{Sign}, \mathrm{Vfy})$ to prevent modifications of valid proofs. The prover generates a key $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}}) \leftarrow K_{\mathrm{sots}}(1^k)$ 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.

Each common reference string will contain a random $2k$-bit value, which in a simulation string will instead 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 PRF 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 adaptive chosen ciphertext attack security of the encryption scheme. Even after many extractions the ciphertexts

still reveal nothing about the witness or whether the seed for a pseudorandom function has been used to simulate a proof.

Common reference string/simulation string: Generate $(pk_1, dk_1), (pk_2, dk_2) \leftarrow K_{\mathrm{CCA2}}(1^k)$; $r \leftarrow \{0,1\}^{2k}$; $\sigma \leftarrow \{0,1\}^{\ell_{\mathrm{zap}}(k)}$. 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 $\xi := dk_1$.
In case the adversary supplies us with malformed common reference strings, we replace them with the default common reference string $\Sigma := K(1^k; 0)$, so in the following we will without loss of generality assume that all common reference strings have the form (pk_1, pk_2, r, σ).

Proof: $P((\Sigma_1, \ldots, \Sigma_n), x, w)$ where $(x, w) \in R$ runs as follows: Generate a key pair for the 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 $c_{1i} := E_{pk_{1i}}(w_i, vk_{\mathrm{sots}}; r_{1i})$ for $i = 1, \ldots, n$ and encrypt dummy values $c_{2i} \leftarrow E_{pk_{2i}}(0^{2k})$. 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 such that $(x, w) \in R$ *or* there exist at least t_z seeds τ_i such that $r_i = \mathrm{PRF}_{\tau_i}(0)$ and c_{2i} encrypts $\mathrm{PRF}_{\tau_i}(vk_{\mathrm{sots}})$."
Reduce this statement to a polynomial size circuit C and a satisfiability witness W. For all i create a zap $\pi_i \leftarrow P_{\mathrm{zap}}(\sigma_i, C, W)$ for C being satisfiable. Finally, sign everything using the strong 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 of 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 t_s extraction keys in $\vec{\xi}$ to decrypt the corresponding t_s ciphertexts. Check that the plaintexts are of the form $(w_i, vk_{\mathrm{sots}})$ and combine the t_s secret shares to recover the witness w. If any of the checks fail return $\perp$.

Simulated proof: To simulate a proof use the t_z simulation trapdoors in $\vec{\tau}$. These are τ_i such that $r_i = \mathrm{PRF}_{\tau_i}(0)$. 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 $c_{2i} \leftarrow E_{pk_{2i}}(0^{2k})$. Let $w_1, \ldots, w_n$ be a (t_s, n)-threshold secret sharing of 0 and encrypt these values as $c_{1i} \leftarrow E_{pk_{1i}}(w_i, vk_{\mathrm{sots}})$. Let again C be the circuit corresponding to 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 *or* there exist at least t_z seeds τ_i such that $r_i = \mathrm{PRF}_{\tau_i}(0)$ and c_{2i} encrypts $\mathrm{PRF}_{\tau_i}(vk_{\mathrm{sots}})$."
From the creation of the ciphertexts c_{2i} we have a witness W for C being satisfiable. Create zaps $\pi_i \leftarrow P_{\mathrm{zap}}(\sigma_i, C, W)$ for C being satisfiable. Finally, make a strong one-time signature on everything $sig \leftarrow \mathrm{Sign}_{sk_{\mathrm{sots}}}(vk_{\mathrm{sots}}, x, \Sigma_1, c_{11}, c_{21}, \pi_1, \ldots, \Sigma_n, c_{1n}, c_{2n}, \pi_n)$. The simulated proof is $\Pi := (vk_{\mathrm{sots}}, c_{11}, c_{21}, \pi_1, \ldots, c_{1n}, c_{2n}, \pi_n, sig)$.

Lemma 14. *The construction given above is a $(0, t_s, t_z, n)$-NIZK proof of knowledge for Circuit Satisfiability for all choices of $t_s + t_z > n$ with simulation-extractability and*

extraction zero-knowledge. If the public-key encryption scheme has random public keys then the construction has uniform random strings.[2]

Proof. Perfect completeness follows by direct verification. Common reference strings and simulated reference strings are indistinguishable by the pseudorandomness of the pseudorandom function PRF. The (t_z, n)-zero knowledge property follows from the extraction zero-knowledge property that we will now prove.

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 created either by the prover using a real witness or the simulator using the simulation trapdoors. 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 the challenge proof using the witness w as a real prover would do.

Hybrid 2: We modify hybrid 1 by encrypting t_z pseudorandom values in $c_{21}, \ldots, c_{2n}$ when making the challenge proof. We know t_z seeds τ_i such that $r_i = \mathrm{PRF}_{\tau_i}(0)$. Instead of setting $c_{2i} \leftarrow E_{pk_2}(0^{2k})$, we encrypt $c_{2i} \leftarrow E_{pk_2}(\mathrm{PRF}_{\tau_i}(vk_{\mathrm{sots}}))$.
By the semantic security of the encryption scheme, 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 $c_{21}, \ldots, c_{2n}$ to form a witness W for C being satisfiable. We use this witness in the zaps when creating the challenge proof instead of using 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's extraction query is of the form $(\vec{\Sigma}, I, x, \Pi)$, where the index contains some i where Σ_i has been generated by SE'_1 and c_{1i} is recycled from the challenge proof then the extraction oracle returns $\perp$.
To make a valid proof, the adversary has to give a strong one-time signature using her chosen verification key. 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. This means c_{1i} contains the wrong verification key. The regular behavior of the extraction oracle in hybrid 3 would therefore also be to return $\perp$, so there is negligible difference between hybrid 3 and hybrid 4.

Hybrid 5: We modify hybrid 4 by making a (t_s, n)-threshold secret sharing $w_1, \ldots, w_n$ of 0 instead of secret sharing w and encrypting these shares as $c_{1i} \leftarrow E_{pk_{1i}}(w_i, vk_{\mathrm{sots}})$ in the challenge proof.
Hybrid 4 and hybrid 5 are indistinguishable. We have ruled out that the adversary ever makes an extraction query requiring decryption of a ciphertext c_{1i} that has been recycled from the challenge proof. A hybrid argument using the chosen ciphertext attack security of the encryption scheme can now be used to see that 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 whether w or 0 has been secret shared.

[2] CCA2-secure public-key encryption with random public-keys can for instance be constructed as a variant of Cramer–Shoup encryption [17] defined over a suitable prime order group since primes can be sampled from uniform strings and the public key group elements are random.

Hybrid 6: We modify hybrid 5 by switching back to the original extraction oracle. Hybrid 6 is identical to the case where we give the adversary a simulated challenge proof.

Since there is negligible probability of forging a strong one-time signature, we can give a similar argument as we did when going from hybrid 3 to hybrid 4 that either type of extraction oracle with overwhelming probability outputs $\perp$ when facing a query $(\vec{\Sigma}, I, x, \Pi)$ where the index indicates some i where c_{1i} has been recycled from the challenge proof.

Next, let us consider simulation-sound extractability. Here the adversary sees the extraction keys but not the simulation trapdoors of the common reference strings generated by SE'_1. She has access to a simulation oracle and in the end she 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 tell us that $c_{11}, \ldots, c_{1n}$ 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 the construction gives a *statistical* $(0, t_s, t_z, n)$-proof of knowledge. With overwhelming probability a random $2k$-bit value r is not pseudorandom, so by the statistical soundness of the zaps $c_{11}, \ldots, c_{1n}$ must encrypt a (t_s, n)-threshold secret sharing of a witness for $x \in L$. □

Theorem 15. *If NIZK proofs with uniform random strings and CCA2-secure public key encryption with uniformly random public keys exist, then multi-string NIZK proof of knowledge with simulation-extractability and extraction zero-knowledge exist in the uniform random strings model for all languages in NP.*

Proof. CCA2-secure public key encryption implies the existence of one-way functions, from which it is possible to construct both pseudorandom functions [31] and strong one-way functions. NIZK proofs with uniform random strings imply the existence of zaps. Lemma 14 now gives us the existence of multi-string NIZK proofs of knowledge with simulation-extractability and extraction zero-knowledge in the uniform random strings model. □

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 a security parameter specifying the size of the bilinear group elements. Typically, n will be much smaller than $|C|$, so the complexity matches the best known NIZK proofs for circuit satisfiability in the single common reference string model [39] 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 $\mathbb{G}$.
- $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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 efficiently verifiable that indeed it is a bilinear group and that g generates $\mathbb{G}$.
- There is an efficient sampling algorithm that given a random string of $61k$ bits interprets it as 60 statistically close to uniformly random group elements. The sampling algorithm is efficiently reversible, such that given 60 group elements we can pick at random one of the $61k$-bit strings that would lead to sampling them.
- The length of the description of $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is at most $4k$ bits.[3]
- If uniformly random reference strings are desired we will additionally assume $\mathcal{G}$ simply outputs a uniformly random $4k$-bit string from which $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ can be sampled.

The security of our multi-string NIZK proof will rely on the decisional linear (DLIN) assumption introduced by Boneh, Boyen, and Shacham [10], 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 $\mathcal{G}$, which we formally define below.

Definition 16 (DLIN group generator). A bilinear group generator $\mathcal{G}$ as described above is a DLIN group generator if for all non-uniform polynomial time adversaries $\mathcal{A}$

$$\begin{aligned}
&\Pr\big[(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}\big(1^k\big); \phi, \eta, r, s \leftarrow \mathbb{Z}_p; t = r + s : \\
&\qquad \mathcal{A}\big(p, \mathbb{G}, \mathbb{G}_T, e, f, g, h, f^r, g^s, h^t\big) = 1\big] \\
&\approx \Pr\big[(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}\big(1^k\big); \phi, \eta, r, s, t \leftarrow \mathbb{Z}_p : \\
&\qquad \mathcal{A}\big(p, \mathbb{G}, \mathbb{G}_T, e, f, g, h, f^r, g^s, h^t\big) = 1\big].
\end{aligned}$$

Example We will offer a class of candidates for DLIN groups as described above. Consider the elliptic curve $y^2 \equiv x^3 + 1 \bmod q$, where $q \equiv 2 \bmod 3$ is a prime. It is straightforward to check whether a point (x, y) is on the curve. Furthermore, picking $y \in \mathbb{Z}_q$ and computing $x \equiv (y^2 - 1)^{(1+2(q-1))/3} \bmod q$ gives us a point on the curve. The curve has a total of $q + 1$ points after including also the point at infinity.

When generating bilinear groups, we will pick p as a k-bit prime. We then let $q \equiv 2 \bmod 3$ be the smallest prime[4] such that $p|q + 1$ and define $\mathbb{G}$ to be the order p

[3] The constant 4 is chosen to exceed the size of current descriptions of bilinear groups. It is, however, easy to modify our protocol to work whenever the description of the bilinear group is $\mathcal{O}(k)$ bits.

[4] In other words, q is the smallest prime in the arithmetic progression $3p-1, 6p-1, 9p-1, \ldots$. Granville and Pomerance [36] conjecture that it requires $\mathcal{O}(k^2)$ steps in this progression to encounter such a prime q.

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 [9]. Verification of $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ being a bilinear group is straightforward, since it corresponds to checking that p, q are primes such that $p|q+1$ and $q \equiv 2 \bmod 3$ and g is an order p element on the curve. A random point on the curve can be sampled by picking $y \leftarrow \mathbb{Z}_q \cup \{\infty\}$ and solving for the unique x such that $y \equiv x^3 + 1 \bmod q$ or letting it be the point of infinity in case $y = \infty$. A random element of the group $\mathbb{G}$ can be sampled by picking a random point (x, y) on the curve and raising it to $\frac{1+2(q-1)}{p}$. These sampling processes are reversible since multiplying a group element with a random point on the curve of order $\frac{1+2(q-1)}{p}$, i.e., a random point raised to 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 $u, v \leftarrow \mathbb{G}$, 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 [44]. 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 $u_1, v_1, \ldots, u_N, v_N$ 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 $(f_1, h_1, \ldots, f_M, h_M, u_1, v_1, \ldots, u_N, v_N, w_{11}, \ldots, w_{MN})$ 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, $E_{1,1}$ 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 $\pi = u^{y/x}$ and (u, v, w) is a linear tuple if and only if there is a π such that $e(u, h) = e(f, \pi)$ and $e(g, \pi v) = e(w, h)$. In other words, we can provide

MN 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 [37] 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 Sect. 3 could be implemented efficiently using groups with a bilinear map. However, 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 [37].

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 $r_i \oplus s_i$ 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 we are careful and chooses a pseudorandom generator that stretches the input sufficiently then we 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 such that $r_i \oplus s_i$ 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 $s_1, \ldots, s_n$ 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 [37]. Also, pick a random string $r \leftarrow \{0,1\}^{61k}$. Output $\Sigma := (p, \mathbb{G}, \mathbb{G}_T, e, g, \sigma, r)$.

If one can sample DLIN groups from uniformly random strings the common reference strings can be uniformly random bit-strings.

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.

It is possible to efficiently verify the validity of the bilinear group, the correct length of r and that σ gives perfect completeness. If either check fails we will assume the prover and verifier replace it with a default common reference string.

Proof: Given $(\Sigma_1, \ldots, \Sigma_n), C, w$ such that $C(w) = 1$ 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 [37] 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 group element 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 encryption scheme [10], 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 a tuple z_i of 60 group elements.

Make a non-interactive zap π using the group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ and combining techniques of [39] and [37] for the following statement:

$$C \text{ satisfiable } \vee \left(\prod_{i=1}^{n} c_i \text{ encrypts } g^{t_z} \wedge \forall i : \right. \tag{1}$$

$$\left. c_i \text{ encrypts } g^0 \text{ or } g^1 \ \wedge \ \left(z_i \text{ is a pseudorandom value } \vee c_i \text{ encrypts } g^0\right) \right).$$

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

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

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

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 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 $\Sigma_1, \ldots, \Sigma_n$, of which t_z 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 such that 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 such that $r_i \oplus s_i$ 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 for (1) without knowing C's satisfiability witness.

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

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

Theorem 17. *If $\mathcal{G}$ is a DLIN group generator as defined in the beginning of this section, 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 scheme can be set up with uniformly random reference strings if we can sample groups with bilinear maps from uniformly 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 we can use uniformly random strings. 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 sampling pseudorandom 60-tuples of group elements 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 $\mathbb{G}$ such that 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 $\mathbb{G}_i$ and $\mathbb{G}$, respectively.

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 such that $r_i \oplus s_i$ 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 $g^0, \ldots, g^{t_z-1}$. 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 cannot 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 encryption scheme, the adversary cannot detect this change. We already select s_i such that $r_i \oplus s_i$ specifies a pseudorandom value for the reference strings not generated by S_1. Let us switch to also selecting s_i such that $r_i \oplus s_i$ 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. □

6. Multi-party Computation in the UC Framework

In the rest of the paper, we will work in Canetti's UC framework. The universal composability (UC) framework, see [11] for a detailed description, is a strong security model capturing security of a protocol under concurrent execution of arbitrary protocols. We model everything happening concurrently with the protocol but not directly related to it 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 may 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. She can adaptively corrupt parties, in which case she learns the entire history of the party and gains complete control over the actions of the 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 and when doing so it learns the history of the party. As the real world adversary, $\mathcal{S}$ can freely communicate with the environment.

Comparing these two models we 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.

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

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

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

String selection: On input (**multi**, *sid*, $\Sigma_1, \ldots, \Sigma_n$) where $\Sigma_1, \ldots, \Sigma_n \in \{0,1\}^{\ell_{\text{mcrs}}(k)}$ from $\mathcal{S}$ check that more than half of the strings $\Sigma_1, \ldots, \Sigma_n$ match stored strings. In that case output (**multi**, *sid*, $\Sigma_1, \ldots, \Sigma_n$) to all parties and halt.

Fig. 1. The ideal multi-string generator.

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

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

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

Fig. 2. The ideal authenticated broadcast functionality.

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

We will assume parties can broadcast their messages and we make this assumption explicit by giving them access to the ideal broadcast functionality in Fig. 2. 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 [33]. 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.

6.1. *Tools*

We will now present a few tools that we will need in our constructions.

ENCRYPTION SCHEME WITH PSEUDORANDOM PUBLIC KEYS AND PSEUDORANDOM CIPHERTEXTS An encryption scheme $(K_{\text{pseudo}}, E, D)$ has pseudorandom ciphertexts of length $\ell_E(k)$ if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

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

where $R_{pk}(m)$ runs $c \leftarrow \{0,1\}^{\ell_E(k)}$ and returns c. Moreover, an encryption scheme has pseudorandom public keys of length $\ell_K(k)$ if correctly generated public keys are computationally indistinguishable from random strings of length $\ell_K(k)$ bits.

We will use an encryption scheme with pseudorandom public keys and pseudorandom ciphertexts with errorless decryption. Damgård and Nielsen [19] show that such encryption schemes, which they call simulatable encryption schemes, exist under standard assumptions such as RSA or DDH.

TAG-BASED SIMULATION-SOUND TRAPDOOR COMMITMENT A tag-based simulation sound trapdoor commitment scheme [42] has four probabilistic polynomial time algorithms $(K_{\text{tag-com}}, \text{Com}, \text{TCom}, \text{Topen})$. The key generation algorithm $K_{\text{tag-com}}$ produces a commitment key ck as well as a trapdoor key tk. The commitment key specifies amongst other things the message space, which we will assume is of the form $\{0,1\}^{\ell}$ for some suitable ℓ polynomially bounded by k. There is a commitment algorithm that takes as input the commitment key ck, a message $m \in \{0,1\}^{\ell}$ and any tag $tag \in \{0,1\}^*$ 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.

The algorithms Tcom, Topen 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 $(c, ek) \leftarrow \text{Tcom}_{tk}(tag)$. Later we can open it to any message m as $r \leftarrow \text{Topen}_{ek}(tag; m)$, 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 the following trapdoor property:

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

where $\mathcal{R}(m, tag)$ returns uniformly random coins r and $\mathcal{O}(m, tag)$ computes $(c, ek) \leftarrow \text{Tcom}_{tk}(tag)$; $r \leftarrow \text{Topen}_{ek}(tag, m)$ and returns r. Both oracles ignore queries with tags that have already been queried.

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 the following simulation-soundness property:

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

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 commitments were implicitly used in Di Crescenzo, Ishai and Ostrovsky [25]. The explicit term was coined by Garay, MacKenzie, and Yang [27], while the definition presented here is from MacKenzie and Yang [42]. The latter paper offers a construction based on one-way functions as well as more efficient constructions based on the DSA or the strong RSA assumptions.

Since we are working over random strings, we want $K_{\text{tag-com}}$ to output public keys that are random or pseudorandom. We say the commitment scheme has pseudorandom keys if there is a polynomial $\ell_{\text{tag-com}}$ such that $ck \leftarrow \{0,1\}^{\ell_{\text{tag-com}}(k)}$ is computationally indistinguishable from correctly generated commitment keys. This property can also be achieved based on the existence of one-way functions.

TAG-BASED SIMULATION-EXTRACTABLE COMMITMENT SCHEME We will need something stronger than tag-based simulation-sound trapdoor commitments, namely tag-based simulation-extractable commitments. A tag-based simulation-extractable commitment is a tag-based simulation-sound trapdoor commitment scheme ($K_{\text{se-com}}$, SCom, STcom, STopen) with an additional algorithm Extract that given an extraction key ξ is able to extract the committed message. More precisely, with the trapdoor key we can make equivocal commitments, however, for all other tags the adversary will with overwhelming probability make commitments from which a unique possible message can be extracted. For all non-uniform polynomial time adversaries $\mathcal{A}$ we have the following simulation-extractability property:

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

where $\mathcal{O}(\text{Com}, tag)$ computes $(c, ek) \leftarrow \text{STcom}_\tau(tag)$, returns c and stores (c, tag, ek), and $\mathcal{O}(\text{Open}, c, m, tag)$ returns $r \leftarrow \text{STopen}_{ek}(tag, m)$ if some (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 tag-based simulation-sound trapdoor commitments and encryption schemes with pseudorandom public keys and pseudorandom ciphertexts. 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 $c_1, \ldots, c_\ell$. When making equivocal commitments, we can use the trapdoor key to create equivocal commitments $c_1, \ldots, c_\ell$ that can be opened to any bit we like.

We need to make this commitment scheme extractable. We therefore encrypt the openings of the commitments. Now we can extract messages but we have reintroduced the problem of equivocation. An equivocal commitment may have 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 encryption scheme 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 [14].

To recap, the idea so far is to commit to a bit b by making 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. We now present the full commitment scheme in Fig. 3.

Theorem 18. *Tag-based simulation-extractable commitment schemes exist if encryption schemes with pseudorandom ciphertexts exist. If encryption schemes with pseudorandom public keys and pseudorandom ciphertexts exist then tag-based simulation-extractable commitments with pseudorandom public keys exist.*

Proof. Tag-based simulation-sound trapdoor commitments with pseudorandom keys can be built from one-way functions [42] so the assumptions give us the tools needed

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), where $m = m_1 m_2 \ldots m_\ell$ when written in binary, 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 $C := (c_1, C_{10}, C_{11}, \ldots, c_\ell, C_{\ell 0}, C_{\ell 1})$

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

Equivocal 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 such that $c_i = \text{Com}_{ck}((tag, i); 0; r_{i0}) = \text{Com}_{ck}((tag, i); 1; r_{i1})$.
2. For $i = 1$ to ℓ select randomness $R_{i,b}$ and set $C_{ib} := E_{pk}(r_{ib}; R_{i,b})$.
3. Return $C := (c_1, C_{10}, C_{11}, \ldots, c_\ell, C_{\ell 0}, C_{\ell 1})$ and $EK := (\sigma, r_1, R_{1,0}, R_{1,1}, \ldots, r_\ell, R_{\ell,0}, R_{\ell,1})$.

Equivocation: On input (tag, EK, C, m) return $(r_{1,m_1}, R_{1,m_1}, \ldots, r_{\ell,m_\ell}, R_{\ell,m_\ell})$.
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} such that $c_i = \text{Com}_\sigma((tag, i); b; r_{ib})$, we set $m_i := b$. In case all these processes succeed, we return the concatenation m, else we return $\perp$.

Fig. 3. Tag based simulation-extractable commitment.

in the construction in Fig. 3. This also shows that we have pseudorandom keys for the tag-based simulation-extractable commitment scheme if the encryption scheme has pseudorandom public keys.

We now need to prove that even after seeing equivocal commitments and equivocations it is hard to come up with a commitment with a different tag and open this commitment to a message that differs from the extracted message. Consider first the case where the adversary for some index i creates c_i, C_{i0}, C_{i1} such that 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 equivocated before so this breaks the simulation-sound binding property of the tag-based simulation-sound trapdoor commitment. The errorless decryption property of the pseudorandom encryption scheme 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 oracle in several steps and show that $\mathcal{A}$ cannot tell the difference. Let us start with the oracle $R(\cdot,\cdot)$ that on input (tag, m) returns a randomly chosen randomizer $r_1, R_{1,m_1}, C_{1,1-m_1}, \ldots, r_\ell, R_{\ell,m_\ell}, C_{1,1-m_\ell}$. Instead of making commitments $c_i := \text{Com}_{ck}((tag, i); m_i; r_i)$, we may instead run $(c_i, ek_i) \leftarrow \text{Tcom}_{tk}(tag, i); r_i \leftarrow$

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

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

Commitment: On input (**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 (**commit**, *sid*, P_i) to all parties and $\mathcal{S}$. Ignore future (**commit**, *sid*, ·) inputs from P_i.

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

Fig. 4. The ideal commitment functionality.

$\text{Topen}_{ek_i}(m_i)$ 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 oracle $O(\cdot,\cdot)$ where we make equivocations to both r_{i0} and r_{i1} such that $c_i = \text{Com}_{ck}((tag, i); b; r_{i,b})$ 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 previously modified $R(\cdot,\cdot)$ oracle. □

6.2. *UC Commitment in the Multi-string Model*

We will now show how to securely realize the ideal UC commitment functionality $\mathcal{F}_{\text{COM}}^{1:N}$ described in Fig. 4 in the multi-string model. Let us start by giving some intuition behind our construction. 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 $\mathcal{F}_{\text{BC}}$, $\mathcal{F}_{\text{MCRS}}$ and $P_1, \ldots, P_N$. There are two general types of issues that can come up in the ideal process simulation. First, when $\mathcal{F}_{\text{COM}}^{1:N}$ tells $\mathcal{S}$ that a party has committed to some message, $\mathcal{S}$ does not know which message it is but has to simulate to $\mathcal{A}$ that this party makes a UC commitment. Therefore, we want to be able to make equivocal commitments and later open them to any value. Second, if 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 need to extract the message from the UC commitment.

As a tool to get both the equivocation/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, $\mathcal{S}$ simulates $\mathcal{F}_{\text{MCRS}}$ and can therefore pick the honest 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 an equivocal commitment, $\mathcal{S}$ makes honest commitments to $n - t$ random shares for the adversarial keys and equivocal commitments with the t simulation-extractable keys. Since the adversary knows at most $n - t < t$ shares, $\mathcal{S}$ can later open the commitment to any message by making suitable trapdoor openings of the latter t shares. To extract a message m from a UC commitment made by the adversary, $\mathcal{S}$ ex-

tracts t shares from the simulation-extractable commitments and combines the shares to get the only possible adversarial message m.

One remaining issue is when the adversary recycles a commitment or parts of it. We may risk that it uses an equivocal commitment made by an honest party, in which case $\mathcal{S}$ is unable to extract a message. To guard against this problem, we 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 and enabling $\mathcal{S}$ to extract.

Another problem arises when $\mathcal{A}$ corrupts a party, which enables it to send messages on behalf of this party. At this point, however, $\mathcal{S}$ learns the message so we just need to force $\mathcal{A}$ to reuse the same message if it reuses parts of the equivocal 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 in the tag, which in turn binds her to use the same share as the one the party committed to before being corrupted. We now describe the full UC commitment scheme, which has the additional benefit of being non-interactive.

Commitment: On input $(\textbf{multi}, sid, (ck_1, \sigma_1), \ldots, (ck_n, \sigma_n))$ from $\mathcal{F}_{\text{MCRS}}$ and $(\textbf{commit}, sid, m)$ from $\mathcal{Z}$, the party P_i does the following. She makes a (t, n)-threshold secret sharing $s_1, \ldots, s_n$ 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 $C_j := \text{SCom}_{\sigma_j}((P_i, c_j); s_j; R_j)$. The commitment is $c := (c_1, C_1, \ldots, c_n, C_n)$. She broadcasts $(\textbf{broadcast}, sid, \text{commit}, c)$ (using $ssid = \text{commit}$).

Receiving commitment: A party on input $(\textbf{multi}, sid, (ck_1, \sigma_1), \ldots, (ck_n, \sigma_n))$ from $\mathcal{F}_{\text{MCRS}}$ and $(\textbf{broadcast}, sid, \text{commit}, P_i, c)$ from $\mathcal{F}_{\text{BC}}$ outputs $(\textbf{commit}, sid, P_i)$ to the environment.

Opening commitment: Party P_i wishing to open her commitment broadcasts $(\textbf{broadcast}, sid, \text{open}, s_1, r_1, R_1, \ldots, s_n, r_n, R_n)$ (using $ssid = \text{open}$).

Receiving opening: A party receiving an opening $(\textbf{broadcast}, sid, \text{open}, P_i, s_1, r_1, R_1, \ldots, s_n, r_n, R_n)$ from $\mathcal{F}_{\text{BC}}$ to a commitment she received earlier checks that all commitments are correctly formed $c_j = \text{Com}_{ck_j}(s_j; r_j)$ and $C_j = \text{SCom}_{\sigma_j}((P_i, c_j); s_j; R_j)$. She also checks that $s_1, \ldots, s_n$ are valid shares of a (t, n)-threshold secret sharing of some message $m \in \{0, 1\}^{\ell(k)}$. In that case she outputs $(\textbf{open}, sid, P_i, m)$ to the environment.

Theorem 19. *$\mathcal{F}_{\text{COM}}^{1:N}$ can be securely realized in the $(\mathcal{F}_{\text{BC}}, \mathcal{F}_{\text{MCRS}})$-hybrid model assuming tag-based simulation-extractable commitment schemes with pseudorandom public commitment keys exist.*

Proof. We claim that the protocol described above is a secure realization of $\mathcal{F}_{\text{COM}}^{1:N}$. Let us describe the ideal-process adversary $\mathcal{S}$, which runs a black-box simulation of $\mathcal{A}$. In particular, $\mathcal{S}$ simulates the parties $P_1, \ldots, P_N$ and the ideal functionalities $\mathcal{F}_{\text{MCRS}}$ and $\mathcal{F}_{\text{BC}}$. The dummy parties that are actually involved in the protocol and communicate with $\mathcal{Z}$ are written as $\tilde{P}_1, \ldots, \tilde{P}_N$.

Communication: $\mathcal{S}$ forwards all communication between the simulated adversary $\mathcal{A}$ and the environment $\mathcal{Z}$. Also, whenever $\mathcal{A}$ delivers a message to a party P_i she simulates this message 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 (**crs**, *sid*, (ck, σ)), while storing the corresponding keys $(ck, tk, \sigma, \tau, \xi)$.

When $\mathcal{A}$ inputs (**multi**, *sid*, $(ck_1, \sigma_1), \ldots, (ck_n, \sigma_n)$) to $\mathcal{F}_{\text{MCRS}}$ $\mathcal{S}$ checks that more than half the pairs $(ck_1, \sigma_1), \ldots, (ck_n, \sigma_n)$ match the stored public keys. In that case, $\mathcal{S}$ sends (**multi**, *sid*, $(ck_1, \sigma_1), \ldots, (ck_n, \sigma_n)$) to all parties and halts the simulation of $\mathcal{F}_{\text{MCRS}}$. Note, we only need t stored keys so if there are more than t honest public key pairs $\mathcal{S}$ just uses t of the secret key tuples.

Commitment by honest party: On receiving (**commit**, *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. If the multi-string has not yet been delivered, $\mathcal{S}$ waits until $\mathcal{A}$ has submitted reference strings to $\mathcal{F}_{\text{MCRS}}$ and delivers them to P_i.

$\mathcal{S}$ makes a (t, n)-threshold secret sharing $s_1, \ldots, s_n$ of 0. For the $n - t$ reference strings where she does not know the trapdoors she commits to s_j as $c_j := \text{Com}_{tk}(s_j; r_j)$ and $C_j := \text{SCom}_{\sigma_j}(P_i, c_j; s_j; R_j)$. For the t reference strings where she does know the trapdoors, she makes trapdoor commitments $(c_j, ek_j) \leftarrow \text{Tcom}(tk)$ and equivocal commitments $(C_j, EK_j) \leftarrow \text{STcom}_{\tau_j}(P_i, c_j)$. She simulates broadcasting (**broadcast**, *sid*, commit, $c_1, C_1, \ldots, c_n, C_n$).

Opening by honest party: When $\mathcal{S}$ receives (**open**, *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. $\mathcal{S}$ recalls the $n - t$ shares for adversarial reference strings and fit them into a (t, n)-threshold secret sharing $s_1, \ldots, s_n$ of m. She opens the $n - t$ commitment pairs c_j, C_j correctly and equivocates the t commitment pairs c_j, C_j where she knows the corresponding equivocation keys as $r_j \leftarrow \text{Topen}_{ek_j}(s_j)$ and $R_j \leftarrow \text{STopen}_{EK_j}((P_i, c_j), s_j)$. She simulates P_i broadcasting (**broadcast**, *sid*, open, $s_1, r_1, R_1, \ldots, s_n, r_n, R_n$).

Honest party receiving commitment: On $\mathcal{A}$ delivering (**broadcast**, *sid*, commit, $P_i, c_1, C_1, \ldots, c_n, C_n$) from $\mathcal{F}_{\text{BC}}$ to a party P_j deliver the corresponding (**commit**, *sid*, P_i) message from $\mathcal{F}_{\text{COM}}^{1:N}$ to the dummy party $\tilde{P}_j$.

Honest party receiving opening: On $\mathcal{A}$ delivering (**broadcast**, *sid*, open, $P_i, s_1, r_1, R_1, \ldots, s_n, r_n, R_n$) from $\mathcal{F}_{\text{BC}}$ to a party P_j check that she has received a commitment from P_i and continue if she already has received a commitment or after she receives a commitment.

Check that the commitments contain a consistent (t, n)-threshold secret sharing of $s_1, \ldots, s_n$ of a message m and for all j we have $c_j = \text{Com}_{ck_j}(s_j; r_j)$ and $C_j = \text{SCom}_{\sigma_j}(P_i, c_j; s_j; R_j)$. If the checks pass deliver (**open**, *sid*, P_i, m) from $\mathcal{F}_{\text{COM}}^{1:N}$ to the dummy party $\tilde{P}_j$ that outputs the opening to $\mathcal{Z}$.

Corruption: If $\mathcal{A}$ corrupts a party P_i, $\mathcal{S}$ corrupts the corresponding dummy party $\tilde{P}_i$. $\mathcal{S}$ needs to simulate the history of this party. Receipt of commitments and openings is already known to $\mathcal{A}$. If the party has not yet made a commitment there is therefore no history to simulate. If the party has made a commitment and already opened it $\mathcal{A}$ already has a simulated history. If the party has made a commitment but not yet

opened it, $\mathcal{S}$ must simulate an opening of the commitment. On corrupting $\tilde{P}_i$ she learns the message and she can now use the opening simulation for honest parties described earlier.

Once a party is corrupted it is controlled by $\mathcal{A}$. $\mathcal{S}$ will forward communication between $\tilde{P}_i$ and the environment $\mathcal{Z}$ as prescribed by the simulated $\mathcal{A}$.

Commitment by corrupt party: When a corrupt party P_i makes a commitment (**broadcast**, *sid*, commit, $P_i, c_1, C_1, \ldots, c_n, C_n$), $\mathcal{S}$ must input some message to $\mathcal{F}_{\text{COM}}^{1:N}$.

After simulating the receipt by $\mathcal{F}_{\text{BC}}$ $\mathcal{S}$ uses the extraction keys to extract t committed values $s_j \leftarrow \text{Extract}_{\xi_j}((P_i, c_j), C_j)$. The only case, where she cannot extract a value s_j from C_j may be when the tag (P_i, c_j) has been used before by $\mathcal{S}$ in making an equivocal commitment. However, this can only happen if P_i used (P_i, c_j) as a tag when it was honest and then upon corruption $\mathcal{S}$ made an equivocation of c_j to some s_j. But then $\mathcal{S}$ does not need to extract from C_j since it already knows the purported value s_j contained in the commitment.

After having obtained t shares $\mathcal{S}$ reconstructs m and inputs (**commit**, *sid*, m) to $\mathcal{F}_{\text{COM}}^{1:N}$ on behalf of the dummy party $\tilde{P}_i$. In case she did not manage to extract a message, she inputs $m := 0$ to $\mathcal{F}_{\text{COM}}^{1:N}$. Note that inputting a dummy value 0 is fine since as we will show in the proof we will not end up in a situation where $\mathcal{S}$ needs to ask $\mathcal{F}_{\text{COM}}^{1:N}$ to open the dummy commitment.

Opening by corrupt party: When a corrupt party wants to open a commitment by broadcasting (**broadcast**, *sid*, open, $s_1, r_1, R_1, \ldots, s_n, r_n, R_n$), $\mathcal{S}$ simulates the receipt by $\mathcal{F}_{\text{BC}}$, checks the opening and if acceptable inputs (**open**, *sid*, m) to $\mathcal{F}_{\text{COM}}^{1:N}$.

To see that this gives us a good simulation, consider the following hybrid experiments.

Hybrid 1: This is the protocol executed with $\mathcal{A}$ and environment $\mathcal{Z}$ and parties $P_1, \ldots, P_n$ in the $(\mathcal{F}_{\text{BC}}, \mathcal{F}_{\text{MCRS}})$-hybrid model.

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

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

Hybrid 3: This is hybrid 2 modified such that honest party P_i uses the t known trapdoors to make equivocal commitments corresponding to these keys to the secret shares instead of making real commitments. When opening the commitments P_i uses the equivocation keys to generate randomizers so the commitments open to the (t, n) secret sharing of the message.

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 uses a (t, n)-threshold secret sharing of 0 instead of a threshold secret sharing of m when making a commitment. In the opening phase it opens the $n - t$ pairs (c_j, C_j) where it does not know the trapdoors to the s_j's it committed to. It reconstructs shares s_j for the t equivocal commitments so $s_1, \ldots, s_n$ 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.

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

Parameterized with polynomial ℓ and running with parties $P_1, \ldots, P_N$ and adversary $\mathcal{S}$.
CRS generation: Generate random $\Sigma \leftarrow \{0,1\}^{\ell(k)}$ and output (**crs**, *sid*, Σ) to all parties and $\mathcal{S}$. Halt.

Fig. 5. The ideal common random string generator.

Hybrid 5: We now turn to modify the way we handle corrupt parties. Whenever a corrupt party P_i submits a commitment (**broadcast**, *sid*, commit, $c_1, C_1, \ldots, c_n, C_n$) to $\mathcal{F}_{\text{BC}}$ we want to extract a message.
For any of the t C_j's where we know the extraction 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 secret shares that we combine to get a possible message m. We abort the simulation if the commitment is ever successfully opened to a different message. Hybrid 4 and hybrid 5 are indistinguishable since there is negligible probability of aborting. An abort happens 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. □

6.3. *Multi-party Computation*

We will now show how to generate a common random string on the fly using UC commitments. More precisely, we will securely realize the ideal functionality $\mathcal{F}_{\text{CRS}}$ in Fig. 5 that produces a random bit-string.

COIN-FLIPPING The parties will use the natural coin-flipping protocol where all parties commit to random strings 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 (**commit**, *sid*, r_i) to $\mathcal{F}_{\text{COM}}^{1:N}$. $\mathcal{F}_{\text{COM}}^{1:N}$ on this input sends (**commit**, *sid*, P_i) to all parties.
Opening: Once P_i sees (**commit**, *sid*, P_j) for all j, it sends (**open**, *sid*) to $\mathcal{F}_{\text{COM}}^{1:N}$. $\mathcal{F}_{\text{COM}}^{1:N}$ on this input sends (**open**, *sid*, P_i, r_i) to all parties.
Output: Once P_i sees (**open**, *sid*, P_j, r_j) for all j, it outputs (**crs**, *sid*, $\bigoplus_{j=1}^{N} r_j$) and halts.

Theorem 20. *The ideal common reference string generator $\mathcal{F}_{\text{CRS}}$ can be (perfectly) securely realized in the $\mathcal{F}_{\text{COM}}^{1:N}$-hybrid model.*

Proof. We claim the protocol above securely realizes $\mathcal{F}_{\text{CRS}}$. Consider the following ideal process adversary $\mathcal{S}$ working in the $\mathcal{F}_{\text{CRS}}$-hybrid model, giving her a common ref-

erence string Σ. She runs a black-box simulation of $\mathcal{A}$, a simulated copy of $\mathcal{F}_{\text{COM}}^{1:N}$ and simulated parties $P_1, \ldots, P_N$, not to be confused with the dummy parties $\tilde{P}_1, \ldots, \tilde{P}_N$ that interact with $\mathcal{Z}$ and $\mathcal{F}_{\text{CRS}}$. Whenever the simulated $\mathcal{A}$ communicates with the environment $\mathcal{Z}$, $\mathcal{S}$ simply forwards those messages. We now list the events that can happen in the protocol.

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

On delivery of commitments from all parties to an honest party P_i, she selects r_i at random, subject to the continued satisfiability of condition $\Sigma = \bigoplus_{j=1}^{N} r_j$ and stores it. She 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, the simulator corrupts the corresponding dummy party $\tilde{P}_i$. If P_i has made a commitment but it has not yet been opened, $\mathcal{S}$ selects r_i at random, subject to the continued satisfiability of the condition $\Sigma = \bigoplus_{j=1}^{N} r_j$, and simulates 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 $P_1, \ldots, P_N$ in the $\mathcal{F}_{\text{COM}}^{1:N}$-hybrid model, and $\mathcal{S}$ running with dummy parties $\tilde{P}_1, \ldots, \tilde{P}_N$ in the $\mathcal{F}_{\text{CRS}}$-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 $\Sigma = \bigoplus_{j=1}^{N} r_j$. □

MULTI-PARTY COMPUTATION Armed with a coin-flipping protocol we can generate random strings. Canetti, Lindell, Ostrovsky, and Sahai [14] demonstrated that with access to a common random string it is possible to do any kind of multi-party computation in the presence of a malicious adaptive adversary that can corrupt any number of parties. The multi-party computation protocol of Canetti, Lindell, Ostrovsky, and Sahai [14] assumes the existence of encryption schemes with pseudorandom public keys and pseudorandom ciphertexts and enhanced trapdoor permutation, and when using a uniformly random string it also assumes the existence of encryption schemes with dense public keys and dense ciphertexts [21].

The requirement for encryption schemes with pseudorandom public keys and pseudorandom ciphertexts is not explicitly stated in [14], however, a careful reading reveals that such encryption schemes are used. Encryption Schemes with pseudorandom public keys and pseudorandom ciphertexts imply the existence of augmented non-committing encryption [19], which is also used in their construction.

Enhanced trapdoor permutation are trapdoor permutations that remain hard to invert even given the random coins of the domain sampler. Enhanced trapdoor permutations can for instance be built from the RSA assumption [13]. The construction in [13] has the additional feature that the public keys are dense so via a standard hardcore-bit transformation [29] this also gives an example of an encryption scheme with dense public keys and dense ciphertexts.

We now have the following corollary to Theorems 18, 19, and 20.

Theorem 21. *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 assuming the existence of encryption schemes with pseudorandom public keys and pseudorandom ciphertexts and assuming the existence of enhanced trapdoor permutations with dense public keys.*

Proof. From [14] we find that any well-formed functionality $\mathcal{F}$ has a non-trivial protocol that securely realizes it in the $(\mathcal{F}_{\mathrm{BC}}, \mathcal{F}_{\mathrm{CRS}})$-hybrid model assuming the existence of encryption schemes with pseudorandom public keys and pseudorandom ciphertexts and enhanced trapdoor permutations with dense public keys.

Theorem 20 shows that we can securely realize $\mathcal{F}_{\mathrm{CRS}}$ in the $\mathcal{F}_{\mathrm{COM}}^{1:N}$-hybrid model. Therefore, by the universal composability theorem [11], we can securely realize $\mathcal{F}$ in the $(\mathcal{F}_{\mathrm{BC}}, \mathcal{F}_{\mathrm{COM}}^{1:N})$-hybrid model.

Theorem 19 shows that we can securely realize $\mathcal{F}_{\mathrm{COM}}^{1:N}$ in the $(\mathcal{F}_{\mathrm{BC}}, \mathcal{F}_{\mathrm{MCRS}})$-hybrid model assuming the existence of simulation-extractable commitments, which Theorem 18 shows can be built from encryption schemes with pseudorandom public keys and pseudorandom ciphertexts. By the universal composability theorem [11] we see that $\mathcal{F}$ can be securely realized in the $(\mathcal{F}_{\mathrm{BC}}, \mathcal{F}_{\mathrm{MCRS}})$-hybrid model under the stated cryptographic assumptions. □

Acknowledgements

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

References

[1] B. Barak, R. Pass, On the possibility of one-message weak zero-knowledge, in *TCC*. Lecture Notes in Computer Science, vol. 2951 (2004), pp. 121–132
[2] B. Barak, R. Canetti, J.B. Nielsen, R. Pass, Universally composable protocols with relaxed set-up assumptions, in *FOCS* (2004), pp. 186–195
[3] B. Barak, S.J. Ong, S.P. Vadhan, Derandomization in cryptography. *SIAM J. Comput.* **37**(2), 380–400 (2007)
[4] D. Beaver, Commodity-based cryptography (extended abstract), in *STOC* (1997), pp. 446–455
[5] D. Beaver, Server-assisted cryptography, in *Workshop on New Security Paradigms* (1998), pp. 92–106
[6] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, in *STOC* (1988), pp. 1–10
[7] M. Blum, P. Feldman, S. Micali, Non-interactive zero-knowledge and its applications, in *STOC* (1988), pp. 103–112
[8] M. Blum, A. De Santis, S. Micali, G. Persiano, Noninteractive zero-knowledge. *SIAM J. Comput.* **20**(6), 1084–1118 (1991)
[9] D. Boneh, M.K. Franklin, Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **32**(3), 586–615 (2003)
[10] D. Boneh, X. Boyen, H. Shacham, Short group signatures, in *CRYPTO*. Lecture Notes in Computer Science, vol. 3152 (2004), pp. 41–55
[11] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in *FOCS* (2001), pp. 136–145
[12] R. Canetti, M. Fischlin, Universally composable commitments, in *CRYPTO*. Lecture Notes in Computer Science, vol. 2139 (2001), pp. 19–40

[13] R. Canetti, U. Feige, O. Goldreich, M. Naor, Adaptively secure multi-party computation, in *STOC* (1996), pp. 639–648
[14] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally composable two-party and multi-party secure computation, in *STOC* (2002), pp. 494–503
[15] R. Canetti, Y. Dodis, R. Pass, S. Walfish, Universally composable security with pre-existing setup, in *TCC*. Lecture Notes in Computer Science, vol. 4392 (2007), pp. 61–85
[16] D. Chaum, C. Crépeau, I. Damgård, Multiparty unconditionally secure protocols (extended abstract), in *STOC* (1988), pp. 11–19
[17] R. Cramer, V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, in *CRYPTO*. Lecture Notes in Computer Science, vol. 1462 (1998), pp. 13–25
[18] I. Damgård, Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 658 (1992), pp. 341–355
[19] I. Damgård, J.B. Nielsen, Improved non-committing encryption schemes based on a general complexity assumption, in *CRYPTO*. Lecture Notes in Computer Science, vol. 1880 (2000), pp. 432–450
[20] I. Damgård, J.B. Nielsen, Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor, in *CRYPTO*. Lecture Notes in Computer Science, vol. 2442 (2002), pp. 581–596
[21] A. De Santis, G. Persiano, Zero-knowledge proofs of knowledge without interaction, in *FOCS* (1992), pp. 427–436
[22] A. De Santis, G. Di Crescenzo, G. Persiano, Non-interactive zero-knowledge: a low-randomness characterization of NP, in *ICALP*. Lecture Notes in Computer Science, vol. 1644 (1999), pp. 271–280
[23] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, A. Sahai, Robust non-interactive zero knowledge, in *CRYPTO*. Lecture Notes in Computer Science, vol. 2139 (2002), pp. 566–598
[24] A. De Santis, G. Di Crescenzo, G. Persiano, Randomness-optimal characterization of two NP proof systems, in *RANDOM*. Lecture Notes in Computer Science, vol. 2483 (2002), pp. 179–193
[25] G. Di Crescenzo, Y. Ishai, R. Ostrovsky, Non-interactive and non-malleable commitment, in *STOC* (1998), pp. 141–150
[26] U. Feige, D. Lapidot, A. Shamir, Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Comput.* **29**(1), 1–28 (1999)
[27] J.A. Garay, P.D. MacKenzie, K. Yang, Strengthening zero-knowledge protocols using signatures. *J. Cryptol.* **19**(2), 169–209 (2006)
[28] S. Garg, V. Goyal, A. Jain, A. Sahai, Bringing people of different beliefs together to do UC, in *TCC*. Lecture Notes in Computer Science, vol. 6597 (2011), pp. 311–328
[29] O. Goldreich, L.A. Levin, A hard-core predicate for all one-way functions, in *STOC* (1989), pp. 25–32
[30] O. Goldreich, Y. Oren, Definitions and properties of zero-knowledge proof systems. *J. Cryptol.* **7**(1), 1–32 (1994)
[31] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
[32] O. Goldreich, S. Micali, A. Wigderson, How to play ANY mental game, or A completeness theorem for protocols with honest majority, in *STOC* (1987), pp. 218–229
[33] S. Goldwasser, Y. Lindell, Secure multi-party computation without agreement. *J. Cryptol.* **18**(3), 247–287 (2005)
[34] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proofs. *SIAM J. Comput.* **18**(1), 186–208 (1989)
[35] V. Goyal, J. Katz, Universally composable multi-party computation with an unreliable common reference string, in *TCC*. Lecture Notes in Computer Science, vol. 4948 (2008), pp. 142–154
[36] A. Granville, C. Pomerance, On the least prime in certain arithmetic progressions. *J. Lond. Math. Soc.* **s2–41**(2), 193–200 (1990)
[37] J. Groth, Simulation-sound NIZK proofs for a practical language and constant size group signatures, in *ASIACRYPT*. Lecture Notes in Computer Science, vol. 4248 (2006), pp. 444–459
[38] J. Groth, R. Ostrovsky, Cryptography in the multi-string model, in *CRYPTO*. Lecture Notes in Computer Science, vol. 4622 (2007), pp. 323–341
[39] J. Groth, R. Ostrovsky, A. Sahai, New techniques for noninteractive zero-knowledge. *J. ACM* **59**(3), 11:1–11:35 (2012)

[40] J. Håstad, R. Impagliazzo, L.A. Levin, M. Luby, A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)
[41] J. Kilian, E. Petrank, An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *J. Cryptol.* **11**(1), 1–27 (1998)
[42] P.D. MacKenzie, K. Yang, On simulation-sound trapdoor commitments, in *EUROCRYPT*. Lecture Notes in Computer Science, vol. 3027 (2004), pp. 382–400
[43] M. Naor, Bit commitment using pseudorandomness. *J. Cryptol.* **4**(2), 151–158 (1991)
[44] M. Naor, O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.* **58**(2), 336–375 (1999)
[45] R. Ostrovsky, One-way functions, hard on average problems, and statistical zero-knowledge proofs, in *Structure in Complexity Theory Conference* (1991), pp. 133–138
[46] R. Ostrovsky, A. Wigderson, One-way functions are essential for non-trivial zero-knowledge, in *ISTCS* (1993), pp. 3–17