Very-efficient simulatable flipping of many coins into a well*
(and a new universally-composable commitment scheme)

Luís T. A. N. Brandão**

University of Lisbon and Carnegie Mellon University
luis.papers@gmail.com

Abstract. This paper presents new cryptographic protocols for a stand-alone simulatable two-party parallel coin-flipping (into a well) and a universally composable commitment scheme, with near optimal asymptotic communication rate, in the static and computational malicious model. The approach, denoted *expand-mask-hash*, uses in both protocols a pseudo-random generator (PRG) and a collision-resistant hash function (CR-Hash) to combine separate extractable commitments and equivocable commitments (associated with short bit-strings) into a unified extractable-and-equivocable property amplified to a larger target length, amortizing the cost of base commitments. The new stand-alone coin-flipping protocol is based on a simple augmentation of the traditional coin-flipping template. To the knowledge of the author, it is the first proposal simultaneously shown to be two-side-simulatable and having an asymptotic (as the target length increases) communication rate converging to two bits per flipped coin and computation rate per party converging to that of PRG-generating and CR-hashing a bit-string with the target length. The new universally composable commitment scheme has efficiency comparable to very recent state-of-the-art constructions – namely asymptotic communication rate as close to 1 as desired, for each phase (commit and open) – while following a distinct design approach. Notably it does not require explicit use of oblivious transfer and it uses an erasure encoding instead of stronger error correction codes.

Keywords: coin-flipping, commitments, simulation, extractability, equivocability, rewinding, universal composability, cryptographic protocols.

1 Introduction

Secure two-party parallel coin-flipping is a probabilistic functionality that allows two mutually distrustful parties to agree on a common random bit-string of a certain *target* length. Using a coin-flipping protocol, both parties provide and combine independent *contributions* so that the output bit-string of an honest party is indistinguishable from random even if at most one party is malicious.

* Extended abstract – full version is at the Cryptology ePrint Archive, Report 2015/640.

** The author was supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Carnegie Mellon Portugal Program under Grant SFRH/BD/33770/2009, as a Ph.D. student at FCUL-DI and CMU-ECE.

The coin-flipping is denoted *simulatable* if it can be proven secure within the ideal/real simulation paradigm, showing that it *emulates* a protocol in an ideal world where an *ideal functionality* would decide and deliver the random bit-string to the two parties. Achieving simulatability is useful for the design of larger protocols, as it guarantees security under some type of *composition* operation, e.g., non-concurrent modular self-composition [Can00] (a.k.a. the stand-alone setting) or *universal composability* (UC) [Can01], depending on the type of achievable simulation, namely *with-rewinding* or *one-pass*, respectively.

Motivation for this functionality can be found directly in the real-world usefulness of "coin-flipping," enabling parties to jointly make random decisions (e.g., "who gets the car" [Blu83]). A more-technical motivation for simulatability is the security enhancement of larger cryptographic protocols. An important application is the joint decision of a large *common reference string* needed as setup condition of one or several follow-up protocols [CR03]. It is also useful for protocols whose probabilistic output needs to directly depend on random bit-strings, such as in S2PC-with-commitments (e.g., [Bra13]), where both parties may want to jointly generate many random commitments.

1.1 Coin-flipping and primitives

A protocol for two-party coin-flipping ("by telephone") was early proposed by Blum [Blu83]. It uses the fundamental notion of commitment scheme, allowing one party ($\mathrm{P_A}$) to *commit* her own contribution before knowing anything about the contribution of the other party ($\mathrm{P_B}$), but *hiding* it until the contribution of $\mathrm{P_B}$ is revealed, and *binding* $\mathrm{P_A}$ to only being able to *open* the committed value. The solution, emulating a coin-flipping into a well, sets the basis for what is hereinafter denoted as the *traditional template*:

- **Step 1.** (*Commit* phase) $\mathrm{P_A}$ commits to a contribution, hiding it from $\mathrm{P_B}$.
- **Step 2.** $\mathrm{P_B}$ selects and sends his random contribution to $\mathrm{P_A}$.
- **Step 3.** (*Open* phase) $\mathrm{P_A}$ opens her contribution to $\mathrm{P_B}$ in a convincing way.
- **Step 4.** Each party outputs a combination of both random contributions.

The simulatability of a coin-flipping protocol within this template may depend on the number of coins flipped in parallel, i.e., the length of the contributions, and the type of commitment scheme. When flipping a single coin, any hiding and binding commitment scheme is enough if rewinding is allowed in the simulation [Gol04, §7.4.3.1]. Conversely, when doing parallel flipping of coins in number at least linear in the security parameter, or when considering a setting without rewinding, simulatability is facilitated by commitment (Com) schemes with special *extractable* (Ext) and *equivocable* (Equiv) properties. In an Ext-Com scheme [SCP00], a *simulator* is able to *extract* a contribution that has been committed by another party, in apparent conflict with the *hiding* property. In an Equiv-Com scheme [Bea96], a *simulator* is able to *equivocate* the opening to any contribution, namely to a value different from what had been committed, in apparent conflict with the *binding* property. The conflict is only apparent, as in

comparison with a real party the simulator has extra power, such as capability to rewind the other party in the simulated execution, and/or knowledge of secret information (a trapdoor) obtained from some specially selected setup.

Traditionally, achieving simultaneous Ext and Equiv properties is costly as a function of the target length. For example: in the plain model and when allowing rewinding, by requiring zero-knowledge (ZK) proofs (or ZK proofs of knowledge) about elements of size or in number linear with the target length [Lin03], or cut-and-choose techniques with high communication cost [PW09]; or, in a model with setup assumptions but not allowing rewinding, by requiring Com-schemes based on computationally expensive operations (e.g., exponentiations) in number or size dependent on the target length [CF01, BCPV13].

This paper explores efficiency improvements in two ways: (i) augmenting the traditional template into a new structure that requires less sophisticated commitments (i.e., not necessarily Ext&Equiv); (ii) devising a more efficient Ext&Equiv-Com scheme that can be directly used within the traditional template. Both cases benefit from a *pseudo-random generator* (PRG) (naturally associated with the generation of bit-strings indistinguishable from random) and a *collision-resistant hash function* (CR-Hash) (naturally associated with compressing commitments). As the target length increases, the asymptotic communication rate: converges to 1 for each contribution of a party in the stand-alone coin-flipping; converges to a rate close to 1 (i.e., closer than any desired distance), for each phase (commit and open) of the UC-Com scheme. The computational complexity for each party approximates that of applying a PRG and a CR-Hash to produce an output and hash an input, respectively, with length expansion rate asymptotically as close as desired to 1. This is useful given the high efficiency of standardized PRG [Nat14b] (e.g., based on block or stream ciphers) and CR-Hash [Nat14a] constructions. In the UC-Com scheme each party also uses an erasure code to encode a string of length approximately equal to the target length.

The initial (incomplete) intuition comes from the observation that: the Ext of a large string can be reduced to the Ext of one short seed, whose PRG-expansion is used to mask (with a one-time-pad) the large string; the Equiv of a large string can be reduced to the Equiv of a short hash of whatever large string (e.g., the mask) the simulator wants to equivocate. However, a simple triplet composed of a masking of a string, an Ext-but-not-Equiv-Com of the *seed* of the mask, and an Equiv-but-not-Ext-Com of a *hash* of the mask does not result in an Ext&Equiv-Com of the string. For example, opening the Ext-Com would disallow equivocability. This paper devises two ways in which to very-efficiently and securely combine the two separate properties, associated with a few commitments of short seeds and hashes (in number independent of the target length), into a unified property extended to a much larger string.

Contributions. In summary, two novel constant-round protocols are devised for two-party parallel coin-flipping (the second stemming from a new UC-Com scheme). They are proven secure in a *static*, *active* and *computational* model; i.e., at most one party is corrupted at the onset of the protocol execution, the corrupted party may deviate from the protocol specification, and both parties

are limited to probabilistic polynomial time computations. For simplicity and generality, the protocols and proofs are defined in a hybrid model with access to ideal commitment functionalities $\mathcal{F}_{\mathrm{X}}$ and $\mathcal{F}_{\mathrm{Q}}$, from each of which the simulator only needs to use either the Ext or the Equiv property, respectively, but not complementary property (Equiv or Ext, respectively).

1.2 Intuition and overview of Protocol #1

The first protocol (§4) is simulatable-with-rewinding. It augments the traditional template with a simple preamble, in order to avoid a simulatability difficulty (related with unknown adversarial probabilities of abort) found in the protocol of Blum [Blu83], due to the use of an Equiv-but-not-Ext-Com scheme in the traditional template. The new solution also avoids a full-fledged Ext&Equiv-Com scheme, whose (older) constructions have a larger associated complexity: explicit ZK proof/argument sub-protocols about a committed long-contribution, as required in Lindell's protocol [Lin03]; a high communication cost, as incurred in Pass and Wee protocol [PW09].

$\mathrm{P_A}$ is still the first party to learn the final bit-string. However, the new protocol starts with $\mathrm{P_B}$ producing an Equiv-Com of his contribution and only then proceeds to the traditional template. This allows the simulator in the role of $\mathrm{P_A}$ in the simulated execution to *non-locally* extract the contribution of a malicious $\mathrm{P_B}$ (i.e., upon rewinding beyond the respective *commit* phase), because said value cannot change across rewinding attempts, namely because $\mathrm{P_B}$ commits to it before the contribution of $\mathrm{P_A}$ is committed, and because the decision to open it (vs. aborting) is done while the contribution of $\mathrm{P_A}$ is still semantically hidden. The significant benefit is that now the commitment by $\mathrm{P_A}$ (part of the traditional template) no longer needs to be Equiv, but rather only Ext. Correspondingly, using the Ext property, the simulator in the role of $\mathrm{P_B}$ in the simulated execution can extract the contribution of a malicious $\mathrm{P_A}$, without $\mathrm{P_A}$ opening it.

To the knowledge of the author: this construction has not been analyzed before (which is surprising given its simplicity), and in the mentioned simulatability setting it allows, asymptotically, the most efficient instantiation to date of two-side-simulatable coin-flipping in the plain model (assuming a PRG and CR-Hash instantiation with computational cost linear in the target length). The simulatability motivation to depart from the traditional template is subtle and the analysis is challenging for the case of corrupted $\mathrm{P_B}$ (the simulator is allowed expected-polynomial time). Asymptotically, the protocol requires communication of only *two bits per flipped coin*. Computationally, each party has to commit and open a short value, and compute a PRG and a CR-Hash of a string with the target length. Assuming intractability of the Decision Diffie-Hellman (DDH) problem, an instantiation is possible with only five exponentiations per party in a setup phase (allowing the simulator to extract a trapdoor), and four (or six) exponentiations in the online phase. Exponentiations can be avoided altogether, by using PRG-based commitments of short strings or even just bit-commitments (e.g., as in [PW09] or others analyzed in the full version of this paper). In the later example, the simulator exercises Ext and Equiv over the Ext-Com and the

Equiv-Com, respectively, using rewinding, and the construction requires more communication rounds and larger concrete communication complexity of the short commitments but is still amortizable.

1.3 Intuition and overview of Protocol #2

The second protocol (§5) is a new UC-Com scheme (thus Ext&Equiv) for large bit-strings, with asymptotic communication rate as close to 1 as desired, and computational complexity linear in the string size. It is based on a *cut-and-choose* method, where the size of each instance in the cut-and-choose is (approximately) inversely proportional to the number of instances. Each instance is a triplet containing: the Ext-Com of a seed; a masking of an "authenticated" *fragment* (produced by an erasure code) of the string being committed; and an Equiv-Com of the hash of the mask. This allows the simulator to anticipate (before the actual open phase) whether each extracted fragment is correct or not, and reconstruct the original message using only correct fragments. The fragments are also equivocable because the respective pseudo-random masks are equivocable.

The ideal commitment functionalities used for separate Ext and Equiv simulatability properties can also be instantiated with a full-fledged Ext&Equiv-Com functionality. Assuming the existence of a PRG and a CR-Hash, this represents a UC-Com length extension, where a few (*commit* and *open*) calls to an Ext&Equiv-Com scheme for short bit-strings enable an Ext&Equiv-Com (*commit* and *open*) of a polynomially larger size. At the cost of more interactivity, the Equiv-Coms can be based on Ext-Coms.

Similar amortized asymptotic communication complexity is also achieved by very recent UC-Com scheme proposals [GIKW14, DDGN14, CDD$^+$15]. They explicitly use oblivious transfer (OT), i.e., as an ideal functionality in a hybrid model. In contrast, the protocol in this paper avoids explicit use of OT, and instead uses base Ext-Com and Equiv-Com schemes (besides a PRG and a CR-Hash). Also, [GIKW14, DDGN14] rely on *secret sharing schemes* with error-correction or verifiability requirements ([CDD$^+$15] works with any linear code), whereas this paper uses a simpler erasure code, facilitated by the *authenticator* mechanism, with corresponding benefits in terms of encoding parameters. A comparison of tradeoffs allowed by each design is left for future work.

1.4 Roadmap

The paper proceeds as follows: Section 2 reviews related work; Section 3 mentions background notions about the security model and ideal functionalities; Section 4 describes the new protocol for coin-flipping simulatable-with-rewinding; Section 5 specifies the new UC commitment scheme.

2 Related work

2.1 Basic primitives

One-way permutations or functions are enough in theory to achieve many useful cryptographic primitives, such as PRGs [HILL99, VZ12], one-way hash functions [NY89, Rom90], some types of commitment schemes [Nao91, DCO99] and ZK proofs of knowledge (ZKPoK) [FS90]. CR-Hash functions can also be built from other primitives [Sim98], such as claw-free sets of permutations [Dam88] or pseudo-permutations [Rus95]. Based on such primitives, coin-flipping can be achieved in different ways, e.g., based solely on one-way functions [Lin03, PW09] (with rewinding). In different simulatability settings, coin-flipping can be more directly based on higher level primitives, such as bit or multi-bit Ext&Equiv-Com schemes (e.g., [CF01, DN02, Cre03]) and even from coin-flipping protocols with weaker properties [HMQU06, LN11].

In the computational model (the one considered in this paper), there are known theoretical feasibility results about coin-flipping, covering the stand-alone and the UC security settings. For example, in the UC setting it is possible to achieve coin-flipping extension, i.e., coin-flip a large bit-string when having as basis a single invocation of an ideal functionality realizing coin-flipping of a shorter length [HMQU06]. This paper shares the concern of achieving properties in large strings based on functionalities associated with short strings, but focuses on a base of a few short commitments (not needing to be simultaneously Ext and Equiv) and has a motivation of improving efficiency. The paper does not delve into analyzing implications between different primitives (e.g., see [DNO10] for relations between OT and commitments, under several setup assumptions).

Only in very recent research works (including this one) have UC commitment schemes been devised with an amortized communication cost, with asymptotic communication rate close to 1. In contrast, similar attention has not been given to coin-flipping in the stand-alone setting, where the most efficient protocols known to be two-side simulatable would not be highly efficient for large strings. While the new results for UC-com schemes are directly applicable to stand-alone secure coin-flipping, with a corresponding asymptotic efficiency benefit (3 bits per flipped coin), an yet different and more efficient approach (2 bits per flipped coin) is herein devised for the stand-alone setting, without requiring an explicit Ext&Equiv-Com scheme.

In spite of very-efficient realizations of OT-extension [ALSZ15] and free-XOR techniques [KS08] for garbled circuits, a coin-flipping based on a direct (generic) approach of S2PC of bit-wise-XOR would still induce, in communication and computation, a multiplicative cost proportional to the security parameter, by requiring one *minicrypt* block operation (e.g., block-cipher evaluation) per flipped coin. In contrast, in the approach in this paper each block of bits (e.g., equal to the security parameter) requires a unitary number of minicrypt block operations (e.g, close to 1 block-cipher for the PRG and 1 CR-Hash).

The idea of combining commitments with a CR-Hash (*hash then commit*) and commitments with a PRG for efficiency reasons is not new. The former

resembles the *hash then sign* paradigm, and it also has applications to non-malleable commitments [DCKOS01]. The later resembles hybrid encryption, where a symmetric key (the analogous to the PRG seed) would be encrypted with a public key system (the analogous to the commitment) and then the message would be encrypted with a symmetric scheme (the analogous to the one-time-pad masking using the PRG expansion). This paper explores ways of combining both techniques, aimed at achieving simulatability in coin-flipping and UC commitment schemes.

2.2 Parallel coin-flipping simulatable-with-rewinding

A parallel coin-flipping using the traditional template is simulatable if the base commitment scheme is Ext&Equiv. Lindell achieved this (in two variant protocols [Lin03, §5.3 & §6]) by augmenting the *commit* and *open* phases with ZK sub-protocols that enable the respective Ext and Equiv properties: an Ext-commit phase (step 1) is a regular commitment followed by a ZK argument of knowledge of the committed value, from which the simulator in the role of receiver can extract the value; an Equiv-open phase (step 3) consists on sending the intended (equivocated) contribution of P_A (which on its own cannot be verified against the respective commitment) and giving a *fake* ZK argument that it was the valid committed value. The solution provides a feasibility result for constant-round simulatable parallel coin-flipping. However, for a general commitment scheme applied to a long bit-string, either a ZK proof/argument of knowledge for *extraction* or a ZK proof/argument for *equivocation* is typically expensive, if not both. Note: it is worth noticing that the protocols by Lindell also address an augmented version of coin-flipping into a well, where P_A receives a random bit-string and P_B receives the result of applying a known function to such bit-string – the case of the identity function is the one considered in this paper.

In a different approach, Pass and Wee [PW09] use a cut-and-choose approach to achieve Ext and Equiv properties directly from regular commitment schemes (and thus from one-way functions). They show simulatability of coin-flipping in the traditional template, based on an Ext&Equiv-Com scheme constructed from regular commitments in number proportional to the target length multiplied by the statistical parameter. In contrast with the two above referred constructions, protocol #1 in this paper integrates the Ext and Equiv properties in different commitments, in order to improve efficiency, amortizing the cost of base commitment schemes to that of a PRG and CR-Hash.

Goldreich and Kahan [GK96] also joined two types of commitment schemes in a protocol to achieve (what this paper calls) *non-local* extraction. Their application is *constant round ZK interactive proof systems*, rather than coin-flipping. They also augment the protocol by introducing an unconditional hiding commitment as preamble, but their goal is achieving statistical soundness in an interactive proof system, rather than providing *local* equivocability or achieving a communication complexity amortization as in protocol #1 in this paper. They define a simulator that estimates the probability of non-abort of the malicious party, in order to dynamically determine an upper bound on the number of

rewindings that should be tried before giving up on obtaining a (second) non-abort by the malicious party. The estimation works because the commitments are used in a way that prevents the probability of abort from depending (i.e., up to a negligible variation) on the value committed by the honest party. This simulation strategy was also used by [Lin03] and [PW09] for the simulation of ZK sub-protocols, and can also be used to simulate the coin-flipping protocol #1 in this paper, with an expected polynomial number of rewindings. However, the technique is not applicable in the coin-flipping protocol of Blum [Blu83], because there the decision of abort by the party that produced the Equiv-Com (i.e., the decision to open her contribution vs. to abort without opening) is made once already knowing the contribution of the other party.

A similar subtle problem of simulatability derived from unknown probabilities of abort has also been addressed by Rosen [Ros04]. With the goal of simplifying the analysis of simulatability of ZK proofs, Rosen introduces a preamble stage involving an unconditionally-hiding Ext-Com, allowing a prover in a ZK proof system to initially (and locally) extract the challenge of the verifier. Such augmentation is different from the one in this paper. First, the preamble commitment in their ZK proof (Ext-)commits a value (the *challenge*) that does not influence the actual honest output bit (accept vs. reject) of the ZK. Conversely, herein the value (Equiv-)committed (by $\mathrm{P_B}$) in the preamble is a *contribution* with direct impact in the bit-string outputted by the coin-flipping execution. Second, in their ZK application the use of the preamble with the Ext-Com by one party (the verifier) relieves the simulator in the role of the other party (the prover) from having to do non-local equivocation in the subsequent part of the execution. Conversely, herein the preamble (with an Equiv-Com by $\mathrm{P_B}$) does not relieve the simulator in the role of the other party ($\mathrm{P_A}$) from having to non-locally equivocate the contribution that it commits to in the remainder of the execution. Third, their proposed Ext-Com scheme is unconditionally hiding, whereas the PRG-based Ext-Com construction used in protocol #1 to commit the contribution of $\mathrm{P_A}$ is (motivated by efficiency) inherently non-unconditionally hiding.

2.3 UC commitment schemes

When rewinding simulations are not possible, the simulatability of flipping even a single coin using the traditional template requires simultaneous Ext and Equiv properties of the underlying commitment scheme [CR03]. Canetti and Fischlin [CF01] developed non-interactive UC commitments, requiring a unitary number of asymmetric operations per committed bit. The construction assumes a CRS setup and is based on the equivocable bit-commitment from Crescenzo, Ishai and Ostrovsky [DCIO98]. Canetti, Lindell, Ostrovsky, and Sahai [CLOS02] proposed other non-interactive schemes from general primitives, with adaptive security without erasures. Damgård and Nielsen [DN02] then improved with a construction denoted *mixed commitment scheme*, that is able to commit a linear number of bits using only a unitary number of asymmetric operations, and using a linear number of communicated bits. For some keys they are unconditionally-hiding and *equivocable*, whereas for other keys they are unconditionally-binding

and *extractable*. Crescenzo [Cre03] devised two non-interactive Ext&Equiv-Com schemes for individual bits, in the public random string model (suitable to UC). One construction is based on Equiv-Com schemes and NIZKs, the other is based on one Ext-Com and one Equiv-Com schemes. Damgård and Lunemann [DL09] consider UC in a quantum setting and solve the problem of flipping a single bit, based on UC-Coms from [CF01]. Lunemann and Nielsen [LN11] consider also the quantum setting and achieve secure flipping of a bit-string based on mixed commitments from [DN02]. They consider how to amplify security from weaker security notions of coin-flipping (uncontrollable, random) up to full simulatable (enforceable). The use of Ext-Com and Equiv-Com schemes, together with a cut-and-choose and encoding scheme has been previously considered by Damgård and Orlandi [DO10] to enable efficient constructions. They combine these techniques to enhance security from the passive to the active model for secure computation of arithmetic circuits, in a model where a trusted dealer is able to generate correlated triplets. While they achieve efficient constructions for multiparty computation (also including more than two parties), the efficiency is not amortized to communication rate 1.

In another line of work, several more efficient commitment schemes have been proposed for short strings, based on specific assumptions such as DDH intractability, e.g., [Lin11, FLM11, BCPV13] achieving a *low* constant number (but grater than one) of group elements of communication and of exponentiations to commit to a group element. Still, the trivial extension of these protocols for larger strings would imply a linear increase in said number of asymmetric operations (modular exponentiations), without amortization. Some of the above scheme achieve adaptive security (with erasures), whereas this paper considers only static security.

In regard to the more recent independent works that also achieve asymptotic communicate rate 1: [GIKW14] additionally considers selective openings; [DDGN14, CDD$^+$15] additionally consider homomorphic properties and verification of linear relations between committed values; [CDD$^+$15] achieves, comparably to this paper, linear computational complexity. All these protocols are based on a hybrid model with an ideal OT functionality. In contrast to OT, the cut-and-choose mechanism in protocol #2 in this paper does not hide from the sender the partition of (check) instances. In practice, the *authenticator* mechanism allows the simulator to recover the fragmented message using an erasure code, thus allowing a cut-and-choose with less instances than what an error correction code would imply (e.g., see Table 1). A more recent concurrent result [FJNT16] improves the complexity of the OT-based protocols (also for additively homomorphic commitments), using an additional *consistency check* mechanism to also allow a simpler erasure code.

A concrete comparison between different methods – qualitative (e.g., implications between primitives) and quantitative (actual instantiations and implementations) – is left for future work. For example, [GIKW14]) reports 640 exponentiations for a concrete instantiation of the OT setup phase. A concrete instantiation of Ext or Equiv commitments has not yet been explored herein,

though their complexity is naturally upper bounded by that of instantiations of full-fledged UC-Coms for short strings, e.g., requiring less than a dozen group elements per base commitment [BCPV13]. The overall number of commitments of short strings will depend on the erasure code parameters, defined to meet the goals of statistical security and communication rate.

In summary, this paper is focused on the design of protocols that explore the duality between Ext and Equiv commitments, without considering OT as a primitive. About OT only two notes are mentioned here from other work: it is known that UC-OT implies UC-Coms in myriad setup models [DNO10, Fig. 1], e.g., in the *uniform*, the *chosen* and the *any common reference string* models (U/C/A-CRS), and in the *chosen* and *any key registration authority* models (C/A-KRA), whereas the reverse implication is proven only in a narrower set of models (e.g., U/A-CRS, A-KRA) [DNO10, Table 1]; while [GIKW14] shows that "the existence of a semi-honest OT protocol is necessary (and sufficient) for UC-Com length extension," the UC scheme in this paper does not make explicit use of OT and can also be seen as a UC-Com length extension (if replacing the Ext-Com and Equiv-Com schemes with an Ext&Equiv-Com scheme) – these two results do not superpose, since [GIKW14] only allows a single call to the ideal Com-scheme, whereas the extension herein requires several calls.

3 Background notions

It is here assumed that the reader is familiar with the ideal/real simulation paradigm, as developed in the work of Canetti on composability of protocols [Can00, Can01]. Familiarity is also assumed with the standard ideal functionalities of commitment schemes ($\mathcal{F}_{\text{MCOM}}$) and coin-flipping ($\mathcal{F}_{\text{MCF}}$), namely in the UC framework. For example, instances can be found in [CF01, Fig. 3] (multiple bit-commitments), [DN02, §4.2] (multiple message-commitments, there also considering homomorphic relations), [DL09, Fig. 2] (coin-flipping), [Lin03] (general S2PC). A background review of these standard notions and specification of ideal functionalities is given in full version of this paper. For convenience, this section simply states notions underlying extractable [SCP00] and equivocable commitments [Bea96].

Definition 1 (**extractability**). *An* extractable *commitment (Ext-Com) scheme is one whose* commit *phase in a simulated execution allows $\mathcal{S}$ in the role of* receiver*, and indistinguishable from an honest receiver in the view of a possibly malicious sender, to* extract *(i.e., learn) the committed value, with probability equal to or larger than a value negligibly-close to the maximum probability with which the (possibly malicious)* sender *is able to successfully* open *said value.*

Definition 2 (**equivocability**). *An* equivocable *commitment (Equiv-Com) scheme is one whose* open *phase in a simulated execution allows $\mathcal{S}$ in the role of* sender*, and indistinguishable from an honest sender in the view of a possibly malicious receiver, to* equivocate *the opening to any intended value, in the domain of committable values and possibly externally decided only after the* commit *phase.*

Definition 3 (**locality of Ext and Equiv**). *Within a protocol using commitments, namely with both* commit *and* open *phases,* extraction *is characterized as* local *if* $\mathcal{S}$ *can extract the committed value within the respective* commit *phase, i.e., without going beyond that phase in the protocol and without rewinding to a step before that phase.* Local equivocation *is defined analogously in relation to the* open *phase. The properties are characterized as* non-local *if they can be achieved but not* locally*, i.e., involving rewinding beyond the respective phase.*

The protocols hereinafter are described and proven secure in a hybrid model with access to ideal commitment functionalities $\mathcal{F}_{\mathrm{X}}$ and $\mathcal{F}_{\mathrm{Q}}$, with which $\mathcal{S}$ respectively only needs to take advantage of Ext and Equiv, but not both.

4 A new coin-flipping protocol simulatable-with-rewinding

This section devises a new (constant round) parallel coin-flipping protocol, simulatable-with-rewinding. The intuition has already been given (§1.2); a textual description follows, along with a succinct specification in Fig. 1.

4.1 Description of protocol #1

As implicit parameters, the protocol depends on a computational security parameter (1) and a respectively secure PRG and CR-Hash function (2). The execution starts when both parties are activated to initiate a coin-flipping of a certain *target length*, with an appropriate execution context (3), which in particular encodes the roles of the two parties – $\mathrm{P_A}$ will be the first to learn the final outcome ((4)-(5)) – and the target length. After a possibly implicit setup phase (e.g., in the plain model, to allow the simulator to obtain a trapdoor), $\mathrm{P_B}$ selects his contribution (6) with the target length, calculates its hash (7), and uses $\mathcal{F}_{\mathrm{Q}}$ to commit to the hash ((8)-(9)). Then, $\mathrm{P_A}$ selects a seed (10) and *commits* to it using $\mathcal{F}_{\mathrm{X}}$ ((11)-(12)). $\mathrm{P_A}$ also selects a random bit-string (denoted *contribution masking*) with the target length (13) and sends it to $\mathrm{P_B}$ (14). Then, $\mathrm{P_B}$ uses $\mathcal{F}_{\mathrm{Q}}$ to open the committed hash to $\mathrm{P_A}$ ((15)-(16)) and sends his contribution to $\mathrm{P_A}$ (17). $\mathrm{P_A}$ checks that the hash of the contribution of $\mathrm{P_B}$ is equal to the *opened* hash (18). If not, it Aborts; otherwise it proceeds. Then, $\mathrm{P_A}$ uses $\mathcal{F}_{\mathrm{X}}$ to *open* to $\mathrm{P_B}$ the committed seed ((19)-(20)). Finally, each party proceeds concurrently with local computations: expanding the seed of $\mathrm{P_A}$ into a bit-string of the target length (21) (i.e., the *mask*); computing the *bit-wise exclusive-OR* (XOR) combination of the *mask* and the *contribution masking*, thus determining the contribution of $\mathrm{P_A}$ (22); and locally computing the final outcome as the XOR of the two contributions (23), and deciding that as the final output (24)-(25).

4.2 Concrete instantiations in the plain model

In the *plain* model, $\mathcal{F}_{\mathrm{X}}$ and $\mathcal{F}_{\mathrm{Q}}$ can be respectively replaced by actual Ext-Com and Equiv-Com schemes. They can be agreed upon in a *setup* phase, with Ext-Com being non-malleable with respect to opening of Equiv-Com. An intuition is given here for possible concrete instantiations (more details in the full version).

Implicit parameters.
Security parameters: 1^{κ} (1)
Primitives: $(\text{PRG}, \kappa_{\text{PRG}})$, CR-Hash (2)

0. Initial input.
$ctx \equiv (sid, cfid, \text{P}_\text{A}, \text{P}_\text{B})$ (3)
$input_A \rightarrow \text{P}_\text{A} : (\texttt{cf-start-1}, ctx, \ell)$ (4)
$input_B \rightarrow \text{P}_\text{B} : (\texttt{cf-start-2}, ctx, \ell)$ (5)

1. Commit contribution of P_B.
$\text{P}_\text{B} : \chi_B \leftarrow^{\$} \{0,1\}^{\ell}$ (contribution of P_B) (6)
$\text{P}_\text{B} : h_B = \text{CR-Hash}(\chi_B)$ (short hash) (7)
$\text{P}_\text{B} \rightarrow \mathcal{F}_\text{Q} : (\texttt{commit}, (ctx, \text{Q}), h_B)$ (8)
$\mathcal{F}_\text{Q} \rightarrow \text{P}_\text{A} : (\texttt{receipt}, (ctx, \text{Q}), |h_B|)$ (9)

2. Commit contribution of P_A (extractable).
$\text{P}_\text{A} : s_A \leftarrow^{\$} \{0,1\}^{\kappa_{\text{PRG}}}$ (short seed) (10)
$\text{P}_\text{A} \rightarrow \mathcal{F}_\text{X} : (\texttt{commit}, (ctx, \text{X}), s_A)$ (11)
$\mathcal{F}_\text{X} \rightarrow \text{P}_\text{B} : (\texttt{receipt}, (ctx, \text{X}), |s_A|)$ (12)
$\text{P}_\text{A} : t_A \leftarrow^{\$} \{0,1\}^{\ell}$ (contribution masking) (13)
$\text{P}_\text{A} \rightarrow \text{P}_\text{B} : (\texttt{cf-masking-1}, ctx, t_A)$ (14)

3. Open contribution of P_B (equivocable).
$\text{P}_\text{B} \rightarrow \mathcal{F}_\text{Q} : (\texttt{open-ask}, (ctx, \text{Q}))$ (15)
$\mathcal{F}_\text{Q} \rightarrow \text{P}_\text{A} : (\texttt{open-send}, (ctx, \text{Q}), h_B)$ (16)
$\text{P}_\text{B} \rightarrow \text{P}_\text{A} : (\texttt{cf-contrib-2}, ctx, \chi_B)$ (17)
$\text{P}_\text{A} :$ If $\text{CR-Hash}(\chi_B) \neq h_B$ then ABORT (18)

4. Open contribution of P_A.
$\text{P}_\text{A} \rightarrow \mathcal{F}_\text{X} : (\texttt{open-ask}, (ctx, \text{X}))$ (19)
$\mathcal{F}_\text{X} \rightarrow \text{P}_\text{B} : (\texttt{open-send}, (ctx, \text{X}), s_A)$ (20)
$\text{P}_\text{A}, \text{P}_\text{B} : s'_A = \text{PRG}[s_A](\ell)$ (seed expansion $\equiv$ *mask*) (21)
$\text{P}_\text{A}, \text{P}_\text{B} : \chi_A = t_A \oplus s'_A$ (contribution of P_A) (22)

5. Final output (locally combine contributions).
$\text{P}_\text{A}, \text{P}_\text{B} : \chi = \chi_A \oplus \chi_B$ (23)
$\text{P}_\text{A} \rightarrow output_A : (\texttt{cf-output-1}, ctx, \chi)$ (24)
$\text{P}_\text{B} \rightarrow output_B : (\texttt{cf-output-2}, ctx, \chi)$ (25)

Fig. 1. Protocol #1 (Parallel coin-flipping (simulatable-with-rewinding). Legend: κ (cryptographic security parameter, e.g., $128 \equiv 1^{128}$); ℓ (target length, i.e., number of bits to coin-flip in parallel, e.g., 10^6, satisfying $\ell \in O(poly(\kappa))$); χ_p (contribution of P_p, for $p \in \{A, B\}$); $\text{PRG}[s](\ell)$ (expansion of seed s, using the PRG, into a bit-string of length ℓ); κ_{PRG} (length of PRG input-seed, consistent with κ); X, Q (indices denoting *extractable* and *equivocable*); (ctx, x) (abbreviation for $(sid, (cfid, x), \text{P}_\text{A}, \text{P}_\text{B})$, where $x \in \{\text{X}, \text{Q}\}$ – by including X and Q in the *context* information exchanged with the respective ideal Com functionalities ($\mathcal{F}_\text{X}$, $\mathcal{F}_\text{Q}$), it is syntactically easier to replace them both by a single full-fledged ideal X&Q (multi-)Com functionality $\mathcal{F}_{\text{X\&Q}}$.)

Based on DDH intractability assumption. For the Ext-Com scheme: P_A commits to the seed by sending a simple El-Gamal encryption [ElG85] of the seed; the simulator can extract if it knows the encryption key (a discrete-log); P_A opens the seed by revealing the seed and the encryption randomness, thus letting P_B verify its correctness. For the Equiv-Com scheme: P_A commits to the hash by sending a simple Pedersen commitment [Ped92]; P_B opens the hash by revealing the hash and the commitment randomness. The simulator can equivocate the opening if it knows the trapdoor (a discrete-log). Interestingly, both Com-schemes can have the same trapdoor, because the seed extraction and the hash equivocation are needed by the same simulator (in the role of P_B, when interacting with P^*_A). The parameters can be agreed in a setup phase, with P^*_A proposing them (two generators in a multiplicative group where the DDH assumption holds) and giving a ZKPoK of their relation (the discrete-log between two generators). Basically, this can be a ZK adaptation of Schnorr's protocol [Sch91], e.g., as described in [LPS08, Fig. 3]. Overall, this requires only 9 exponentiations from each party (or 11, using more practical parameters), 5 of which are in the setup phase (amortizable across several coin-flippings).

A concrete application example. The S2PC-with-BitComs protocol presented in [Bra13], simulatable-with-rewinding, requires a simulatable coin-flipping to sample a random group element for each bit of input and output of the regular

S2PC. (Improvements of the protocol can reduce the needed number and size of said group-elements.) There, the benchmark evaluation of S2PC-with-BitComs of AES-128 requires a simulatable flipping of about 1.18 million bits. As suggested therein, using a DDH assumption in groups over elliptic curves, an instantiation of the coin-flipping with the protocol of [Lin03] would require (for practical parameters) 7 exponentiations per party per block of 256 bits, and communication of about 12 blocks per block, i.e., overall about 32 thousand exponentiations and 12 megabits. In contrast, applying the new coin-flipping devised herein would overall require (with the instantiation suggested in the previous paragraph) less than a dozen exponentiations per party and slightly more than 2 megabits of communication, thus reducing the complexity of the coin-flipping sub-protocol by more than 3,000 fold in number of exponentiations and 6 fold in communication.

Based on PRG-based commitments. It is possible to avoid exponentiations by building Ext-Com and Equiv-Com schemes based on more basic primitives, such as regular commitments (i.e., hiding and biding but possibly not Ext and not Equiv). For example, Pass and Wee [PW09] analyze cut-and-choose based constructions (the full version of this paper explores improvements, e.g., using a random-seed-checking type of technique [GMS08]). Comparatively, those constructions require more concrete communication than the DDH based one, but still amortizable because it only applies to two short elements (one seed and one hash), and more online interactivity.

4.3 Security analysis

Proving security (i.e., simulatability) amounts to show a simulator ($\mathcal{S}$) that, with an expected number of rewindings at most polynomial in the security parameter, induces in the ideal world a *global* output whose distribution is indistinguishable from the one in the real world. In the role of each party in a simulation, $\mathcal{S}$ must be able (with overwhelming probability) to learn the contribution of the other possibly-malicious (black-box) party and still be in a position to *open* the *needed complementary contribution*, as if it was honestly random, and at the same time simulate the correct probability of early-abort.

Theorem 1 (security of protocol #1). *Assuming a cryptographically secure PRG and CR-Hash, protocol #1 securely-emulates (with computational indistinguishability) the ideal functionality $\mathcal{F}_{MCF}$ of* long *bit-string coin-flipping between two-parties, in a stand-alone setting and in the ($\mathcal{F}_X$,$\mathcal{F}_Q$)-hybrid model, in the presence of static and computationally active* rewindable *adversaries. For each (polynomially arbitrarily-long) bit-string coin-flipping execution, each phase (*commit *and* open*) of $\mathcal{F}_X$ and $\mathcal{F}_Q$ is invoked only once for a short string; simulation is possible: without rewinding in the case of a malicious P_A^*; with an expected polynomial number of rewindings in the case of a malicious P_B^*.*

One-pass simulation (i.e., without rewinding), for malicious P_A^*. In the simulated execution, $\mathcal{S}$ (in the role of $\mathrm{P_B}$) commits to a random hash value

(8). Then, $\mathcal{S}$ impersonates $\mathcal{F}_{\mathrm{X}}$ to extract from $\mathrm{P}_{\mathrm{A}}^*$ the seed that $\mathrm{P}_{\mathrm{A}}^*$ intended to commit (11). $\mathcal{S}$ computes the PRG expansion of the seed (as in (21)). Then, upon receiving the contribution masking of $\mathrm{P}_{\mathrm{A}}^*$ (14), $\mathcal{S}$ combines it with the PRG-expansion of the extracted seed (as in (22)), in order to learn the contribution of $\mathrm{P}_{\mathrm{A}}^*$. Then, in the ideal world, $\mathcal{S}$ in the role of the ideal $\widehat{\mathrm{P}}_{\mathrm{A}}^*$ receives from the ideal coin-flipping functionality $\mathcal{F}_{\mathrm{MCF}}$ the random target coin-flipping bit-string. $\mathcal{S}$ then computes the needed complementary contribution of P_{B}, as the XOR between the target outcome and the contribution of $\mathrm{P}_{\mathrm{A}}^*$. $\mathcal{S}$ computes the hash of this complementary contribution (as in (7)) and in the role of $\mathcal{F}_{\mathrm{Q}}$ it equivocates its opening to be such hash value (16). Finally, $\mathcal{S}$ also sends the complementary contribution to $\mathrm{P}_{\mathrm{A}}^*$ (17). Since the ideal $\mathcal{F}_{\mathrm{X}}$ is impersonated by $\mathcal{S}$ (respectively, in the plain model, since Equiv-Com is non-malleable with respect to opening of Ext-Com), it follows that $\mathrm{P}_{\mathrm{A}}^*$ can only either open the contribution (19) that has been extracted by $\mathcal{S}$, or abort without successfully opening her contribution. In case of abort by $\mathrm{P}_{\mathrm{A}}^*$, $\mathcal{S}$ *emulates an abort*; otherwise, $\mathcal{S}$ lets $\mathcal{F}_{\mathrm{MCF}}$ continue the execution in the ideal world (i.e., send the bit-string to the ideal $\widehat{\mathrm{P}}_{\mathrm{B}}$) and $\mathcal{S}$ outputs in the ideal world what $\mathrm{P}_{\mathrm{A}}^*$ outputs in the simulation. (In the plain model, equivocability and/or extractability of Ext-Com and/or Equiv-Com may require either local rewinding or rewinding in a setup phase, but that is irrelevant in the hybrid model.)

Simulation with explicit rewinding, for malicious $\mathrm{P}_{\boldsymbol{B}}^*$.

- **First iteration.** In the simulated execution, $\mathcal{S}$ in the role of an honest P_{A} interacts until receiving the contribution of $\mathrm{P}_{\mathrm{B}}^*$ and verifying its hash against the respective opening (18). If $\mathrm{P}_{\mathrm{B}}^*$ aborts until this step (including by an invalid opening), then $\mathcal{S}$ emulates an abort, otherwise it proceeds.
- **Get target outcome.** $\mathcal{S}$ in the role of ideal $\widehat{\mathrm{P}}_{\mathrm{B}}^*$ receives from $\mathcal{F}_{\mathrm{MCF}}$ in the ideal world the target outcome and uses it to compute the needed complementary contribution of P_{A} in the simulated execution, namely the XOR between the target outcome and the contribution of $\mathrm{P}_{\mathrm{B}}^*$.
- **Determine upper-bound of rewindings.** $\mathcal{S}$ determines an upper bound number of rewindings (`#rw-bound`) needed for the next simulation stage. This can be based on the strategy of Goldreich and Kahan [GK96], which in this case means rewinding, possibly a super-polynomial number of times, to repeat committing a random contribution of P_{A} ((11)-(14)). and expecting to obtain an opening of the contribution of P_{B} ((16)-(17)), until indeed obtaining a successful opening (18) an adequate polynomial (e.g., quadratic) number of times, and estimating therefrom an adequate probability of non-abort by P_{A}, and defining `#rw-bound` as the inverse of said estimate. An intuition for the expected polynomial number of rewindings is that a negligible probability of non-abort also implies a negligible probability that the simulation reaches this estimation stage. (Using a more involved argument about the hiding property of the PRG-based Ext-Com of the contribution of P_{A}, the full version of the paper explores the possibility of a different simulation strategy, with a static

super-polynomial upper bound `#rw-bound` instead of one depending on a dynamic estimation of the non-abort probability.)

– **Induce target outcome.** $\mathcal{S}$ rewinds and selects (10) and commits (11) to a new random seed of P_A. Then, $\mathcal{S}$ computes and sends to P_B^* a contribution masking of P_A (14), computed as the XOR combination of the needed complementary contribution and the PRG-expansion of the seed (instead of a random *contribution masking* (13)). Since the Ext-Com+PRG-based commitment of the contribution of P_A is semantically hiding, the probability of abort by P_B^* changes at most by a negligible amount in comparison with the previous stage. If P_B^* subsequently opens his contribution successfully ((16)-(18)), then $\mathcal{S}$ continues the simulation until the end and outputs in the ideal world whatever P_B^* outputs in the real world, even if P_B^* aborts before receiving the opening of the seed of P_A (20). Otherwise, if P_B^* aborts without successfully opening his contribution, $\mathcal{S}$ rewinds and replays again as just described, again and again until either obtaining a successful opening of the contribution of P_B^* (equal to the one already known by $\mathcal{S}$) and in that case leading the simulation to an end, or until reaching the `#rw-bound` bound, and in that case it emulates an abort in the ideal world.

5 A new UC commitment scheme

This section devises a new UC commitment scheme, i.e., one-pass-simulatable and with local Ext and Equiv properties, usable in the traditional template of coin-flipping to *commit* and *open* the contribution of P_A.

5.1 More intuition

Besides the Ext-Com, Equiv-Com, PRG and CR-Hash, the new protocol embeds three main ingredients, in a sequence of optimizations:

– a **cut-and-choose**: P_A builds several *instances* of short commitments and then P_B checks the correctness of some (the *check* instances) to gain *some* confidence that a majority of the others (the *evaluation* instances) are correct;
– **authenticators**: allows the simulator to anticipate whether individual *instances* are *good* or *bad*, thus gaining assurance about correct extraction;
– an **information dispersal algorithm** (IDA): allows *spliting* the target message m into smaller *fragments*, and allows *recovery*) of the original message from a sufficient portion of those fragments (essentially, based on a threshold erasure code); using an IDA enables the size of each *instance* of the cut-and-choose to be reduced proportionally to the number of instances.

5.1.1 Cut-and-choose warmup.

A simple (yet innefficient) UC-com scheme:

– **Commit phase.** P_A produces several seeds, builds an Ext-Com of each, and also an Equiv-Com of a CR-Hash (hereafter denoted *global hash*) of the

sequence of PRG-expansions of all seeds. Then, P_B *cuts* the set of instances of seed-commitments into two random complementary subsets, and *chooses* one for a *check* operation and the other for an *evaluation* operation. For each *evaluation* instance, P_A uses the respective PRG-expansion to XOR-mask the *target message*, and sends the respective *message masking* to P_B.
– **Open phase.** P_A reveals the message m, letting P_B compute all used masks, one for each *evaluation* instance, namely the XOR of the message with each respective masking. P_A also opens all *check* seeds, letting P_B compute the respective PRG-expansions and verify that they are equal to the computed masks. Finally, P_A opens the committed global hash, letting P_B verify that it is equal to the one that can be obtained from all PRG-expansions. Otherwise, if the global hash verification fails, P_A rejects the opening of m.

This has the needed simulatability properties (though high communication complexity: target length ℓ multiplied by number e of *evaluation* instances):

– **Hiding.** In the *commit* phase, the message is hidden from P_B, by a one-time-pad of PRG-expansions (the masks).
– **Binding.** In the *open* phase, P_A is bound to open a single message: by collision resistance of CR-Hash, P_A can only know one mask per evaluation instance that leads to the correct global hash; thus, P_A can only successfully open the message that for all evaluation instances is equal to the XOR of such mask and the respective *masking*.
– **Equivocation.** In the *open* phase, the equivocator-simulator ($\mathcal{S}^Q$) in the role of P_A can open any desired fake message, by revealing the message, opening the correct seeds of check instances and then *equivocating* the needed fake global hash (without revealing the respective seeds of evaluation instances).
– **Extractability.** In the *commit* phase, the extractor-simulator ($\mathcal{S}^X$) in the role of P_B *extracts* the seed of each evaluation instance, then uses its PRG-expansion to unmask the respective masking into a tentative message. If a majority of the tentative messages are equal, then $\mathcal{S}^X$ chooses their value as the correct one. Otherwise $\mathcal{S}^X$ guesses that P_A will not be able to successfully open any message in the later open phase. Conditioned on a future successful verification of the global hash, the probability that the majority of the extracted seeds are correct is, with adequate cut-and-choose parameters [SS11, §A], overwhelming in the total number of instances. For example, slightly more than 40 bits of statistical security, i.e., a probability of wrong extraction less that two to the minus 40, is obtained using 123 instances, 74 of which for *check* and 49 of which for *evaluation* [Bra13, Table 2].

5.1.2 Authenticator aid. Statistical security can be improved by giving $\mathcal{S}^X$ the ability to decide whether isolated evaluation instances are *good* or *bad*. With such capability, $\mathcal{S}^X$ extracts an incorrect message only if all *check* instances are *good* and all *evaluation* instances are *bad*, i.e., only if a malicious P_A^* anticipates the exact cut-and-choose partition. The new rationale about probabilities is similar to that of the forge-and-lose type of technique recently devised for more general S2PC

protocols based on a cut-and-choose of garbled-circuits [Bra13, Lin13, HKE13]. Essentially, the success criterion changes from "at least a majority of correct instances" to "at least one correct instance." For example, 40 bits of statistical security can now be obtained with 41 or 123 instances, by respectively limiting *evaluation* instances to be at most 20 or 8. Since only evaluation instances are relevant in terms of communication, with 123 instances this corresponds to a 6-fold reduction in communication.

The intended verifiability is achieved by augmenting each *evaluation* instance with a short *authenticator* that allows $\mathcal{S}^X$ to verify whether or not each extracted seed is consistent with each respective anticipated tentative message. Specifically, when $\mathcal{S}^X$ extracts a seed and uses its seed-expansion to unmask the respective masking received from $\mathrm{P_A}$, only two things may happen: either (i) $\mathcal{S}^X$ gets a correctly *authenticated* message, which must be the only one that $\mathrm{P_A}$ can later successfully open, i.e., this is a *good* instance; or (ii) $\mathcal{S}^X$ gets an incorrectly *authenticated* message, implying that a successful *opening* by a malicious $\mathrm{P_A^*}$ will reveal a mask different from the seed-expansion, i.e., this is a *bad* instance.

The authenticator is implemented as a function that relates the message and the nonce in a non-trivial way, to ensure that it is infeasible for $\mathrm{P_A^*}$ to produce a masking for which two different unmaskings yield authenticated messages. Also, in order to allow equivocation by $\mathcal{S}^Q$ (when in the role of $\mathrm{P_A}$), the authenticator is masked by an equivocable mask. This means that the authenticator cannot simply be a CR-Hash function (i.e., without an unpredictable input) of the masked fragment, least $\mathrm{P_A^*}$ would in that case (by maliciously using a mask different from the seed-expansion) be able to induce a collision by crafting a special mask different from the seed-expansion. Instead, the authenticator can be achieved by means of a *universal hash family*, such that he probability of collision is independent of the choices of $\mathrm{P_A^*}$. This can be implemented by introducing a random unpredictable value (a *nonce*) that $\mathrm{P_B}$ discloses to $\mathrm{P_A^*}$ only after $\mathrm{P_A^*}$ becomes bound to her choices, e.g., after committing to the seeds and global hash. This nonce acts like an identifier of the hash from the universal hash family.

In concrete, the authenticator can for example be an algebraic field-multiplication between the nonce and a CR-hash of the message. If the image space of the CR-Hash is the set of bit-strings of some fixed length (e.g, 256 bits), the nonce can be uniformly selected from the non-null elements of a Galois field with characteristic 2, modulo an irreducible polynomial of degree equal to the hash length. This ensures that the authenticators of any two known messages (which by assumption would necessarily have different CR-Hash) would have an unpredictably offset. Conversely, a successful forgery by $\mathrm{P_A^*}$ would require guessing this offset, in order to make the real mask have such (bit-wise XOR) offset with the seed-expansion. (Some optimizations are possible, requiring a more involved explanation and/or correlation-robust type of assumptions – details in the full version of the paper.)

5.1.3 IDA support Communication is drastically reduced by using a threshold *information dispersal algorithm* (IDA) [Rab89]. The IDA enables splitting

(i.e., *dispersing*) the original message m into several (e) fragments, such that m can be reconstructed from any subset with at least a threshold number t of good fragments, each with a *reduced length* me/t. Essentially, any t-out-of-e erasure-code can be used. based on XOR operations and with linear time encoding and decoding. This IDA does not need to hide the original message, as would a full-fledged secret-sharing scheme [Sha79, Kra94], because for each of the e *evaluation* instances of the cut-and-choose, $\mathrm{P_B}$ only receives a masked version of the respective authenticated fragment, instead of several masked versions of the same authenticated message. It also does not need to support error-correction [RS60] (i.e., of semantic errors), because the *authenticator* mechanism already gives $\mathcal{S}^X$ (in the role of $\mathrm{P_B}$) the ability to detect errors and thus simply discard badly-authenticated fragments. $\mathcal{S}^X$ *reconstructs* m from any subset of at least t well authenticated fragments. The communication complexity is thus proportional to e/t. This can be made asymptotically close to 1 as desired, as ℓ increases.

It is interesting to notice that parties only need to encode; only the simulator needs to decode. A rateless code is also possible, with appropriate probabilistic considerations – there are very efficient instantiations, e.g., [Lub02, Sho06].

The statistical security is again changed, with the new criterion for successful extraction requiring a number of *good* evaluation instances at least as high as the recovery threshold. Furthermore, the fragmentation also reduces the sum of all PRG-expansion lengths, as well as the length of the sequence of masks whose hash needs to be calculated. Concrete parameters are given in Table 1.

5.2 Description of protocol #2

The protocol is succinctly described in Fig. 2. For further intuition, a pictorial sketch is provided in Fig. 3. The parties agree on security parameters (computational and statistic) and other consistent elements: the cut-and-choose parameters (with a fixed number of *check* and *evaluation* instances) (26); a PRG and a CR-Hash functions (27); the IDA scheme and parameters (28); and an authenticator mode (29) (in Fig. 2, the STRICT mode corresponds to the description given in §5.1.2) and respective parameters (30). The LOOSE mode (discussed in the full version of the paper) allows removing some steps of the protocol (namely avoiding the Equiv-Com of the hash of the message being committed) but requires a stronger assumption about the authenticator function.

5.2.1 X-commit phase ($\mathrm{P_A}$ commits a message to $\mathrm{P_B}$)

- **1.a. Commit instances.** Upon being initialized to commit m (31), $\mathrm{P_A}$ selects n random seeds (32) (e.g., 119) and uses $\mathcal{F}_{\mathrm{X}}$ to commit individually to each of them ((33),(34)). $\mathrm{P_A}$ uses the PRG to expand each seed s_j into a string s'_j with a *reduced-length* (equal to the target length ℓ divided by the IDA recovery-*threshold* t) extended by an *authenticator-length* ℓ_a (35). $\mathrm{P_A}$ calculates the *global hash* h as the CR-hash of the concatenation of all seed-expansions (36). $\mathrm{P_A}$ then uses $\mathcal{F}_{\mathrm{Q}}$ to commit to h ((37),(38)). If in the

Implicit parameters.

Security parameters: $1^{\kappa}, (1^{\sigma}, n, v, e)$ (26)

Primitives: $(\mathrm{PRG}, \kappa_{\mathrm{PRG}}), \text{CR-Hash}$ (27)

IDA: $(t, \mathrm{IDA}[t]_{\mathrm{split}}, \mathrm{IDA}[t]_{\mathrm{recover}})$ (28)

$\textsc{AuthMode} \in \{\textsc{strict}, \textsc{loose}\}$ (29)

Authenticator parameters: $\{\alpha, \ell_a = |\alpha|, \ell_z\}$ (30)

1. X-Commit phase.

$input_A \to \mathrm{P_A} : (\texttt{commit}, sid, cid, \mathrm{P_A}, \mathrm{P_B}, m)$ (31)

1.a. Commit instances. For $j \in [n]$:

$\mathrm{P_A} : s_j \leftarrow^{\$} \{0,1\}^{\kappa_{\mathrm{PRG}}}$ (seed) (32)

$\mathrm{P_A} \to \mathcal{F}_{\mathrm{X}} : (\texttt{commit}, (ctx, (X, j)), s_j)$ (33)

$\mathcal{F}_{\mathrm{X}} \to \mathrm{P_B} : (\texttt{receipt}, (ctx, (X, j)), |s_j|)$ (34)

$\mathrm{P_A} : s'_j = \mathrm{PRG}[s_j](\lceil |m|/t + \ell_a \rceil)$ (35)

$\mathrm{P_A} : h = \text{CR-Hash}(||_{j\in[n]} s'_j)$ (global hash) (36)

$\mathrm{P_A} \to \mathcal{F}_{\mathrm{Q}} : (\texttt{commit}, (ctx, Q), h)$ (37)

$\mathcal{F}_{\mathrm{Q}} \to \mathrm{P_B} : (\texttt{receipt}, (ctx, Q), |h|)$ (38)

If $\textsc{AuthMode} =^{?} \textsc{strict}$, then:

$\mathrm{P_A} : \eta = \text{CR-Hash}\,(m)$ (hash of message) (39)

$\mathrm{P_A} \to \mathcal{F}_{\mathrm{Q}} : (\texttt{commit}, (ctx, (Q, +)), \eta)$ (40)

$\mathcal{F}_{\mathrm{Q}} \to \mathrm{P_B} : (\texttt{receipt}, (ctx, (Q, +)), |\eta|)$ (41)

1.b. Cut-and-choose. $(n = e + v)$

$\mathrm{P_B} : (J_V, J_E) \leftarrow^{\$} \mathrm{Partition}[v, e](n)$ (42)

$\mathrm{P_B} : z \leftarrow^{\$} \{0,1\}^{\ell_z}$ (nonce) (43)

$\mathrm{P_B} \to \mathrm{P_A} : (\texttt{c\&c}, sid, cid, \mathrm{P_B}, \mathrm{P_A}, (J_V, J_E, z))$ (44)

1.c. Message masking.

$\mathrm{P_A} : \langle m'_j : j \in J_E \rangle \leftarrow \mathrm{IDA}[t]_{\mathrm{split}}\,(m, J_E)$ (45)

$\mathrm{P_A} : a_j = \alpha(m'_j, z) : j \in J_E$ (authenticators) (46)

$\mathrm{P_A} : t_j = \left(m'_j || a_j\right) \oplus s'_j : j \in J_E$ (maskings) (47)

$\mathrm{P_A} \to \mathrm{P_B} : (\texttt{maskings}, sid, cid, \mathrm{P_A}, \mathrm{P_B}, ||_{j\in J_E} t_j)$ (48)

2. Q-Open phase.

$input_A \to \mathrm{P_A} : (\texttt{open}, sid, cid, \mathrm{P_A}, \mathrm{P_B})$ (49)

2.a. Reveal message.

$\mathrm{P_A} \to \mathrm{P_B} : (\texttt{reveal}, sid, cid, \mathrm{P_A}, \mathrm{P_B}, m)$ (50)

If $\textsc{AuthMode} =^{?} \textsc{strict}$, then:

$\mathrm{P_A} \to \mathcal{F}_{\mathrm{Q}} : (\texttt{open-ask}, (ctx, (Q, +)))$ (51)

$\mathcal{F}_{\mathrm{Q}} \to \mathrm{P_B} : (\texttt{open-send}, (ctx, (Q, +)), \eta)$ (52)

$\mathrm{P_B}$: If CR-Hash$(m) \neq \eta$ then $\textsc{Abort}$ (53)

2.b. Obtain evaluation maskings.

$\mathrm{P_B} : \langle m'_j : j \in J_E \rangle \leftarrow \mathrm{IDA}[t]_{\mathrm{split}}\,(m, J_E)$ (54)

$\mathrm{P_B} : a_j = \alpha\left(m'_j, z\right) : j \in J_E$ (authenticator) (55)

$\mathrm{P_B} : s'_j = t_j \oplus \left(m'_j || a_j\right) : j \in J_E$ (tentative masks) (56)

2.b. Obtain check maskings.

$\mathrm{P_A} \to \mathcal{F}_{\mathrm{X}} : (\texttt{open-ask}, (ctx, (X, j))) : j \in J_V$ (57)

$\mathcal{F}_{\mathrm{X}} \to \mathrm{P_B} : (\texttt{open-send}, (ctx, (X, j)), s_j) : j \in J_V$ (58)

$\mathrm{P_B} : s'_j = \mathrm{PRG}[s_j](\lceil |m|/t + \ell_a \rceil) : j \in J_V$ (59)

2.d. Verify global hash.

$\mathrm{P_A} \to \mathcal{F}_{\mathrm{Q}} : (\texttt{open-ask}, (ctx, Q))$ (60)

$\mathcal{F}_{\mathrm{Q}} \to \mathrm{P_B} : (\texttt{open-send}, (ctx, Q), h)$ (61)

$\mathrm{P_B}$: If CR-Hash$(||_{j\in[n]} s'_j) \neq h$ then $\textsc{Abort}$ (62)

$\mathrm{P_B} \to output_B : (\texttt{accept}, sid, cid, \mathrm{P_A}, \mathrm{P_B}, m)$ (63)

Fig. 2. Protocol #2 (UC commitment scheme). Legend: legend of Fig. 1 also applies; σ (statistical security parameter, e.g., $40 \equiv 1^{40}$); n, v, e (numbers of *total* instances, *check* instances and *evaluation* instances); [n] (set of the first n positive integers); Partition$[v, e](n)$ (set of possible partitions of [n], into a pair of complementary subsets, the first with v elements, and the second with the remaining e). IDA$[t]$ (*information dispersal algorithm* (erasure code) with *recovery threshold* of t fragments; it has sub-algorithms *split* and *recover*; if e and v are fixed in a setup phase they must satisfy $((n-b)!e!) \,/\, ((e-b)s!) \leq 2^{-\sigma}$, where $b = e - t + 1$ is the number of *bad* instances in an optimal attack); α (authenticator function); ℓ_z (length of nonce); ℓ_a (length of authenticator output, e.g., 256 bits).

$\textsc{strict}$ mode, $\mathrm{P_A}$ also computes the hash of the message m (39) and then uses $\mathcal{F}_{\mathrm{Q}}$ to commit to said hash ((40),(41)).

- **1.b. Cut-and-choose.** $\mathrm{P_B}$ decides a random cut-and-choose partition (42) (e.g., identifying 73 instances for *check* and 46 for *evaluation*) and a random nonce z (43) and sends them both to $\mathrm{P_A}$ (44).
- **1.c. Message masking.** $\mathrm{P_A}$ uses the threshold IDA to *split* her message into as many fragments as the number of *evaluation* instances (45), each with a *reduced length*. Then, $\mathrm{P_A}$ computes the authenticator a_j of each

fragment m'_j as an appropriate function α of the fragment and the nonce (46); $\mathrm{P_A}$ then uses the extended mask s'_j to compute the masking t_j of the fragment concatenated with the authenticator (47). Finally, $\mathrm{P_A}$ sends to $\mathrm{P_B}$ the maskings associated with all *evaluation* instances (48).

5.2.2 Q-open phase ($\mathbf{P_A}$ opens a message to $\mathbf{P_B}$)

- **2.a. Reveal message.** Upon being initialized to open the committed message m (49), $\mathrm{P_A}$ sends m to $\mathrm{P_B}$ (50). If using the STRICT authenticator mode, then $\mathrm{P_A}$ also asks $\mathcal{F}_\mathrm{Q}$ to open to $\mathrm{P_B}$ the hash of the message ((51),(52)). $\mathrm{P_B}$ then verifies that it is consistent with the hash of the received message (53). If not, it Aborts; otherwise it proceeds.
- **2.b. Obtain *evaluation* masks.** $\mathrm{P_B}$ uses the IDA to obtain the same fragments that an honest $\mathrm{P_A}$ would (54). $\mathrm{P_B}$ computes the authenticator of the fragment in the same way that an honest $\mathrm{P_A}$ would have, based on the fragment and the nonce (55). Then, $\mathrm{P_B}$ concatenates the tentative fragment and the tentative authenticator, and computes the XOR combination of the resulting string with the extended masking, thus obtaining the tentative extended mask s'_j, supposedly used by $\mathrm{P_A}$ (56).
- **2.c. Obtain *check* masks.** $\mathrm{P_A}$ uses $\mathcal{F}_\mathrm{X}$ to *open* to $\mathrm{P_B}$ the seeds of *check* instances (but not those of *evaluation* instances) ((57),(58)). $\mathrm{P_B}$ locally computes the PRG-expansion (with the appropriate length) of each *check* seed (59).
- **2.d. Verify global hash.** $\mathrm{P_A}$ uses $\mathcal{F}_\mathrm{Q}$ to *open* to $\mathrm{P_B}$ the previously committed global hash ((60), (61)). Then, $\mathrm{P_B}$ verifies that the global hash of all concatenated masks is equal to the one just opened by $\mathrm{P_A}$ (62). If some verification has failed, then $\mathrm{P_B}$ aborts, otherwise it accepts the message of $\mathrm{P_A}$ as a correct *opening* (63).

5.3 Concrete configurations

Table 1 shows optimal configurations of the cut-and-choose and IDA parameters to achieve 40 bits of statistical security, for different goals of communication rate. Asymptotically as ℓ increases, it is possible to configure the parameters to yield arbitrary high levels of statistical security and at the same time reduce the expansion-rate to values arbitrarily close to 1. With $(n; e; t) = (119; 46; 23)$, the scheme achieves 40 bits of statistical security and an asymptotic communication expansion-rate $r = 2$ in the *commit* phase (the open phase always has an asymptotic rate 1). With $(n; e; t) = (775; 275; 250)$, the rate becomes $r = 1.1$, with the computed PRG output and the hash input being $r' = 3.1$ times the message length. Both r and r' can be brought arbitrarily close to 1. In comparison, for a communication expansion rate of $r = 1.1$, the protocol from [GIKW14] would require encoding m into 53,020 blocks, and using an error correcting code capable of correcting more than 1198 semantic errors. Table 1 also describes parameters for optimizations of [GIKW14], namely by using k-out-of-n OT instead of δ-Rabin OT, reducing the number of instances by up to a factor slightly larger than two.

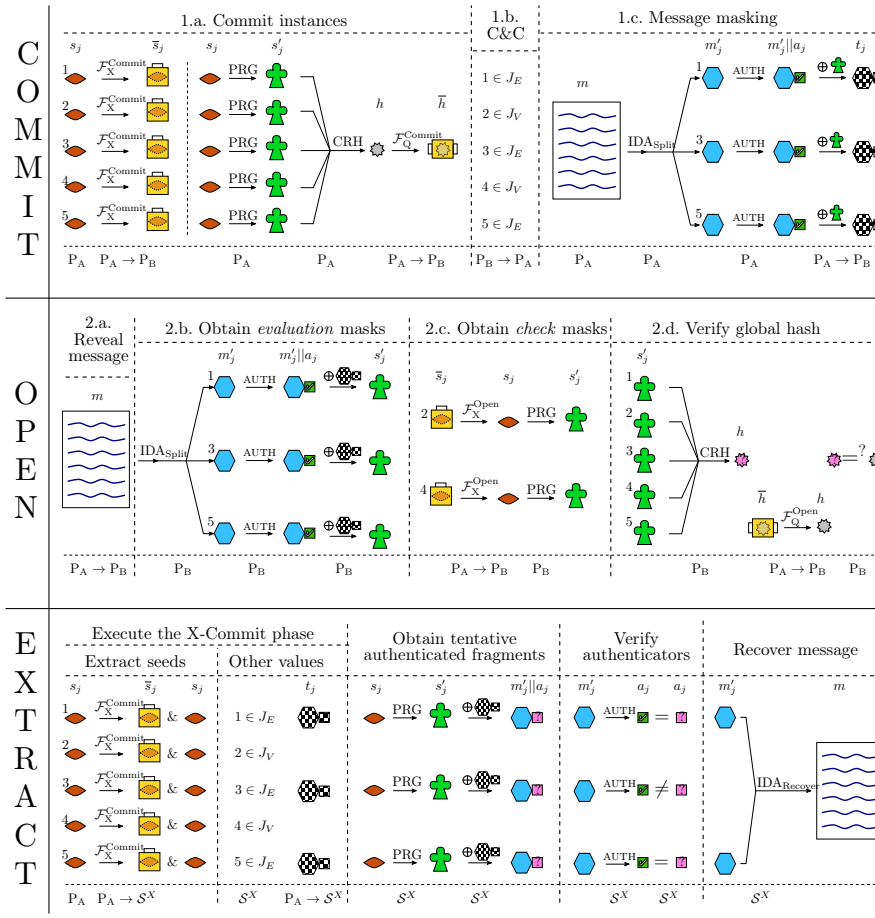

Fig. 3. Sketch of UC commitment scheme. Legend: (seed s_j); (Ext-Com $\overline{s}_j$ – like a vault with a single opening); (seed expansion s'_j – like a tree growing from a seed); (global hash – like a smashed paper); (Equiv-Com $\overline{h}$ – like a vault with several openings); (message m being committed – like a text file); (message fragment m'_j – can be combined with other fragments to recover the initial message); (authenticator a_j – vouches for the correctness of the respective fragment); (masking t_j – the chess pattern represents something that is masked); AUTH (authenticator function); C&C (cut-and-choose); $\mathscr{C}$ (commitment scheme); $\mathscr{C}_{\mathrm{X}}$ (extractable $\mathscr{C}$); $\mathscr{C}_{\mathrm{Q}}$ (equivocable $\mathscr{C}$); $\mathcal{S}^X$ (simulator with extraction goal). This is a toy example with a cut-and-choose with $n = 5$ instances, of which $v = 2$ are selected for *check* and $e = 3$ are selected for *evaluation*. In the *extraction* example, a malicious $\mathrm{P}^*_{\mathrm{A}}$ constructed one *bad* instance ($j = 3$), selected for the *check* subset. $\mathcal{S}^X$ detects the bad instance and thus ignores it when using the IDA to reconstruct the message from only $t = 2$ (the recovery threshold) fragments.

Remark (interactivity tradeoffs). The use of an Equiv-Com scheme with $\mathrm{P_A}$ as sender and $\mathrm{P_B}$ as receiver can be replaced by an Ext-Com scheme with

Table 1. UC commitment scheme parameters for 40 bits of statistical security

A	B	C	D	E	F	
Maximum allowed expansion rate	**This work**		**[GIKW14]** (original)	Variations of **[GIKW14]**		1
	$r = e/t \leq r_{\max}$	$r' = n/t \leq r_{\max}$	$\delta = t_0/(2n')$ $r = n'/n$	Optimal δ $r = n'/n$	t_0-out-of-n' OT $r = n'/n$	2
$r_{\max} \leq 2$	$n = 119$ $v = 73$ $e = 46$ $t = 23$ $r' \approx 5.17$ $r = 2$	$n = 324$ $v = 87$ $e = 237$ $t = 162$ $r' = 2$ $r \approx 1.46$	$n = 826$ $n' = 1652$ $t_0 = 428$ $t_{\text{error}} = \lfloor 399/2 \rfloor$ $\delta = 107/826 \approx 0.1295$ $r = 2$	$n = 577$ $n' = 1154$ $t_0 = 339$ $t_{\text{error}} = \lfloor 239/2 \rfloor$ $\delta \approx 0.2064$ $r = 2$	$n = 352$ $n' = 704$ $t_0 = 186$ $t_{\text{error}} = \lfloor 167/2 \rfloor$ $r = 2$	3
$r_{\max} \leq 3/2$	$n = 193$ $v = 121$ $e = 72$ $t = 48$ $r' \approx 4.02$ $r = 1.5$	$n = 822$ $v = 144$ $e = 678$ $t = 548$ $r' = 1.5$ $r \approx 1.237$	$n = 2540$ $n' = 3810$ $t_0 = 650$ $t_{\text{error}} = \lfloor 621/2 \rfloor$ $\delta = 65/762 \approx 0.0853$ $r = 1.5$	$n = 1706$ $n' = 2559$ $t_0 = 481$ $t_{\text{error}} = \lfloor 373/2 \rfloor$ $\delta \approx 0.1379$ $r = 1.5$	$n = 1152$ $n' = 1728$ $t_0 = 296$ $t_{\text{error}} = \lfloor 281/2 \rfloor$ $r = 1.5$	4
$r_{\max} \leq 11/10$	$n = 775$ $v = 500$ $e = 275$ $t = 250$ $r' = 3.1$ $r = 1.1$	$n = 12,793$ $v = 598$ $e = 12,195$ $t = 11,630$ $r' = 1.1$ $r \approx 1.0489$	$n = 48,200$ $n' = 53,020$ $t_0 = 2424$ $t_{\text{error}} = \lfloor 2397/2 \rfloor$ $\delta = \frac{303}{13255} \approx 0.0229$ $r = 1.1$	$n = 28,740$ $n' = 31,614$ $t_0 = 1498$ $t_{\text{error}} = \lfloor 1377/2 \rfloor$ $\delta \approx 0.03945$ $r = 1.1$	$n = 23,530$ $n' = 25,883$ $t_0 = 1185$ $t_{\text{error}} = \lfloor 1169/2 \rfloor$ $r = 1.1$	5
$r_{\max} \leq 101/100$	$n = 7310$ $v = 4684$ $e = 2626$ $t = 2600$ $r' = 2.81$ $r = 1.01$	$n = 1,125,645$ $v = 5631$ $e = 1,120,014$ $t = 1,114,500$ $r' = 1.01$ $r \approx 1.00495$	$n = 4,474,600$ $n' = 4,519,346$ $t_0 = 22,388$ $t_{\text{error}} = \lfloor 22,359/2 \rfloor$ $\delta = \frac{5597}{2,259,673} \approx 0.00248$ $r = 1.01$	$n = 2,384,200$ $n' = 2,408,042$ $t_0 = 12,166$ $t_{\text{error}} = \lfloor 11,677/2 \rfloor$ $\delta \approx 0.004737$ $r = 1.01$	$n = 2,231,600$ $n' = 2,253,916$ $t_0 = 11,166$ $t_{\text{error}} = \lfloor 11,151/2 \rfloor$ $r = 1.01$	6

Common legend for columns B-F. r (communication expansion rate in the commit phase, relative to the target length, i.e., to the length of the value being committed – it is asymptotic in that it does not account with the base short commitments (columns B-C) or the OT implementation (columns D-F).

Legend for columns B-C ("This work"). r' (overall length of PRG output, divided by the target length (at $\mathrm{P_A}$ – it is smaller at $\mathrm{P_B}$, because $\mathrm{P_B}$ does not evaluate the PRG for *evaluation* instances); also the overall length of CR-Hash input, divided by the target length); n (total number of instances in the cut-and-choose); e (number of evaluation instances = number of fragments); t (recovery threshold = number of fragments necessary to recover message). The parameters were chosen to minimize the total number of instances n, while satisfying the *maximum allowed rate* ($r_{\max}$, identified in column A), as follows: in column B ("$r = e/t \leq r_{\max}$"), the communication expansion rate r is limited to $r_{\max}$ (in this case the PRG and the CR-Hash can be applied to bigger lengths – see r'); in column C ($t = \lceil n/r \rceil$), the computation expansion rate r' determined by the length of PRG output and CR-Hash input are limited to $r_{\max}$ (and in this case the overall communication rate r is smaller). After minimizing n, the remaining parameters were chosen to minimize e.

Legend for columns D-F ("[GIKW14]" and variations). n (number of blocks before encoding, i.e., number of symbols in which the target message is partitioned); t_0 (0-info threshold (the original notation was t), i.e., number of blocks whose knowledge does not reveal anything about the original message); t_{error} (error-recovery threshold – the original notation is $\Delta/2$); δ (probability of message passing through the δ-Rabin-OT – the original version uses $t_0 = 2\delta n'$); n' (total number of blocks after encoding, satisfying $n' = t + n + \Delta - 1$). For each value $r = n'/n$, the values of other parameters were chosen to minimize n. In column F, where the equivocator-simulator can always equivocate, statistical security depends only on the probability that a malicious $\mathrm{P_A}$ can guess $t_{\text{error}} + 1$ positions that $\mathrm{P_B}$ will not select in the OT.

P_B as sender and P_A as receiver, and a regular Com scheme (i.e., possibly neither Ext nor Equiv) and further interaction. Once an Ext-Com scheme is assumed available (either by an ideal functionality or by a trapdoor obtained by the simulator in a setup phase, e.g., from a CRS), equivocation is possible without rewinding, as follows. Basically, the Ext of a short bit-string committed by P_B^* with an Ext-Com scheme would allow $\mathcal{S}$ (in the role of P_A in the simulated execution) to decide (within the overall open phase of the UC scheme) any desired outcome of a (single-side simulatable) short coin-flipping played between P_A and P_B. Each bit of this short coin-flipping can be set to determine one-out-of-two positions to open from each pair of (supposedly) copies of a committed bit (and additional redundant checksum bits included to prevent malicious behavior). This allows $\mathcal{S}$ to equivocate the short-bit string because it could undetectably commit to two different bits in each position (instead of two copies of the same bit) and then open only the convenient ones. In a direction of less interaction, it is conceivably possibly to let the cut-and-choose partition and nonce values be computed by P_A non-interactively, if willing to accept an assumption of a non-programmable random oracle model [Lin15]. This would make all interactivity of the commit phase (commit and open) become implicit in the instantiations of the base commitment schemes (Ext and Equiv). However, concrete instantiations are not yet explored in this paper. Furthermore, the cut-and-choose and IDA (erasure code) parameters would have to increase, letting the statistical security parameter become equal to the cryptographic security parameter, to mitigate the new possibility that P_A could computationally try a brute-force trial-and-error attempt to exploit the probability of error that would otherwise be negligible only in a low statistical parameter.

5.4 Security analysis

Proving security amounts to show, without rewiding, that the new commitment scheme is Ext&Equiv, i.e., the *commit* phase is Ext and the *open* phase is Equiv. The analysis assumes that the PRG and CR-Hash are cryptographically secure and that the underlying Ext-Com ($\mathscr{C}_X$) and Equiv-Com ($\mathscr{C}_Q$) are replaced (in a hybrid model) by respective ideal functionalities ($\mathcal{F}_X$, $\mathcal{F}_Q$). The proof of security is accomplished by defining respective simulators.

Theorem 2 (security of protocol #2). *Assuming a cryptographically secure PRG and CR-Hash, and an adequate authenticator, protocol #2 UC-realizes the ideal functionality $\mathcal{F}_{MCOM}$ of* long *bit-string commitments in the ($\mathcal{F}_X$,$\mathcal{F}_Q$)-hybrid model, in the presence of static and computationally active adversaries. Each phase of $\mathcal{F}_Q$ and $\mathcal{F}_X$ is invoked for short bit-strings only a number of times that is independent of the polynomial target length.*

5.4.1 Extractability – simulatability with corrupted P_A^*. The extractor-simulator $\mathcal{S}^X$ initiates a simulation, with black-box access to $\mathcal{A}$, letting it believes that it is in the real world controlling P_A^*.

Simulation of the* commit *phase. Once the protocol starts, $\mathcal{S}^X$ (in the role of honest $\mathrm{P_B}$ and also in the role of $\mathcal{F}_\mathrm{X}$ in the simulated execution) extracts the seeds committed by $\mathrm{P_A^*}$ (33) and later receives from $\mathrm{P_A^*}$ the maskings of authenticated fragments of the message being committed (48). $\mathcal{S}^X$ then unmasks each masking, using the PRG-expansion of the respective extracted seed, obtaining from each a respective *tentative* authenticated fragment. $\mathcal{S}^X$ verifies whether the authentication is correct or not, thus identifying which instances are *good*. (The security of the described authenticator is statistically derived from the properties of a universal hash family.) If the number of good fragments is at least t (the recovery threshold) then $\mathcal{S}^X$ uses the IDA recovery algorithm to reconstruct the message from t (the recovery threshold) *good* fragments. Otherwise, if there are less than t good fragments, then $\mathcal{S}^X$ realizes that it cannot extract the message from $\mathrm{P_A^*}$, but it does not complain. Instead, $\mathcal{S}^X$ computes a random message as the assumed extracted message, and in addition it memorizes that the extracted message is corrupted. Finally (in either of the two above cases), in the ideal world, $\mathcal{S}^X$ (in the role of the ideal $\widehat{\mathrm{P}}_\mathrm{A}^*$) sends the extracted message to the ideal functionality $\mathcal{F}_\mathrm{MCOM}$, thus committing to it.

Simulation of the* open *phase. Once $\mathrm{P_A^*}$ opens the message to $\mathrm{P_B}$ in the simulated execution, $\mathcal{S}^X$ checks that the opening is successful and that it corresponds to the previously extracted message. If the opening is unsuccessful, e.g., if the global hash verification fails (62), then $\mathcal{S}^X$ emulates an abort, leading $\mathcal{F}_\mathrm{MCOM}$ to halt the execution associated with this commitment, consequently leading the ideal party $\widehat{\mathrm{P}}_\mathrm{B}$ to never receive any opening. If (with negligible probability) the opening is successful but different from the value previously extracted from $\mathcal{S}^X$, then $\mathcal{S}^X$ outputs `Fail` (i.e., in this case the simulation fails). Otherwise, if the opening of the expected message is done successfully, then $\mathcal{S}$ asks $\mathcal{F}_\mathrm{MCOM}$ in the ideal world to *open* the committed message.

Analysis of the simulation (statistical security). In the commit phase, $\mathcal{S}$ makes a perfect emulation of the abort distribution, since it only aborts early if and only if $\mathrm{P_A^*}$ also aborts. Thus, distinguishability (by the environment) between real and simulated executions can only happen if $\mathrm{P_A^*}$ is able (with non-negligible probability) to successfully open a message different from the one $\mathcal{S}^X$ has extracted. However, this is not possible. Based on the (described) authenticator mechanism security (derived directly from the collision-resistance of a CR-Hash, and the statistical properties of a universal hash family), $\mathrm{P_A^*}$ cannot forge a bad authentication, i.e., lead $\mathcal{S}^X$ to believe that a *bad* fragment is actually *good*. Also, based on the default binding property of all underlying commitments, $\mathrm{P_A^*}$ is not able to equivocate any of the Ext-Com or Equiv-Com. It can thus be assumed impossible for $\mathcal{S}$ to unknowingly mark as *good* an evaluation fragment (i.e., the result upon unmasking) that is actually *bad*. Now, a malicious successful opening by $\mathrm{P_A^*}$ requires that all check instances are good selected and at least $n - t + 1$ evaluation instances are bad. However, the probability of this event can be made negligible for appropriate cut-and-choose and IDA parameters (see Table 1). As an example, in the trivial case where $\mathrm{P_A^*}$ would build all check

and evaluation instances as bad, $\mathcal{S}^X$ in the ideal world would still commit to a random valid value, but later in the open phase it would never let the ideal functionality open the value to the honest $\mathrm{P_B}$.

Remark. There is an interesting subtle difference between two types of real UC-Com schemes. There are Com-schemes where the receiver is ensured that the committer is *technically able* to open the commitment (if it "wants" to). For example, this is the case of schemes that contain in the commit phase a ZKPoK of the committed value. There are other schemes where the commit phase is not enough to let the receiver know about the actual ability of the committer to later open a value. It is possible that a maliciously played commit phase prevents the sender ($\mathrm{P_A^*}$) in advance from being able to later open the commitment accepted by the receiver ($\mathrm{P_B}$). Protocol #2 is of this second kind, i.e., $\mathcal{S}$ (who can nonetheless extract the contribution timely) needs to wait for the open phase before aborting. The protocol can be easily changed to become of the first type (if desired), at the cost of increasing the calls to the Equiv-Com functionality, namely in number equal to the cut-and-choose instances, while nonetheless retaining an amortized communication complexity. The idea is simple: instead of just producing one Equiv-Com of the global hash, $\mathrm{P_A^*}$ would produce one Equiv-Com for each mask of a fragment; then, after the cut-and-choose partition is determined, but still within the overall commit phase, $\mathrm{P_A^*}$ would open the check seeds and the check hashes. In this way, $\mathcal{S}$ immediately knows whether some bad check instance was bad. If any bad check instance is detected, then $\mathcal{S}$ can immediately emulate an abort (assuming an appropriate ideal functionality contemplating such abort in the commit phase); otherwise, $\mathcal{S}$ can now base its final choice on the verification of the authentication of extracted evaluation masks and the associated anticipated fragments. In this construction there is a negligible probability that the number of good instances is less than the recovery threshold.

5.4.2 Equivocability – simulatability with corrupted $\mathbf{P_B^*}$. The equivocator-simulator $\mathcal{S}^Q$ initiates a simulation, with black-box access to $\mathcal{A}$, letting it believe that it is in the real world controlling $\mathrm{P_B^*}$.

Simulation of the* commit *phase. In the ideal world, $\mathcal{S}^Q$ in the role of $\widehat{\mathrm{P}}_\mathrm{B}^*$, waits to receive from $\mathcal{F}_{\mathrm{MCOM}}$ a receipt of commitment done by the ideal $\widehat{\mathrm{P}}_\mathrm{A}$. Then, in the role of $\mathrm{P_A}$ in the simulated execution, $\mathcal{S}^Q$ plays the whole commit phase to commit a random message to $\mathrm{P_B^*}$. This involves keeping state about the seeds (32) and their Ext-Coms (33), about the Equiv-Com of the global hash of masks (38), possibly about the Equiv-Com of the hash of the random message (41) (i.e., if in the STRICT mode), about the cut-and-choose partition and the nounce, and about the maskings of authenticated fragments (48). If $\mathrm{P_B^*}$ aborts at any point before the end of the overall *commit* phase, then $\mathcal{S}^Q$ emulates an abort, i.e., in the role of $\widehat{\mathrm{P}}_\mathrm{B}^*$ in the ideal world sends `abort` to $\mathcal{F}_{\mathrm{MCOM}}$, thus making it ignore further actions related with this commitment sub-session.

Simulation of the* open *phase. $\mathcal{S}^Q$ waits in the ideal world to receive from $\mathcal{F}_{\text{MCOM}}$ the *opening* of the target message (i.e., the one initially committed by the ideal $\widehat{\text{P}}_\text{A}$). Then, $\mathcal{S}^Q$, in the role of P_A and also in the role of $\mathcal{F}_\text{Q}$ in the simulated execution, sends to P_B^* the target message (50), instead of the previously committed random message. If in the STRICT mode, then $\mathcal{S}^Q$ in the role of $\mathcal{F}_\text{Q}$ equivocates the opening of the needed hash of the message (52). Then, $\mathcal{S}^Q$ computes what are the *alternative masks* s'_j needed to unmask (the maskings t_j previously sent) into the target message received from $\mathcal{F}_{\text{MCOM}}$. This is done in the exact same way that P_B does as receiver: $\mathcal{S}^Q$ computes the message fragments (54), then their authenticators (55), and then takes the XOR with the maskings t_j (56) that were transmitted in the commit phase. Finally, $\mathcal{S}^Q$ computes the global hash (as in (36), but now using the updated masks), and then impersonates $\mathcal{F}_\text{Q}$ and equivocates the opening of said global hash (61). This allows P_B^* to perform all verifications as if $\mathcal{S}^Q$ was in fact an honest P_A. Finally, $\mathcal{S}^Q$ outputs in the ideal world whatever P_B^* outputs in the simulated execution (63).

Analysis of the simulation. The only difference between a real protocol execution and the simulated execution is that $\mathcal{S}^Q$ commits to a random message and later equivocates it. However, detection by P_B^* of equivocation would require differentiating the random masks from seed-expansions, which is contrary to the pseudo-randomness assumption of the PRG. Thus, in case of corrupted P_B^* the distributions between real and ideal world are computationally indistinguishable.

Remark. The cut-and-choose partition does not need to be decided via a simulatable coin-flipping, because equivocation is directly based on the assumed ability to equivocate the global hash (committed with an Equiv-Com), which directly allows equivocation of the masks of all evaluation instances. Thus, to P_B^*, the actions of $\mathcal{S}^Q$ "appear" as correct independently of the partition. $\mathcal{S}^Q$ simply produces all commitments of seeds and all maskings correctly (for a random value), so that later all check instances are consistent.

Acknowledgments. The author would like to thank the anonymous referees for their useful reviewing comments.

References

[ALSZ15] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More Efficient Oblivious Transfer Extensions with Security for Malicious Adversaries. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015*, vol. 9056 of *LNCS*, pages 673–701. Springer Berlin Heidelberg, 2015. Also at ia.cr/2015/061. (p. 6.)

[BCPV13] O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Analysis and Improvement of Lindell's UC-Secure Commitment Schemes. In M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS 2013*, vol. 7954 of *LNCS*, pages 534–551. Springer, Berlin Heidelberg, 2013. Also at ia.cr/2013/123. (pp. 3, 9, and 10.)

[Bea96] D. Beaver. Adaptive Zero Knowledge and Computational Equivocation (Extended Abstract). In *STOC 1996*, pages 629–638. ACM, New York USA, 1996. (pp. 2 and 10.)

[Blu83] M. Blum. Coin flipping by telephone – a protocol for solving impossible problems. *SIGACT News*, 15:23–27, 1983. Appeared also at CRYPTO 1981. (pp. 2, 4, and 8.)

[Bra13] L. T. A. N. Brandão. Secure Two-Party Computation with Reusable Bit-Commitments, via a Cut-and-Choose with Forge-and-Lose Technique. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013*, vol. 8270 of *LNCS*, pages 441–463. Springer-Verlag, Berlin Heidelberg, 2013. Also at ia.cr/2013/577. (pp. 2, 12, 16, and 17.)

[Can00] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *J. Cryptology*, 13:143–202, 2000. Also at ia.cr/1998/018. (pp. 2 and 10.)

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *FOCS 2001*, pages 136–145, 2001. Also at ia.cr/2000/067. (pp. 2 and 10.)

[CDD+15] I. Cascudo, I. Damgård, B. David, I. Giacomelli, J. Nielsen, and R. Trifiletti. Additively Homomorphic UC Commitments with Optimal Amortized Overhead. In J. Katz, editor, *Public-Key Cryptography – PKC 2015*, vol. 9020 of *LNCS*, pages 495–515. Springer Berlin Heidelberg, 2015. (pp. 5 and 9.)

[CF01] R. Canetti and M. Fischlin. Universally Composable Commitments. In J. Kilian, editor, *CRYPTO 2001*, vol. 2139 of *LNCS*, pages 19–40. Springer, Berlin Heidelberg, 2001. Also at ia.cr/2001/055. (pp. 3, 6, 8, 9, and 10.)

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proc. STOC 2002*, pages 494–503. ACM, New York USA, 2002. Also at ia.cr/2002/140. (p. 8.)

[CR03] R. Canetti and T. Rabin. Universal Composition with Joint State. In D. Boneh, editor, *CRYPTO 2003*, vol. 2729 of *LNCS*, pages 265–281. Springer, Berlin Heidelberg, 2003. Also at ia.cr/2002/047. (pp. 2 and 8.)

[Cre03] G. D. Crescenzo. Equivocable and Extractable Commitment Schemes. In S. Cimato, G. Persiano, and C. Galdi, editors, *SCN 2002*, vol. 2576 of *LNCS*, pages 74–87. Springer, Berlin Heidelberg, 2003. (pp. 6 and 9.)

[Dam88] I. B. Damgård. Collision Free Hash Functions and Public Key Signature Schemes. In D. Chaum and W. L. Price, editors, *EUROCRYPT 1987*, vol. 304 of *LNCS*, pages 203–216. Springer, Berlin Heidelberg, 1988. (p. 6.)

[DCIO98] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and Non-malleable Commitment. In *STOC 1998*, pages 141–150. ACM, New York USA, 1998. (p. 8.)

[DCKOS01] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and Non-interactive Non-malleable Commitment. In B. Pfitzmann, editor, *EUROCRYPT 2001*, vol. 2045 of *LNCS*, pages 40–59. Springer, Berlin Heidelberg, 2001. Also at ia.cr/2001/032. (p. 7.)

[DCO99] G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Pre-processing. In M. Wiener, editor, *CRYPTO 1999*, vol. 1666 of *LNCS*, pages 485–502. Springer, Berlin Heidelberg, 1999. (p. 6.)

[DDGN14] I. Damgård, B. David, I. Giacomelli, and J. B. Nielsen. Compact VSS and Efficient Homomorphic UC Commitments. In P. Sarkar and T. Iwata,

editors, *ASIACRYPT 2014*, vol. 8874 of *LNCS*, pages 213–232. Springer, Berlin Heidelberg, 2014. Also at ia.cr/2014/370. (pp. 5 and 9.)

[DL09] I. Damgård and C. Lunemann. Quantum-Secure Coin-Flipping and Applications. In M. Matsui, editor, *ASIACRYPT 2009*, vol. 5912 of *LNCS*, pages 52–69. Springer, Berlin Heidelberg, 2009. Also at arXiv:0903.3118. (pp. 9 and 10.)

[DN02] I. Damgård and J. B. Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In M. Yung, editor, *CRYPTO 2002*, vol. 2442 of *LNCS*, pages 581–596. Springer, Berlin Heidelberg, 2002. Also at ia.cr/2001/091. (pp. 6, 8, 9, and 10.)

[DNO10] I. Damgård, J. B. Nielsen, and C. Orlandi. On the Necessary and Sufficient Assumptions for UC Computation. In D. Micciancio, editor, *TCC 2010*, vol. 5978 of *LNCS*, pages 109–127. Springer, Berlin Heidelberg, 2010. (pp. 6 and 10.)

[DO10] I. Damgard and C. Orlandi. Multiparty Computation for Dishonest Majority: From Passive to Active Security at Low Cost. In T. Rabin, editor, *CRYPTO 2010*, vol. 6223 of *LNCS*, pages 558–576. SBH, 2010. (p. 9.)

[ElG85] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. Blakley and D. Chaum, editors, *Advances in Cryptology*, vol. 196 of *LNCS*, pages 10–18. Springer-Verlag, Berlin Heidelberg, 1985. (p. 12.)

[FJNT16] T. Frederiksen, T. Jakobsen, J. Nielsen, and R. Trifiletti. On the Complexity of Additively Homomorphic UC Commitments. In E. Kushilevitz and T. Malkin, editors, *Theory of Cryptography*, vol. 9562 of *LNCS*, pages 542–565. Springer Berlin Heidelberg, 2016. Also at ia.cr/2015/694. (p. 9.)

[FLM11] M. Fischlin, B. Libert, and M. Manulis. Non-interactive and Re-usable Universally Composable String Commitments with Adaptive Security. In D. Lee and X. Wang, editors, *ASIACRYPT 2011*, vol. 7073 of *LNCS*, pages 468–485. Springer, Berlin Heidelberg, 2011. (p. 9.)

[FS90] U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. In G. Brassard, editor, *CRYPTO 1989*, vol. 435 of *LNCS*, pages 526–544. Springer New York, 1990. (p. 6.)

[GIKW14] J. A. Garay, Y. Ishai, R. Kumaresan, and H. Wee. On the Complexity of UC Commitments. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, vol. 8441 of *LNCS*, pages 677–694. Springer, Berlin Heidelberg, 2014. (pp. 5, 9, 10, 20, and 22.)

[GK96] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *J. Cryptology*, 9(3):167–189, 1996. (pp. 7 and 14.)

[GMS08] V. Goyal, P. Mohassel, and A. Smith. Efficient Two Party and Multi Party Computation Against Covert Adversaries. In N. Smart, editor, *EUROCRYPT 2008*, vol. 4965 of *LNCS*, pages 289–306. Springer-Verlag, Berlin Heidelberg, 2008. (p. 13.)

[Gol04] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004. ISBN: 0521830842. (p. 2.)

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (p. 6.)

[HKE13] Y. Huang, J. Katz, and D. Evans. Efficient Secure Two-Party Computation Using Symmetric Cut-and-Choose. In R. Canetti and J. Garay, editors, *CRYPTO 2013*, vol. 8043 of *LNCS*, pages 18–35. Springer-Verlag, Berlin Heidelberg, 2013. Also at ia.cr/2013/081. (p. 17.)

[HMQU06] D. Hofheinz, J. Müller-Quade, and D. Unruh. On the (Im-)Possibility of Extending Coin Toss. In S. Vaudenay, editor, *EUROCRYPT 2006*, vol. 4004 of *lncs*, pages 504–521. Springer, Berlin Heidelberg, 2006. Also at ia.cr/2006/177. (p. 6.)

[Kra94] H. Krawczyk. Secret Sharing Made Short. In D. Stinson, editor, *CRYPTO 1993*, vol. 773 of *LNCS*, pages 136–146. Springer, Berlin Heidelberg, 1994. (p. 18.)

[KS08] V. Kolesnikov and T. Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming*, vol. 5126 of *LNCS*, pages 486–498. Springer Berlin Heidelberg, 2008. (p. 6.)

[Lin03] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *J. Cryptology*, 16(3):143–184, 2003. Also at ia.cr/2001/107. (pp. 3, 4, 6, 7, 8, 10, and 13.)

[Lin11] Y. Lindell. Highly-Efficient Universally-Composable Commitments Based on the DDH Assumption. In K. Paterson, editor, *EUROCRYPT 2011*, vol. 6632 of *LNCS*, pages 446–466. Springer, Berlin Heidelberg, 2011. Also at ia.cr/2011/180. (p. 9.)

[Lin13] Y. Lindell. Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries. In R. Canetti and J. Garay, editors, *CRYPTO 2013*, vol. 8043 of *LNCS*, pages 1–17. Springer-Verlag, Berlin Heidelberg, 2013. Also at ia.cr/2013/079. (p. 17.)

[Lin15] Y. Lindell. An Efficient Transform from Sigma Protocols to NIZK with a CRS and Non-programmable Random Oracle. In Y. Dodis and J. Nielsen, editors, *Theory of Cryptography*, vol. 9014 of *LNCS*, pages 93–109. Springer Berlin Heidelberg, 2015. (p. 23.)

[LN11] C. Lunemann and J. B. Nielsen. Fully Simulatable Quantum-Secure Coin-Flipping and Applications. In A. Nitaj and D. Pointcheval, editors, *AFRICACRYPT 2011*, vol. 6737 of *LNCS*, pages 21–40. Springer, Berlin Heidelberg, 2011. Also at ia.cr/2011/065. (pp. 6 and 9.)

[LPS08] Y. Lindell, B. Pinkas, and N. Smart. Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In R. Ostrovsky, R. De Prisco, and I. Visconti, editors, *SCN '08*, vol. 5229 of *LNCS*, pages 2–20. Springer-Verlag, Berlin Heidelberg, 2008. (p. 12.)

[Lub02] M. Luby. LT codes. In *Proc. 43rd annual IEEE symposium on FOCS 2002*, pages 271–280, 2002. (p. 18.)

[Nao91] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. (p. 6.)

[Nat14a] National Institute of Standards and Technology. FIPS 202 – DRAFT SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. U.S. Department of Commerce, NIST-ITL-CSD, May 2014. (p. 3.)

[Nat14b] National Institute of Standards and Technology. SP800-90 A Rev. 1 – DRAFT Recommendation for Random Number Generation Using Deterministic Random Bit Generators. U.S. Department of Commerce, NIST-ITL-CSD, April 2014. (p. 3.)

[NY89] M. Naor and M. Yung. Universal One-way Hash Functions and Their Cryptographic Applications. In *STOC 1989*, pages 33–43. ACM, New York USA, 1989. (p. 6.)

[Ped92] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *CRYPTO 1991*, vol. 576 of *LNCS*, pages 129–140. Springer-Verlag, Berlin Heidelberg, 1992. (p. 12.)

[PW09] R. Pass and H. Wee. Black-Box Constructions of Two-Party Protocols from One-Way Functions. In O. Reingold, editor, *TCC 2009*, vol. 5444 of *LNCS*, pages 403–418. Springer-Verlag, Berlin Heidelberg, 2009. Also at *IACR Online Proceedings* for TCC 2009. (pp. 3, 4, 6, 7, 8, and 13.)

[Rab89] M. O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *J. ACM*, 36(2):335–348, 1989. (p. 17.)

[Rom90] J. Rompel. One-way Functions Are Necessary and Sufficient for Secure Signatures. In *Proc. STOC 1990*, pages 387–394. ACM, New York USA, 1990. (p. 6.)

[Ros04] A. Rosen. A Note on Constant-Round Zero-Knowledge Proofs for NP. In M. Naor, editor, *Theory of Cryptography*, vol. 2951 of *LNCS*, pages 191–202. Springer, Berlin Heidelberg, 2004. (p. 8.)

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the SIAM*, 8(2):300–304, 1960. (p. 18.)

[Rus95] A. Russell. Necessary and sufficient conditions for collision-free hashing. *J. Cryptology*, 8(2):87–99, 1995. (p. 6.)

[Sch91] C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991. See also extended abstract at EUROCRYPT 1989. (p. 12.)

[SCP00] A. Santis, G. Crescenzo, and G. Persiano. Necessary and Sufficient Assumptions for Non-interactive Zero-Knowledge Proofs of Knowledge for All NP Relations. In U. Montanari, J. Rolim, and E. Welzl, editors, *ICALP 2000*, vol. 1853 of *LNCS*, pages 451–462. Springer, Berlin Heidelberg, 2000. (pp. 2 and 10.)

[Sha79] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979. (p. 18.)

[Sho06] A. Shokrollahi. Raptor Codes. *IEEE/ACM Trans. Netw.*, 14(SI):2551–2567, 2006. (p. 18.)

[Sim98] D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In K. Nyberg, editor, *EUROCRYPT 1998*, vol. 1403 of *LNCS*, pages 334–345. Springer, Berlin Heidelberg, 1998. (p. 6.)

[SS11] A. Shelat and C.-h. Shen. Two-Output Secure Computation with Malicious Adversaries. In K. Paterson, editor, *EUROCRYPT 2011*, vol. 6632 of *LNCS*, pages 386–405. Springer-Verlag, Berlin Heidelberg, 2011. Also at ia.cr/2011/533. (p. 16.)

[VZ12] S. Vadhan and C. J. Zheng. Characterizing Pseudoentropy and Simplifying Pseudorandom Generator Constructions. In *STOC 2012*, pages 817–836, New York, NY, USA, 2012. ACM. (p. 6.)