Very-efficient simulatable flipping of many coins into a well

(and a new very-efficient extractable-and-equivocable commitment scheme)

Luís T. A. N. Brandão*

(Technical report – June 28, 2015)

Abstract. *Secure two-party parallel coin-flipping* is a cryptographic functionality that allows two mutually distrustful parties to agree on a common random bit-string of a certain *target length*. In coin-flipping *into-a-well*, one party learns the bit-string and then decides whether to abort or to allow the other party to learn it. It is well known that this functionality can be securely achieved in the ideal/real simulation paradigm, using commitment schemes that are simultaneously *extractable* (X) and *equivocable* (Q). This paper presents two new constant-round simulatable coin-flipping protocols, based explicitly on one or a few X-commitments of short seeds and a Q-commitment of a short hash, independently of the large target length. A *pseudo-random generator* and a *collision-resistant hash function* are used to combine the separate X and Q properties (associated with short bit-strings) into a unified X&Q property amplified to the target length, thus amortizing the cost of the base commitments. In this way, the new protocols are significantly more efficient than an obvious batching or extension of coin-flippings designed (in the same security setting) for short bit-strings and based on inefficient X&Q commitments. The first protocol, simulatable with rewinding, deviates from the traditional coin-flipping template in order to improve simulatability in case of unknown adversarial probabilities of abort, without having to use a X&Q commitment scheme. The second protocol, one-pass simulatable, derives from a new construction of a universally composable X&Q commitment scheme for large bit-strings, achieving communication-rate asymptotically close to 1. Besides the base X and Q commitments, the new commitment scheme only requires corresponding collision-resistant hashing, pseudo-random generation and application of a threshold erasure code. Alternative constructions found in recent work with comparable communication complexity require explicit use of oblivious transfer and use different encodings of the committed value.

Keywords: coin-flipping, commitments, simulatability, extractability, equivocability, rewinding, universal composability, efficient protocols.

* Ph.D. student at University of Lisbon and Carnegie Mellon University. Contact email: luis.papers@gmail.com

Index

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. Leaving implicit the target length and other elements needed for possible setup conditions, e.g., a short *common reference string*, both parties start with a nil input and then interact to obtain the same bit-string as output, uniformly random across different executions. It is intuitive that the output bit-string must be obtained as a *combination* of independent *contributions* from both parties, in a way that any party is able to induce the necessary randomness, even if the other party is malicious. In particular, each party must prevent the other from distinguishing the final bit-string from one that would have been obtained from a uniform random selection.

In a further sophistication, a coin-flipping protocol 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 provide the random bit-string directly 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* [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 deciding a random bit, or bit-string, enabling parties to make fair decisions (e.g., "who gets the car" [Blu83]). A more-technical motivation is the security enhancement of larger cryptographic protocols. A fundamental application, requiring simulatability, may be the joint decision of a large *common reference string* needed as setup condition of one or several follow-up protocols [CR03]. Another general application, within larger constant-round *secure two-party computation* (S2PC) protocols, may be to decide random bit-strings that have an explicit representation or influence in the final (honest) output of an outer protocol. A practical example is *S2PC with bit-commitments* (e.g., [Bra13]), where both parties need to jointly decide as many random bits as the number of bit-commitments multiplied by the size of bit-commitments.

1.1 Coin-flipping and primitives

Commitment schemes and the* traditional *coin-flipping template. 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 (say, P_A) to *commit* her own contribution before knowing anything about the contribution of the other party (say, P_B), but *hiding* it until the contribution of P_B is revealed, and *binding* P_A to only being able to *open* the committed value. The solution, designed to emulate a coin-flipping into a well,[1] sets the basis for what is hereafter denoted as the *traditional template*:

[1] The expression "into a well" (hereafter implicit) appears in the title of an unpublished manuscript [BM81] cited by Blum [Blu83]. It alludes to a party flipping a coin into a well, such that another party close to the well is able to see the outcome of the coin-flip, but not change it, whereas the sender who is far away from the well cannot see the result (explanation based on [Sal96]).

- **Step 1.** In a *commit* phase, P_A commits to her contribution, while hiding it from P_B.
- **Step 2.** P_B selects and sends his random contribution to P_A.
- **Step 3.** In an *open* phase, P_A reveals her contribution to P_B in a convincing way.
- **Step 4.** Each party computes the final output as a combination of both contributions.

The simulatability of a protocol within this template depends 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 using commitment schemes with special simulatability properties, such as *extractability* (X) and *equivocability* (Q). In a X 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 a Q 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 is endowed with extra power, such as capability to rewind the other party in the simulated execution, or knowledge of some secret information (a trapdoor) obtained from some specially selected setup.

Traditionally, achieving simultaneous X and Q (i.e., X&Q) 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 commitment schemes based on computationally expensive operations (e.g., exponentiations) associated with the target length [CF01, BCPV13]. The high cost noticed in traditional approaches serves as motivation in this paper to explore improvements, following two approaches: (i) deviate from the traditional template to find a new structure that requires less sophisticated commitments (i.e., not necessarily X&Q); (ii) devise a more efficient X&Q commitment scheme that can be directly used within the traditional template.

***Very-efficient primitives* (PRG and CR-Hash).** This paper is interested in flipping of *many* coins (e.g., a bit-string with target length larger than a million), in contrast with a number small enough (e.g., up to 256) to fit into a (relatively inefficient) X&Q commitment of a group-element of practical size. Thus, it is pertinent to ask how very-efficient primitives can be used to increase the computational throughput and reduce the communication cost of parallel flipping of many coins, improving the efficiency per flipped coin. The suggestive expression *many coins* denotes a *target length* large enough to make relevant the communication and computational amortization of the expensive operations. Intuitively, a solution may involve applying the very-efficient primitives to strings with the large target length and leave other expensive operations (e.g, commitments with special simulatability properties) to only a few short strings.

This paper devises solutions with the aid of 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). The underlying practical assumption is that there are constructions, directly assumed to be cryptographically secure, such that, for sufficiently large target length, the respective *generation* and *hashing* are much more efficient than an available base of X&Q commitment scheme for bit-strings with similar length. Indeed, there are standardized highly-efficient PRGs [Nat14b] (e.g., based on block or stream ciphers) and CR-Hash functions [Nat14a], allowing a throughput in the order of gigabits per second using current commodity hardware and software.

The advantage of using these two primitives is hinted by the following initial intuition, here denoted *expand-mask-hash*: the *extractability* of a large string can be reduced to the *extractability* (X) of one short seed, whose PRG-expansion is used to mask (with a one-time-pad) the large string; the *equivocability* of a large string can be reduced to the *equivocability* (Q) of a short hash of whatever large string (e.g., the mask) the simulator wants to equivocate. However, a simple concatenation of these elements (a X commitment of the *seed* of a mask, a masked message and a Q commitment of a *hash* of the mask) would not produce a X&Q commitment (as it would not preserve equivocability). This paper devises two ways in which to very-efficiently combine the two properties within a coin-flipping protocol.

1.2 Contributions

This paper introduces a new *expand-mask-hash* approach for parallel coin-flipping, harnessing the "power" of a PRG and a CR-Hash, to improve the throughput of flipping *many coins* (i.e., bit-strings with a large target length). The approach relies on base commitment schemes with separate extractable (X) and equivocable (Q) properties, applied only to a few short strings independently of the polynomial target length. This shows that X and Q properties in separate and in a few short strings can be extended to enable a unified X&Q property (i.e., combining X and Q) in a very long string, while at the same time significantly amortizing the computational cost of the base commitments and asymptotically achieving optimal communication complexity. More specifically, two new *constant-round simulatable* protocols are devised for *two-party parallel coin-flipping*, in two different simulation settings: *with rewinding* and *one-pass*, respectively. The protocols 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 (in the first protocol, the simulator in the ideal world is allowed expected-polynomial time).

Protocol #1. The first protocol (§4) devised in this paper is designed to be simulatable-with-rewinding. It deviates from the traditional template, in order to avoid a simulatability difficulty (related with unknown adversarial probabilities of abort) found in Blum's protocol [Blu83], due to the use of a Q-but-not-X commitment scheme, and at the same time avoid large complexity: it does not require explicit ZK proof/argument sub-protocols[2] about a committed long-contribution, as required in Lindell's protocol

[2] ZK sub-protocols may still be implicitly used in the new protocol to instantiate the underlying commitment schemes (used for a short seed and a short hash). However, they would not be used

[Lin03]; it does not incur a high communication cost, as incurred in Pass and Wee protocol [PW09]. A key aspect of the new protocol is that it does not require augmenting the base commitment schemes to become full-fledged X&Q, and instead it achieves these properties (for the coin-flipping) based on a notion of *non-local extraction and equivocation*. Specifically, extraction of the contribution of P_B and equivocation of the contribution of P_A are possible via black-box rewinding of P_B outside of the respective *commit* and *open* phases. 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 associated with the target length.

The protocol and its proof are defined in a hybrid model with access to separate ideal functionalities of a X commitment scheme and a Q commitment scheme. The Appendix shows a very-efficient concrete instantiation in the plain model, assuming intractability of the decision Diffie-Hellman problem (§D.1), requiring only five exponentiations per party in a setup phase, and four exponentiations in the online phase (or six exponentiations, considering practical parameters where an exponent only fits half of a hash length). This is an improvement in comparison with protocols that require a linear number of exponentiations (or exponentiations in a group with elements of size at least as large as the target length). The exponentiations can be avoided altogether, using solely PRG-based commitments and achieving X and Q based on rewinding. This is exemplified with concrete instantiations of X and Q commitments based only on regular bit-commitments (§D.3 and §D.4), at the cost of extra communication rounds and larger concrete communication complexity – though still amortized as the target length grows.

Protocol #2. The second protocol (§5) devised in this paper is designed for one-pass simulatability. It follows the traditional template of coin-flipping, with a commitment used only for the contribution of P_A. Here, the innovative technical contribution is a new X&Q UC commitment scheme for large strings. The new construction is based on a *cut-and-choose* method, where the size of each instance in the cut-and-choose is reduced proportionally to the number of instances. This achieves communication rate as close as desired to 1, i.e., the number of bits transmitted in each phase (*commit* and *open*) being *roughly* the same as (i.e., just slightly larger than) the number of committed bits – asymptotically, this means *three bits per flipped coin* in the traditional template.

The scheme is defined in a hybrid model using separate ideal functionalities for a X commitment scheme and a Q commitment scheme. Even though UC commitments schemes are simultaneously X and Q, security can be proven with a simulator that, for each type of base commitment scheme (X or Q), does not use the complementary property (Q or X, respectively). This means that a X&Q UC commitment scheme for long bit-strings can be implemented based on commitment schemes for short strings and with suppressed properties, e.g., X&$\overline{\text{Q}}$ (X-but-not-Q) and $\overline{\text{X}}$&Q (not-X-but-Q). Naturally, the protocol remains secure if the underlying schemes are indeed full-fledged X&Q – in such case the protocol represents a UC commitment extension, where a few (*commit* and *open*) calls to X&Q commitments for short bit-strings achieve a X&Q commitment

to directly extract and equivocate the *long* contribution of a party, but rather to enable extraction or equivocation of each *short* string.

(*commit* and *open*) for a string of independently larger polynomial size (and assuming the existence of a PRG and a CR-Hash).

In comparison, very recent and independently developed works [GIKW14, DDGN14, CDD+15], using a different approach, also focus on committing many bits in parallel, amortizing the cost to the throughput of a PRG and capable of achieving rate-1 asymptotic communication complexity. One noteworthy difference with the protocol devised in this paper is that they explicitly use oblivious transfer, instead of other base commitments and a CR-Hash.[3] Also, [GIKW14, DDGN14] rely on *secret sharing schemes* with error-correction or verifiability requirements ([CDD+15] works with any linar code), whereas this paper uses a simpler erasure code (i.e., an *information dispersal scheme* where reconstruction only uses correct fragments). In addition, [GIKW14] considers selective openings and [DDGN14, CDD+15] considers homomorphic properties and verification of linear relations between committed values. [CDD+15] also achieves linear computational complexity, which may be comparable with this work.

Remark. While the main focus of the paper is achieving very-efficient protocols, the analysis is also relevant by the discovery of new approaches and relations between properties, such as: using a new coin-flipping template structure, different from the traditional template; understanding the distinctive role of rewinding for some simulatability properties, clarified by the contrast made between definitions of *local* vs. *non-local* X and Q (in a rewinding setting); achieving a rate-1 UC commitment scheme, resolving the duality between X and Q via a cut-and-choose approach without explicit use of OT; exploring the concept of ideal commitment schemes with suppressed properties (X-but-not-Q and Q-but-not-X). While the direct applicability of commitment schemes might be broader than coin-flipping, this paper justifiably focuses on coin-flipping as an application that better contextualizes the sequence of contributions. In particular, the reflection is made in a sequence of simulation settings (first *with-rewinding*, then *one-pass*), with the goal of reducing the need of *rewinding* in simulation of coin-flipping.

1.3 Roadmap

The paper proceeds as follows: Section 2 reviews related work; Section 3 introduces preliminary definitions useful in the remainder of the paper; Section 4 describes the new protocol for coin-flipping simulatable-with-rewinding; Section 5 specifies the new UC commitment scheme (X&Q), directly usable for *one-pass simulatable* coin-flipping; the Appendix includes formal definitions of the ideal functionalities of commitment schemes and coin-flipping (§A, §B), concrete instantiations for the coin-flipping protocol #1 (§C, §D), and further details about the new UC commitment scheme (§E, §F).

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

[3] "Explicit use" is here meant in the sense of being used as a primitive in the protocol description. This paper does not explore further which primitives imply which in a theoretical sense.

[NY89, Rom90], some types of commitment schemes [Nao91, DCO99] and ZK proofs of knowledge (ZKPoK) [FS90]. Collision-resistant 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 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 X&Q commitment schemes (e.g., [CF01, DN02, Cre03]) and even from coin-flipping protocols with weaker properties [HMQU06, LN11].

In practice, parallel flipping of a few coins (i.e., a short bit-string) can be achieved somewhat efficiently from commitment schemes based on specific algebraic instantiations of standard cryptographic assumptions, such as the intractability of the *decision Diffie-Hellman* problem (e.g., [Lin11, BCPV13]). Naturally, *flipping many coins* can be devised by batching several executions of a protocol for flipping a short bit-string (a *block*), thus achieving a throughput linear in the total number of coins, but still inheriting the inefficiencies of the underlying repeated protocol. This is not satisfactory when the inefficiency of the base commitment scheme multiplied by the number of *blocks* leads to a cost larger than what can be supported, namely computation proportional to a factor super-linear in the block size (e.g., from exponentiations with respective modulus size) and communication complexity with a constant multiplicative factor larger than one (e.g., from communicating several group elements for each flipped *block*). In contrast, this paper devises a construction where the cost of base commitments is amortized.

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 (assuming enhanced trapdoor permutations with dense public descriptions), 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 X and Q) and has the motivation of improving efficiency. The paper does not delve into analyzing further implications between different primitives (e.g., see [DNO10] for relations between OT and commitments, under several setup assumptions).

2.2 Constant-round parallel coin-flipping (with rewinding)

When flipping many coins, the traditional template enables simulatability if the base commitment scheme is X&Q. Lindell achieved this (in two variant protocols [Lin03, §5.3 & §6])[4] by augmenting the *commit* and *open* phases with ZK sub-protocols that endow the phases with respective X and Q properties. In particular, a X-*commit* phase (step 1) can be achieved by a regular commitment followed by a ZK argument of knowledge of the committed value, from which the simulator in the role of receiver

[4] Actually, the two protocols devised by Lindell solve a more general functionality: P_A receives a bit-string as outcome of the coin-flipping but P_B receives the result of applying a known function to such bit-string. In the case of the identity function, the functionality collapses to the regular coin-flipping considered in this paper.

can extract the value. Then, a Q-*open* phase (step 3) can be achieved by 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 is a valid decommitment. This 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.

A different approach, followed by Pass and Wee [PW09], is to use a cut-and-choose approach (in a rewinding setting) to achieve X and Q properties directly from a regular commitment scheme (or from one with only X or only Q properties). The goal of their construction is not efficiency, but rather showing that one-way functions are sufficient to achieve X&Q commitment schemes and consequently also coin-flipping (based on the traditional template protocol). In particular, their technique requires communication proportional to the target length multiplied by the statistical security parameter of the cut-and-choose approach.

In contrast with the two referred constructions in a simulation-with-rewinding setting, a main goal in this paper is to improve efficiency when flipping many coins. This is achieved by focusing on X and Q properties of short strings, and then using a PRG and hash functions to combine and expand them into a larger bit-string, thus amortizing the cost of the underlying commitment schemes.

2.3 UC commitment schemes

Within the scope of the traditional template, simultaneous X and Q properties of the underlying commitment scheme are essential if the simulation setting does not allow rewinding [CR03], e.g., in the UC framework, even for a single-coin flip. 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*, also X&Q, 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 X&Q commitment schemes for individual bits, in the public random string model (suitable to UC). One construction is based on Q commitment schemes and NIZKs, the other is based on one X and one Q commitment schemes. Damgård and Lunemann [DL09] consider UC in a quantum setting and solve the problem of flipping a single bit, based on UC-commitments 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 distinguish several flavors of coin-flipping security (uncontrollable, random and enforceable) and then amplify security from a weaker notion up to the stronger notion – simulatability (on both sides). In a recent line of work, several "highly" efficient commitment schemes have been proposed, based

on specific assumptions, such as the Decision Diffie-Helman (DDH) intractability, e.g., [Lin11, FLM11, BCPV13] achieving a *low* (constant) number of group elements of communication and exponentiations to commit to a group element. Still, the trivial extension of these protocols for a large number of commitments would imply a linear increase in said number of asymmetric operations (modular exponentiations), without amortization. Some of the above-mentioned proposals achieve adaptive security (with erasures), whereas this paper is focused only on static security.

This paper is focused on achieving high efficiency when flipping (and committing) many coins, making use of base X or Q commitments and erasure codes – as already mentioned, three recent independent works [GIKW14, DDGN14, CDD+15] devise UC commitment schemes with comparable asymptotic rate-1 communication, but following different approaches (namely explicitly using oblivious transfer and encoding the committed value in different ways). Future work may be useful to identify concrete tradeoffs between different techniques.

3 Preliminaries

3.1 Ideal/real simulation paradigm

This paper considers the ideal/real simulation paradigm, as developed in the work of Canetti on composability of protocols [Can00, Can01]. A basis to understand the ideal functionalities of commitment schemes, namely in the UC framework, can also be found in the work of Canetti et al. [CF01, CLOS02]. A protocol in the real world is considered secure if it *emulates* an intended ideal functionality defined in an ideal world. In the case of parallel coin-flipping (into a well), the ideal functionality[5] $\mathcal{F}_{\text{MCF}}$ decides the random coin-flipping outcome (the *target bit-string*), then sends it to P_A and only then (if P_A allows it) to P_B, as suggested by the traditional template. Formally, the concept of emulation is equivalent to the inability of an external observer (denoted environment $\mathcal{Z}$) to distinguish executions in the ideal world from those in the real world, when the adversary in the ideal world is replaced by a suitable simulator $\mathcal{S}$ that has black-box access to the adversary $\mathcal{A}$ of the real world.

This paper considers a *static* and *active* corruption model, where an adversary can corrupt at most one party (i.e., make it *malicious*) only before the beginning of the protocol execution.[6] The protocol is said to be simulatable if the proof of emulation can be carried out for each of the two possible corruptions. The protocols are proven secure in the *computational* model, with all parties being restricted to probabilistic polynomial time computations. In the protocol for simulatable-with-rewinding, $\mathcal{S}$ is allowed expected probabilistic polynomial time.

Security is proven by constructively defining a suitable simulator $\mathcal{S}$, who performs indeed a simulation in order to learn how the adversary $\mathcal{A}$ in the real world would behave.

[5] The notation "MCF" in $\mathcal{F}_{\text{MCF}}$ denotes "multiple coin-flipping", meaning that the functionality can be invoked multiple times, each time providing a random bit-string.

[6] This is in contrast to an *adaptive* model, where the *adversary* would be able to choose during the execution of the protocol which party to corrupt. The term *malicious party* is used to denote a party under the control of the adversary.

In the case of coin-flipping, the essential intuition is that $\mathcal{S}$ needs to be able to induce the final coin-flipping result of a simulated execution (as received by $\mathcal{A}$) to be the value that $\mathcal{F}_{\text{MCF}}$ has decided in the ideal world (the *target bit-string*). This is necessary so that $\mathcal{S}$ can learn what $\mathcal{A}$ would output (or possibly abort) when facing the perspective of obtaining such target bit-string as the result of a real protocol. This is the reason to require from $\mathcal{S}$ the ability to either *extract* or *equivocate* the contribution of a party in a simulation. Furthermore, since an abort is also a possible outcome of a protocol, $\mathcal{S}$ must also be able to emulate said probability in the ideal world, namely the case of early-abort, where P_A aborts after learning the coin-flipping outcome but before P_B learns it.

Definition 1 (**needed complementary contribution**). *When $\mathcal{S}$ has obtained in the ideal world the* target bit-string *from $\mathcal{F}_{MCF}$, and has already obtained in the simulated execution (possibly by means of* extraction*) the contribution of the malicious party, the* needed complementary contribution *denotes the contribution that $\mathcal{S}$, in the role of the other (honest) party in the simulated execution, must present (possibly by means of* equivocation*) to the malicious party, so that the outcome of the simulated execution (in case there is no abort) is the same as the target bit-string decided by $\mathcal{F}_{MCF}$.*

Definition 2 (**emulate an abort**). *To* emulate an abort *means that $\mathcal{S}$ aborts the execution in the ideal world (e.g., by sending an* abort *message to $\mathcal{F}_{MCF}$) and then outputs in the ideal world whatever the black-box adversary outputs in the simulated execution.*

3.2 Commitment schemes

A commitment scheme is a two-phase protocol played between two parties, a *sender* and a *receiver*. In a *commit* phase, the parties interact in a way that the sender becomes *bound* to a value, while *hiding* it from the receiver. In an *open* phase, the parties interact in a way that the *receiver* learns the committed value and becomes convinced that it is the correct value. A phase is called non-interactive if the communication consists on a single message from the *sender* to the *receiver*.

Definition 3 (**extractability**). *An* extractable *commitment scheme $\mathscr{C}_X$ is one whose* commit *phase allows $\mathcal{S}$ in the role of* receiver *to* extract *(i.e., learn) the committed value, with probability equal to or larger than a value negligibly-close to the probability with which the (possibly malicious)* sender *is able to successfully* open *the value in the respective* open *phase.*

Definition 4 (**equivocability**). *An* equivocable *commitment scheme $\mathscr{C}_Q$ is one whose* open *phase allows $\mathcal{S}$ in the role of* sender *to* equivocate *the opening to any desired value (in the domain of committable values), regardless of the value that had been committed, such that the view of the execution by the (possibly malicious)* receiver *is indistinguishable from one where the* commit *and* open *phase had been honestly associated with such equivocated value.*

The proofs of security presented in this paper consider a ($\mathcal{F}_\text{X}$,$\mathcal{F}_\text{Q}$)-hybrid model, where the two underlying commitments schemes of the protocol, i.e., the extractable

scheme ($\mathscr{C}_{\mathrm{X}}$) and the equivocable scheme ($\mathscr{C}_{\mathrm{Q}}$), are replaced by ideal commitment functionalities from which the simulator does not take advantage of Q or X, respectively.[7]

3.3 Coin-flipping

For the purpose of a coin-flipping protocol simulatable-with-rewinding, extracting or equivocating the contribution of a party might be achievable (e.g., with the help of rewinding) even if commitment scheme(s) used as primitive(s) is(are) not on its own X or Q, respectively. Thus, it is useful to introduce a characterization (locality vs. non-locality), of the type of X and Q, in order to allow differentiating the context in which the respective *extraction* and *equivocation* can be achieved. This characterization will facilitate conveying the intuition for the new structure of protocol #1, in comparison with the traditional template.

Definition 5 (**locality of X and Q**). *Within a protocol making use of commitments, namely including both* commit *and* open *phases,* extraction *is characterized as* local *if the simulator 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).*

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

This section devises a new parallel coin-flipping protocol, simulatable-with-rewinding.

4.1 Intuition

In the new (constant round) protocol, the malicious $\mathrm{P}_{\mathrm{A}}^{*}$ is still the first party to learn the final bit-string. However, in fundamental contrast with Blum's protocol (which uses a Q-but-not-X commitment scheme in the traditional template), in the new protocol a simulator in the role of P_{B} in the simulated execution can extract the contribution of $\mathrm{P}_{\mathrm{A}}^{*}$ while still semantically-hiding from $\mathrm{P}_{\mathrm{A}}^{*}$ (even in the execution before any *explicit rewinding*) the contribution of P_{B}. Intuitively, this prevents the probability of an *unfair-abort* of $\mathrm{P}_{\mathrm{A}}^{*}$ from depending on the contribution of P_{B} and the final bit-string. Furthermore, in contrast with the protocols of Lindell [Lin03] and Pass and Wee [PW09], where P_{A} commits her contribution using a X&Q commitment, in the new protocol the extraction of the contribution of P_{B} is achieved even though P_{A} uses a Q-but-not-X commitment scheme. In particular, the new protocol achieves simulatability based on *non-local extraction* of the contribution of one party (P_{B}) and *non-local equivocation* of the other (P_{A}).

The protocol uses a new template that requires only one X-commitment of a short seed by P_{A} and one Q-commitment of a hash by P_{B}, plus a PRG (to expand the seed) and

[7] Appendix B formalizes a *nested hybrid model*, technically preventing the simulator from making use of said capabilities. This helps conceptualizing the notion of Q-but-not-X and X-but-not-Q commitment schemes, i.e., ideal commitment schemes with suppressed properties.

a collision-resistant hash function (to compress the contribution of P_B). Thus, as the target length grows the throughput is amortized to that of one PRG and one hash function. The devised solution corresponds to an augmentation of the traditional template, involving the combination of three key aspects: (1) *extraction* of the contribution of P_A is obtained by extraction of a small seed, which once expanded serves as mask of the full contribution – this reduces the input size of the base X-commitment; (2) *extraction* of the contribution of P_B is obtained non-locally (i.e., far away from the respective *commit* phase), but before P_B learns anything about the contribution of P_A in any rewinding attempt – this bypasses the simulatability difficult found in Blum's protocol; (3) *equivocation* of the contribution of P_B is applied to the hash of the contribution, rather than to the full contribution – this reduces the input size of the base Q-commitment. Despite the simplicity of the new protocol, the proof of security is challenging for the case of corrupted P_B^*.

4.2 Detailed description (protocol #1)

The protocol is specified using succinct notation in Fig. 1. As implicit parameters, the protocol depends on a computational security parameter (1) and a respectively secure PRG and CR-Hash function (2). The protocol is here defined and proven secure in a hybrid model where a X commitment scheme ($\mathscr{C}_X$) and a Q commitment scheme ($\mathscr{C}_Q$) (3) are replaced by respective ideal functionalities $\mathcal{F}_X$ and $\mathcal{F}_Q$ with respective X and Q properties. In the *plain* model (§D), $\mathscr{C}_X$ and $\mathscr{C}_Q$ would be agreed between the two parties in a *setup* phase, requiring that $\mathscr{C}_X$ is *non-malleable with respect to opening* of $\mathscr{C}_Q$.

The execution starts when both parties are activated to initiate a coin-flipping of a certain *target length*, with the parties assuming asymmetric roles in the protocol, e.g., w.l.o.g., with P_A being the first to learn the final outcome ((4)-(5)). After a possible implicit setup phase, P_B selects his contribution (6), with the target length, and *commits* to its hash (7) using $\mathcal{F}_Q$ (8). Then, P_A selects a seed (9) and *commits* to it using $\mathcal{F}_X$ (10). P_A also selects a random bit-string (denoted *masked contribution*) with the target length (11) and sends it to P_B (12). Then, P_B asks $\mathcal{F}_Q$ to *open* to P_A the committed hash (13), and then P_B sends his contribution to P_A (14). P_A checks that the hash of the contribution of P_B is equal to the *opened* hash (15). Then, P_A asks $\mathcal{F}_X$ to *open* to P_B the committed seed (16). Finally, each party proceeds concurrently with local computations: expanding the seed of P_A into a bit-string of the target length (17) (i.e., the *mask*); computing the *bit-wise exclusive-OR* (XOR) combination of the *mask* and the *masked contribution*, thus determining the contribution of P_A (18); and locally computing the final outcome as the XOR of the two contributions (19), and deciding that as the final output (20)-(21).

Appendix D describes efficient concrete instantiations in the plain model. One requires an unitary number of exponentiations in a short group; the other does not need any exponentiation but it requires a slightly larger communication complexity (also amortizable) and one commitment scheme becomes explicitly interactive.

4.3 Security analysis

Theorem 1 (security of protocol #1). *Assuming a cryptographically secure PRG and CR-Hash, the coin-flipping protocol #1 securely-emulates (with computational indistinguishability) the ideal functionality of* long *bit-string coin-flipping between two-parties,*

Implicit parameters.

Security parameters: 1^{κ} (1)

Primitives: $(\text{PRG}, \kappa_{\text{PRG}})$, CR-Hash (2)

Sub-protocols: $(\mathscr{C}_{\text{X}}, \mathscr{C}_{\text{Q}})$ (3)

0. Initial input.

$input_A \rightarrow \text{P}_\text{A} : (\texttt{start-1}, sid, cfid, \text{P}_\text{A}, \text{P}_\text{B}, \ell)$ (4)

$input_B \rightarrow \text{P}_\text{B} : (\texttt{start-2}, sid, cfid, \text{P}_\text{A}, \text{P}_\text{B}, \ell)$ (5)

1. Commit contribution of P_B.

$\text{P}_\text{B} : \chi_B \leftarrow^{\$} \{0,1\}^{\ell}$ (select contribution of P_B) (6)

$\text{P}_\text{B} : h_B = \text{CR-Hash}(\chi_B)$ (hash of contribution) (7)

$\mathscr{C}^{\text{Commit}}_{\text{Q},sid,cfid}\left[\text{P}_\text{A} \leftarrow \overline{h_B}; \text{P}_\text{B}(h_B) \leftarrow (\underline{h_B}, \overline{h_B})\right]$ (8)

2. Commit contribution of P_A.

$\text{P}_\text{A} : s_A \leftarrow^{\$} \{0,1\}^{\kappa_{\text{PRG}}}$ (seed) (9)

$\mathscr{C}^{\text{X-Commit}}_{\text{X},sid,cfid}\left[\text{P}_\text{A}(s_A) \leftarrow (\underline{s_A}, \overline{s_A}); \text{P}_\text{B} \leftarrow \overline{s_A}\right]$ (10)

$\text{P}_\text{A} : t_A \leftarrow^{\$} \{0,1\}^{\ell}$ (masked contribution) (11)

$\text{P}_\text{A} \rightarrow \text{P}_\text{B} : (\texttt{masking-1}, sid, cfid, \text{P}_\text{A}, \text{P}_\text{B}, t_A)$ (12)

3. Reveal contribution of P_B.

$\mathscr{C}^{\text{Q-Open}}_{\text{Q},sid,cfid}\left(\overline{h_B}\right)\left[\text{P}_\text{A} \leftarrow h_B; \text{P}_\text{B}(h_B, \underline{h_B})\right]$ (13)

$\text{P}_\text{B} \rightarrow \text{P}_\text{A} : (\texttt{send-contrib-2}, sid, cfid, \text{P}_\text{A}, \text{P}_\text{B}, \chi_B)$ (14)

P_A : If CR-Hash$(\chi_B) \neq h_B$ then *halt* (15)

4. Open contribution of P_A.

$\mathscr{C}^{\text{Open}}_{\text{X},sid,cfid}\left(\overline{s_A}\right)\left[\text{P}_\text{A}(s_A, \underline{s_A}); \text{P}_\text{B} \leftarrow s_A\right]$ (16)

$\text{P}_\text{A}, \text{P}_\text{B} : s'_A = \text{PRG}[s_A](\ell)$ (seed expansion $\equiv$ *mask*) (17)

$\text{P}_\text{A}, \text{P}_\text{B} : \chi_A = t_A \oplus s'_A$ (contribution of P_A) (18)

5. Final output (locally combine contributions).

$\text{P}_\text{A}, \text{P}_\text{B} : \chi = \chi_A \oplus \chi_B$ (19)

$\text{P}_\text{A} \rightarrow output_A : (\texttt{output-1}, sid, cfid, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (20)

$\text{P}_\text{B} \rightarrow output_B : (\texttt{output-2}, sid, cfid, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (21)

Fig. 1. Protocol #1 – Parallel coin-flipping. 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 $p \in \{A, B\}$); CR-Hash (compressive collision-resistant hash function); PRG$[s](\ell)$ (expansion of seed s, using a pseudo-random generator, into a bit-string of length ℓ); κ_{PRG} (input-seed length sufficient to guarantee security consistent with parameter κ); $\overline{x}$ (commitment of x); $\underline{x}$ (randomness needed to decommit x from $\overline{x}$); X, Q, X&Q (extractable, equivocable, extractable-and-equivocable – properties of commitment schemes, which may be present in some phases); ${\mathscr{C}_t}^p$ (phase p (*Commit* or X-*Commit* or *Open* or Q-*Open*) of a commitment scheme of type t (X or Q or X&Q); ${\mathscr{C}_t}^p(z)\left[\text{P}_\text{A}(x_a) \leftarrow y_a; \text{P}_\text{B}(x_b) \leftarrow y_b\right]$ (execution of phase ${\mathscr{C}_t}^p$, between P_A and P_B, starting with common input z and with respective private inputs x_a and x_b, and ending with respective private outputs y_a and y_b).

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. Furthermore: 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; in the case of a malicious P^*_A, simulation is possible without explicit rewinding; in the case of a malicious P^*_B, simulation is possible with an expected number of explicit rewindings less than two.*

Intuition for proof of security. Proving security (i.e., simulatability) amounts to show a simulator that, with an expected number of rewindings at most polynomial in the security parameter, is able to induce in the ideal world a *global* output whose distribution is indistinguishable from the one in the real world. Specifically, in the role of each party in a simulated execution, $\mathcal{S}$ 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 probability of *early-abort* of the malicious party.

The simulator ($\mathcal{S}$) starts a simulation, with rewindable black-box access to $\mathcal{A}$ (who corrupts a party P^*_p at the onset of the simulated execution). $\mathcal{S}$ relays to $\mathcal{A}$ the input received from $\mathcal{Z}$ in the ideal world, and relays to the simulated P^*_p the input that $\mathcal{Z}$ sends to $\widehat{\text{P}}_p$ in the ideal world.

One-pass simulation (i.e., without explicit*[8] *rewinding), for malicious P^*_A. In the simulated execution, $\mathcal{S}$ in the role of P_B commits to a random hash value (8). The execution

[8] It is acceptable that the underlying commitment schemes are equivocable or extractable based on local rewinding, but that is irrelevant for the proof in the hybrid model with ideal commitments.

proceeds with $\mathcal{S}$ receiving the X-commitment of a seed from P_A^* (10), and extracting the respective seed. Then, once receiving from P_A^* a masked contribution (12), $\mathcal{S}$ combines it with the PRG-expansion of the seed in order to learn the contribution of P_A^* ((17)-(18)). 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 bit-string (i.e., the target outcome). $\mathcal{S}$ determines what is the *needed complementary contribution* for P_B in the simulated execution, namely the XOR between the target outcome and the contribution of P_A^*. $\mathcal{S}$ computes the hash of this complementary contribution (7) and uses its power to equivocate such hash value from the respective Q-commitment (13). Finally, $\mathcal{S}$ also sends the complementary contribution to P_A^* (14). Since $\mathscr{C}_\mathrm{Q}$ is *non-malleable with respect to opening* of $\mathscr{C}_\mathrm{X}$, it follows that P_A^* can only either open the contribution that has been extracted by $\mathcal{S}$, or decide to abort without sending her contribution to P_B (played by $\mathcal{S}$) (16). In case of abort, then $\mathcal{S}$ *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 P_A^* outputs in the simulation.

Simulation with rewinding, for malicious P_B^*.

- **First iteration.** In the simulated execution, $\mathcal{S}$ in the role of an honest P_A proceeds the simulation until receiving the contribution of P_B^* and verifying its hash against the respective opened commitment (15). If P_B^* aborts until this step (including if it reveals a contribution inconsistent with the opened hash), then $\mathcal{S}$ *emulates an abort*. Otherwise, $\mathcal{S}$ learns the contribution of P_B^*.
- **Get target outcome.** $\mathcal{S}$ then gets 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 P_B^*. Then, $\mathcal{S}$ defines for the remainder of the simulation an appropriate exponential upper-bound (`#rw-bound`) of number of rewindings. There is a subtle issue related with probability of early abort: if there is no bound, then this simulator does not guarantee an expected polynomial number of rewindings; if the bound is polynomial, then there is an adversary for which the probability of early abort is distinguishable between the ideal and real worlds (namely if the probability is noticeable and noticeably-smaller than the inverse of `#rw-bound`). An appropriate bound is discussed in Appendix C.3.4
- **Induce target outcome.** Then, $\mathcal{S}$ rewinds until the step of selecting the seed of P_A (9) and resumes the execution by selecting a new random seed and committing to it (10). In the next step of the protocol, instead of selecting a random *masked contribution* (12), $\mathcal{S}$ computes the contribution of P_A as the XOR combination of the *needed complementary contribution* and the PRG-expansion of the seed. If P_B^* does not abort before the step where it reveals her contribution, 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 while P_A is opening her seed (16)). Otherwise, if P_B^* aborts before correctly revealing his contribution, $\mathcal{S}$ rewinds again until the step of selecting the seed of P_A and replays the simulation procedure just described, again and again until receiving the contribution of P_B^* (equal to the one already known by $\mathcal{S}$) and thus leading the simulation to an end, or until reaching the `#rw-bound` bound and in that case it *emulates an abort*.

5 A new UC commitment scheme

This section devises a new UC commitment scheme, i.e., one-pass-simulatable, thus with the X and Q properties being local. To flip many coins, it is then enough to use this scheme to *commit* and *open* the contribution of P_A in the traditional template. The section starts by conveying an intuition (§5.1), then gives a precise description using concise notation (§5.2), and then presents a security analysis (§5.3).

5.1 Intuition in a sequence of optimizations

Besides the primitives already discussed (X and Q commitments, PRG, CR-Hash), the new protocol embeds three main ingredients:

- a **cut-and-choose** approach, where 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 remaining (the *evaluation* instances) are correct;
- **authenticators**, allowing the simulator to anticipate whether individual *instances* are *good* or *bad*, thus having better assurance about correct extraction;
- an ***information dispersal scheme*** (IDS), used to partition (i.e., *split*) the target message into smaller components (i.e., *fragments*), and allowing a portion of those fragments to be combined into (i.e., to *recover*) the original message; the IDS enables the size of each *instance* to be reduced proportionally to the number of instances.

The combination of these elements is described below, in a sequence of progressive optimizations, and is complementary also sketched in Fig. 2.

5.1.1 Cut-and-choose warmup

A non-example. It is instructive to first look at a non-X or non-Q (and thus non-UC) scheme. Consider a *commit* phase as follows: i) P_A X-commits to a seed; ii) calculates a *mask* as the PRG-expansion of the seed; iii) sends the respective message *masking* to P_B; iv) and Q-commits to the hash of the *mask*. Then: if the *open* phase involves opening the seed, then the scheme is X-but-not-Q, because each seed (which not even a simulator can equivocate) leads each masking to a unique message;[9] if, instead, the *open* phase involves revealing the message and opening the hash of the mask, but not opening the seed, then the scheme is not-X, because a malicious P_A^* may have (undetectably to P_B) used a mask different from the seed-expansion. Conversely, suppose that in the latter case it would somehow be possible to ensure that the used mask was indeed the PRG-expansion of the committed-but-not-opened seed. Then the simulator would be able to extract the message – by extracting the seed and then use its PRG-expansion to unmask the *masking*. The method hereafter uses a cut-and-choose approach, based on several instances of commitments of seeds, to obtain the needed assurance in a statistical manner, namely ensuring an overwhelming probability of correct extraction.

[9] Even if the seed had been committed with a X&Q scheme, it would not be possible to equivocate most messages, because there are much less *short* seeds than *large* messages or masks.

A simple but not-efficient cut-and-choose based scheme:

- **Commit phase.** P_A produces several seeds, builds a X-commitment of each, and also builds a Q-commitment 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 in two random complementary subsets, and *chooses* one for a *check* operation and the other for an *evaluation* operation. For each instance selected for *evaluation*, P_A uses the respective PRG-expansion to mask (using XOR) the *target message*, and sends the respective *message masking* to P_B.
- **Open phase.** P_A reveals the message – this allows P_B to compute all masks (i.e., the PRG-expansions) used with evaluation instances. Also, P_A opens all *check* seeds, thus letting P_B compute the respective PRG-expansions. Finally, P_A opens the commitment of the 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 the target message.

This above solution has a high communication complexity: target length multiplied by number of *evaluation* instances. Nonetheless, it has the needed simulatability properties:

- **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 previously sent *masking*.
- **Equivocation.** In the *open* phase, the equivocator-simulator ($\mathcal{S}^Q$) in the role of P_A can successfully 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 PRG-expands it and uses it as a tentative mask to unmask the masked message. This leads $\mathcal{S}^X$ to obtain several unmaskings, all of which are tentative messages. If a majority of tentative messages is consistent (i.e., all of them are equal), then $\mathcal{S}^X$ chooses such message 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.

Statistical security. Above, only extractability depends on a statistical argument. Extraction can still fail if all check instances are good but a majority of the evaluation instances are *bad* – *bad* in the sense that the PRG-expansion of the extractable seed is not the value that was used as a mask. However, when using so many *bad* instances, P_A^* risks having any of them selected for the *check* subset, case in which the verification of the global hash would fail. Thus, a successful verification of the global hash induces a (statistically based) probability that a majority of the committed-but-not-opened seeds of the evaluation instances is also correct. Correspondingly, conditioned on a future successful verification of the global hash, if follows a probability that the majority of the

extracted seeds are correct. With adequate cut-and-choose parameters, this probability is overwhelming in the total number n of instances, i.e., the commitment scheme is extractable. For fixed n, statistical security (the additive symmetric of the logarithm base 2 of the probability of failed extraction) is maximized with the optimal proportions of approximately three fifths for *check* and two fifths for *evaluation* [SS11, §A], leading to about 1 bit of statistical security per every three instances. For example, using 123 instances, 74 of which for *check* and 49 of which for *evaluation* [Bra13, Table 2], leads to approximately (slightly larger than) 40 bits of statistical security, i.e., the probability of wrong extraction is less that two to the minus 40.

5.1.2 Authenticator aid

Improving statistical security. In comparison with the above-described method, statistical security can be improved by including a *verifiability mechanism* that endows $\mathcal{S}^X$ with the ability to verify which evaluation instances are *good* and which ones are *bad*, without comparison between instances. With such mechanism, $\mathcal{S}^X$ 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. For appropriate cut-and-choose parameters, the (un)likelihood of this guess corresponds to about one bit of statistical security per instance, thus reducing the number of instances about three-fold. For example, 40 bits of statistical security can be obtained with 41 or 76 instances, while respectively limiting the number of *evaluation* instances to be at most 20 or 10. The rationale about probabilities is similar to that used in recent improvements obtained for more general secure two-party computation protocols based on a cut-and-choose of garbled-circuits approach, where the criterion for success changed from at least a majority to just at least a single correct evaluation garbled-circuit [Bra13, Lin13, HKE13].

Authenticators. The intended verifiability can be achieved by augmenting each *evaluation* instance with an *authenticator* element 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 masked string received from P_A, only two things should be possible: either (i) $\mathcal{S}^X$ gets a correctly *authenticated* message, which must be the only one that 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 P_A^* will have to use a mask different from the seed-expansion, i.e., this is a *bad* instance.

As a starting point, P_A^* becomes bound to her choices, by committing her seeds and the global hash (which commits the masks), before even learning how to compute *authenticators*. Only then is the authenticator function α defined, dependent on a random unpredictable value (a *nonce*) that P_B discloses to P_A. The function α will relate the message and the nonce in a non-trivial way, e.g., with a suitable collision-resistant type of property, such that it is infeasible for P_A^* to produce a masking (i.e., a masked version of the message) for which two different unmaskings (the one using the seed-expansion as mask; and another using the maliciously arbitrary pre-selected mask) yield authenticated messages. In a *strict* authenticator mode, the seeds, the (global hash of the) masks and

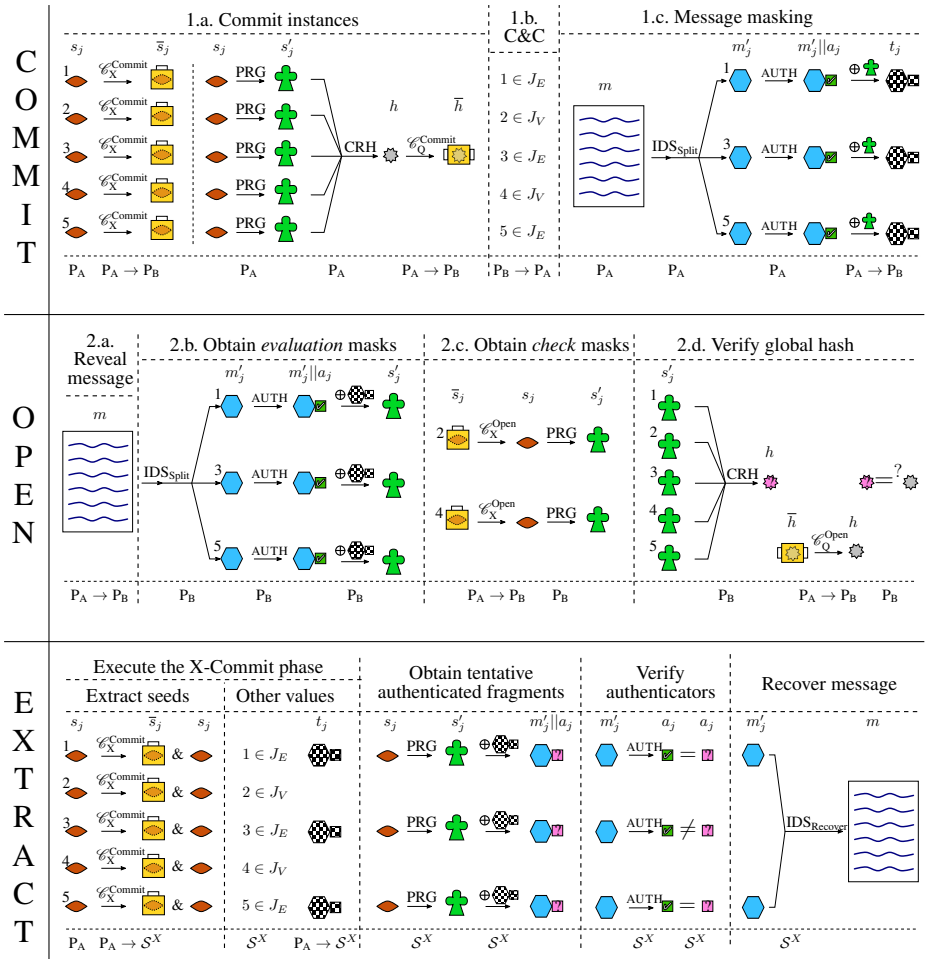

Fig. 2. Sketch of UC commitment scheme. Legend: (seed s_j); (extractable commitment $\overline{s}_j$ – like a vault with a single opening); (seed expansion s'_j – like a tree growing from a seed); (global hash – like a smashed paper); (equivocable commitment $\overline{h}$ – like a vault with several 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); $\mathscr{C}$ (extractable commitment scheme); $\mathscr{C}_{\mathrm{X}}$ (extractable $\mathscr{C}$); $\mathscr{C}_{\mathrm{Q}}$ (equivocable $\mathscr{C}$); CRH (collision-resistant hash); IDS (information dispersal scheme); PRG (pseudo-random generator); $\mathcal{S}^X$ (simulator with extraction goal). This sketch shows 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 IDS to reconstruct the message from only $t = 2$ (the recovery threshold) fragments. Equivocation (not shown) is trivial, as it involves only revealing the equivocated message (2.a), then calculating the intermediate values as a receiver ($\mathrm{P_B}$) would (2.b, 2.c and 2.d), and then simply equivocate the global hash needed for a successful final verification (2.d).

(additionally) even the (hash of the) message are committed before the nonce is disclosed to $\mathrm{P_A}$. In a *loose* authenticator mode, the (hash of the) message does not need to be committed, thus allowing α to be defined still in a setup phase.

The properties and concrete instantiations of α are further analyzed in Appendix E.[10] One *strict-authenticator* function, proposed as an algebraic field-multiplication between the nonce and a CR-hash of the message, is proven secure based solely on a collision resistance of the hash. One *loose-authenticator* function, proposed as the hash of the concatenation of the message and the nonce, is secure if it is assumed to be infeasible to find four pre-images whose respective hashes cancel out when XORed. In spite of the authenticator mechanism, equivocation is still possible by the simulator $\mathcal{S}^Q$ in the role of P_A because the authenticator is also masked and the respective mask is also equivocable.

5.1.3 Communication reduction

Using an IDS. In the above method, the *commit* phase still requires communicating bits in number proportional to the number of *evaluation* instances multiplied by the target length. This can be further reduced, by taking advantage of a threshold *information dispersal scheme* (IDS) [Rab89] to *split* (i.e., disperse) the message into as many fragments as the number of *evaluation* instances, each fragment with a *reduced length*. Then, the original message can be *reconstructed* from proper subsets (of the set of fragments) with at least a certain *threshold* number of elements.[11] In the improved method, the message is initially split into as many fragments as the number of *evaluation* instances. Then, the authenticator scheme is applied to every fragment, instead of only to the full message. Finally, P_A sends to P_B, in the commit phase, the authenticated masked fragments (each with a *reduced length*) instead of several masked versions of the same message (where each would have the original target length). This makes the communication complexity become proportional to the number of *evaluation* instances divided by the IDS *threshold*, instead of just proportional to the number of *evaluation* instances.

The new method 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, thus also obtaining a computational improvement in compensation to the new IDS split operation. Asymptotically, as the target length increases to infinite, the communication expansion-rate can be made as close to 1 as desired. 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. For example, with 119 instances, 46 of which for evaluation, and with an IDS threshold 23, the scheme achieves 40 bits of

[10] Another authenticator mechanism is considered there, based on a X&Q commitment scheme, providing a conceptually simple UC commitment extension; i.e., a protocol where a few UC commitments applied to short strings enable a X&Q commitment of a large string.

[11] The IDS does not need to hide the original message, as would a full-fledged secret-sharing scheme [Sha79, Kra94], because the necessary hiding is already achieved with the (XOR) masking. Also, the IDS does not need to support error-correction [RS60], because the *authenticator* mechanism gives $\mathcal{S}^X$ (in the role of P_B) the ability to detect errors. Yet, the IDS needs to be an *erasure code*, in the sense that it can recover the original contribution from proper subsets of fragments, namely when omitting the fragments that $\mathcal{S}^X$ anticipates as being *bad*. There are sufficiently-efficient erasure codes [Lub02, Sho06], possible to implement based on XOR operations. Interestingly, besides the communication efficiency gain, the *splitting* into fragments of reduced length also reduces the computation required from the PRG and the hash function, as they also need to be applied to an overall shorter length.

statistical security and an asymptotic communication expansion-rate 2 in the *commit* phase. The communication in the open phase has rate 1, as only the committed message needs to be revealed (along with opening several short commitments). Other concrete parametrizations are mentioned in Appendix F (Tables 3 and 5).

5.2 Detailed description (protocol #2)

The protocol is described using succinct notation in Fig. 3. In a setup phase, the parties agree on the security parameters (computational and statistic) and consistently agree on remaining elements: the cut-and-choose parameters (including a fixed number of *check* and *evaluation* instances) (22), a PRG and a CR-Hash function (23), a X commitment scheme ($\mathscr{C}_{\mathrm{X}}$) and a Q commitment scheme ($\mathscr{C}_{\mathrm{Q}}$) (24), a mode of authentication (STRICT or LOOSE) (25) and respective function and parameters (26), and the IDS scheme and respective parameters (27). The proof of security will be made in a hybrid model where the $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ are replaced by respective ideal functionalities $\mathcal{F}_{\mathrm{X}}$ and $\mathcal{F}_{\mathrm{Q}}$.

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

- **1.a. Commit instances.** $\mathrm{P_A}$ selects n random seeds (29) (e.g., 119) and commits individually to each of them using (the X-*commit* phase of) $\mathcal{F}_{\mathrm{X}}$ (30). $\mathrm{P_A}$ 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 IDS recovery-*threshold* t) extended by an *authenticator-length* ℓ_a (31). $\mathrm{P_A}$ calculates the *global hash* h as the CR-hash of the concatenation of all seed-expansions (32). $\mathrm{P_A}$ then uses $\mathcal{F}_{\mathrm{Q}}$ to commit to h (33). If in the STRICT mode, $\mathrm{P_A}$ also uses $\mathcal{F}_{\mathrm{Q}}$ to commit to the hash of the target message m (34)-(35).
- **1.b. Cut-and-choose.** $\mathrm{P_B}$ decides a random cut-and-choose partition (36) (e.g., identifying 73 instances for *check* and 46 for *evaluation*) and a random nonce z (37) of adequate length (ℓ_z) and sends them both to $\mathrm{P_A}$ (38).
- **1.c. Message masking.** $\mathrm{P_A}$ uses the threshold IDS to *split* her message into as many fragments as the number of *evaluation* instances (39), 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 (40); $\mathrm{P_A}$ then uses the extended mask s'_j to compute the masking t_j of the fragment concatenated with the authenticator (41). Finally, $\mathrm{P_A}$ sends to $\mathrm{P_B}$ the maskings associated with all *evaluation* instances (42).

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

- **2.a. Reveal message.** $\mathrm{P_A}$ sends the (previously committed) message m to $\mathrm{P_B}$ (44). 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 (45), and $\mathrm{P_B}$ verifies that it is consistent with the hash of the received message (46).
- **2.b. Obtain *evaluation* masks.** The phase continues regardless of the authenticator mode being STRICT or LOOSE. $\mathrm{P_B}$ uses the IDS to obtain the same fragments that an honest $\mathrm{P_A}$ would (47).[12] $\mathrm{P_B}$ computes the authenticator of the fragment in the

[12] The IDS can be probabilistic, as long as $\mathrm{P_A}$ also sends the random coins used in its previous *split* operation, so that $\mathrm{P_B}$ can deterministically reproduce the fragmentation.

Implicit parameters.

Security parameters: $1^{\kappa}, (1^{\sigma}, n, v, e)$ (22)

Primitives: $(\text{PRG}, \kappa_{\text{PRG}}), \text{CR-Hash}$ (23)

Sub-protocols: $(\mathscr{C}_{\text{X}}, \mathscr{C}_{\text{Q}})$ (24)

$\text{AUTHMODE} \in \{\text{STRICT}, \text{LOOSE}\}$ (25)

Authenticator parameters: $\{\alpha, \ell_a = |\alpha|, \ell_z\}$ (26)

IDS: $(t, \text{IDS}[t]_{\text{split}}, \text{IDS}[t]_{\text{recover}})$ (27)

1. X-Commit phase.

$input_A \rightarrow \text{P}_\text{A} : (\texttt{commit}, sid, cid, \text{P}_\text{A}, \text{P}_\text{B}, m)$ (28)

1.a. Commit instances. For $j \in \{1, ..., n\}$

$\text{P}_\text{A} : s_j \leftarrow^{\$} \{0,1\}^{\kappa_{\text{PRG}}}$ (seed) (29)

$\mathscr{C}^{\text{X-Commit}}_{\text{X},sid,(cid,j)} \left[\text{P}_\text{A}(s_j) \leftarrow \left(\underline{s_j}, \overline{s_j}\right) ; \text{P}_\text{B} \leftarrow \overline{s_j}\right]$ (30)

$\text{P}_\text{A} : s'_j = \text{PRG}[s_j](\lceil |m|/t + \ell_a \rceil)$ (31)

$\text{P}_\text{A} : h = \text{CR-Hash}(||_{j\in[n]} s'_j)$ (global hash) (32)

$\mathscr{C}^{\text{Commit}}_{\text{Q},sid,cid} \left[\text{P}_\text{A}(h) \leftarrow \left(\underline{h}, \overline{h}\right) ; \text{P}_\text{B} \leftarrow \overline{h}\right]$ (33)

If $\text{AUTHMODE} =^{?} \text{STRICT}$, then:

$\text{P}_\text{A} : \eta = \text{CR-Hash}\,(m)$ (hash of message) (34)

$\mathscr{C}^{\text{Commit}}_{\text{Q},sid,(cid,+)} \left[\text{P}_\text{A}(\eta) \leftarrow \left(\underline{\eta}, \overline{\eta}\right) ; \text{P}_\text{B} \leftarrow \overline{\eta}\right]$ (35)

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

$\text{P}_\text{B} : (J_V, J_E) \leftarrow^{\$} \text{Partition}[v, e](n)$ (36)

$\text{P}_\text{B} : z \leftarrow^{\$} \{0,1\}^{\ell_z}$ (nonce) (37)

$\text{P}_\text{B} \rightarrow \text{P}_\text{A} : (\texttt{c\&c}, sid, cid, \text{P}_\text{B}, \text{P}_\text{A}, (J_V, J_E, z))$ (38)

1.c. Message masking.

$\text{P}_\text{A} : \langle m'_j : j \in J_E \rangle \leftarrow \text{IDS}[t]_{\text{split}}\,(m, J_E)$ (39)

$\text{P}_\text{A} : a_j = \alpha(m'_j, z) : j \in J_E$ (authenticators) (40)

$\text{P}_\text{A} : t_j = \left(m'_j || a_j\right) \oplus s'_j : j \in J_E$ (maskings) (41)

$\text{P}_\text{A} \rightarrow \text{P}_\text{B} : (\texttt{maskings}, sid, cid, \text{P}_\text{A}, \text{P}_\text{B}, ||_{j\in J_E} t_j)$ (42)

2. Q-Open phase.

$input_A \rightarrow \text{P}_\text{A} : (\texttt{open}, sid, cid, \text{P}_\text{A}, \text{P}_\text{B})$ (43)

2.a. Reveal message.

$\text{P}_\text{A} \rightarrow \text{P}_\text{B} : (\texttt{open}, sid, cid, \text{P}_\text{A}, \text{P}_\text{B}, m)$ (44)

If $\text{AUTHMODE} =^{?} \text{STRICT}$, then:

$\mathscr{C}^{\text{Q-Open}}_{\text{Q},sid,(cid,+)}(\overline{\eta}) \left[\text{P}_\text{A}(\eta, \underline{\eta}); \text{P}_\text{B} \leftarrow \eta\right]$ (45)

P_B : If CR-Hash$(m) \neq \eta$ then ABORT (46)

2.b. Obtain evaluation maskings.

$\text{P}_\text{B} : \langle m'_j : j \in J_E \rangle \leftarrow \text{IDS}[t]_{\text{split}}\,(m, J_E)$ (47)

$\text{P}_\text{B} : a_j = \alpha\left(m'_j, z\right) : j \in J_E$ (authenticator) (48)

$\text{P}_\text{B} : s'_j = t_j \oplus \left(m'_j || a_j\right) : j \in J_E$ (tentative masks) (49)

2.b. Obtain check maskings.

$\mathscr{C}^{\text{Open}}_{\text{X},sid,(cid,j)}\,(\overline{s_j}) \left[\text{P}_\text{A}\left(s_j, \underline{s_j}\right) ; \text{P}_\text{B} \leftarrow s_j\right] : j \in J_V$ (50)

$\text{P}_\text{B} : s'_j = \text{PRG}[s_j](\lceil |m|/t + \ell_a \rceil) : j \in J_V$ (51)

2.d. Verify global hash.

$\mathscr{C}^{\text{Q-Open}}_{\text{Q},sid,cid}\,(\overline{h})\,[\text{P}_\text{A}\,(h, \underline{h})\,; \text{P}_\text{B} \leftarrow h]$ (52)

P_B : If CR-Hash$(||_{j\in[n]} s'_j) \neq h$ then *halt* (53)

$\text{P}_\text{B} \rightarrow output_B : (\texttt{accept}, sid, cid, \text{P}_\text{A}, \text{P}_\text{B}, m)$ (54)

Fig. 3. Protocol #2 – X&Q bit-string commitment scheme. Legend: legend of Fig. 1 also applies; σ (statistical security parameter, e.g., $40 \equiv 1^{40}$); n, v, e (number of *total* instances, number of *check* instances, number of *evaluation* instances); Partition$[v, e](n)$ (set of possible partitions of the set of the first n positive integers into a pair of complementary subsets, the first with v elements, and the second with the remaining e). IDS$[t]$ (*information dispersion scheme* 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 $((s-b)!e!) \,/\, ((e-b)w!s!) \leq 2^{-\sigma}$, where $b = e - t + 1$ (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).

same way that an honest P_A would have, based on the fragment and the nonce (48). Then, 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 P_A (49).

- **2.c. Obtain *check* masks.** P_A asks $\mathcal{F}_\text{X}$ to *open* to P_B the seeds associated with *check* instances (but not those of *evaluation* instances) (50). P_B computes by himself the PRG-expansion (with the appropriate length) of each *check* seed (51).
- **2.d. Verify global hash.** P_A asks $\mathcal{F}_\text{Q}$ to *open* to P_B the previously committed hash (52). The, P_B verifies that the global hash of all concatenated masks is equal to the one just opened by P_A (53). If some verification has failed, then P_B aborts, otherwise it accepts the message of P_A as a correct *opening* (54).

5.3 Security analysis

Theorem 2 (security of protocol #2). *Assuming a cryptographically secure PRG and CR-Hash, the protocol defined in this section (using an adequate authenticator in the* STRICT *mode) UC-realizes the ideal functionality of* long *bit-string commitment in the ($\mathcal{F}_X$,$\mathcal{F}_Q$)-hybrid model, in the presence of static and computationally active adversaries. Furthermore: for the combined* commitment *and* opening *of a bit-string, with polynomially arbitrarily-long target length, each phase of $\mathcal{F}_Q$ is invoked for a short-string only once, and each phase of $\mathcal{F}_X$ is invoked for short strings only a number of times that is independent of the target length.*

If $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ are directly instantiated by an ideal UC commitment scheme (Fig. 4), respectively limited to messages of length κ_{PRG} and κ_{Hash}, then this becomes a statement about UC commitment length extension (i.e., that a few X&Q-commitments of short strings can be extended to X&Q-commitments of long strings).

Intuition for proof of security. Proving security amounts to show that the new commitment scheme is X&Q; more specifically, that the *commit* phase is extractable (X) and the *open* phase is equivocable (Q). The analysis assumes that the PRG and CR-Hash are cryptographically secure and that $\mathscr{C}_{\mathrm{X}}$ is extractable and $\mathscr{C}_{\mathrm{Q}}$ is equivocable. More precisely, an hybrid model is considered, with the underlying commitment schemes ($\mathscr{C}_{\mathrm{X}}$, $\mathscr{C}_{\mathrm{Q}}$) being replaced by ideal functionalities ($\mathcal{F}_{\mathrm{X}}$, $\mathcal{F}_{\mathrm{Q}}$) with respective X and Q properties, while suppressing the respectively complementary properties (Q and X). The proof of security is accomplished by defining respective simulators:

- For equivocation, in the *open* phase the equivocator-simulator ($\mathcal{S}^Q$, in the role of $\mathrm{P_A}$ in the simulated execution) (i) reveals the intended equivocated message, then (ii) computes the masks that would need to have been used to unmask the previously sent masked fragments into the needed fragments, and finally (iii) equivocates the opening of the needed global hash. This procedure is always successful.
- For extractability, in the *commit* phase the extractor-simulator ($\mathcal{S}^X$, in the role of $\mathrm{P_B}$ in the simulated execution) (i) extracts the seeds committed by $\mathrm{P_A}$, then (ii) uses their PRG-expansions to unmask the masked-fragments sent by $\mathrm{P_A}$, then (iii) uses the *authenticator* mechanism to find which ones are *good*, and then (iv) uses the IDS-recovery procedure to recover the committed message from a set of enough *good* (i.e., authenticated) fragments. This procedure is also depicted in Fig. 2. For suitable cut-and-choose and IDS parameters, a wrong *extraction* by $\mathcal{S}^X$ during the *commit* phase can happen only with probability negligible in the statistical parameter – exemplified configurations are detailed in Appendix F (Table 5).

Acknowledgments. This research was performed while the author was a Ph.D. student at University of Lisbon (FCUL-DI) and Carnegie Mellon University (CMU-CIT-ECE) 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. The author thanks the anonymous referees of CRYPTO 2014, ASIACRYPT 2014 and TCC 2015 conferences for their useful reviewing comments about earlier versions of this paper.

References

[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. See also Cryptology ePrint Archive, Report 2013/123. (Cited on pages 4, 8, and 10.)

[Bea96] D. Beaver. Adaptive Zero Knowledge and Computational Equivocation (Extended Abstract). In *STOC 1996*, pages 629–638. ACM, New York USA, 1996. (Cited on pages 4 and 31.)

[Blu83] M. Blum. Coin flipping by telephone – a protocol for solving impossible problems. *SIGACT News*, 15:23–27, 1983. Appeared also at CRYPTO 1981. (Cited on pages 3, 5, 45, and 48.)

[BM81] M. Blum and S. Micali. Coin-Flipping into a Well. Unpublished, 1981. (Cited on page 3.)

[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. See also Cryptology ePrint Archive, Report 2013/577. (Cited on pages 3 and 18.)

[Can00] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *J. Cryptology*, 13:143–202, 2000. See also Cryptology ePrint Archive, Report 1998/018. (Cited on pages 3, 10, and 28.)

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *FOCS 2001*, pages 136–145, 2001. See also Cryptology ePrint Archive, Report 2000/067. (Cited on pages 3, 10, 28, and 82.)

[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. (Cited on pages 7 and 10.)

[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. See also Cryptology ePrint Archive, Report 2001/055. (Cited on pages 4, 8, 9, 10, and 32.)

[Cle86] R. Cleve. Limits on the Security of Coin Flips when Half the Processors Are Faulty. In *STOC 1986*, pages 364–369. ACM, New York USA, 1986. (Cited on page 47.)

[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. See also Cryptology ePrint Archive, Report 2002/140. (Cited on pages 9, 10, and 30.)

[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. See also Cryptology ePrint Archive, Report 2002/047. (Cited on pages 3 and 9.)

[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. (Cited on pages 8 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. (Cited on page 8.)

[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. (Cited on pages 9 and 61.)

[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. See also Cryptology ePrint Archive, Report 2001/032. (Cited on page 61.)

[DCO99] G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Preprocessing. In M. Wiener, editor, *CRYPTO 1999*, vol. 1666 of *LNCS*, pages 485–502. Springer, Berlin Heidelberg, 1999. (Cited on page 8.)

[DDGN14] I. Damgård, B. David, I. Giacomelli, and J. B. Nielsen. Compact VSS and Efficient Homomorphic UC Commitments. In *ASIACRYPT 2014 – to appear*, LNCS. Springer, Berlin Heidelberg, 2014. See also Cryptology ePrint Archive, Report 2014/370. (Cited on pages 7 and 10.)

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. See also Non-Malleable Cryptography at STOC 1991. (Cited on page 61.)

[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. See also arXiv:0903.3118. (Cited on pages 9, 33, and 53.)

[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. See also Cryptology ePrint Archive, Report 2001/091. (Cited on pages 8, 9, and 32.)

[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. (Cited on page 8.)

[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. (Cited on page 60.)

[FF09] M. Fischlin and R. Fischlin. Efficient Non-malleable Commitment Schemes. *J. Cryptology*, 22(4):530–571, 2009. See also Crypto 2000. (Cited on pages 61 and 62.)

[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. (Cited on page 10.)

[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. (Cited on page 8.)

[FY92] M. Franklin and M. Yung. Communication Complexity of Secure Computation (Extended Abstract). In *STOC 1992*, pages 699–710. ACM, New York USA, 1992. (Cited on page 94.)

[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. (Cited on pages 7, 10, 91, 94, 95, and 96.)

[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. (Cited on page 64.)

[Gol04] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004. ISBN: 0521830842. (Cited on pages 4 and 47.)

[Hal95] S. Halevi. Efficient Commitment Schemes with Bounded Sender and Unbounded Receiver. In D. Coppersmith, editor, *CRYPT0 1995*, vol. 963 of *LNCS*, pages 84–96. Springer, Berlin Heidelberg, 1995. (Cited on page 48.)

[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. (Cited on page 7.)

[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. See also Cryptology ePrint Archive, Report 2013/081. (Cited on page 18.)

[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. See also Cryptology ePrint Archive, Report 2006/177. (Cited on page 8.)

[Kil94] J. Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *FOCS 1994*, pages 466–477, Nov 1994. (Cited on page 75.)

[Kra94] H. Krawczyk. Secret Sharing Made Short. In D. Stinson, editor, *CRYPTO 1993*, vol. 773 of *LNCS*, pages 136–146. Springer, Berlin Heidelberg, 1994. (Cited on page 20.)

[Lin03] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *J. Cryptology*, 16(3):143–184, 2003. See also Cryptology ePrint Archive, Report 2001/107. (Cited on pages 4, 6, 8, 12, and 49.)

[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. See also Cryptology ePrint Archive, Report 2011/180. (Cited on pages 8 and 10.)

[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. See also Cryptology ePrint Archive, Report 2013/079. (Cited on page 18.)

[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. See also Cryptology ePrint Archive, Report 2011/065. (Cited on pages 8 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. (Cited on pages 59 and 60.)

[Lub02] M. Luby. LT codes. In *Proc. 43rd annual IEEE symposium on FOCS 2002.*, pages 271–280, 2002. (Cited on page 20.)

[MNS09] T. Moran, M. Naor, and G. Segev. An Optimally Fair Coin Toss. In O. Reingold, editor, *TCC 2009*, vol. 5444 of *LNCS*, pages 1–18. Springer, Berlin Heidelberg, 2009. (Cited on page 47.)

[Nao91] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. (Cited on pages 8, 59, and 65.)

[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. (Cited on page 5.)

[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. (Cited on page 5.)

[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. (Cited on page 8.)

[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. (Cited on pages 59 and 60.)

[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. See also the *IACR Online Proceedings* for TCC 2009. (Cited on pages 4, 6, 8, 9, 12, 59, 64, 65, and 75.)

[Rab89] M. O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *J. ACM*, 36(2):335–348, 1989. (Cited on page 20.)

[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. (Cited on page 8.)

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the SIAM*, 8(2):300–304, 1960. (Cited on page 20.)

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

[Sal96] A. Salomaa. Cryptographic Protocols: Surprising Vistas for Communication. In *Public-Key Cryptography*, pages 181–244. Springer, Berlin Heidelberg, 1996. (Cited on page 3.)

[Sch91] C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991. See also extended abstract at EUROCRYPT 1989. (Cited on page 59.)

[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. (Cited on pages 4 and 31.)

[Sha79] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, November 1979. (Cited on page 20.)

[Sho06] A. Shokrollahi. Raptor Codes. *IEEE/ACM Trans. Netw.*, 14(SI):2551–2567, 2006. (Cited on page 20.)

[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. (Cited on page 8.)

[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. See also Cryptology ePrint Archive, Report 2011/533. (Cited on page 18.)

[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. (Cited on page 7.)

Appendix

Appendix A reviews background notions about the ideal/real simulation paradigm and describes the ideal functionalities of X&Q commitment schemes and coin-flipping of bit-strings. Appendix B introduces alternative notions of commitment schemes, isolating on purpose either the X or the Q properties. Appendix C analyzes in further detail the coin-flipping protocol simulatable-with-rewinding defined in Section 4. Appendix D considers instantiations in the plain model. Appendix E examines more thoroughly the security of the UC commitment scheme, including the authenticator mechanisms that allow the extractor simulator to anticipate whether a masking instance is good or bad. Appendix F compares protocol adjustments and parametrizations. Appendix G contains an indexed list of Figures and Tables presented throughout the paper.

A Ideal functionalities

This section starts with a high-level review of the ideal/real simulation model (§A.1). Then it defines the ideal functionality for multiple (X&Q) bit-string commitments (§A.2) and for multiple bit-string coin-flippings (§A.3). Finally, it also describes the *traditional* template for coin-flipping, based on a single X&Q commitment scheme (§A.4).

A.1 Background on ideal/real simulation paradigm

Ideal functionality. Within the ideal/real simulation paradigm [Can00, Can01], a real protocol Π is considered secure if it *emulates* a respectively intended *ideal functionality*. The ideal functionality ($\mathcal{F}$) defines the behavior of a *trusted third party* that in an ideal world mediates the communication and computation between the other regular parties ($\widehat{\mathrm{P}}_\mathrm{A}$ and $\widehat{\mathrm{P}}_\mathrm{B}$). For simplicity, the term "ideal functionality" is used to designate the trusted third party and also its set of rules of interaction with the other parties, thus defining the context of a corresponding protocol in the ideal world. For example, for the case of coin-flipping in the ideal world, the regular ideal parties simply send a coin-flipping request to an ideal functionality $\mathcal{F}_{\mathrm{MCF}}$, who then chooses the random bit-string and sends it to the two parties (one at a time). Conversely, in the real world, the coin-flipping is achieving by means of direct interaction between the regular parties. In any of the two worlds (ideal or real), an external entity denoted *environment* ($\mathcal{Z}$) interacts with the regular parties, namely activating the execution of the protocol, by sending a chosen initial private input (z) to the two regular parties and later receiving their final output. Furthermore, in any of the two worlds there exists an adversary that is able to corrupt and control a party in the respective world and also interact with $\mathcal{Z}$. In a standalone setting, $\mathcal{Z}$ interacts with the parties and the adversary only in the beginning and in the end of the protocol, by means of delivering their input and receiving their output. In the UC framework, $\mathcal{Z}$ is also able to interact with the adversary throughout the protocol execution. However, $\mathcal{Z}$ does not have access to $\mathcal{F}$ in the ideal world. In fact, security is based on the condition that $\mathcal{Z}$ is not able to know whether the regular parties are playing a real protocol or if they are interacting with intermediation by an ideal functionality.

Emulation and simulation. Informally, a real protocol (Π) is said to securely-emulate an ideal functionality ($\mathcal{F}$) if $\mathcal{Z}$ cannot distinguish the ideal and the real worlds apart, for some adversary in the ideal world (denoted *simulator*) with black-box access to the adversary of the real world. This is proven by defining the said simulator ($\mathcal{S}$), such that, for every $\mathcal{Z}$ and for every adversary ($\mathcal{A}$) in the real world, $\mathcal{Z}$ is not capable of distinguishing executions in the ideal world from executions in the real world. $\mathcal{S}$ is given black-box access to $\mathcal{A}$ and uses such capability to make a controlled internal simulation of a real world protocol execution. In the simulation, $\mathcal{S}$ activates the black-box $\mathcal{A}$ by relaying the respective input message coming from $\mathcal{Z}$, and in the end of the simulation also relaying the output of $\mathcal{A}$ to $\mathcal{Z}$. In the simulation, $\mathcal{S}$ also plays the role of the non-corrupted parties, such that $\mathcal{A}$ *believes* to be in the real world, i.e., such that it cannot distinguish the simulated execution from a real (not simulated) one. Since $\mathcal{A}$ behaves as it would in the real world, $\mathcal{S}$ gains knowledge about how to behave as an adversary in the ideal world, in order to induce in the ideal world a *global output* (i.e., the joint output of all the parties and the adversary) with a probabilistic distribution indistinguishable (by $\mathcal{Z}$) from the one that $\mathcal{A}$ would induce in the real world. In the specific case of coin-flipping, $\mathcal{S}$ tries to learn how $\mathcal{A}$ reacts (e.g., aborting or not aborting, and which value it outputs) when facing the perspective of the protocol terminating in a final bit-string equal to one that $\mathcal{S}$ obtains from $\mathcal{F}_{\text{MCF}}$ in the ideal world.

Indistinguishability of distributions. The aim of a proof of security is to show that the probability distributions are indistinguishable. In a rewinding setting, considering only rewindable adversaries, $\mathcal{Z}$ only interacts with the parties and the adversary by means of defining their inputs and reading their final outputs. In this case, the *global output* distribution in the ideal world and the distribution in the real world can be parametrized simply by the input value (z) with which $\mathcal{Z}$ activates the execution of the protocol. These distributions are respectively denoted by $\text{REAL}_{\Pi,\mathcal{A}}(z)$ and $\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}}}(z)$. Conversely, in the UC setting, where $\mathcal{Z}$ may interact with the adversary during the protocol, and so rewinding cannot be used in the simulation,[13] the distributions are directly parametrized by $\mathcal{Z}$. These distributions are respectively denoted by $\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}$ and $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$. In the *plain model*, a protocol Π is said to securely emulate an ideal functionality $\mathcal{F}$ if the IDEAL and the REAL distributions are indistinguishable by $\mathcal{Z}$, i.e., if the *advantage* that $\mathcal{Z}$ has in distinguishing the two worlds is at most negligible in the security parameter. Alternatively, a proof of emulation may be done for a protocol defined in a *hybrid world*, where in comparison with the real world some primitives or sub-protocols are replaced by respective ideal (sub-)functionalities. In particular, this paper considers a hybrid model where the real protocol is defined with the help of separate ideal functionalities that implement an extractable (Q) commitment scheme and an equivocable (Q) commitment scheme. In a simulation of the protocol in the hybrid model, these ideal (sub-)functionalities are impersonated by $\mathcal{S}$ whenever $\mathcal{A}$ attempts to interact with them. In such case, the goal of the proof becomes to show that the IDEAL and the HYBRID distributions are indistinguishable by $\mathcal{Z}$.

Adversarial model. Emulation is considered within the context of a certain class of adversaries and external environments. This paper is interested in computational indis-

[13] In the sense that it would allow $\mathcal{Z}$ to distinguish the two worlds, even if $\mathcal{A}$ is actually rewindable.

tinguishability, i.e., when $\mathcal{Z}$ is computationally bounded to *probabilistic polynomial time*. The adversaries are assumed to be *active*, i.e., may deviate from the protocol specification, namely with the adversary $\mathcal{A}$ in the real world also being computationally bounded to probabilistic polynomial time. This allows instantiating primitives with standard cryptographic assumptions, such as a PRG and a CR-Hash function.[14] In the case of $\mathcal{S}$ in the ideal world, and when considering simulation-with-rewinding, the assumption is more *liberal* by allowing computation in probabilistic expected-polynomial time. A *static* corruption model is assumed, where the adversary is able to corrupt a single party only at the onset of the execution, i.e., before the execution begins, e.g., based on a request that the adversary receives from $\mathcal{Z}$. It is assumed that, in the simulated execution, $\mathcal{S}$ is able to detect which party is corrupted by $\mathcal{A}$, for example by detecting a respective special interaction with the party being corrupted.

Simulatability. The term *simulatability* (essentially denoting *security* in the ideal/real paradigm) is used to denote simulatability on both sides (namely achieving extractability and equivocability), i.e., for whatever party out of two may be corrupted in the two-party computation. The term highlights the contrast with the needed primitives that intuitively require simulatability on just one side (extractability or equivocability). Two settings of simulatability are considering in this paper, namely *with rewinding* and *one-pass*.

Message syntax. Using standard notation (e.g., see [CLOS02]), messages sent between each party and the ideal functionality are composed of a public header and an optional private content. The public header is typically composed of: a *message-type identifier*, e.g., a text-string suggestive of the role of the message in the protocol; a session identifier *sid*, so that in concurrent executions each party can know to which session the message refers to; a sub-session identifier, so that in a modular construction of a larger protocol an ideal functionality can be initialized only once (in the first call) instead of having to be initialized in every call, e.g., *cid* for commitment identifier, of *cfid* for coin-flipping identifier; the parties associated with the message (e.g., sender and receiver), so that the message can be properly forwarded and processed; additional contextual information that does not need to be private, e.g., the length of the bit-string being decided in a coin-flipping protocol. The messages exchanged are ideally authenticated, which means that an adversary cannot change them without being noticed. In order to model an adversary that in the ideal world can read the public content of every message exchanged between the regular parties and the ideal functionality, the ideal functionality sends to $\mathcal{S}$ a *receipt* of message exchanges, including their public content.

For simplicity, throughout the protocols defined in this paper it is assumed that the activation messages from $\mathcal{Z}$ to the honest ideal parties are equal to the ones with which the ideal parties are supposed to activate the ideal functionality, and the same thing can be assumed in reverse order for the final outputs. For example for coin-flipping (Fig. 6), an honest ideal party $\widehat{\mathrm{P}}_{\mathrm{A}}$ is expected to send an initial message of the form (`start-1`, *sid*, (*cfid*, P_{A}, P_{B}), ℓ) to $\mathcal{F}_{\mathrm{MCF}}$ (here being assumed w.l.o.g. that $\mathcal{Z}$ is asking for an execution

[14] More formally, but left implicit, it is assumed that there is a family of PRGs and CR-Hash functions, parametrized by the security parameter κ, such that for sufficiently large κ the respective intractabilities are satisfied: indistinguishability from random and collision resistance.

between P_A and P_B, with P_A taking the role of the first party to learn the bit-string outcome). Thus, it is simply assumed that an honest party will send such message to $\mathcal{F}_{\text{MCF}}$ only upon receiving in her external input tape the exact same message from $\mathcal{Z}$.

***A note on* rewinding.** *Rewinding* refers to the ability of the simulator $\mathcal{S}$ to rewind the state of the real adversary $\mathcal{A}$. For example, the notion is intuitive when considering adversaries that can be virtualized as a computer algorithm execution, accessible in a black-box manner, with recoverable past states (e.g., by taking snapshots of the state of the party across state transitions), and interacting with the external environment $\mathcal{Z}$ only at the beginning and end of a protocol execution. In contrast, in other settings (which may include concurrent protocol executions) it may be necessary to consider *one-pass simulatability* (i.e., without rewinding), namely if (within the the adversarial class being considered) there exists an adversary $\mathcal{A}$ that cannot be rewound, or if its rewinding can be directly detected by $\mathcal{Z}$. The latter case may happen when $\mathcal{Z}$ is able to interact with $\mathcal{A}$ at arbitrary moments of an execution, as foreseen in the UC framework, e.g., keeping track of the state of $\mathcal{A}$. $\mathcal{Z}$ would then be able to distinguish the ideal from the real world by simply determining whether or not a rewinding operation has taken place.

A.2 X&Q bit-string commitments

A commitment scheme is a protocol with two parties, denoted *sender* and *receiver*,[15] and two phases, denoted *commit* and *open* (or *reveal*). In the *commit* phase, the *sender* binds to a value that is *hidden* from the receiver. In the *open* phase, the *sender* reveals (in a convincing way) the committed value to the receiver. The scheme is denoted *interactive* if any of the two phases requires one or several rounds of communication, and *non-interactive* if the communication in each phase consists only on sending a message from the sender to the receiver. While the properties of *hiding* and *binding* directly reflect inabilities of the parties, there are more subtle properties related to the simulation paradigm of security, where the simulator is endowed with extra power in comparison with a regular party. A commitment scheme is called *equivocable* (Q) if the simulator in the role of sender is able to open any commitment into any value in the domain of allowed committed values [Bea96]. A commitment scheme is called *extractable* (X) if the simulator in the role of receiver is able to extract the value that has been committed by an honest *sender* [SCP00].

Notation. The following notation is used throughout the paper to denote phases of a commitment scheme $\mathscr{C}$, between a committer party P_s (the sender) and a receiver P_r.

$$\textbf{Commit phase: } \mathscr{C}^{\text{Commit}}\left[P_s(x) \leftarrow^{\$} (\underline{x}, \overline{x})\,; P_r \leftarrow \overline{x}\right] \tag{55}$$

$$\textbf{Open phase: } \mathscr{C}^{\text{Open}}\,(\overline{x})\,[P_s(x, \underline{x}); P_r \leftarrow x] \tag{56}$$

In a *commit* phase (55), P_s and P_s interact, with P_s starting with private input x, from which it obtains the commitment $\overline{x}$ of x, and also the randomness $\underline{x}$ needed

[15] More generally, a set with more parties can be considered, of which two parties assume the role of sender and receiver for a particular execution.

later to decommit (i.e., *open*) x from $\overline{x}$, and with P_r ending with output $\overline{x}$. In an *open* phase (56), P_s and P_r interact, with common input $\overline{x}$, with P_s having $(x, \underline{x})$ as private input, and with P_r ending with x as final output. A possible variant of the commitment scheme is one where in the *open* phase the receiver also receives $\underline{x}$. It is implicit that there is a non-interactive operation $\mathscr{C}^{\text{Verify}}$ that can be used to verify the correct relation between x, $\underline{x}$ and $\overline{x}$ (i.e., for whoever is in possession of such three elements). The notation may be augmented with more information about the commitment scheme properties or the type of phase. For example, ${\mathscr{C}_t}^p$ may be used to indicate a phase p (*Commit* or *Open*) of a commitment scheme of type t (X or Q or X&Q) where X and Q respectively denote extractable and equivocable properties of simulatability. More generally, ${\mathscr{C}_t}^p(z)\,[P_s(x_a) \leftarrow y_a; P_r(x_b) \leftarrow y_b]$ indicates the execution of phase ${\mathscr{C}_t}^p$, between parties P_s and P_r, starting with common input z and with respective private inputs x_a and x_b, and ending with respective private outputs y_a and y_b. Hereafter, the superscript denoting the phase will also include a X or Q as prefix whenever the respective action (X or Q) is supposed to be achieved by the simulator. In other words, *X-Commit* and *Q-Open* will respectively denote *commit* and *open* phases where the simulator is supposed to be able to perform *extraction* and *equivocation*.

The notation can be simplified in case of a non-interactive scheme, as follows:

Commit phase		**Open phase**	
$P_s : (\underline{x}, \overline{x}) \leftarrow^{\$} \mathscr{C}[x]$	(57)	$P_s \rightarrow P_r : (\underline{x}, x)$	(59)
$P_s \rightarrow P_r : \overline{x}$	(58)	$P_r : \text{Verify}[\mathscr{C}](x, \underline{x}, \overline{x})$	(60)

The *commit* phase starts with a local computation by the sender, who computes alone the public commitment and the private randomness needed to open it (57). Then, P_s send the commitment to P_r. In the *open* phase, the communication consists simply on P_s sending to P_r the committed value x and the respective randomness $\underline{x}$ needed to verify it. Then, P_r locally verifies the commitment against the respective value and randomness and outputs the value if the verification is successful.

Ideal X&Q functionality. Ideal functionalities for X&Q commitments have been defined in prior work, e.g., see [CF01, Fig. 3] for multiple bit-commitments and [DN02, §4.2] for multiple message-commitments (there also considering homomorphic relations). The ideal functionality $\mathcal{F}_{\text{MCOM(X\&Q)}}$ in Fig. 4 is the immediate generalization of [CF01, Fig. 3] from individual bits to bit-strings and allowing the bit-string length to be defined at the moment of commitment (so that the same functionality can be used several times to commit messages of different length). In this ideal world, the parties do not need to execute any kind of computation, as in essence the *commit* and *open* phases are intermediated by $\mathcal{F}_{\text{MCOM(X\&Q)}}$ that is fully trusted to store a value transmitted by the sender (P_s), and later, when allowed by P_s, to reveal it to the receiver (P_r). When considering a simulation, $\mathcal{S}$ impersonates the ideal functionality, and so it takes advantage of the trust that the other parties place in it. In particular: *extractability* (X) in the commit phase derives from P_s sending the message in clear to $\mathcal{F}_{\text{MCOM(X\&Q)}}$; *equivocability* (Q) in the open phase derives from the acceptance by P_r of any (well formed) message received from $\mathcal{F}_{\text{MCOM(X\&Q)}}$. Commitments that are simultaneously X and Q are specially

Functionality $\mathcal{F}_{\text{MCOM(X\&Q)}}$

$\mathcal{F}_{\text{MCOM(X\&Q)}}$ activated with session identifier *sid* proceeds as follows, running with parties P_1, ..., P_n, an adversary $\mathcal{S}$, and a computational security parameter 1^κ. (Note: the suffix $[\ell]$ may be added to the notation to denote the obvious restriction to commitments of bit-strings of length ℓ.)

1. **Commit phase.** Upon receiving a message (`commit`, *sid*, *cid*, P_s, P_r, (ℓ, m)) from P_s (the sender), if $\ell \in O(poly(\kappa))$, $m \in \{0,1\}^\ell$ and $s, r \in [n]$, then record the tuple (*cid*, P_s, P_r, m), send the message (`receipt`, *sid*, *cid*, P_s, P_r, ℓ) to P_r (the receiver) and $\mathcal{S}$ and ignore subsequent messages with header (`commit`, *sid*, *cid*, P_s, P_r,...); otherwise ignore the message.
2. **Open phase.** Upon receiving message (`open`, *sid*, *cid*, P_s, P_r) from P_s: if for some m a tuple (*cid*, P_s, P_r, m) has been recorded, then send the message (`open`, *sid*, *cid*, P_s, P_r, m) to P_r and $\mathcal{S}$ and ignore any subsequent messages with identifiers (..., *sid*, *cid*, P_s, P_r, ...); otherwise ignore the message.
3. **Abort.** Upon receiving a message (`abort`, *sid*, *cid*, P_s, P_r) from P_s or P_r, then ignore any subsequent messages with identifiers (..., *sid*, *cid*, P_s, P_r, ...).

Fig. 4. Ideal functionality – multiple bit-string commitments (X and Q)

important, as they enable simulatability (i.e., simulatability from both sides) of other protocols, as is the case of coin-flipping. Section B defines alternative ideal notions of commitment schemes, where either X or Q properties are suppressed on purpose.

A.3 Parallel coin-flipping into a well

In an ideal two-party coin-flipping (*into a well*), the ideal functionality $\mathcal{F}_{\text{MCF}}$ uniformly selects a bit-string of the target length and sends it to the parties, one at a time, and with the first receiver being able to decide whether or not the other party can learn the bit-string. A prior definition can be found for example in [DL09, Fig. 2], though in this paper this is generalized from a single-coin to bit-strings, and from a single coin-flipping call to multiple coin-flips calls of specifiable length (Fig. 5, Fig. 6). In this ideal functionality, the parties choose which one learns the random bit-string in the first place. This is motivated by typical real protocols (e.g., within the traditional template) where the parties have asymmetric roles, one of which guarantees being the first receiver of the bit-string. This has a correspondence to the adversary (the simulator) in the ideal world. If the adversary is controlling the ideal party that receives the string in second place, then the adversary is also not able to discover the bit-string before the first party. This is based on the assumption that the adversary is not able to see the messages exchanged between $\widehat{\text{P}}_\text{A}$ and the $\mathcal{F}_{\text{MCF}}$.[16]

The ideal coin-flipping functionality $\mathcal{F}_{\text{MCF}}$ is here initially and informally described in the form of a protocol in the ideal world, considering honest interaction between the parties Fig. 5. As initial input, the two regular parties are activated with an input that instructs them to start the protocol for coin-flipping of a certain target length, and distinguishing the role of the parties (namely the order in which they should learn the

[16] An alternative assumption, if it is preferable to let the adversary eavesdrop, is to consider that messages have a private component (e.g., using encryption) and include the bit-string therein.

Setup – private inputs.

$\text{input}_A \rightarrow \widehat{\text{P}}_\text{A} : (\texttt{cf-start-1}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \ell_A)$ (61)

$\text{input}_B \rightarrow \widehat{\text{P}}_\text{A} : (\texttt{cf-start-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \ell_B)$ (62)

Send inputs to $\mathcal{F}_{\text{MCF}}$.

$\widehat{\text{P}}_\text{A} \rightarrow \mathcal{F}_{\text{MCF}} : (\texttt{cf-start-1}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \ell_A)$ (63)

$\widehat{\text{P}}_\text{B} \rightarrow \mathcal{F}_{\text{MCF}} : (\texttt{cf-start-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \ell_B)$ (64)

Local computation at $\mathcal{F}_{\text{MCF}}$.

If $(\ell_A \neq \ell_B)$, then : (65)

For $p \in \{A, B\}$:

$\mathcal{F}_{\text{MCF}} \rightarrow \widehat{\text{P}}_p : (\texttt{cf-fail}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B})$ (66)

$\mathcal{F}_{\text{MCF}}$: ignore further (...,*sid*,*cfid*,P_A,P_B,...) (67)

$\mathcal{F}_{\text{MCF}} : \chi \leftarrow^{\$} \{0,1\}^{\ell}$ (68)

Send output to parties.

$\mathcal{F}_{\text{MCF}} \rightarrow \widehat{\text{P}}_\text{A} : (\texttt{cf-deliver-1}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (69)

$\widehat{\text{P}}_\text{A} \rightarrow \mathcal{F}_{\text{MCF}} : (\texttt{cf-OK}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B})$ (70)

$\mathcal{F}_{\text{MCF}} \rightarrow \widehat{\text{P}}_\text{B} : (\texttt{cf-deliver-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (71)

Final outputs.

$\widehat{\text{P}}_\text{A} \rightarrow \text{output}_A : (\texttt{cf-output-1}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (72)

$\widehat{\text{P}}_\text{B} \rightarrow \text{output}_B : (\texttt{cf-output-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (73)

Concurrent process. For $p \in \{A, B\}$:

$\mathcal{F}_{\text{MCF}}$: If receiving $(\texttt{cf-abort-i}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B})$ (74)

from $\widehat{\text{P}}_p$, then relay it to $\widehat{\text{P}}_{\bar{p}}$ and (75)

ignore further (..., *sid*, *cfid*, P_A, P_B,...) (76)

$\widehat{\text{P}}_p$: If receiving $(\texttt{cf-abort-i}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B})$ or

$(\texttt{cf-fail}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B})$ from $\mathcal{F}_{\text{MCF}}$ (77)

then $\widehat{\text{P}}_p \rightarrow \text{output}_p$:

$(\texttt{cf-abort}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B})$ (78)

Fig. 5. Ideal parallel coin-flipping into a well (succinct notation)

output) (61)-(62). $\mathcal{F}_{\text{MCF}}$ awaits receiving a `start` type of request from both parties, expecting that they have consistently agreed on who should be the first to receive the bit-string (63), e.g., P_A, and who should be the second (64). If the target length requested by both parties is not the same (65), then $\mathcal{F}_{\text{MCF}}$ sends a `fail` message to both parties (66) and ignores further messages related with this coin-flipping instance (67). Otherwise, if the target lengths are the same[17], $\mathcal{F}_{\text{MCF}}$ computes a random bit-string with such length (68) and sends it first to $\widehat{\text{P}}_\text{A}$ (69) (the party that activated $\mathcal{F}_{\text{MCF}}$ with a message of type `start-1`). $\widehat{\text{P}}_\text{A}$ then confirms acceptance of the bit-string by sending an `OK` message to $\mathcal{F}_{\text{MCF}}$ (70). Finally, $\mathcal{F}_{\text{MCF}}$ sends the bit-string to P_B (71). Each of the two parties decides as final output the bit-string received from $\mathcal{F}_{\text{MCF}}$ (72)-(73). If during the protocol execution $\mathcal{F}_{\text{MCF}}$ receives an *abort* message from one (necessarily corrupted) party (74), then it relays it to the other party (75) and it ignores further messages related with this coin-flipping instance (76). If a party receives an `abort` or `fail` message from $\mathcal{F}_{\text{MCF}}$ (77), then it aborts the execution of this ideal coin-flipping protocol (78).

The ideal functionality is specified using more conventional notation in Fig. 6, considering also the messages received sent from the ideal functionality to the simulator (i.e., the adversary that may be controlling one of the two parties).

A.4 A base protocol template

The traditional template (possibly the simplest) for a simulatable protocol of two-party coin-flipping into-the-well requires a single party to use an extractable and equivocable commitment scheme to commit to her contribution. This is concisely described in Fig. 7, with the two parties being called P_A and P_B. In a setup phase, P_A and P_B are activated to start a coin-flipping for a given target length and agreeing that P_A is the one to learn the bit-string in the first place (79)-(80). P_A selects her contribution randomly (81) and then commits to it using a X-*commit* phase (82); i.e., such that a simulator in the role of

[17] Other verifications (e.g,. *sid*, *cid*, parties' identifiers) are left implicit in the description.

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

$\mathcal{F}_{\text{MCF}}$ activated with session identifier *sid* proceeds as follows, running with parties P_1, ..., P_n, an adversary $\mathcal{S}$, and a computational security parameter 1^κ. (Note: the suffix $[\ell]$ may be added to the notation to denote the obvious restriction to commitments of bit-strings of length ℓ.)

Start instructions.

- When receiving a message (`cf-start-1`, *sid*, *cfid*, P_A, P_B, ℓ) from a running party P_A : $A \in [n]$, with sub-session identifier (*cfid*, P_A, P_B), where P_B : $B \in [n]$, and specifying a polynomial *target*-length $\ell \in O(poly(\kappa))$, record the tuple (`cf-start-1`, *cfid*, P_A, P_B, ℓ), send (`cf-start-1-receipt`, *sid*, *cfid*, P_A, P_B, ℓ) to the adversary $\mathcal{S}$, and ignore subsequent messages of type (`cf-start-1`, *sid*, *cfid*, P_A, P_B, ...).
- When receiving a message (`cf-start-2`, *sid*, *cfid*, P_A, P_B, ℓ) from P_B, do the same as in the previous item but replacing `cf-start-1` by `cf-start-2` and `cf-start-1-receipt` by `cf-start-2-receipt`.

First delivery. After receiving a pair of correlated messages as specified in the previous two items, i.e., with the same sub-session identifier (*cfid*, P_A, P_B) and covering the message-types `cf-start-1` and `cf-start-2`, if the target-length specified in the two messages is the same:

- *then* uniformly sample a bit-string $m \in \{0,1\}^\ell$ and then send (`cf-deliver-1`, *sid*, *cfid*, P_A, P_B, m) to P_A (i.e., to the party whose initial message was of type `cf-start-1`) and send (`cf-deliver-1-receipt`, *sid*, *cfid*, P_A, P_B) to the adversary $\mathcal{S}$,[a] and record (`cf-deliver-1`, (*cfid*, P_A, P_B));
- *else* send (`cf-fail`, *sid*, *cfid*, P_A, P_B) to both parties P_A, P_B and to the adversary $\mathcal{S}$, and record the tuple (`cf-end`, (*cfid*, P_A, P_B)).

Second delivery. Upon receiving (`cf-OK`, *sid*, *cfid*, P_A, P_B) from P_A, if (`cf-deliver-1`, *cfid*, P_A, P_B) has been recorded and (`cf-end`, *cfid*, P_A, P_B) has not been recorded, then send (`cf-deliver-2`, *sid*, *cfid*, P_A, P_B, m) to P_B and (`cf-deliver-2-receipt`, *sid*, *cfid*, P_A, P_B) to the adversary $\mathcal{S}$, and record (`cf-end`, *cfid*, P_A, P_B).

Early abort requests.

- Upon receiving (`cf-abort-1`, *sid*, (*cfid*, P_A, P_B)) from P_A, if (`cf-start-1`, *cfid*, P_A, P_B) has been recorded and (`cf-end`, *cfid*, P_A, P_B) has not been recorded, then send (`abort-1`, *sid*, (*cfid*, P_A, P_B)) to P_B and $\mathcal{S}$ and record (`cf-end`, *cfid*, P_A, P_B).
- Upon receiving (`cf-abort-2`, *sid*, *cfid*, P_A, P_B) from P_B, do the same as in the previous item but replacing `cf-abort-1` with `cf-abort-2`, and replacing the recipient P_B by P_A.

[a] Notice that $\mathcal{S}$ does not obtain the message m from the receipt. However, if $\mathcal{S}$ is controlling P_A then it will be able to read such value inside the `cf-deliver-1` message.

Fig. 6. Ideal functionality for multiple bit-string coin-flippings.

P_B would be able to extract the contribution of P_A in this step. Then, P_B simply decides (83) and sends his random contribution to P_A (84). Then, using a Q-*open* phase, P_A opens her contribution to P_B (85); i.e., such that a simulator in the role of P_A would be able to successfully open any contribution of his choice, namely one decided only after knowing the contribution of P_B. Finally, each party locally computes the final output as a combination of both contributions (86), and each party outputs the result (87)-(88).

0. Setup – private inputs.

$\text{input}_A \to \text{P}_\text{A} : (\texttt{cf-start-1}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \ell)$ (79)

$\text{input}_B \to \text{P}_\text{A} : (\texttt{cf-start-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \ell)$ (80)

1. X-Commit contribution of P_A.

$\text{P}_\text{A} : \chi_A \leftarrow^{\$} \{0,1\}^\ell$ (81)

$\mathscr{C}^{\text{X-Commit}}_{\text{XQ},\mathit{sid},\mathit{cfid},\text{P}_\text{A},\text{P}_\text{B}}(\text{P}_\text{A}(\chi_A) \leftarrow^{\$} (\underline{\chi}_A, \overline{\chi}_A), \text{P}_\text{B} \leftarrow \overline{\chi}_A)$ (82)

2. Send clear contribution of P_B.

$\text{P}_\text{B} : \chi_B \leftarrow^{\$} \{0,1\}^\ell$ (83)

$\text{P}_\text{B} \to \text{P}_\text{A} : (\texttt{cf-contrib-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi_B)$ (84)

3. Q-Open contribution of P_A.

$\mathscr{C}^{\text{Q-Open}}_{\text{XQ},\mathit{sid},\mathit{cfid},\text{P}_\text{A},\text{P}_\text{B}}(\overline{\chi}_A)(\text{P}_\text{A}(\chi_A, \underline{\chi}_A), \text{P}_\text{B} \leftarrow \chi_A)$ (85)

4. Locally combine contributions.

$\text{P}_\text{A}, \text{P}_\text{B} : \chi = \chi_A \oplus \chi_B$ (86)

$\text{P}_\text{A} \to \text{output}_A : (\texttt{cf-output-1}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (87)

$\text{P}_\text{B} \to \text{output}_B : (\texttt{cf-output-2}, \mathit{sid}, \mathit{cfid}, \text{P}_\text{A}, \text{P}_\text{B}, \chi)$ (88)

Fig. 7. Traditional template for coin-flipping

Proposition 1. *The* multiple bit-string coin-flipping *protocol Π_{MCF} (Fig. 7) UC-realizes the ideal* multiple bit-string coin-flipping *functionality $\mathcal{F}_{MCF}$ (Fig. 6) in the $\mathcal{F}_{X\&Q}$-hybrid model in the presence of static computationally active adversaries.*

If the base commitment scheme is indeed X&Q, then the protocol is simulatable because the simulator is able to use the X or Q properties (depending on which party is being simulated) to induce the outcome of the simulated execution to be the target *bit-string* decided by $\mathcal{F}_{\text{MCF}}$ in the ideal world. Specifically, after learning the contribution of the malicious party in the simulated real world, $\mathcal{S}$ is able to calculate the *needed complementary contribution*, i.e., the one that once combined with the other contribution leads to the target bit-string, and use it as the contribution of the honest party in the simulated execution. Naturally, it is here assumed that the *combination* operation is efficiently invertible in respect to any fixed contribution of a party, e.g., as in the case of bit-wise XOR or modular integer multiplication. The possibility of a malicious party aborting the simulated execution is also contemplated in simulatability. An *abort* is not an issue in this template, because it can only happen in an execution that otherwise (i.e., if not *aborted* by the malicious party and not *rewound* by the simulator) would lead to the target bit-string outcome. Thus, in case of *abort* in the simulated execution the simulator simply *emulates an abort* in the ideal world. Section C.2 highlights an issue with *abort* if the commitment scheme is not X or not Q.

B Ideal commitments with suppressed properties

While this section is not essential for the remaining analysis of the new protocols (and its reading can be safely skipped), it is helpful by providing a complementary perspective of the dual nature of X and Q. First, a motivation is given for the formalization of X and Q in isolation, namely by suppressing the respective complementary property (§B.1). Then, as an initial attempt, the properties are suppressed based on a circular definition of a new type of ideal commitment functionalities (§B.2). The circularity is then removed by nesting the hybrid model inside another hybrid model (§B.3). Finally, in respect to their use in proofs of security, a brief comparison is made between the plain model and the other two mentioned models (§B.4).

B.1 Motivation for simulation based security

The ideal/real simulation paradigm provides a conceptually simple framework to understand certain security properties, namely when considering a hybrid model with access to ideal functionalities. For example, properties like extractability (X) and equivocability (Q) derive automatically from the power that a simulator gains by impersonating an ideal commitment functionality $\mathcal{F}_{\text{MCOM}}$ during a simulated execution. In particular: X is a consequence of the sender in the ideal world transmitting in clear to $\mathcal{F}_{\text{MCOM}}$ the value that is being committed; Q is a consequence of the receiver in the ideal world accepting whatever value the ideal $\mathcal{F}_{\text{MCOM}}$ reveals.

In contrast with the above observation, this paper aims at emphasizing that a X&Q commitment scheme (for *large* strings) can be built from a X-but-possibly-not-Q commitment scheme and a separate Q-but-possibly-not-X commitment scheme (both for *short* strings and invoked only a *few* times). Thus, it is relevant to formally consider the notions of non-extractability and non-equivocability and directly embed them into the ideal world, namely into respective ideal commitment functionalities. These functionalities must be different from the typical ideal commitment functionality $\mathcal{F}_{\text{MCOM}}$ that is simultaneously X and Q. The pertinent question is: *how to formalize an ideal commitment functionality $\mathcal{F}_X$ or $\mathcal{F}_Q$ whose impersonation by a simulator does not provide the simulator with a Q or X capability, respectively, but still allows the use of the functionality in a hybrid model*.

B.2 An initial attempt (using a definition with circularity)

As a step to consolidate intuition, Fig. 8 informally sketches a way in which to suppress Q or X properties from the ideal functionality. To suppress Q (in order to get X&$\overline{\text{Q}}$), in the *commit* phase the ideal functionality $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ sends a (stand-alone secure) commitment to the receiver, such that $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ becomes *bound* to a single message for the later *open* phase, but temporarily hiding it from the receiver. To suppress X (in order to get $\overline{\text{X}}$&Q), in the *commit* phase the committer does not send the message in clear text to the ideal functionality $\mathcal{F}_{\overline{\text{X}}\text{\&Q}}$, but instead sends a (stand-alone secure) commitment to the message. These ideal functionalities with suppressed properties require an internal instantiation of a commitment scheme, hereafter denoted as *virtual*. Thus, when questioning if a real commitment scheme $\mathscr{C}_{\text{R}}$ emulates one of these ideal functionalities, it is also necessary to consider an additional virtual commitment scheme $\mathscr{C}_{\text{V}}$.

When attempting a simulation, it becomes clear that $\mathscr{C}_{\text{R}}$ and $\mathscr{C}_{\text{V}}$ must be somehow related to each other. For example, consider a simulator ($\mathcal{S}$) acting as a sender P_s in a simulated execution of a real world extractable commitment scheme $\mathscr{C}_{\text{R}}$ (intended to be extractable), trying to emulate the ideal functionality $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}[\mathscr{C}_{\text{V}}]$ and for that purpose acting concurrently as receiver $\widehat{\text{P}}_r$ in a respective ideal world execution. In the role $\widehat{\text{P}}_r$, during the commit phase $\mathcal{S}$ receives from the ideal functionality a *virtual* commitment (99). Then, in the simulated execution, while impersonating a real sender P_s, $\mathcal{S}$ relays this commitment to the black-box receiver P_r^* (58) who is expecting to receive a *real* commitment (i.e., something produced with $\mathscr{C}_{\text{R}}$). In particular, in the later *open* phase, when $\mathcal{S}$ (in the role of ideal receiver $\widehat{\text{P}}_r$) receives the opening from the ideal functionality (102), $\mathcal{S}$ (in the role of real sender P_s) must be able to open the same value (59) for the

Templates for different ideal commitment functionalities

Note: in the X&$\overline{\text{Q}}$ and the $\overline{\text{X}}$&Q cases, $\mathscr{C}_{\text{V}}$ is a commitment scheme (i.e., with at least computational hiding and binding properties) that requires instantiation by a real commitment scheme; nonetheless, considering that it is implemented in the ideal world, it is hereafter denoted as a *virtual* commitment scheme.

Ideal functionality $\mathcal{F}_{\text{X\&Q}} \equiv \mathcal{F}_{\text{MCOM(X\&Q)}} \equiv \mathcal{F}_{\text{MCOM(X\&Q)}}$ of X&Q commitment scheme

Commit phase

$$\text{input}_s \to \widehat{\text{P}}_s : (\texttt{commit}, \ell, m) \quad (89)$$
$$\widehat{\text{P}}_s \to \mathcal{F}_{\text{X\&Q}} : (\texttt{commit}, \ell, m) \quad (90)$$
$$\mathcal{F}_{\text{X\&Q}} \to \widehat{\text{P}}_r : (\texttt{OK}, \ell) \quad (91)$$

Open phase

$$\text{input}_s \to \widehat{\text{P}}_s : \texttt{open} \quad (92)$$
$$\widehat{\text{P}}_s \to \mathcal{F}_{\text{X\&Q}} : \texttt{open} \quad (93)$$
$$\mathcal{F}_{\text{X\&Q}} \to \widehat{\text{P}}_r : (\texttt{open}, m) \quad (94)$$
$$\widehat{\text{P}}_r \to \text{output}_r : (\texttt{open}, m) \quad (95)$$

Template for ideal functionality $\mathcal{F}_{\text{X\&}\overline{\text{Q}}} \equiv \mathcal{F}_{\text{MCOM}(\text{X\&}\overline{\text{Q}})}^{[\mathscr{C}_{\text{V}}]}$ of X&$\overline{\text{Q}}$ commitment scheme

Commit phase

$$\text{input}_s \to \widehat{\text{P}}_s : (\texttt{commit}, \ell, m) \quad (96)$$
$$\widehat{\text{P}}_s \to \mathcal{F}_{\text{X\&}\overline{\text{Q}}} : (\texttt{commit}, \ell, m) \quad (97)$$
$$\mathcal{F}_{\text{X\&}\overline{\text{Q}}} : (\underline{m}, \overline{m}) \leftarrow^{\$} \mathscr{C}_{\text{V}}[m] \quad (98)$$
$$\mathcal{F}_{\text{X\&}\overline{\text{Q}}} \to \widehat{\text{P}}_r : (\texttt{OK}, \ell, \overline{m}) \quad (99)$$

Open phase

$$\text{input}_s \to \widehat{\text{P}}_s : \texttt{open} \quad (100)$$
$$\widehat{\text{P}}_s \to \mathcal{F}_{\text{X\&}\overline{\text{Q}}} : \texttt{open} \quad (101)$$
$$\mathcal{F}_{\text{X\&}\overline{\text{Q}}} \to \widehat{\text{P}}_r : (\texttt{open}, m, \underline{m}) \quad (102)$$
$$\widehat{\text{P}}_r : \text{Verify}[\mathscr{C}_{\text{V}}](m, \underline{m}, \overline{m}) \quad (103)$$
$$\widehat{\text{P}}_r \to \text{output}_r : (\texttt{open}, m) \quad (104)$$

Template for ideal functionality $\mathcal{F}_{\overline{\text{X}}\text{\&Q}} \equiv \mathcal{F}_{\text{MCOM}(\overline{\text{X}}\text{\&Q})}^{[\mathscr{C}_{\text{V}}]}$ of $\overline{\text{X}}$&Q commitment scheme

Commit phase

$$\text{input}_s \to \widehat{\text{P}}_s : (\texttt{commit}, \ell, m) \quad (105)$$
$$\widehat{\text{P}}_s : (\underline{m}, \overline{m}) \leftarrow^{\$} \mathscr{C}_{\text{V}}[m] \quad (106)$$
$$\widehat{\text{P}}_s \to \mathcal{F}_{\overline{\text{X}}\text{\&Q}} : (\texttt{commit}, \ell, \overline{m}) \quad (107)$$
$$\mathcal{F}_{\overline{\text{X}}\text{\&Q}} \to \widehat{\text{P}}_r : (\texttt{OK}, \ell) \quad (108)$$

Open phase

$$\text{input}_s \to \widehat{\text{P}}_s : \texttt{open} \quad (109)$$
$$\widehat{\text{P}}_s \to \mathcal{F}_{\overline{\text{X}}\text{\&Q}} : (\texttt{open}, m, \underline{m}) \quad (110)$$
$$\mathcal{F}_{\overline{\text{X}}\text{\&Q}} : \text{Verify}[\mathscr{C}_{\text{V}}](m, \underline{m}, \overline{m}) \quad (111)$$
$$\mathcal{F}_{\text{X\&Q}} \to \widehat{\text{P}}_r : (\texttt{open}, m) \quad (112)$$
$$\widehat{\text{P}}_r \to \text{output}_r : (\texttt{open}, m) \quad (113)$$

Fig. 8. Templates for ideal commitment schemes (using succinct notation). For simplicity, the notation assumes that $\mathscr{C}_{\text{V}}$ is non-interactive – it would also be possible to use interactive schemes. Given an implicit security parameter 1^{κ}, it is assumed that the ideal functionality verifies that $|m| \in O(poly(\kappa))$ (if verification fails then it ignores the message). For simplicity, the session identifier (*sid*) and the sub-session identifier (*cid*, for commitment identifier) are left implicit, as well as the headers containing the source and destination of messages. In the X&$\overline{\text{Q}}$ case, $\widehat{\text{P}}_r$ accepts the *open* phase only if the opening of the underlying virtual commitment is correct (i.e., the "Verify" operation is performed even though the ideal functionality is trusted by default) (103). In the $\overline{\text{X}}$&Q case, the ideal functionality proceeds with the *open* phase only if $\widehat{\text{P}}_s$ correctly opens the underlying virtual commitment, i.e., if the respective verification is successful (111).

real commitment that was previously sent to the black-box P_r^*. Thus, the simulation is valid only if the *real* and *virtual* commitments are indistinguishable, e.g., if they are the same. A corresponding relation (with obvious adjustments) could also be analyzed for the case of $\mathcal{F}_{\overline{\text{X}}\text{\&Q}}$.

In spite of the circularity that arises from defining $\mathscr{C}_{\text{V}}$ equal to $\mathscr{C}_{\text{R}}$, the defined ideal functionalities are useful to differentiate real commitment schemes in terms of the ideal functionalities that they emulate. In particular, this enables a definition of X and Q commitment schemes, as follows.

Definition 6 (extractable commitment scheme). *A real commitment scheme $\mathscr{C}$ (by definition already* hiding *and* biding*)*[18] *is said to be extractable if it securely emulates the ideal $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}[\mathscr{C}]$. This is succinctly expressed in the equation below (114).*

$$(\mathscr{C} \text{ is extractable}) \equiv \left((\exists \mathcal{S})(\forall \mathcal{Z}, \mathcal{A}) \texttt{IDEAL}_{\mathcal{F}_{\text{X\&}\overline{\text{Q}}}[\mathscr{C}], \mathcal{S}^{\mathcal{A}}, \mathcal{Z}} \overset{c}{\approx} \texttt{REAL}_{\mathscr{C}, \mathcal{A}, \mathcal{Z}} \right) \quad (114)$$

[18] This definition is still implicitly based on the *hiding* and *binding* notions of a commitment scheme. In the next subsection this is replaced by an ideal (virtual) functionality.

Template for ideal functionality $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}[\mathscr{C}] \equiv \mathcal{F}_{\text{MCOM}(\text{X\&}\overline{\text{Q}})}[\mathscr{C}]$

Let $\mathscr{C}$ be a commitment scheme (i.e., at least computationally hiding and binding in respect to the implicit security parameter 1^κ), with respective *commit* and *open* phases. $\mathcal{F}_{\text{MCOM}(\text{X\&}\overline{\text{Q}})}[\mathscr{C}]$ activated with session identifier *sid* proceeds as follows, running with parties P_1, ..., P_n, an adversary $\mathcal{S}$. (Note: the suffix $[\ell]$ may be added to the notation to denote the obvious restriction to commitments of bit-strings of length ℓ.)

1. **Commit phase.** Upon receiving a message (`commit`, *sid*, *cid*, P_s, P_r, (ℓ, m)) from P_s (the sender), if $\ell \in O(poly(\kappa))$, $m \in \{0,1\}^\ell$ and $s, r \in [n]$, then: compute a commitment of m as $(\overline{m}, \underline{m}) \leftarrow^{\$} \mathscr{C}[m]$, where $\overline{m}$ is the public commitment and $\underline{m}$ is the respective private information needed for *opening*; then record the tuple (*cid*, P_s, P_r, $(m, \underline{m})$), send the message (`receipt`, *sid*, *cid*, P_s, P_r, ℓ, $\overline{m}$) to P_r (the receiver) and $\mathcal{S}$ and ignore subsequent messages with header (`commit`, *sid*, *cid*, P_s, P_r, ...); otherwise ignore the message.
2. **Open phase.** Upon receiving message (`open`, *sid*, *cid*, P_s, P_r) from P_s: if for some m a tuple (*cid*, P_s, P_r, $(m, \underline{m})$) has been recorded, then send the message (`open`, *sid*, *cid*, P_s, P_r, $(m, \underline{m})$) to P_r and $\mathcal{S}$ and ignore any subsequent messages with identifiers (...,*sid*, *cid*, P_s, P_r,...); otherwise ignore the message.
3. **Abort.** Upon receiving a message (`abort`, *sid*, *cid*, P_s, P_r) from P_s or P_r, then ignore any subsequent messages with identifiers (..., *sid*, *cid*, P_s, P_r, ...).

Fig. 9. Ideal functionality – multiple bit-string commitments X-but-not-Q

For example, consider a real X commitment scheme, where a setup phase endows the simulator with a trapdoor that enables extraction directly from the commitments received in the *commit* phase. The emulation is straightforward:

- **Simulator for malicious P_s^* (simulator can extract).**
  - ***Commit* phase.** The extractor-simulator $\mathcal{S}^X$ (in the role of P_r in the simulated execution) receives from P_s^* a (real) commitment (58). Then it uses its power (in this example the trapdoor) to extract the committed value. Then, in the role of $\widehat{\text{P}}_s^*$ in the ideal world it sends the extracted value to $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ (97) – this is similar to what would happen if using the usual $\mathcal{F}_{\text{MCOM}}$ (90).
  - ***Open* phase.** If P_s^* in the simulated execution successfully opens the previously extracted value (59), then $\mathcal{S}^X$ asks $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ to open the commitment (101) – this is similar to what would happen if using the usual $\mathcal{F}_{\text{MCOM}}$ (93). Otherwise, if P_s^* aborts without opening, then $\mathcal{S}^X$ *emulates an abort*.
- **Simulator for malicious P_r^* (simulator cannot equivocate).**
  - ***Commit* phase.** In the ideal world, after $\widehat{\text{P}}_s$ interacts with $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ to commit a value, the simulator ($\mathcal{S}$) in the role of $\widehat{\text{P}}_r^*$ receives from $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ a (virtual) commitment (99) – compare this against the simple "OK" receipt that would have been received if using the usual $\mathcal{F}_{\text{MCOM}}$ (91). Then, $\mathcal{S}$ in the role of P_s in the simulated execution sends the commitment to the malicious P_r^* (58).
  - ***Open* phase.** In the ideal world, $\mathcal{S}$ receives from $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ the opening of the value (102) and verifies its correctness (103) – compare this against simply receiving the value and trusting on its correctness, as would happen if using the usual $\mathcal{F}_{\text{MCOM}}$ (94). Then, $\mathcal{S}$ in the simulated execution sends the same opening to

Template for ideal functionality $\mathcal{F}_{\overline{\text{X\&Q}}}[\mathscr{C}] \equiv \mathcal{F}_{\text{MCOM}(\overline{\text{X\&Q}})}[\mathscr{C}]$

Let $\mathscr{C}$ be a commitment scheme (i.e., at least computationally hiding and binding in respect to the implicit security parameter 1^κ), with respective *commit* and *open* phases. $\mathcal{F}_{\text{MCOM}(\overline{\text{X\&Q}})}[\mathscr{C}]$ activated with session identifier *sid* proceeds as follows, running with parties P_1, ..., P_n, an adversary $\mathcal{S}$, and a computational security parameter 1^κ. (Note: the suffix $[\ell]$ may be added to the notation to denote the obvious restriction to commitments of bit-strings of length ℓ.)

1. **Commit phase.** Upon receiving a message (`commit`, *sid*, *cid*, P_s, P_r, $(\ell, \overline{m})$) from P_s (the sender), if $\ell \in O(poly(\kappa))$, $\overline{m}$ (from the image set of commitments of messages of size ℓ) and $s, r \in [n]$, then record the tuple (*cid*, P_s, P_r, $(\ell, \overline{m})$), send the message (`receipt`, *sid*, *cid*, P_s, P_r, ℓ) to P_r (the receiver) and $\mathcal{S}$ and ignore subsequent messages with header (`commit`, *sid*, *cid*, P_s, P_r, ...); otherwise ignore the message.
2. **Open phase.** Upon receiving message (`open`, *sid*, *cid*, P_s, P_r, $(m, \underline{m})$) from P_s: if for some ℓ and $\overline{m}$ a tuple (*cid*, P_s, P_r, ℓ, m) has been recorded, then: verify $\mathscr{C}^{\text{Verify}}[1^\kappa](m, \underline{m}, \overline{m})$ (i.e., that the commitment and opening sent by P_s is consistent with the commitment scheme, the security parameter and the allowed length); then send the message (`open`, *sid*, *cid*, P_s, P_r, m) to P_r and $\mathcal{S}$ and ignore any subsequent messages with identifiers (...,*sid*,*cid*,P_s,P_r,...); otherwise ignore the message.
3. **Abort.** Upon receiving a message (`abort`, *sid*, *cid*, P_s, P_r) from P_s or P_r, then ignore any subsequent messages with identifiers (..., *sid*, *cid*, P_s, P_r, ...).

Fig. 10. Ideal functionality for multiple bit-string commitments not-X-but-Q

the malicious P_r^* (59). $\mathcal{S}$ outputs in the end whatever the simulated P_r^* outputs, including a possible early abort.

Below is the corresponding analysis (with obvious adjustments) for the $\overline{\text{X}}$&Q case.

Definition 7 (**equivocable commitment scheme**). *A real commitment scheme $\mathscr{C}$ (by definition already* hiding *and* biding*) is said to be equivocable if it securely emulates the ideal $\mathcal{F}_{\overline{\text{X\&Q}}}[\mathscr{C}]$. This is succinctly expressed in the equation below (115).*

$$(\mathscr{C} \text{ is equivocable}) \equiv \left((\exists \mathcal{S})(\forall \mathcal{Z}, \mathcal{A}) \texttt{IDEAL}_{\mathcal{F}_{\overline{\text{X\&Q}}}[\mathscr{C}], \mathcal{S}^{\mathcal{A}}, \mathcal{Z}} \overset{c}{\approx} \texttt{REAL}_{\mathscr{C}, \mathcal{A}, \mathcal{Z}} \right) \qquad (115)$$

For example, consider a real Q commitment scheme, where a setup phase endows the simulator with a trapdoor that enables equivocation directly in the *open* phase. The emulation is straightforward:

– **Simulator for malicious P_s^* (simulator cannot extract).**
  - ***Commit* phase.** The simulator ($\mathcal{S}$) in the role of receiver P_r in the simulated execution receives a commitment from P_s^* (58). Then, in the role of sender $\widehat{\text{P}}_s^*$ in the ideal world, $\mathcal{S}$ relays the commitment to $\mathcal{F}_{\overline{\text{X\&Q}}}$ (107) – compare against simply revealing the value as would happen if using the usual $\mathcal{F}_{\text{MCOM}}$ (90).
  - ***Open* phase.** If the sender P_s^* in the simulated execution successfully opens the committed value (59), which requires a successful verification (60), then $\mathcal{S}$ in the ideal world sends the same opening to $\mathcal{F}_{\overline{\text{X\&Q}}}$ (110) – compare against simply asking the usual $\mathcal{F}_{\text{MCOM}}$ to open the previously learned value (93). Otherwise, if P_s^* aborts without opening, then $\mathcal{S}$ *emulates an abort* in the ideal world.

– **Simulator for malicious P_r^* (simulator can equivocate).**
  - ***Commit* phase.** The equivocator-simulator ($\mathcal{S}^Q$) in the role of sender $\widehat{\mathrm{P}}_s^*$ in the ideal world receives from $\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}$ a receipt that a value has been committed (108) – this is similar to what would happen if using the usual $\mathcal{F}_{\mathrm{MCOM}}$ (91). Then, $\mathcal{S}^Q$ in the role of P_s in the simulated execution computes a commitment to a random message (of adequate length) and sends it to P_r^* (59).
  - ***Open* phase.** $\mathcal{S}^Q$ in the ideal world receives the value in clear from $\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}$ (112) – this is similar to what would happen if using the usual $\mathcal{F}_{\mathrm{MCOM}}$ (94). Then, $\mathcal{S}^Q$ in the simulated execution uses its equivocation power (in this example the trapdoor) to *open* such value to the malicious P_r^* (59). $\mathcal{S}$ outputs in the ideal world whatever P_r^* outputs in the simulated execution.

Remark (length of committed values). A subtle alternative for a $\overline{\mathrm{X}}\&\mathrm{Q}$ commitment scheme is to have the length ℓ not be revealed to the receiver during the *commit* phase (108), but only in the *open* phase. By revealing it directly in the *commit* phase, this definition becomes closer (in this aspect) to the ideal X&Q commitment functionality, for which (in the UC framework, where rewinding is not possible) the X property requires that the commit phase reveals some information about the length of the committed value.

Remark (separable X and Q). The suppression of properties is defined in the ideal world, not in the real world. Thus, a particular real commitment scheme $\mathscr{C}$ that emulates $\mathcal{F}_{\mathrm{X}\&\overline{\mathrm{Q}}}[\mathscr{C}_{\mathrm{X}}]$ (i.e., which by definition is *extractable*) might also emulate $\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}[\mathscr{C}_{\mathrm{X}}]$ (i.e., it might also be *equivocable*), and vice-versa. In particular, a commitment scheme $\mathscr{C}_{\mathrm{XQ}}$ that by definition is X&Q, i.e., one that emulates $\mathcal{F}_{\mathrm{MCOM}} \equiv \mathcal{F}_{\mathrm{X}\&\mathrm{Q}}$, is simultaneously (as expected) both *extractable* and *equivocable*, as it respectively emulates $\mathcal{F}_{\mathrm{X}\&\overline{\mathrm{Q}}}[\mathscr{C}_{\mathrm{XQ}}]$ and $\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}[\mathscr{C}_{\mathrm{XQ}}]$. Conversely (and also as intuitively expected), there are extractable commitment schemes and equivocable commitment schemes that are not simultaneously extractable-and-equivocable, i.e., that do not securely-emulate $\mathcal{F}_{\mathrm{X}\&\mathrm{Q}}$.

Remark (hiding and binding properties). Definitions 6 and 7 would make sense *per se* if the expression "real commitment" in the preamble was omitted, i.e., would make sense beyond the scope of commitment schemes. For example, a two phase scheme where both "commit" and "open" phases correspond to P_s sending the value in clear to P_r (and P_r then simply verifying that the values are equal) is extractable and binding, but is not a commitment scheme because it is not hiding (and for that reason also not equivocable). Similarly, a two phase scheme where the "commit" phase corresponds to P_s simply informing that a value is committed (but without actually sending anything else), and the "open" phase corresponds to P_s simply sending the value (and P_r simply accepting it), is hiding and equivocable, but is not a commitment scheme because it is not binding (and for that reason also not extractable). For these reasons it is mandatory to explicitly enforce that the X schemes and Q schemes are indeed commitment schemes.

It is interesting to notice that *equivocability* of the *open* phase implies that the *commit* phase is *hiding* and the opening phase is *revealing*; correspondingly, *extractability* of the *commit* phase implies *binding*. In other words, the combination of X-and-Q implies that a scheme is indeed a commitment scheme (i.e., hiding and binding).

B.3 A nested hybrid model

Even though the above conceptualization allows a differentiation of X and Q, there are potential problems associated with having to instantiate the virtual commitment scheme with a real (cryptographic) commitment scheme.

- **Syntactic differences.** Each of $\mathcal{F}_{\mathrm{X\&}\overline{\mathrm{Q}}}[\mathscr{C}_{\mathrm{X}}]$ and $\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}[\mathscr{C}_{\mathrm{Q}}]$ explicitly introduce a specific syntax of interaction, requiring the sender or receiver to interact differently from the case of an ideal $\mathcal{F}_{\mathrm{X\&Q}}$. Thus, if a protocol is proven secure in a hybrid model that uses one of the above defined functionalities, in practice it is not syntactically possible to just replace them with $\mathcal{F}_{\mathrm{X\&Q}}$. This may apparently contrast with the logic that a protocol secure without Q or X should also be secure when adding extra power (Q or X) to the simulator. After all, a commitment scheme $\mathscr{C}_{\mathrm{XQ}}$ that emulates $\mathcal{F}_{\mathrm{X\&Q}}$ also emulates $\mathcal{F}_{\mathrm{X\&}\overline{\mathrm{Q}}}[\mathscr{C}_{\mathrm{XQ}}]$ and $\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}[\mathscr{C}_{\mathrm{XQ}}]$. The problem is minor if when changing functionalities one is careful to consider the necessary syntactic transformations.
- **Interfering commitment schemes.** When using several ideal functionalities as defined above, a more relevant problem may arise from interference between the several underlying virtual commitment schemes, e.g., if a virtual scheme is malleable with respect to another virtual scheme (or even itself). This type of problem would not be present if using full-fledged X&Q ideal functionalities instead of real (non-ideal) commitment schemes. However, the goal set forth in this section is about suppressing an isolated property (e.g., Q or X) from the ideal functionality, but without relinquishing other useful properties (e.g., non-malleability) present in the typical X&Q ideal functionality $\mathcal{F}_{\mathrm{X\&Q}}$.

This subsection proposes a way to resolve the two above-mentioned problems. Instead of instantiating the virtual commitment scheme $\mathscr{C}_{\mathrm{V}}$ with a real commitment scheme, it can be instantiated itself with the standard ideal $\mathcal{F}_{\mathrm{MCOM}} \equiv \mathcal{F}_{\mathrm{X\&Q}}$ commitment functionality. This corresponds to a nested application of an ideal functionality, in the sense that the new $\mathscr{C}_{\mathrm{V}}$ (an element of the intended ideal functionality) is itself instantiated with an ideal functionality (itself also denoted hereafter as *virtual*, and also as a *second-level* ideal functionality $\widehat{\widehat{\mathcal{F}}}$). This is depicted in Fig. 11. In this new *nested-hybrid* model, it is mandated by definition that the simulator ($\mathcal{S}$) is not able to impersonate the virtual / second-level ideal functionality. Thus, whether or not the simulator is able to extract or equivocate a value in a simulation depends on whether or not such capability can be achieved without interaction with $\widehat{\widehat{\mathcal{F}}}$. Moreover, from the perspective of a real party, the interaction with any of these ideal functionalities is similar to that of an interaction with the standard ideal commitment functionality $\mathcal{F}_{\mathrm{X\&Q}}$. Intuitively, this means that if a protocol is proven secure in a type of ($\mathcal{F}_{\mathrm{X\&}\overline{\mathrm{Q}}}$,$\mathcal{F}_{\overline{\mathrm{X}}\&\mathrm{Q}}$)-nested-hybrid model, it is guaranteed to also be secure if any of the respective functionalities is directly replaced by a full-fledged $\mathcal{F}_{\mathrm{X\&Q}}$.

For example, consider the case of a ($\mathcal{F}_{\mathrm{X}}$)-nested-hybrid model, i.e., corresponding to a normal ($\mathcal{F}_{\mathrm{X\&}\overline{\mathrm{Q}}}[\mathscr{C}_{\mathrm{V}}]$)-hybrid model but where $\mathscr{C}_{\mathrm{V}}$ is replaced by a *non-impersonatable* $\mathcal{F}_{\mathrm{X\&Q}}$. Here, a simulator with black-box access to a malicious sender P_s^* can extract a value committed by P_s^*, because the value is sent in clear to the first-level functionality

Commitments in a nested-hybrid model

Note: $\widehat{\widehat{\mathcal{F}}}$ is the second-level ideal functionality (a.k.a. *virtual* functionality) that cannot be impersonated by a simulator. $\mathcal{F}_{\text{X}}$ and $\mathcal{F}_{\text{Q}}$ represent the ideal functionalities in the first-level, which can be impersonated by a simulator.

Nested-hybrid X-but-bot-Q functionality

$\mathcal{F}_{\text{X}} \equiv \mathcal{F}_{\text{X\&}\overline{\text{Q}}}[\widehat{\widehat{\mathcal{F}}}] \equiv \mathcal{F}_{\text{MCOM}(\text{X\&}\overline{\text{Q}})}[\widehat{\widehat{\mathcal{F}}}]$

Commit phase

$\text{input}_s \to \text{P}_s : (\texttt{commit}, \ell, m)$ (116)

$\text{P}_s \to \mathcal{F}_{\text{X\&}\overline{\text{Q}}} : (\texttt{commit}, \ell, m)$ (117)

$\mathcal{F}_{\text{X\&}\overline{\text{Q}}} \to \widehat{\widehat{\mathcal{F}}} : (\texttt{commit}, \ell, m)$ (118)

$\widehat{\widehat{\mathcal{F}}} \to \text{P}_r : (\texttt{OK}, \ell, \overline{m})$ (119)

Open phase

$\text{input}_s \to \text{P}_s : \texttt{open}$ (120)

$\text{P}_s \to \mathcal{F}_{\text{X\&}\overline{\text{Q}}} : \texttt{open}$ (121)

$\mathcal{F}_{\text{X\&}\overline{\text{Q}}} \to \widehat{\widehat{\mathcal{F}}} : \texttt{open}$ (122)

$\widehat{\widehat{\mathcal{F}}} \to \text{P}_r : (\texttt{open}, m)$ (123)

$\text{P}_r \to \text{output}_r : (\texttt{open}, m)$ (124)

Nested-hybrid not-X-but-Q functionality

$\mathcal{F}_{\text{Q}} \equiv \mathcal{F}_{\overline{\text{X}}\text{\&Q}}[\widehat{\widehat{\mathcal{F}}}] \equiv \mathcal{F}_{\text{MCOM}(\overline{\text{X}}\text{\&Q})}[\widehat{\widehat{\mathcal{F}}}]$

Commit phase

$\text{input}_s \to \text{P}_s : (\texttt{commit}, \ell, m)$ (125)

$\text{P}_s \to \widehat{\widehat{\mathcal{F}}} : (\texttt{commit}, \ell, m)$ (126)

$\widehat{\widehat{\mathcal{F}}} \to \mathcal{F}_{\overline{\text{X}}\text{\&Q}} : (\texttt{OK}, \ell)$ (127)

$\mathcal{F}_{\overline{\text{X}}\text{\&Q}} \to \text{P}_r : (\texttt{OK}, \ell)$ (128)

Open phase

$\text{input}_s \to \text{P}_s : \texttt{open}$ (129)

$\text{P}_s \to \widehat{\widehat{\mathcal{F}}} : \texttt{open}$ (130)

$\widehat{\widehat{\mathcal{F}}} \to \mathcal{F}_{\overline{\text{X}}\text{\&Q}} : (\texttt{open}, m)$ (131)

$\mathcal{F}_{\overline{\text{X}}\text{\&Q}} \to \text{P}_r : (\texttt{open}, m)$ (132)

$\text{P}_r \to \text{output}_r : (\texttt{open}, m)$ (133)

Fig. 11. Commitments in a nested-hybrid model (using succinct notation). For simplicity, the session identifier (*sid*), the sub-session identifier (*cid*, for commitment identifier) and other message context are left implicit. Simulation notes:

- In X&$\overline{\text{Q}}$ case: in the *commit* phase, $\mathcal{S}$ can impersonate $\mathcal{F}_{\text{X\&}\overline{\text{Q}}}$ to *extract* the value committed by P_s^* (117); in the *open* phase, $\mathcal{S}$ **cannot** *equivocate* the opening of m (123), as it cannot impersonate $\widehat{\widehat{\mathcal{F}}}$ (e.g., as if the channel between $\widehat{\widehat{\mathcal{F}}}$ and P_r^* were authenticated).
- In $\overline{\text{X}}$&Q case: in the *commit* phase, $\mathcal{S}$ **cannot** *extract* the value committed by P_s^* (126), as it cannot impersonate $\widehat{\widehat{\mathcal{F}}}$ (e.g., as if the channel between $\widehat{\widehat{\mathcal{F}}}$ and P_s^* were authenticated and encrypted); in the *open* phase, $\mathcal{S}$ can impersonate $\mathcal{F}_{\overline{\text{X}}\text{\&Q}}$ to *equivocate* the opening of m (132).

(117), which can be impersonated by the simulator. Conversely, a simulator with black-box access to a malicious receiver P_r^* cannot equivocate the opening of a value, because the opening of the value is sent from the second-level functionality to P_r^* (119).

B.4 Different models of simulation

The above discussion has considered alternative conceptualizations of X and Q (and non-X and non-Q), when a simulator interacts with ideal commitment functionalities during

a simulation. This conveys that proofs of security may consider different simulatability models, in what concerns the hybrid use of ideal functionalities.

- **Plain model.** When rewinding is allowed, then it is possible to consider the *plain model*, without any kind of ideal functionalities. Here, either no setup assumptions are required, or the protocol is augmented with a setup phase that results from interaction between the two parties, without reliance on ideal functionalities. In such model, $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ are assumed to be implemented by real sub-protocols between the two parties, in a way that allows $\mathcal{S}$ to have the necessary X and Q capabilities, respectively. In practice, these capabilities may derive either from implicit local rewinding in the respective phases (e.g., within some ZK sub-protocol) or/and from use of a secret trapdoor obtained upon rewinding in some implicit prior *setup* phase.
- **Hybrid model.** In a *hybrid model*, $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ are replaced by respective ideal functionalities $\mathcal{F}_{\mathrm{X\&}\overline{\mathrm{Q}}} \equiv \mathcal{F}_{\mathrm{MCOM}(\mathrm{X\&}\overline{\mathrm{Q}})}[\mathscr{C}_{\mathrm{X}}]$ and $\mathcal{F}_{\overline{\mathrm{X}}\mathrm{\&Q}} \equiv \mathcal{F}_{\mathrm{MCOM}(\overline{\mathrm{X}}\mathrm{\&Q})}[\mathscr{C}_{\mathrm{X}}]$. These two are impersonated by $\mathcal{S}$ when interacting with the black-box malicious party in the simulated execution. As mentioned before, there are two concerns here: i) the protocol must be specified with a syntax of interaction consistent with the used ideal functionalities; ii) the commitments must not cause undesired interferences between themselves, e.g., some kind of malleability (see further notes ahead).
- **Nested hybrid model.** In the *nested hybrid model*, $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ are replaced by ideal functionalities $\mathcal{F}_{\mathrm{X}} \equiv \mathcal{F}_{\mathrm{MCOM}(\mathrm{X\&}\overline{\mathrm{Q}})}[\mathcal{F}_{\mathrm{X\&Q}}]$ and $\mathcal{F}_{\mathrm{Q}} \equiv \mathcal{F}_{\mathrm{MCOM}(\overline{\mathrm{X}}\mathrm{\&Q})}[\mathcal{F}_{\mathrm{X\&Q}}]$, with the underlying virtual $\mathcal{F}_{\mathrm{X\&Q}}$ not being impersonatable by the simulator. As mentioned, in practice this model is syntactically equivalent to the $\mathcal{F}_{\mathrm{X\&Q}}$-hybrid model, but with the constructive (intended) limitation that X and Q are actively isolated, i.e., the simulator cannot take advantage of certain X or Q capabilities. Thus, as desired, this model does not leave room to argue that the proof might require a full-fledged X&Q commitment scheme, as could be argued if one would only know that a proof had been made in the usual $\mathcal{F}_{\mathrm{MCOM}}$-hybrid model.

In spite of the distinction between hybrid and nested-hybrid model, for the sake of simplicity the remainder of the paper uses a single notation when defining and analyzing simulations in any sort of hybrid model. In particular, the security analysis will simply refer to a hybrid model using ideal commitment functionalities $\mathcal{F}_{\mathrm{X}}$ and $\mathcal{F}_{\mathrm{Q}}$. Also, the syntax of interaction will be as if $\mathcal{F}_{\mathrm{X}}$ and $\mathcal{F}_{\mathrm{Q}}$ were actually both equivalent to $\mathcal{F}_{\mathrm{MCOM}} \equiv \mathcal{F}_{\mathrm{X\&Q}}$. Thus, it will remain implicit that the simulation corresponds to what was here defined as the nested-hybrid model. Specifically, in such interpretation one may be assured that when using $\mathcal{F}_{\mathrm{X}}$ or $\mathcal{F}_{\mathrm{Q}}$ the simulator will not even "try" to use Q or X, respectively. This simplification allows this section to not be essential for the understanding of the remaining analysis of the paper, but rather makes it a complementary source of intuition for the interested reader.

Remark (*interfering *commitment schemes). The new protocols devised in this paper make use of two underlying commitment schemes ($\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$). When using the nested-hybrid model, each of these schemes is replaced by an ideal commitment functionality that ensures complete independence between commitments – as if one were actually

using the $\mathcal{F}_{\text{X\&Q}}$-hybrid model. However, when considering real cryptographic instantiations, there may arise *interferences* that jeopardize the intended properties, e.g., *hiding*, *binding*, *extractability* or *equivocability* of an individual commitment scheme, or even *independence* of committed/opened values (broken by means of *malleability*). One notable exception, where such questioning is not needed, is if the instantiations are already proven to be universally composable (UC). However, one goal of this paper is precisely to allow use of underlying commitment schemes that are not necessarily full-fledged UC, namely that are not simultaneously X and Q. Also, there may be advantage in considering instantiations of $\mathscr{C}_{\text{X}}$ and $\mathscr{C}_{\text{Q}}$ that are different but related (e.g., same trapdoor – see §D.1). The matter of interfering commitment schemes is further discussed in §D.2.

C Coin-flipping simulatable-with-rewinding

This section analyzes coin-flipping in a stand-alone setting where simulation with rewinding is allowed. Subsection §C.1 clarifies several aspects about *coin-flipping into a well* – it reviews the early protocol proposed by Blum for coin-flipping by telephone [Blu83], it discusses the matter of *bias* and the security in case of a single coin-flip. Then, still as a "warm-up", Subsection §C.2 comments on the (non-)simulatability of protocols in the traditional template, in case the underlying commitment scheme lacks extractability (X) or equivocability (Q): the case of X&$\overline{\text{Q}}$ is shown to be non-simulatable; the case of $\overline{\text{X}}$&Q (as in Blum's protocol) is left in doubt, but an issue of unknown probability of abort is raised. Subsection §C.3 provides a proof of security (i.e., simulatability) of protocol #1: it specifies a simulator for the case of each corrupted party and analyzes the respective simulation, to argue the indistinguishability of distributions between the ideal and the real worlds. The somewhat intricate analysis of a super-polynomial upper bound to the number of rewindings for the simulation in case of a corrupted P_{B}^{*} is left to Subsection §C.4, showing that it leads to an expected polynomial number of rewindings.

C.1 Coin-flipping into a well

Blum's coin-flipping. The "coin-flipping by telephone" [Blu83] proposed by Blum uses a non-interactive unconditionally-hiding bit-commitment scheme, with trapdoor known by P_{B}.[19] The asymmetry of the protocol can be characterized by saying that "P_{B} flips coins to P_{A}", namely because P_{A} learns the result of the coin-flipping and then can *decide* whether or not to let P_{B} learn it as well. In a setup phase of the protocol, P_{B} chooses the commitment scheme parameter (a Blum integer) and convinces P_{A} that it is correct. This is achieved with P_{B} giving a (honest-verifier) ZKPoK of the respective trapdoor (the integer factorization of the Blum integer), which P_{B} keeps hidden from P_{A}. The protocol then proceeds with P_{A} committing to her random contribution (a vector of bits) and then P_{B} sending his random contribution (a vector of bits with the same length) to P_{A}. At this

[19] In the specific proposal, a commitment of bits is a vector of squares modulo a Blum integer; the respective opening is a vector of square-roots with appropriate Jacobi Symbols. A Blum integer is the product of two prime powers, where each prime is congruent with 3 modulo 4, and each power has an odd exponent. For a fixed Blum integer, the Jacobi Symbol is a completely multiplicative function that maps any group element into 1 or -1.

point, P_A can privately obtain the coin-flipping result as the XOR of the contribution of P_A and the contribution of P_B. Eventually, "whenever ... [P_A] wants to prove to P_B what sequence of ... random bits was flipped to her", P_A opens her commitments, thus also letting P_B compute the final bit-string.

It might be left to interpretation whether or not the final step (P_A opening her contribution to P_B) is necessary to consider the execution successful. Actually, in the original proposed protocol, the parties sign and timestamp some exchanged messages, thus allowing a complementary "Judge's protocol", where all signed messages can be subpoenaed and then the judge can either assert that the "protocol is declared terminated" or it can "enforce completion of the protocol". This setting is not considered in this paper, as it goes beyond the two-party case, but it interestingly solves the *bias* problem discussed ahead.

Definition 8 (**early-abort**). *In the context of a coin-flipping protocol (namely one following the traditional template),* early-abort *denotes the action of a (malicious) party (P_p) aborting the execution before revealing her own contribution, conditioned to not aborting before the other party ($P_{\bar{p}}$) becoming bound to her respective contribution.*

Definition 9 (**unfair-abort**). *In the context of a coin-flipping protocol,* unfair-abort *denotes the action of a (malicious) party (P_p) aborting the execution after* learning something *about the bit-string outcome (i.e., in the sense of breaking* semantic hiding*), but before letting the other party learn* something *about it.*

Clearly, an *unfair-abort* is an *early-abort*, and in the case of the traditional template only P_A^* is able to perform it (by aborting after (84), but before (85)). P_B is also able to do an *early-abort* in the traditional template (by aborting after (82) but before (84)). The consideration of *unfair-abort* and *early-abort* (and respective probabilities) was a main motivation to devise protocol #1 in this paper, in a setting where simulation with rewinding is allowed. In particular, the new protocol is devised in a different template where *early-abort* is still possible by P_A but not in the form of an *unfair-abort*.

***A note on* bias.** The traditional template, which fits (and is actually suggested by) Blum's protocol, inherently allows an *unfair-abort* by a malicious P_A^*, i.e., aborting before opening her contribution (step 3) but after seeing the contribution of the honest P_B (step 2), consequently allowing the output of P_B to be *biased*. In fact, this is allowed in the ideal functionality of coin-flipping into a well (Fig. 5, Fig. 6). In the case of flipping a single coin, *bias* is defined as the absolute value of the distance between one-half and the probability that the honest party outputs a particular bit value.[20] For example, if P_A^* aborts the coin-flipping execution whenever realizing that the final result is an undesired output, then the output of P_B becomes biased. This happens even if the protocol defines a default output mechanism for P_B when facing an *early-abort* by P_A^*. For example, the *bias* is one-forth if P_B outputs a default random bit in case of *early-abort* by P_A^*, which means P_A^* can induce her desired output with probability three-fourths (ignoring

[20] If flipping many coins, this can be generalized to the *bias* of some predicate of the bit-string.

the aborted executions). This *bias* is not a security violation when considering an ideal functionality that also allows *unfair-abort*, as considered in this paper.[21]

Single coin-flip into a well (with rewinding). The traditional template is a prototype for a *coin-flipping into the well* protocol [Gol04, §7.4.3.1], i.e., allowing two parties to obtain as output the same uniformly random bit except if a malicious party performs *early-abort*. In a simulation setting allowing rewinding, and when flipping a single coin, a protocol following the traditional template can be proven secure (i.e., simulatable) regardless of local X or Q properties of the commitment scheme, i.e., it holds for any commitment scheme (hiding and binding). In particular, in the case of a malicious P_A^*, the simulator $\mathcal{S}$ (in the role of P_B in a simulated execution) can use rewinding to test P_A^* with both possible bits in step 2. If P_A^* does *early-abort* in both cases, then $\mathcal{S}$ can safely *emulate an abort* in the ideal world; if P_A^* does not *early-abort* in at least one case, then $\mathcal{S}$ can determine whether or not to abort in the ideal world. More generally, the same type of proof can be used for bit-strings of length logarithmic in the computational security parameter, i.e., whenever the space of possible contributions of P_B is of polynomial size. However, when flipping many coins (e.g., linear in the security parameter) in parallel, some extra properties are necessary from the commitment scheme.

C.2 (Non-)simulatability of the traditional template

Definition 10 (explicit rewinding). *If simulation with rewinding is allowed,* explicit rewinding *denotes the action, performed by $\mathcal{S}$, of rewinding the black-box adversary in a simulated execution in a hybrid model where the underlying commitment schemes are replaced by respective ideal functionalities. In other words, these rewindings do not take in consideration possible* implicit *rewindings that might be necessary in a simulation where the ideal commitment functionalities are replaced by real sub-protocols.*

For a coin-flipping protocol following the traditional template, if the underlying commitment scheme is X&Q then simulatability is trivially possible without *explicit rewinding*. The situation is different if either X or Q properties are missing from the commitment scheme, namely in regard to the *expected number of explicit rewindings* ($E[\#rw]$). For a successful simulation, once $\mathcal{S}$ in the ideal world receives the random bit-string from $\mathcal{F}_{\text{MCF}}$, $\mathcal{S}$ must be able to induce in the simulated execution the perspective of the other black-box party obtaining the same final bit-string (the XOR of two contributions), regardless of then it being rejected (e.g., by means of abort) or accepted (and possibly altered when outputted to $\mathcal{Z}$) by the party.

When using an extractable-but-not-equivocable (X&$\overline{Q}$) commitment scheme. In case of a malicious P_A^*, a one-pass simulator (in the role of P_B in the simulated execution) can always succeed, by locally extracting the contribution of P_A^* (step 1) and then still be in a position to calculate and send the *needed complementary contribution* of P_B (step 2). However, the case of a malicious P_B^* is more problematic, even if P_B^* never

[21] By increasing the number of rounds in the protocol (i.e., necessarily deviating from the traditional template), *bias* could be reduced at most approximately proportionally to the inverse of the number of rounds [Cle86, MNS09].

aborts. This is because the contribution of P_B^* (step 2) may depend on the commitment of the contribution of P_A (step 1), e.g., based on some one-way function.[22] Thus, with overwhelming probability the simulator is not able to induce the intended final bit-string in the simulated execution, except perhaps after an expected number of rewindings that is super-polynomial in the target length.

When using a not-extractable-but-equivocable ($\overline{X}$&Q) commitment scheme. The original protocol of Blum for coin-flipping by telephone [Blu83] fits the traditional template instantiated with a $\overline{X}$&Q commitment scheme, namely an unconditionally-hiding commitment scheme with trapdoor.

- *local equivocation* of the contribution of P_A is possible by endowing the equivocator-simulator ($\mathcal{S}^Q$, in the role of P_A in the simulated execution) with knowledge of the trapdoor of the bit-commitment scheme. The trapdoor can be extracted in the setup phase, with $\mathcal{S}^Q$ rewinding the state of P_B in order to get responses to different challenges of the ZKPoK. (This setup phase and respective rewinding are implicit in the traditional template.) Then, in the phase of *opening* the contribution of P_A (step 3), $\mathcal{S}^Q$ uses the trapdoor to *locally* equivocate each bit-commitment of P_A to any needed bit value (namely such that the combination of contributions of both parties is equal to the one decided by $\mathcal{F}_{\text{MCF}}$ in the ideal world).[23]
- *local extraction* of the contribution of P_A is not possible, since the unconditionally-hiding commitment does not contain any extractable information. Nonetheless, rewinding allows *non-local extraction* if P_A does not do *early-abort*. First, the extractor-simulator ($\mathcal{S}^X$, in the role of P_B in the simulated execution) proceeds the simulation until the step where P_A *opens* her contribution (step 3). Then, $\mathcal{S}^X$ rewinds to immediately before the step where P_B has to decide a contribution (step 2). This means that, *after* extracting the contribution of P_A, $\mathcal{S}^X$ is still able to *go back in time* and choose a new contribution for P_B. At first glance this could seem equivalent to *local extraction*, as if $\mathcal{S}^X$ had extracted the value when P_A committed to it (step 1), but there is an essential difference related with *unfair-abort*.

Even though the rewinding used for *non-local* extraction does not affect the contribution that P_A can *open* (step 3), because by definition a commitment is *binding*, it may affect the willingness of a malicious P_A^* to abort without *opening* her contribution. Specifically, between two executions (the first and the one after rewinding) with different contributions by P_B (and thus also with different perspectives of a final bit-string), the probability of *unfair-abort* by P_A^* may vary. In particular, an (arbitrary and probabilistic) decision-criterion of P_A^* to do *unfair-abort* may be unknown to $\mathcal{S}^X$ and dependent on

[22] Or uniformly sampled using *fresh* (i.e., non-rewindable) randomness (if so allowed by the computational model) after each rewinding – this is not a standard model in the literature.

[23] It is worth noticing that equivocation could be simply based on an equivocable commitment applied to a collision-resistant hash of the contribution of P_A. A related idea appears in [Hal95] based on claw-free permutation pairs. Then, the *opening* would simply consist on sending the full contribution of P_A and equivocating the opening of the respective (short) hash. This idea is used in protocol #1 directly for the contribution of P_B, and in protocol #2 for other large elements that require equivocation.

the value of the contributions of P_A and P_B. Intuitively, this poses a difficulty to defining a suitable $\mathcal{S}^X$, namely (as also pointed out by Lindell [Lin03]) if different execution paths mix negligible and noticeable probabilities of non-abort, i.e., if the probability of non-abort (when taken across different execution paths) is neither negligible nor noticeable. Furthermore, even for certain restricted classes of malicious behavior for which the author of this paper can find a suitable $\mathcal{S}^X$, the simulation found requires $E[\#rw]$ to be at least of the order of the inverse of the *initial-probability-of-no-early-abort*, whereas the new protocol #1 (Fig. 1) has a simulator with $E[\#rw]$ being less than two.

It is worth pointing out that Blum's coin-flipping protocol is adequate for many practical purposes. In particular, it is simulatable with a single *explicit rewinding* if it is assumed that *early-abort* never occurs. Still, the observed difficulty, combining the process of *early-abort* with the non-locality of extraction of the contribution of P_A^*, was the motivation to devise in this paper a new protocol, bypassing the mentioned difficulty by using a different template.

It is left as open problem proving the non-simulatability of Blum's protocol (i.e., when using a $\overline{\text{X}}$&Q commitment scheme within the traditional template). Alternatively, if the protocol is simulatable, then it remains to find a suitable simulator for the case of corrupted P_A^* and calculate the respective (polynomial) $E[\#rw]$.

C.3 Proof of security of coin-flipping protocol #1

Protocol #1 was described in §4.2 (Fig. 1). P_A is the first party to learn the bit-string outcome; i.e.: when P_A is not corrupted it receives `start-1` from $\mathcal{Z}$ (4); when P_B is not corrupted it receives `start-2` from $\mathcal{Z}$ (5).

The proof of security assumes that the PRG and the CR-Hash are cryptographically secure and that the adversary that corrupts P_B^* is rewindable. For the case of a corrupted P_A^* there is no need for *explicit rewinding*, because the simulator is able to *locally* extract the contribution of P_A^* (10) and *locally* equivocate the contribution of P_B (13). Nonetheless, when instantiating in practice the commitment schemes it may be necessary or desirable, in a setup phase or/and during the protocol, to use rewinding of P_A^* in order to bring about the X and Q properties.

The proof shows simulatability on the side of each possible corrupted party. The plain model considers underlying commitment schemes $\mathscr{C}_{\text{X}}$ and $\mathscr{C}_{\text{Q}}$ that are directly assumed to have the needed X or Q properties. The hybrid model uses ideal commitment functionalities $\mathcal{F}_{\text{X}}$ and $\mathcal{F}_{\text{Q}}$, from which the simulator does not take advantage of Q and X, respectively. See §B.4 for a possible different interpretation of the suppression of X and Q properties from ideal functionalities (the nested-hybrid model, applicable but not made explicit hereafter). The plain and hybrid models are described side by side.

Simulation prologue. The following prologue applies to the case of corruption on any side (i.e., for corruption of P_A^* or P_B^*): $\mathcal{S}$ prepares a simulation of the real world, controlling all communication links to and from the adversary $\mathcal{A}$, and having black-box access to and rewinding capability over $\mathcal{A}$; once the simulator $\mathcal{S}$ receives an input from $\mathcal{Z}$, it relays it to $\mathcal{A}$ embedded in the simulated context; in the perspective of $\mathcal{A}$, $\mathcal{S}$ does not exist and the communication takes place directly with $\mathcal{Z}$. if $\mathcal{A}$ corrupts P_p^* in the simulated execution, for some $p \in \{A, B\}$, making it a malicious, then $\mathcal{S}$

correspondingly corrupts $\widehat{\mathrm{P}}_p^*$ in the ideal world; the input sent from $\mathcal{Z}$ to the respective corrupted ideal party $\widehat{\mathrm{P}}_p$ (61) is also intercepted by $\mathcal{A}$ and relayed to the simulated P_p^* (4), thus being intercepted by $\mathcal{A}$.

C.3.1 Simulator for corrupted P_A^*

- **Begin simulation.** In the ideal world, once $\mathcal{S}$ receives a message (`cf-start-2-receipt`, *sid*, (*cfid*, $\mathrm{P_A}$, $\mathrm{P_B}$), ℓ) from $\mathcal{F}_{\mathrm{MCF}}$, it knows that the honest $\widehat{\mathrm{P}}_\mathrm{B}$ has been correspondingly activated by $\mathcal{Z}$ (62). Thus, $\mathcal{S}$ is capable of impersonating an honest $\mathrm{P_B}$ in the simulated execution.
- **Commit contribution of $\mathrm{P_B}$.** $\mathcal{S}$ selects a random hash value (7). Then: in the plain model, $\mathcal{S}$ in the role of $\mathrm{P_B}$ acts as sender in the commit phase of $\mathscr{C}_\mathrm{Q}$ to commit the hash to the receiver P_A^* (8); in the hybrid model, $\mathcal{S}$ impersonates $\mathcal{F}_\mathrm{Q}$ notifying the receiver P_A^* that a value has been committed ((91), (128)) (it is OK to leak the length of the committed hash).
- **Get contribution of $\mathrm{P}_\mathbf{A}^*$.** In the plain model, $\mathcal{S}$ (impersonating $\mathrm{P_B}$ as receiver in a X-*commit* phase of $\mathscr{C}_\mathrm{X}$) waits to receive from P_A^* the commitment of a seed (10). In this plain model, $\mathcal{S}$ uses its X power to extract the seed. In the hybrid model, $\mathcal{S}$ impersonates $\mathcal{F}_\mathrm{X}$ directly receiving the seed of $\mathrm{P_A}$ ((90), (117)). Then, $\mathcal{S}$ in the role of $\mathrm{P_B}$ receives from P_A^* a masked version of her contribution (12). $\mathcal{S}$ calculates the PRG expansion of the seed (as specified in (17)) and uses the result to unmask the masked contribution of P_A^*, thus obtaining the contribution of P_A^*.
- **Get bit-string from $\mathcal{F}_{\mathrm{MCF}}$ (the ideal coin-flipping functionality).** If the simulated execution has aborted at any point, due to some malicious behavior by P_A^* that would have been detected by $\mathrm{P_B}$, then $\mathcal{S}$ *emulates an abort* in the ideal world, namely by sending (`cf-abort-1`, *sid*, (*cfid*, $\mathrm{P_A}$, $\mathrm{P_B}$)) to $\mathcal{F}_{\mathrm{MCF}}$ (76) (which will lead the ideal $\widehat{\mathrm{P}}_\mathrm{B}$ to also abort (78)) and then outputting in the ideal world whatever $\mathcal{A}$ outputs in the simulated execution. Otherwise, if the previous simulation steps were successful, then $\mathcal{S}$ in the role of $\widehat{\mathrm{P}}_\mathrm{A}^*$ in the ideal world sends an honest message (`cf-start-1`, *sid*, (*cfid*, $\mathrm{P_A}$, $\mathrm{P_B}$), ℓ) to $\mathcal{F}_{\mathrm{MCF}}$ (63). $\mathcal{S}$ then receives back from $\mathcal{F}_{\mathrm{MCF}}$ a random bit-string (i.e., the flipped coins) (69), namely in a message of the form (`cf-deliver-1`, *sid*, (*cfid*, $\mathrm{P_A}$,$\mathrm{P_B}$), m), because the honest $\widehat{\mathrm{P}}_\mathrm{B}$ has also requested a respective start of such coin-flipping. In possession of the bit-string decided by $\mathcal{F}_{\mathrm{MCF}}$, henceforth denoted as *target outcome*, $\mathcal{S}$ computes the XOR of the target outcome with the contribution extracted from P_A^*, thus determining what it the *needed complementary contribution* of $\mathrm{P_B}$ in the simulated execution.
- **Equivocate contribution of $\mathrm{P_B}$.** $\mathcal{S}$ in the role of $\mathrm{P_B}$ in the simulated execution sends this *needed complementary contribution* to P_A^* (14) and computes its hash (the *needed hash*), as if it had been done in (7). Then, in the plain model, $\mathcal{S}$ uses its Q power to equivocate the *open* of the *needed hash* (13). In the hybrid model, $\mathcal{S}$ simply impersonates $\mathcal{F}_\mathrm{Q}$ sending the hash to P_A^* ((95), (132)).
- **Determine abort vs. non-abort.** In the plain model, $\mathcal{S}$ waits to receive from P_A^* the opening of the seed of P_A^* (16). In the hybrid model, P_A^* simply asks $\mathcal{F}_\mathrm{X}$ (impersonated by $\mathcal{S}$) to open the seed ((93),(117)) that had been previously stored by $\mathcal{S}$, which means that $\mathcal{S}$ just accepts the value that it had previously extracted. If the

seed is not properly opened (in the plain model) or not opened at all (in the hybrid model), then $\mathcal{S}$ in the role of $\widehat{\mathrm{P}}_{\mathrm{A}}^{*}$ in the ideal world sends (`cf-abort-1`, *sid*, (*cfid*, $\mathrm{P_A}$, $\mathrm{P_B}$)) to $\mathcal{F}_{\mathrm{MCF}}$ (75). Otherwise, if the seed is correctly opened (to the value already previously extracted by $\mathcal{S}$), then $\mathcal{S}$ sends the message (`cf-OK`, *sid*, (*cfid*, $\mathrm{P_A}$, $\mathrm{P_B}$)) to $\mathcal{F}_{\mathrm{MCF}}$ (70). Finally, $\mathcal{S}$ outputs (to $\mathcal{Z}$) in the ideal world whatever $\mathcal{A}$ outputs in the simulated execution.

C.3.2 Simulation analysis when corrupted $\mathbf{P_A^*}$

The above simulation requires a *single pass*, i.e., it is made without *explicit rewinding*, thus making trivial the analysis of the simulation in the ($\mathcal{F}_{\mathrm{X}}$,$\mathcal{F}_{\mathrm{Q}}$)-hybrid model. Essentially, $\mathcal{S}$ in the role of $\mathrm{P_B}$ in the simulated execution is able to directly extract the contribution of $\mathrm{P_A^*}$ and decide the *needed complementary contribution* of $\mathrm{P_B}$ in time to directly equivocate it. Thus, the probabilistic distribution of the *global* output is equal to the one in the ideal world, because $\mathcal{S}$ always induces the needed final output, except if $\mathrm{P_A^*}$ aborts, case in which the emulation of abort happens with the same probability.

In the plain model, when $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ are real individual commitment schemes (i.e., hiding and binding) with respective X and Q properties, the indistinguishability of the global output requires that the commitment schemes retain their properties in spite of there being two commitment schemes in use in the same protocol. Hypothetically, an unfortunate combination of $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ might give extra power to $\mathrm{P_A^*}$ (i.e., to $\mathcal{A}$ or $\mathcal{Z}$), allowing her (with noticeable probability) to prevent $\mathcal{S}$ from inducing the needed target outcome, and/or allowing her to distinguish the equivocated opening from an honest opening. The most obvious necessary condition of non-interference is that $\mathscr{C}_{\mathrm{Q}}$ is *non-malleable with respect to opening* of $\mathscr{C}_{\mathrm{X}}$, or otherwise $\mathrm{P_A^*}$ could potentially equivocate[24] her seed during the *open* phase of $\mathscr{C}_{\mathrm{X}}$, thus preventing $\mathcal{S}$ from inducing the intended final bit-string. A concrete secure instantiation based on the DDH assumption is given in §D.1. The (im)possibility of interferences is discussed in §D.2.

C.3.3 Simulator for corrupted $\mathbf{P_B^*}$

- **Begin simulation.** In the ideal world, once $\mathcal{S}$ receives a message (`cf-start-1-receipt`, *sid*, (*cfid*, $\mathrm{P_A}$, $\mathrm{P_B}$), ℓ) from $\mathcal{F}_{\mathrm{MCF}}$, it knows that the honest $\widehat{\mathrm{P}}_{\mathrm{A}}$ has been correspondingly activated by $\mathcal{Z}$ (61). Thus, $\mathcal{S}$ is capable of impersonating an honest $\mathrm{P_A}$ in the simulated execution.
- **Get commitment to hash of contribution of $\mathbf{P_B^*}$.** $\mathcal{S}$ waits for a commitment of a hash by $\mathrm{P_B^*}$ in the simulated execution. More specifically: in the plain model, $\mathcal{S}$ impersonates $\mathrm{P_A}$ as a *receiver* in the commit phase of $\mathscr{C}_{\mathrm{Q}}$ (8), receiving a commitment to the hash; in the hybrid model, $\mathcal{S}$ impersonates $\mathrm{P_A}$ receiving a notification that a value has been committed ((91),(128)).
- **First iteration of committing the contribution of $\mathbf{P_A}$.** $\mathcal{S}$ selects a random seed for $\mathrm{P_A}$ (9) and impersonating $\mathrm{P_A}$ commits to it, as follows: in the plain model, $\mathcal{S}$ acts as *sender* in the commit phase of $\mathscr{C}_{\mathrm{X}}$ (10); in the hybrid model, $\mathcal{S}$ simply sends to $\mathcal{F}_{\mathrm{X}}$ a request to commit the seed (117), thus leading $\mathcal{F}_{\mathrm{X}}$ to notify $\mathrm{P_B^*}$ that a value has

[24] Here it would be enough to gain the ability to equivocate to a single different value.

been committed (119). After the seed is committed, $\mathcal{S}$ selects a random bit-string of the target length (11), denoted *masked contribution*, and sends it to P_B^* (12).

– **First iteration of opening of hash of contribution of $\mathbf{P}_\mathbf{B}^*$.** $\mathcal{S}$ then waits to receive a correct opening of the committed hash of P_B^* (13), as follows: in the plain model, $\mathcal{S}$ in the role of P_A acts as receiver in the *open* phase of $\mathscr{C}_\mathrm{Q}$ (13); in the hybrid model, $\mathcal{S}$ receives from $\mathcal{F}_\mathrm{Q}$ the hash of the contribution of P_B ((94), (132)). After the opening of the hash, $\mathcal{S}$ in the role of P_A waits to receive the contribution of P_B^* in clear (14). If P_B^* aborts the execution before revealing his contribution (or if the revealed contribution is not consistent with the respective hash (15)) but after receiving the commitment of P_A, then $\mathcal{S}$ *emulates an abort* in the ideal world. More specifically, in this case $\mathcal{S}$ sends (`cf-abort-2`, *sid*, (*cid*, P_A, P_B)) to $\mathcal{F}_\mathrm{MCF}$ (74), thus leading $\widehat{\mathrm{P}}_\mathrm{A}^*$ in the ideal world to receive notice of this *abort* (75) without receiving the target bit-string from $\mathcal{F}_\mathrm{MCF}$, and $\mathcal{S}$ outputs in the ideal world whatever P_B^* outputs in the simulated execution. Otherwise, if P_B^* in the simulated execution reveals his contribution, then $\mathcal{S}$ in the ideal world sends (`cf-start-2`, *sid*, (*cid*, P_A, P_B), ℓ) to $\mathcal{F}_\mathrm{MCF}$ (64).

– **Get bit-string from $\mathcal{F}_\mathbf{MCF}$.** Once $\mathcal{F}_\mathrm{MCF}$ receives the `cf-start-2` message from $\mathcal{S}$ in the role of the ideal $\widehat{\mathrm{P}}_\mathrm{B}^*$, and since an honest ideal $\widehat{\mathrm{P}}_\mathrm{A}$ has also sent a respective `cf-start-1` message, $\mathcal{F}_\mathrm{MCF}$ calculates a random bit-string (the target bit-string) (68) and sends it to the ideal $\widehat{\mathrm{P}}_\mathrm{A}$ (69), namely in a message of the form (`cf-deliver-1`, *sid*, (*cfid*, P_A, P_B), m). $\mathcal{F}_\mathrm{MCF}$ also sends a receipt of such message (but without the random bit-string) to $\mathcal{S}$, namely (`cf-deliver-1-receipt`, *sid*, (*cfid*, P_A, P_B)) (not shown in Fig. 5, but described in Fig. 6). Since $\widehat{\mathrm{P}}_\mathrm{A}$ is assumed to be honest, it will accept the bit-string decided by $\mathcal{F}_\mathrm{MCF}$, confirming it by sending message (`cf-OK`, *sid*, (*cfid*, P_A, P_B)) to $\mathcal{F}_\mathrm{MCF}$ (70). This means that $\mathcal{F}_\mathrm{MCF}$ will then send the target bit-string also to $\widehat{\mathrm{P}}_\mathrm{B}^*$ (71) and respective receipt to $\mathcal{S}$. Furthermore, since $\mathcal{S}$ is controlling the ideal P_B^*, it can read the message sent to P_B^*, namely (`cf-deliver-2`, *sid*, (*cid*, P_A, P_B), m), which includes the decided bit-string m.

– **Induce the target outcome.** $\mathcal{S}$, now knowing the target bit-string and the contribution of P_B^*, computes what is the *needed complementary contribution* of P_A. $\mathcal{S}$ rewinds the state of P_B^* to the step prior to P_A having committed to a seed (10). Then, $\mathcal{S}$ selects a new random seed (9), commits to it (using the adequate model: plain (10) or hybrid ((90),(117))), computes its PRG-expansion (as in (17)) and then XOR's it with the complementary contribution of P_A, thus obtaining a new masked contribution (instead of a random one as in (11)).

  Then, $\mathcal{S}$ in the role of P_A sends the new masked contribution to P_B^* (12) and waits for P_B^* to open again his committed hash (using the adequate model: plain (13) or hybrid ((94), (123)) and to reveal his contribution (14). If the contribution of P_B^* is successfully open without abort, then $\mathcal{S}$ proceeds with the simulation by opening the seed of P_A (using the adequate model: plain (16) or hybrid ((93),(121)), and then it outputs in the ideal world (in the role of the ideal $\widehat{\mathrm{P}}_\mathrm{B}^*$) whatever P_B^* outputs in the simulated execution. Otherwise, if P_B^* aborts without opening again his contribution, then $\mathcal{S}$ rewinds again and again, until: (i) either P_B^* successfully opens the expected contribution, case in which $\mathcal{S}$ outputs in the ideal world whatever $\mathcal{A}$ outputs in the simulated execution); or (ii) the number of rewindings reaches an appropriate

super-polynomial upper-bound `#rw-bound` (see description below, still allowing the expected number of rewindings to be polynomial), case in which $\mathcal{S}$ *emulates an abort* in the ideal world.

C.3.4 Simulation analysis when corrupted P_B^*

In the *hybrid model*, it is clear that, whenever the output does not involve an emulation of abort in the ideal world, $\mathcal{S}$ in the simulated execution is able to induce the target-outcome that $\mathcal{F}_{\text{MCF}}$ decides in the ideal world. Since $\mathcal{S}$ only *emulates an abort* if either P_B^* aborts in the first iteration or if the upper-bound `#rw-bound` is reached, it follows that in the ideal world the probabilities of non-*early-abort* are equal or higher, but not smaller. In order to prove indistinguishability between the two worlds, it needs to be proven that this increase is at most negligible. Furthermore, for the defined $\mathcal{S}$, the only non-null contribution for the $E[\#rw]$ comes from simulation paths whose first iteration does not lead $\mathcal{A}$ to *early-abort*. Thus, an adequate upper-bound `#rw-bound` for the number of rewindings needs to be defined: (i) being sufficiently large such that the increase of probability of *early-abort* in the ideal world is at most by a negligible amount; and (ii) being sufficiently small such that $E[\#rw]$ is polynomial.

In the next subsection it is shown that it is possible to define an appropriate super-polynomial bound `#rw-bound`, if the contribution of P_A is hidden in a *strong* sense before the step when P_B^* reveals his contribution. It is shown as well that such bound leads $E[\#rw]$ to be upper-bounded by a small constant, namely less than 2. It is left as an open problem whether there might be a different suitable simulation requiring only a strict polynomial number of rewinding.

In the *plain* model, the simulation might hypothetically fail only in a contrived case where the combined instantiation of $\mathscr{C}_{\text{X}}$ and $\mathscr{C}_{\text{Q}}$ somehow weakens some of the necessary properties, e.g., breaking *binding* of $\mathscr{C}_{\text{Q}}$ or/and *hiding* of $\mathscr{C}_{\text{X}}$.[25] This is further discussed in §D.2. In this Subsection the proof follows assuming the hybrid model, where the commitments are replaced by ideal commitment functionalities $\mathcal{F}_{\text{X}}$ and $\mathcal{F}_{\text{Q}}$.

C.3.5 Other simulation settings

Is is left for future exploration the consideration of different security settings.

It would be interesting, as a natural extension of this work, to explore protocol adjustments that may achieve security under adaptive corruption, i.e., with an adversary deciding which party to corrupt only after a protocol execution has started, while at the same time retaining the high efficiency: low communication complexity and computational complexity amortized to a PRG and CR-Hash.

As another extension, though in a different computational model, it would be interesting to explore which protocol/simulation adjustments might lead to simulatability in a quantum setting, e.g., using quantum rewinding techniques [DL09], when parties have quantum-computational power and may exchange either classical or quantum messages.

[25] For the given protocol structure, the case of malleability of $\mathscr{C}_{\text{Q}}$ under a malicious behavior by P_B^* would be simply equivalent to losing binding of $\mathscr{C}_{\text{Q}}$, because P_B^* is the first party to commit.

C.4 Bound on number of rewindings

This subsection considers defining a bound for the number of rewinding for the simulation in case of corrupted $\mathrm{P}_{\mathrm{B}}^{*}$.

A problem if there is no upper-bound `#rw-bound` on the number of rewindings. For the specific simulator defined in §C.3.3), if an upper-bound (`#rw-bound`) on the number of rewindings does not exist, i.e., if it is infinite, then $E[\#rw]$ is also infinite whenever there is at least one simulation path that involves an infinite number of rewindings. Such a path exists when, for example, the malicious behaviors by $\mathrm{P}_{\mathrm{B}}^{*}$ determines that it only does not to do *early-abort* if it receives a specific *masked-contribution* bit-string t_A (12). If $\mathrm{P}_{\mathrm{B}}^{*}$ receives this exact bit-string (even though this happens only with probability negligible in the length of the contribution), then it opens his hash h_B (13) and reveals his contribution χ_B (14), thus allowing $\mathcal{S}$ to compute the *needed complementary contribution* $\chi_A = \chi_B \oplus \chi$ (where χ is the value received from $\mathcal{F}_{\mathrm{MCF}}$in the ideal world (69)). Then, there is only a negligible probability (in the target length subtracted by the seed length) that there is any seed s' for which the first masked contribution t_A (the only one to which this adversary would not abort) would unmask to the needed contribution χ_A of P_{A}. In other words, there is an overwhelming probability that $\mathrm{P}_{\mathrm{B}}^{*}$ would do *early-abort* in all subsequent iterations (with $\mathcal{S}$ always committing χ_A). This phenomenon is inherent to the cardinality of the masks (i.e., cardinality of the seeds) being exponentially smaller than the cardinality of the possible contributions, which implies that the contribution of P_{A} is not unconditionally hidden, even if the seed itself is unconditionally hidden and even if the PRG is ideal (e.g., in a hybrid model).

A problem if `#rw-bound` is polynomial. For the specific simulator defined in §C.3.3, if `#rw-bound` is polynomial, then it is easy to construct a malicious $\mathrm{P}_{\mathrm{B}}^{*}$ whose respective simulation induces a noticeably higher probability of *early-abort* in the ideal world, in comparison with the real world. In particular, this happens if the probability of non-*early-abort* of $\mathrm{P}_{\mathrm{B}}^{*}$ is noticeable and at the same time noticeably smaller than the inverse of `#rw-bound`. In such case, an *emulation of abort* would occur not only when $\mathrm{P}_{\mathrm{B}}^{*}$ aborts in the first case, but also with a small but noticeable probability in the remaining cases; i.e., there is a noticeable probability that $\mathrm{P}_{\mathrm{B}}^{*}$ would abort consecutive `#rw-bound` times after a first non-*early-abort*, thus providing a distinguishability criterion between the ideal and real worlds (after a sufficiently large, yet polynomially bounded, number of executions).

The possibility of `#rw-bound` being superpolynomial with negligible inverse is analyzed hereafter, motivated by the two previous paragraphs that showed that it can neither be infinite nor polynomial. The challenge is ensuring that the $E[\#rw]$ remains polynomial. Since a malicious $\mathrm{P}_{\mathrm{B}}^{*}$ might perform *early-abort* based on *something* learned from the contribution of P_{A}, the hiding property associated with the contribution of P_{A} is now analyzed in more detail.

Proposition 2. *Let $\mathscr{C}_X$ be an extractable commitment scheme for (short) seeds and let PRG be at least cryptographically secure. Let $\mathscr{C}'_X$ be defined as follows: (i) the* commit *phase consists on executing a* commit *phase of $\mathscr{C}_X$ to commit a random seed selected by the sender, followed by sending the (long) contribution masked by the XOR with*

the (same-length) PRG-expansion of the seed; (ii) the open *phase consists on using the* open *phase of $\mathscr{C}_X$ to open the seed. Then, it follows that $\mathscr{C}'_X$ is also an* extractable *commitment scheme. Assuming that the cardinality of possible (short) seeds is smaller than the cardinality of possible (long) commitable contributions, then the hiding of $\mathscr{C}_X$' is at most statistical; i.e., $\mathscr{C}_X$' is not perfectly hiding even if the PRG and $\mathscr{C}_X$ are ideal.*

$\mathscr{C}'_X$ in the above proposition is essentially the commitment scheme used to commit the contribution of P_A in protocol #1. The relevant aspect is that even in a hybrid model the hideability of the contribution of P_A depends also on the relation between the length of seeds and the length of contributions. Thus, it is relevant to determine what advantage a malicious P^*_B gains from this, even if just negligibly, to skew the probability of abort in the simulation. The main potential problem is that even a negligible effect on probabilities might have a noticeable effect on $E[\#rw]$.

Semantic hiding *property of a commitment scheme.* The *hiding* property of a commitment scheme can be defined based on an indistinguishability game essentially similar to a definition of semantic security of an encryption scheme (a.k.a., IND-CPA). A *commitment oracle* ($O_{\mathscr{C}}$) is initialized in either a `left` or a `right` mode, with 1/2 probability for each. Then, an adversary ($\mathcal{A}_{\mathscr{C}}$) chooses two messages (*left* and *right*) and sends them to the *commitment oracle*, without knowing the mode of the oracle. The oracle then produces a commitment of the message of its respective mode, either of the *left* or the *right* message, and sends the result to $\mathcal{A}_{\mathscr{C}}$. The adversary can make at most a polynomial number of such queries. Finally, the adversary outputs an attempted guess of the oracle mode: `left` or `right`. The advantage of the adversary in breaking the *hiding* property is defined as the absolute value of the difference between the probability of correctly guessing the `left` world and the probability of incorrectly guessing the `left` world.

$$\mathrm{Adv}(\mathcal{A}_{\mathscr{C}}) = |\Pr(\mathcal{A}_{\mathscr{C}} \Rightarrow \texttt{left} | O_{\mathscr{C}} \Rightarrow \texttt{left}) - \Pr(\mathcal{A}_{\mathscr{C}} \Rightarrow \texttt{left} | O_{\mathscr{C}} \Rightarrow \texttt{right})| \quad (134)$$

Definition 11 (hiding). *A commitment scheme $\mathscr{C}$ is* semantically hiding *(a.k.a.,* hiding*) if the maximum advantage of any polynomially bounded adversary in winning the above-defined game is negligible (in the security parameter).*

Strong hiding. For the purpose of defining an appropriate `#rw-bound`, it is useful to quantify the maximum distinguishability that an adversary ($\mathcal{A}$, a malicious P^*_B) can make about different committed contributions of P_A, when only seeing their commitments, i.e., the maskings t_A (12). This is specifically relevant in terms of how it may affect the decision of *early-abort* (e.g., 0 for abort and 1 for non-abort). For each possible value in the domain of committable values (χ_A), the respective probability of deciding 1 (non-*early-abort*) or 0 (*early-abort*) accounts for all possible probabilistic paths based on the internal state of $\mathcal{A}$ during the execution and all possible probabilistic paths by $\mathcal{S}$ (in the role of P_A) while computing the commitment. In particular, the probability on **non**-*early-abort* by $\mathcal{A}$, conditioned to a particular contribution (χ) decided by P_A (or by $\mathcal{S}$ in the role of P_A) can be calculated as a sum of terms across all possible commitments of such contribution (135). Intuitively, the *hiding* property becomes stronger when reducing the difference between the maximum probability (across all contributions) of

outputting 1 (136) and the minimum probability (across all contributions) of outputting 1 (137). Again, it is worth noticing that there are adversarial behaviors (P_B^*) for which this value is not 0 even if using an ideal PRG and even if the commitment of the seed is unconditionally hiding. Nonetheless, this value is expectably negligible.

$$(\forall\chi)\mathrm{Pr}_1(\chi) = \sum_{\overline{\chi}\in\mathscr{C}[\chi]} \mathrm{Pr}\left(\mathscr{C}[\chi] \Rightarrow \overline{\chi}\right) \cdot \mathrm{Pr}_1(\mathcal{A}(\overline{\chi}) \Rightarrow 1) \tag{135}$$

$$\mathrm{Pr}_1^+ = \max_{\chi\in\{0,1\}^\ell} \left(\mathrm{Pr}_1(\chi)\right) \tag{136}$$

$$\mathrm{Pr}_1^- = \min_{\chi\in\{0,1\}^\ell} \left(\mathrm{Pr}_1(\chi)\right) \tag{137}$$

Definition 12 (strong hiding). *(With respect to some implicit domain of committable values, e.g., all bit-strings of a certain fixed length,) A commitment scheme $\mathscr{C}$ is* ϵ-strongly hiding *if for any feasible adversary it follows that $Pr_1^+ - Pr_1^- < \epsilon$, for any sufficiently large security parameter κ.*[26] *A commitment scheme $\mathscr{C}$ is* strongly hiding *if there is a negligible ϵ such that $\mathscr{C}$ is* ϵ-strongly hiding.

Proposition 3. *If $\mathscr{C}'_X$ is ϵ-strong hiding for some negligible ϵ (138), and if* `#rw-bound` *is super-polynomial but lower than the inverse of ϵ such that the product of α and ϵ is negligible (139) (e.g., with α being the square-root of the inverse of ϵ, e.g., $\epsilon = 2^{-\kappa}$ and $\alpha = 2^{\kappa/2}$), then:*

1. *the probability of* early-abort *is indistinguishable between the ideal and real worlds;*
2. *$E[\#rw]$ is less than two.*

$$(\exists\epsilon : \epsilon > 0)\,(\forall\chi_A, \chi'_A)\,(\mathrm{Pr}_{\neg\perp}(\chi_A) - \mathrm{Pr}_{\neg\perp}(\chi'_A)) < \epsilon \tag{138}$$

$$\texttt{\#rw-bound} = \alpha : (0 < \alpha < 1/\epsilon) \wedge (\alpha \times \epsilon \in \mathrm{Negligible}) \tag{139}$$

The proof of Proposition 3 can be done by analyzing separately different characterizations of the *early-abort* strategy (or equivalently the non-early-abort strategy) of the adversary $\mathcal{A}$, in relation to the factors that determine the hiding-strength of $\mathscr{C}'_X$ (which depends on $\mathscr{C}_X$ and the PRG), and the upper-bound `#rw-bound` of number of rewindings.

Consider an adversary ($\mathcal{A}$) who has corrupted P_B^* in a simulated execution and has not aborted the execution until the step where it waits to receive from P_A (impersonated by the simulator $\mathcal{S}$) the commitment of the contribution of P_A (10)-(12). Upon receiving such commitment, $\mathcal{A}$ either decides to have P_B^* do *early-abort* (i.e., not opening her contribution) or to continue by successfully opening her hash and revealing her contribution (13)-(14). In other words, $\mathcal{A}$ may base its abort vs. non-abort decision on the commitment received from P_A (and possibly also on other aspects of the inner state of $\mathcal{A}$). If $\mathscr{C}'_X$ is *ϵ-strongly hiding* with respect to the space of possible contributions of P_A, then the probability of non-*early-abort* (conditioned to P_B^* not aborting before reaching the step where it waits to receive the commitment of P_A) differs by less than ϵ for any possible pair of contributions. In particular, this negligible distance-bound is valid between the **non**-*early-abort* probabilities in the cases of the random *needed complementary contribution* determined by $\mathcal{S}$ (in case P_A does not abort in the first iteration of the simulation) and the random contribution of P_A in the first iteration.

[26] It is left implicit that ϵ is a function of κ.

*Proof of part 1 (indistinguishability of probability of non-*early-abort*).*[27]

- If the probabilities of non-*early-abort* are negligible, then the distribution is already negligibly close to a distribution where the adversary would always abort. Thus, reaching the upper-bound `#rw-bound` induces at most a negligibly increase in probability of *early-abort* in the ideal world, which is not noticeably distinguishable.
- If the probabilities of non-*early-abort* are noticeable, then with overwhelming probability $\mathcal{A}$ will open the contribution of P_B^* before the super-polynomial bound `#rw-bound` is reached. In such case, if P_B^* opens his contribution in the first iteration it will also (with overwhelming probability) open again after a polynomial number of rewindings, thus inducing a probabilistic distribution with indistinguishable probability of non-*early-abort*.
- Even if the probabilities of non-*early-abort* are non-negligible and non-noticeable, the ϵ-strong hiding assumption remains, with a negligible ϵ. Thus, the ability to noticeably distinguish the ideal world from the real world would imply being able to break said assumption, because there would be different values whose commitments could be distinguished with noticeable probability.

□

Proof of part 2 ($E[\#rw]$ being upper-bounded by a constant). In the role of P_A in the simulated execution, the defined simulator ($\mathcal{S}$) only needs a non-abort from P_B after experiencing a non-abort in the first attempt. This means that, as the non-*early-abort* probability decreases between different malicious strategies by P_B^*, so does decrease the number of execution paths where $\mathcal{S}$ needs to obtain a second opening by P_B^*. However, in each execution path that does not abort in the first attempt, the corresponding conditioned $E[\#rw]$ is proportional to the inverse of the probability of non-abort. In particular, if the probability of non-abort becomes 0 after $\mathcal{S}$ in the role of P_A starts committing the *needed complementary contribution* of P_A, then $\mathcal{S}$ will rewind a number of times equal to the explicit upper bound `#rw-bound` α. Thus, `#rw-bound` needs to be set in a way that the $E[\#rw]$ is polynomial and at the same time does not jeopardize the indistinguishability between ideal and real worlds.

Consider a malicious P_B^* with an average negligible **non**-*early-abort* probability (across the space of possible contributions of P_A). Then, by the ϵ-*strongly hiding* assumption about $\mathscr{C}'_X$ (and assuming a negligible ϵ), there is a positive β such that, for any contribution committed by P_A, the probability of **non**-*early-abort* is bounded between $\beta_- \equiv \beta - \epsilon/2$ and $\beta_+ \equiv \beta + \epsilon/2$ (i.e., within an interval of width ϵ).

In terms of large $E[\#rw]$, the worst case scenario is bounded by the following hypothetical (though impossible) case with two probability characterizations:

[27] An alternative (weaker) proposition would be to simply say that the *early-abort* probability by P_B^* is *almost independent* (i.e., it is not noticeably dependent) of the contribution committed by P_A, and correspondingly also almost independent of the final bit-string value (the XOR of the two contributions). This could be proven directly from the *semantic hiding* of $\mathscr{C}'_X$, namely by showing the counter-positive, i.e., that if the probability of *early-abort* by P_B^* conditioned on the contribution committed by P_A varies noticeably with the contribution of P_A, then the adversary in the simulated execution can be used (as a rewindable black-box) to break the *semantic hiding* property of the commitment scheme $\mathscr{C}'_X$ used by P_A to commit her contribution.

- since rewindings only occur if P_B^* does not abort in the first iteration, then the worst case scenario is that in the first iteration, where $\mathcal{S}$ commits to a random contribution of P_A, the probability of **non**-*early-abort* would be the largest possible, i.e., β_+;
- since the simulation ends either at the subsequent **non**-*early-abort* or when reaching the upper bound, the worst case scenario is that the remaining (at most α) iterations that involve rewinding, with $\mathcal{S}$ always committing to the *needed complementary contribution*, the probability of **non**-*early-abort* in each iteration is the lowest possible, i.e., β_-.

In the mentioned hypothetical case, $E[\#rw]$ is upper-bounded by the product between β_+ (a majorant of the probability of **non**-*early-abort* in the first iteration) and $\min(\alpha, 1/\beta_-)$ (a majorant of the conditioned expected number of rewindings until a new non-*early-abort*.

In regard to how α compares with β_- and β_+, there are three cases to analyze:

1. $1/\beta_+ < 1/\beta_- \leq \alpha < 1/\epsilon$**.** In this case, $E[\#rw]$ is upper-bounded by β_+/β_-, which is negligibly close to 1. This is equal to the quotient between $(\beta + \epsilon/2)$ (a majorant of the probability of **non**-*early-abort* in the first iteration) and $\beta - \epsilon/2$ (a majorant of the expected number of attempts until obtaining a non-*early-abort*), which in turn equals $(1 + \epsilon/(2\beta))/(1 - \epsilon/(2\beta))$. Since $\epsilon/2\beta$ is a negligible function (because by assumption $\alpha \times \epsilon$ is negligible), the result is of the form $(1-x)/(1+x)$ for some value x approaching 0, which means $E[\#rw]$ is negligibly close to 1.
2. $1/\beta_+ \leq \alpha < 1/\beta_-$**.** In this case, $E[\#rw]$ is upper-bounded by $\beta_+ \times \alpha$, which is upper bounded by the previous calculation, i.e., β_+/β_-, i.e., negligibly close to 1.
3. $\alpha < 1/\beta_+ < 1/\beta-$**.** In this case, whenever there is a non-*early-abort* in the first iteration, the remaining simulation will likely reach the upper-bound. Thus, $E[\#rw]$ is bounded by $\beta_+ \times \alpha$, which is lower than 1 because α is less than $1/\beta_+$.

In summary, $E[\#rw]$ is less than two for sufficiently large security parameter κ and assuming a ϵ-*strongly hiding* property of $\mathscr{C}'_X$, for some known ϵ, and an explicit bound `#rw-bound` defined accordingly, e.g., $\alpha = 1/\sqrt{\epsilon}$.

□

It is left for future exploration finding whether, for protocol #1 as is, it is possible to adjust the simulator (in the case of corrupted P_B^*), or adjust its analysis, such that the number of rewindings becomes strictly polynomial (perhaps at the tradeoff of $E[\#rw]$ becoming polynomial but not bounded by a constant). An intuition for this is that $\mathcal{S}$ may always start the simulation with a strict polynomial number of rewindings (e.g., linear in the security parameter) and only then probabilistically decide whether to perform *early-abort* or to continue further with at most a strict polynomial number of rewindings until obtaining the needed opening of the contribution of P_B^* or emulating an abort.

D Instantiations in the plain model (simulatable-with-rewinding)

The coin-flipping protocol #1 requires exactly one X commitment and one Q commitment of short strings, and respective openings, besides the direct communication of two strings

of the target length. In terms of computation per party, it requires: one PRG expansion, one hash function call, two XOR combinations, and one selection of random bit-string (all associated with the target length); and participation in the two commitment schemes (in both *commit* and *open* phases), once as sender and once as receiver.

This section discusses concrete instantiations of protocol #1 in the plain model. Subsection §D.1 considers an instantiation that requires only 9 exponentiations from each party (or 11, using practical parameters), 5 of which can be done in a setup phase. Therein, the commitment schemes have security based on the Decision Diffie-Hellman (DDH) intractability assumption. Subsection §D.2 discusses hypothetical problems in case of other instantiations of pairs of commitment schemes that might have interfering properties (e.g., malleability). Subsections §D.3 and §D.4 devise new instantiations of $\mathscr{C}_{\mathrm{X}}$ and $\mathscr{C}_{\mathrm{Q}}$ without any use of exponentiations, but rather based only on regular bit-commitments (i.e., hiding and binding, but not necessarily X or Q, and assuming that they do not require exponentiations). These instantiations, which come at the cost of more communication rounds, are inspired by (but with differences compared to) techniques described by Pass and Wee [PW09]. While the mentioned prior work has proposed a X scheme based on bit-string commitments and a Q scheme requiring a quadratic number of bit-commitments, this section devises a X scheme based only on bit-commitments, and a Q scheme based only on a linear number of bit-commitments. Going one level further, such regular commitment schemes can be instantiated for example based on a PRG (e.g., using the construction of Naor [Nao91] or subsequent improvements) – a primitive already required by protocol #1.

D.1 Protocol #1 in the plain model – an instantiation based on DDH

A concrete instantiation of the phases of commitment schemes underlying protocol #1 is presented in Fig. 12, secure under the DDH assumption. It is left implicit an initial phase where the parties agree on a computational security parameter (140), a PRG and CR-Hash function (141), and a cyclic group (in multiplicative notation) based on the intractability of the DDH assumption, consistently with the computational security parameter (142). The group order is known by both parties (143), as well as the maximum number of bits that can be committed under each group element (144) (e.g., sufficient to fit a seed and/or a hash value). The parties also agree on a generator of the group (145) (hereafter denoted as *first generator*). All these parameters can be pre-computed.

- **Setup phase.** $\mathrm{P_A}$ raises the *first generator* g to the power of a random exponent α in order to obtain a *second generator* g'. This random exponent, being the discrete log of the second generator in the base of the first generator, is kept hidden from $\mathrm{P_B}$ (146). $\mathrm{P_A}$ sends to $\mathrm{P_B}$ the second generator, thus completing the needed parameters for the two commitment schemes (147). Then, $\mathrm{P_A}$ gives a ZK proof of knowledge of the discrete logarithm of the second generator in the base of the first (148). Basically, this can be a ZK adaptation of Schnorr's protocol [Sch91], as given in [LPS08, Fig. 3] (149)-(159). A simulator in the role of $\mathrm{P_B}$ would be able to extract the discrete log from this ZKPoK.
- **$\mathrm{P_B}$ uses $\mathscr{C}_{\mathrm{Q}}$ to commit hash (8).** $\mathrm{P_B}$ uses the Pedersen commitment scheme [Ped92] to commit to the hash of his contribution. Basically, to commit, $\mathrm{P_B}$ sends the product

An instantiation of $\mathscr{C}_{\mathbf{X}}$ and $\mathscr{C}_{\mathbf{Q}}$ based on DDH intractability

Implicit parameters (common input)

$\mathrm{P_A}, \mathrm{P_B} : 1^{\kappa}$ (computational security parameter) (140)

$\mathrm{P_A}, \mathrm{P_B} : (\mathrm{PRG}, \kappa_{\mathrm{PRG}}), (\mathrm{Hash}, \kappa_{\mathrm{Hash}})$ (141)

$\mathrm{P_A}, \mathrm{P_B} : \mathbb{G} \in \mathrm{CyclicGroups}\left[\mathrm{DDH}\left[1^{\kappa}\right], \# > 2^{\kappa'}\right]$ (142)

$\mathrm{P_A}, \mathrm{P_B} : q = \#(\mathbb{G})$ (143)

$\mathrm{P_A}, \mathrm{P_B} : \kappa' = \lfloor \log_2(q) \rfloor$ (144)

$\mathrm{P_A}, \mathrm{P_B} : g \in \mathrm{Generators}(\mathbb{G})$ (145)

Common setup (i.e., valid for both commitment schemes) (3)

$\mathrm{P_A} : \alpha \leftarrow^{\$} \mathbb{Z}_q, g' = g^{\alpha}$ (decide second generator) (146)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{scheme-params}, \textit{context}, g')$ (147)

$\mathrm{ZKPoK_{DL}}\left[\alpha : g' = g^{\alpha}\right]$ (taken from [LPS08]) (148)

$\mathrm{P_A} : a \leftarrow^{\$} \mathbb{Z}_q, A = g^{a}$ (ephemeral generator) (149)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{ZK-step-ephem-gen}, \textit{context}, A)$ (150)

$\mathrm{P_B} : c, r \leftarrow^{\$} \mathbb{Z}_q, C = g^{c} \cdot A^{r}$ (prepare challenge) (151)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{ZK-step-commit-chal}, \textit{context}, C)$ (152)

$\mathrm{P_A} : k \leftarrow^{\$} \mathbb{Z}_q, K = g^{k}$ (153)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{ZK-step-commit}, \textit{context}, K)$ (154)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{ZK-step-open-chal}, \textit{context}, (c, r))$ (155)

$\mathrm{P_A} : C =^{?} g^{c} \cdot g'^{r}$ (verify challenge c) (156)

$\mathrm{P_A} : z = \alpha \cdot c + k (\mathrm{mod}\ q)$ (compute response) (157)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{ZK-step-respond}, \textit{context}, (z, a))$ (158)

$\mathrm{P_B} : \left(g^{z} =^{?} g'^{c} \cdot K\right) \wedge \left(A =^{?} g^{a}\right)$ (verify) (159)

X commitment scheme (based on ElGamal encryption scheme [ElG85])

$\mathscr{C}^{\mathrm{Commit}}_{\mathbf{X},\textit{sid},\textit{cfid}}[\ldots]$ **– Commit seed (s) of $\mathrm{P_A}$ (10).**

$\mathrm{P_A} : r_A \leftarrow^{\$} \mathbb{Z}_q, c_{A,1} = g^{r_A}, c_{A,2} = s \cdot g'^{r_A}$ (160)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{commit-seed}, \textit{context}, (c_{A,1}, c_{A,2}))$ (161)

$\mathscr{C}^{\mathrm{Open}}_{\mathbf{X},\textit{sid},\textit{cfid}}[\ldots]$ **– Open seed (s) of $\mathrm{P_A}$ (16).**

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{open-seed}, \textit{context}, (r_A, s))$ (162)

$\mathrm{P_B} : c_A =^{?} \left(g^{r_A}, s \cdot g'^{r_A}\right)$ (163)

Q commitment scheme (from Pedersen [Ped92])

$\mathscr{C}^{\mathrm{Commit}}_{\mathbf{Q},\textit{sid},\textit{cfid}}[\ldots]$ **– Commit hash (h) of $\mathrm{P_B}$ (8).**

$\mathrm{P_B} : r_B \leftarrow^{\$} \mathbb{Z}_q, c_B = g^{h} \cdot g'^{r_B}$ (164)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{commit-hash}, \textit{context}, c_B)$ (165)

$\mathscr{C}^{\mathrm{Open}}_{\mathbf{Q},\textit{sid},\textit{cfid}}[\ldots]$ **– Open hash (h) of $\mathrm{P_B}$ (13).**

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{open-hash}, \textit{context}, (r_B, h))$ (166)

$\mathrm{P_A} : c_B =^{?} g^{r_B} \cdot g'^{h}$ (167)

Fig. 12. Instantiation based on DDH. Legend: *context* denotes the message context, including the session identifier (*sid*), the sub-session identifier (*cfid*) and the sender and receiver of the message (e.g., $\mathrm{P_A}$ and $\mathrm{P_B}$).

of powers of the two generators, where the first exponent is the hash and the second is a random value (164)-(165). This is provably an unconditionally hiding commitment, not revealing any information about the hash. Note: if the hash (e.g., 256 bits) is too large to fit into a single commitment, then it can be split into several commitments (e.g., two commitments if the group order allows up to 128 bits).

– **$\mathrm{P_A}$ uses $\mathscr{C}_{\mathbf{X}}$ to commits seed (10).** $\mathrm{P_A}$ uses an ElGamal encryption [ElG85] to commit to her seed s. Basically, $\mathrm{P_A}$ computes the encryption as a pair of elements, where the first element is the first generator g to the power of a random exponent r_A, and the second element is the product of the seed s with the second generator g' to the power of the same random exponent r_A (160)-(161). A simulator in the role of $\mathrm{P_B}$, in possession of the discrete log α between the two generators, would be able to simply decrypt the seed. Note: if the seed (e.g., 128 bits) is too large to fit into a single commitment, then it can be split into several commitments. Nonetheless, it is henceforth assumed that it fits into a single commitment.

- **P_B uses $\mathscr{C}_Q$ to open hash (13).** P_B reveals the two exponents used in the Pedersen commitment, namely the random element and the hash (166), thus allowing P_A to verify correctness (167). A simulator in the role of P_B, knowing the discrete log between the two generators, would be able to determine open any desired contribution, namely by solving a set of two linear equations in order to determine the random value (the exponent of the first generator) corresponding to the intended equivocated value.
- **P_B uses $\mathscr{C}_X$ to open seed (16).** P_A reveals the random exponent and the encrypted seed used in the ElGamal encryption (162), allowing P_B to verify correctness (163).

D.2 Interfering instantiations of two commitment schemes

Since the commitment schemes $\mathscr{C}_X$ and $\mathscr{C}_Q$ might depend on each other in the plain model, it must be analyzed whether their combination in the same protocol jeopardizes the needed properties of the isolated commitment schemes. This subsection discusses the potential problem of *interference* between commitment schemes, specially relevant when two schemes are executed in an interleaved way with parties interchanging roles, as in protocol #1. Perhaps the most obvious problem to consider is that of malleability between $\mathscr{C}_Q$ and $\mathscr{C}_X$. Other types of possible *interference* are also discussed below.

Non-malleability. If a protocol uses more than one commitment scheme, with a party being *receiver* in one and *sender* in the other, then the *hiding* and *binding* properties might be insufficient to guarantee *independence* of opened values. For example, the *receiver* of a first commitment to one value might be able to produce a second commitment, to a *related* value, which it could *open* after seeing the opening of the first commitment. Specifically, this must be prevented in the case of protocol #1, by requiring a type of *non-malleability* property [DDN00, DCIO98, DCKOS01, FF09].

Definition 13 (non-malleability with respect to opening). *Consider an adversary in the role of* receiver *of a commitment scheme (the first), and in the role of* sender *in a possibly different commitment scheme (the second), such that their phases are interleaved as follows: (i) the first* commit *phase is executed to commit a value probabilistically sampled from a publicly known distribution; then (ii) the second* commit *phase is executed to commit a value chosen by the malicious party (possibly dependent on the first commitment), then (iii) the first* open *phase is executed, finally, (iv) the second* open *phase is executed (possibly dependent on the first commitment and opening). The second commitment scheme is* non-malleable with respect to opening *of the first commitment scheme if, for any feasible adversary, the probability that the first and second* opened *values satisfy some* relation *is negligibly close to the probability that the same relation is satisfied if the second value is instead selected by a suitably defined simulator that does not need to see any commitment or opening.*

For protocol #1, this type of *non-malleability* is required between a first Q-scheme used by P_B to commit a hash value and a second X-scheme used by P_A to commit a

short seed value.[28] If $\mathscr{C}_X$ is **not** *non-malleable with respect to opening* of $\mathscr{C}_Q$, then P_A^* could potentially be capable of equivocating[29] her seed during the *open* phase of $\mathscr{C}_X$, thus preventing $\mathcal{S}$ from inducing the intended final bit-string. It is instructive to start by analyzing the instantiation exemplified in Fig. 12, where $\mathscr{C}_Q$ is Pedersen's commitment scheme and $\mathscr{C}_X$ is ElGamal's encryption scheme, both with the same *disrete-log* trapdoor, with overall security based on the DDH assumption (namely what is needed to ensure the hiding property of $\mathscr{C}_X$). The concrete instantiation has the necessary non-malleability property (i.e., for the case of a malicious P_A^*), as can be derived directly from information theoretic arguments. Specifically, since $\mathscr{C}_Q$ is *unconditionally hiding*, the commit phase of $\mathscr{C}_X$ (when P_A commits to a seed) cannot depend on the value committed by P_B using $\mathscr{C}_Q$. Furthermore, since $\mathscr{C}_X$ is *unconditionally binding*, its opening cannot be changed to a value different from what had been committed in the *commit* phase. Interestingly, even though $\mathscr{C}_Q$ is malleable with respect to itself, and $\mathscr{C}_X$ is malleable with respect to itself, $\mathscr{C}_X$ is *non-malleable with respect to opening* of $\mathscr{C}_Q$.

Other types of interference. Malleability is a special aspect in the sense that it may pose problems even if the basic *hiding* and *binding* properties of the commitment schemes are satisfied. Nonetheless, when considering concrete instantiations, there are further aspects that must be verified before establishing that two commitment schemes do not interfere in a way that jeopardizes any of the assumed properties of the isolated commitment schemes. For the structure of protocol #1, it is useful to consider what can go wrong for each possibility of breaking a desirable property of an individual commitment scheme, namely: hiding or binding by the regular parties, extractability (if needed) or (iv) equivocability (if needed) by the simulator.

For the case of a corrupted P_A^*: (i) if hiding of $\mathscr{C}_Q$ is broken, then the Q-capability of $\mathcal{S}$ is also broken – once $\mathcal{S}$ would equivocate, P_A^* could notice that the opened value does not correspond to what it supposed to; (ii) if binding of $\mathscr{C}_X$ is broken, then P_A^* gains some type of Q-capability, thus breaking the X-capability of $\mathcal{S}$ – $\mathcal{S}$ would extract a value in the commit phase, but then P_A^* would be able to open a different value.

For the case of a corrupted P_B^*: (i) if hiding of $\mathscr{C}_X$ is broken, then P_B^* gains a type of X-capability, thus becoming able to adjust his probability of *early-abort* depending on the final output, thus introducing a bias not possible in the case of an ideal $\mathcal{F}_{MCF}$; (ii) if binding of $\mathscr{C}_Q$ is broken, then P_B^* gains some type of Q-capability, thus possibly preventing $\mathcal{S}$ in the role of P_A from inducing the intended target outcome.

There is yet another potential problem relevant in the case of a corrupted P_A^*, when $\mathcal{S}$ (in the role of P_B) needs to have X-capability and Q-capability in respective respect to $\mathscr{C}_X$ and $\mathscr{C}_Q$. In particular, it must also be contemplated the possibility of these properties being broken, in the sense that their application by $\mathcal{S}$ could allow P_A^* to distinguish the presence of $\mathcal{S}$, even though the hiding and binding of $\mathscr{C}_Q$ and $\mathscr{C}_X$ could still remain intact in a computational sense. Specifically, if Q or X are broken, then $\mathcal{S}$ in the role of P_B can no longer guarantee inducing the necessary target outcome in the simulation.

[28] A more general definition adequate to a concurrent setting would consider non-malleability with respect to several messages and an adversary trying to take advantage of the possible interactiveness of each phase [FF09].

[29] Here it would be enough to gain the ability to equivocate to a single different value.

This component is not relevant for the case of a corrupted P_B^*, because in such case the simulator ($\mathcal{S}$ in the role of P_A) does not require X or Q capabilities (i.e., it performs extraction and equivocation via *non-local* rewinding).

In the exemplified instantiation of Fig. 12, none of the mentioned problems occur. This is easy to derive from the fact of sharing a common trapdoor. Specifically:

- breaking the *binding* of $\mathscr{C}_X$ is not possible because it is unconditionally binding;
- breaking the *hiding* of $\mathscr{C}_X$ would mean being able to break the DDH assumption;
- breaking the *binding* of $\mathscr{C}_Q$ would mean being able to compute discrete logarithms;
- breaking the *hiding* of $\mathscr{C}_Q$ is not possible because it is unconditionally hiding;
- breaking the X-indistinguishability of $\mathscr{C}_X$ is not possible, because extraction occurs without interaction with P_A^* (i.e., locally computing the committed value from the commitment, by using the trapdoor obtained in the setup phase);
- breaking the Q-indistinguishability of $\mathscr{C}_Q$ is not possible, because the scheme is unconditionally hiding and $\mathcal{S}$ sends an opening message that is indistinguishable from a correct opening.

Definition 14. *Let $\mathscr{C}$ and $\mathscr{C}$' be two commitment schemes (i.e., hiding and binding). $\mathscr{C}$ and $\mathscr{C}$' are said to be* interference-free with respect to interleaved application *if their interleaved application, namely first the commit phase of $\mathscr{C}$, then the commit phase of $\mathscr{C}$', then the open phase of $\mathscr{C}$, then the open phase of $\mathscr{C}$', does not break the* hiding *and* binding *properties of any of the commitments (namely in the respective* commit *and* open *phases), nor the original X and Q properties (if present) possessed by the individual commitment schemes.*

It is instructive to see an example of interference (different from malleability), even though it is not applicable to protocol #1:

An example of a* failed *combination of commitment schemes. As an example of interference, namely of $\mathscr{C}_Q$ *not being* equivocation-hideable *with respect to* $\mathscr{C}_X$, consider the following pairs of (stand-alone secure) commitment schemes:

- $\mathscr{C}_Q$. P_B is the sender and P_A is the receiver; the *commit* phase consists of an ElGamal encryption; the *open* phase consists of revealing the committed value and then giving a ZK proof that it is a correct opening;
- $\mathscr{C}_X$(designed for a single use). P_A is the sender and P_B is the receiver; a *commit* phase consists of a Pedersen commitment followed by a perfect ZK argument of knowledge of the committed value; the *open* phases consists of opening the Pedersen commitment (i.e., the *sender* P_A sending the two exponents – the committed value and the random value) to the *receiver* P_B, and then having the *receiver* P_B reveal the trapdoor to P_A (i.e., the discrete-log between the two generators).

$\mathscr{C}_Q$ is computationally equivocable, because $\mathcal{S}$ in the role of *sender* P_B can use its rewinding power to succeed in a fake ZK proof in the *open* phase. $\mathscr{C}_X$ is extractable because $\mathcal{S}$ in the role of *receiver* P_B can use its rewinding power to extract the trapdoor from the respective ZK argument of knowledge. However, if both schemes are based on the same trapdoor, i.e., a discrete-log selected by P_B, then P_A gains the ability to

distinguish an equivocated opening of the hash of P_B from an honest opening. Specifically, once learning the trapdoor (that P_B sends to P_A in the *open* phase of $\mathscr{C}_X$), P_A can check whether the value opened with $\mathscr{C}_Q$ is indeed the value encrypted by the respective ElGamal encryption (received in the *commit* phase of $\mathscr{C}_Q$).

Note: This paper does not answer on whether or not there might exist an instantiation of $\mathscr{C}_X$ and $\mathscr{C}_Q$ such that $\mathscr{C}_X$ is malleable with respect to opening of $\mathscr{C}_Q$, within the structure of protocol #1, for a stand-alone execution.

D.3 X commitment scheme from regular bit-commitments

This subsection considers X commitment schemes based on more basic primitives, namely *regular* bit-commitment schemes (i.e., *hiding* and *binding*), which might be neither X nor Q. The subsection starts by describing a prior method [PW09], based on a cut-and-choose approach, that builds a X commitment scheme from regular **bit-string** commitment schemes. Then it motivates and gives an intuition for a variation, devised in the remaining subsection, where **bit**-commitments are used instead of **bit-string** commitments. The description of the new method starts with an unoptimized version (§D.3.1) and then introduces an optimization (§D.3.2) using a *random-seed-checking* (RSC) type of technique [GMS08], which involves a CR-Hash and a PRG (primitives also already used in protocol #1), to allow a reduction of the communication complexity. A comparison in terms of communication is summarized in Table 1.

Note. In the remainder of this section, the word *target* (e.g., in *target value* or *target length*) is used in respect to the value being committed by each of the X or Q commitment schemes. In particular, when considering their application within protocol #1, the value is a *seed* (e.g., 128 bits) in case of the X-scheme, or a *hash* (e.g., 256 bits) in case of the Q-scheme. This should not to be confused with the use of the word *target* in the remainder of the paper, when referring to the *target length* as the number of bits being flipped (protocol #1) or committed (protocol #2).

A prior X scheme. Based on prior work, Pass and Wee [PW09] described cut-and-choose techniques to build X, Q and X&Q commitment schemes from regular commitment schemes (in a setting of simulation with rewinding). The goal of their constructions is not efficiency, but rather showing that one-way functions (which can be used to build PRGs and regular commitment schemes) are enough to build X&Q commitment schemes, and consequently also to build simulatable coin-flipping (based on the traditional template protocol). Even though, in terms of communication, their constructions are inefficient as a function of the length of the values being committed, they can be useful in protocol #1 because there they only need to be applied to short values, e.g., a seed (the X scheme) and or a hash (the Q scheme). In the X scheme [PW09, §4], in the commit phase the sender produces regular bit-string commitments (*commit* and *open*) in number equal to twice the statistical security parameter, with each bit-string commitment being for values as long as the *target value*. Specifically, each pair of such commitments is committing to a pair of random shares of the target value; i.e., each pair of committed values XORs into the target value. Still in the *commit* phase, one out of each two shares is opened –

Table 1. Communication complexity of X commitment schemes

$\mathscr{C}_{\mathrm{X}}(1^{\ell})$ ($\mathrm{P_A}$ as sender)	Direction of communication	Commit phase	Open phase
[PW09, §4]	$\mathrm{P_A} \to \mathrm{P_B}$	$\sigma\left(2\lvert\mathscr{C}^{\mathrm{C}}(1^{\ell})\rvert + \lvert\mathscr{C}^{\mathrm{O}}(1^{\ell})\rvert\right)$	$\sigma\lvert\mathscr{C}^{\mathrm{O}}(1^{\ell})\rvert$
	$\mathrm{P_B} \to \mathrm{P_A}$	σ	–
This paper: unoptimized (Fig. 14)	$\mathrm{P_A} \to \mathrm{P_B}$	$(\ell+\sigma_{\mathrm{X}})\left(2\lvert\mathscr{C}^{\mathrm{C}}(1^{1})\rvert + \lvert\mathscr{C}^{\mathrm{O}}(1^{1})\rvert\right)$	$(\ell+\sigma_{\mathrm{X}})\lvert\mathscr{C}^{\mathrm{O}}(1^{1})\rvert$
	$\mathrm{P_B} \to \mathrm{P_A}$	$\kappa_{\mathrm{PRG}} + (\ell+\sigma_{\mathrm{X}})$	–
This paper: RSC (Fig. 15)	$\mathrm{P_A} \to \mathrm{P_B}$	$\kappa_{\mathrm{Hash}} + (\ell+\sigma_{\mathrm{X}})(1+\kappa_{\mathrm{PRG}} + \lvert\mathscr{C}^{\mathrm{C}}(1^{1})\rvert)$	$\kappa_{\mathrm{PRG}} + \ell$
	$\mathrm{P_B} \to \mathrm{P_A}$	$\kappa_{\mathrm{PRG}} + (\ell+\sigma_{\mathrm{X}})$	–

Legend: $\mathscr{C}$ (regular commitment scheme); $\mathscr{C}_{\mathrm{X}}$ (extractable commitment scheme); ℓ (length of the target value being committed by $\mathscr{C}_{\mathrm{X}}$ – in the context of protocol #1 it is $\ell = \kappa_{\mathrm{PRG}}$); $\lvert\mathscr{C}^{p}(1^{\ell})\rvert$ (communication required by phase p of regular commitment $\mathscr{C}$, where p is either C (*commit*) or O (*open*) – 1^1 is used in the case of a bit-commitment scheme); κ_{PRG} (size of a PRG seed, e.g., 128 bits); κ_{Hash} (size of a CR-Hash output, e.g., 256 bits); σ (statistical security parameter, e.g., 40 bits); σ_{X} (length extension of encoded words (191), ensuring extractability up to statistical security 1^{σ} (see §D.3.1 for further notes)).

this does not reveal any information about the committed value. However, rewinding allows the simulator to obtain at least one such pair, thus extracting the committed value. The open phase consists simply on opening the remaining shares and let the receiver verify they each pair of shares XORs into the same value – the final accepted value.

Intuition for a new X scheme. In contrast with the above mentioned method, this subsection devises a X bit-string commitment scheme based on regular bit-commitments, i.e., such that the underlying regular commitments are only applied to single bits. Trivially, a bit-string commitment could be replaced by directly committing to each bit separately, but this would increase the number of commitment by a multiplicative factor equal to the target length. The method devised below still increases the concrete number of regular commitments, but not as in the trivial method. Instead, the already existing multiplicative factor of the statistical parameter is replaced by a different statistical factor. In spite of the increase, the result might still be potentially compensatory in scenarios where a bit-commitment might be significantly less expensive than a bit-string commitment.[30]

[30] It is worth pointing out that there are efficient ways of instantiating bit-string commitments. For example, Naor [Nao91] described a PRG-based bit-string commitment technique with amortized cost, more efficient than a parallel use of bit-commitments. However, said technique involves an error correction code, whose explanation would make the overall description more cumbersome. Another typical instantiation of bit-string commitment is via application of a cryptographic hash function to the concatenation of the value being committed and an unpredictable suffix – this is *hiding* based on the assumption, not used in this paper, that the hash of an unpredictable value is indistinguishable from a random value.

The intuition follows from the use of an erasure code and a cut-and-choose approach. The parties agree on a binary and linear encoding scheme (i.e., based on XOR), which converts the vector of original bits into a new larger vector of *checksum* bits, with the goal that the message be recoverable in case of *erasure* (i.e., omission) of some checksum bits. Then, the sender splits each resulting checksum bit into a pair of random *shares* (i.e., two bits whose XOR equals the checksum bit), and commits individually to each share. This means that each share bit in isolation does not reveal anything about the respective checksum bit. Then, the receiver (P_B) makes a cut-and-choose, defining for each checksum which share must be opened immediately (i.e., in the overall *commit* phase) by P_A and which one will be opened later (i.e., in the overall *open* phase). This overall *commit* phase does not reveal to P_B anything about the checksum bits, but allows the simulator to use rewinding in order to obtain the information (i.e., the complementary shares) necessary to extract enough checksums such that the target value (the vector of original bits) is recoverable. Later, in the overall *open* phase, P_A opens all remaining shares, thus letting P_B combine the shares in each pair, thus obtaining all checksum bits and verifying that all of them are consistent with the target seed.

In comparison with the example of DDH instantiation of protocol #1, a practical computational advantage of the described instantiations based on regular bit-commitments is completely avoiding the use of exponentiations (i.e., considering the practical assumption that the underlying PRG, the CR-Hash and the regular commitment schemes do not require exponentiations). In contrast, there are two disadvantages (the importance of which may depend on the application and context): the *commit* phase becomes explicitly interactive[31] (i.e., requires communication rounds); it involves a larger concrete communication complexity.

***Implicit setup phase* (Fig. 13).** In any of the subsequently defined X and Q commitment schemes (possibly simultaneously for both, if used within the same protocol, e.g., protocol #1) both parties agree on a computational security parameter (e.g., 128 bits) (168) and a statistical parameter (e.g., 40 bits) (169). Consistently with the security parameters, the parties also agree on a *regular* bit-commitment scheme (i.e., hiding and binding) (170). Furthermore, as needed, the parties know the target length ℓ of elements to be committed in a X or Q manner, respectively the PRG-seed length (e.g., 128 bits) (171) and the CR-Hash output length (e.g., 256 bits) (172), and as needed they also agree on respective PRG and CR-Hash functions.

D.3.1 X commitment scheme – unoptimized

The new X commitment scheme is defined with succinct notation in Fig. 14 and in textual form in the next paragraphs.

Commit phase.

[31] Again, the term *explicit* is used to denote what does not depend on the instantiation of the underlying primitives, in this case the regular commitment scheme. A protocol that is not *explicitly interactive* might still be interactive in practice, namely if the underlying regular commitment scheme is interactive.

Implicit parameters (common input)

$P_A, P_B : 1^\kappa$ (computational security parameter)	(168)	$P_A, P_B :$ (PRG, κ_{PRG}) (PRG function and seed-length)	(171)
$P_A, P_B : 1^\sigma$ (statistical security parameter)	(169)	$P_A, P_B :$ (Hash, κ_{Hash}) (CR-Hash and its output-length)	(172)
$P_A, P_B : \mathscr{C}$ (regular commitment scheme)	(170)		

Fig. 13. Implicit setup parameters for subsequent instantiations of $\mathscr{C}_X$ and $\mathscr{C}_Q$

P_A (sender) uses $\mathscr{C}_X$ to commit (10) and open (16) value v (with $\ell \equiv |v|$) to P_B (receiver).

$\mathscr{C}^{\text{Commit}}_{X,sid,cid}[\mathbf{P_A}(v) \leftarrow (\underline{v}, \overline{v}); \mathbf{P_B} \leftarrow \overline{v}]$**:**

1.a. Decide encoding.

$P_B : s_B \leftarrow^{\$} \{0,1\}^{\kappa_{PRG}}$ (173)

$P_B \rightarrow P_A :$ (`seed-checksums`, *context*, s_B) (174)

$P_A, P_B : \sigma_X = \lceil \alpha(\ell, \sigma) \rceil$ (175)

$P_A, P_B : \ell' = \ell + \sigma_X$ (176)

$P_A, P_B : u = \text{PRG}[s_B](\ell \times \ell')$ (code matrix) (177)

$P_A, P_B : \langle u_{k,i} : k \in [\ell'], i \in \ell \rangle \overset{\text{Parse}}{\leftarrow} u$ (178)

$P_A : t_k = \oplus_{i \in [\ell]}(u_{k,i} \cdot s_i) : k \in [\ell']$ (179)

1.b. Prepare bit-commitments of bit-shares.

$P_A : d^0_k \leftarrow^{\$} \{0,1\} : k \in [\ell']$ (180)

$P_A : d^1_k = d^0_k \oplus t_k : k \in [\ell']$ (181)

For $k \in [\ell'], j \in \{0,1\} : (P_A \rightarrow P_B)$

$\mathscr{C}^{\text{Commit}}_{sid,cid,k,j}[P_A(d^j_k) \leftarrow (\underline{d}^j_k, \overline{d}^j_k); P_B \leftarrow \overline{d}^j_k]$ (182)

1.c. Challenge-response.

$P_B : c \leftarrow^{\$} \{0,1\}^{\ell'}$ (183)

$P_B \rightarrow P_A :$ (`challenge-positions`, *context*, c) (184)

$P_A, P_B : D_k \equiv d^{c_k}_k : k \in [\ell']$

For $k \in [\ell'] : (P_A \rightarrow P_B)$

$\mathscr{C}^{\text{Open}}_{sid,cid,k,c_k}(\overline{D}_k)[P_A(D_k, \underline{D}_k); P_B \leftarrow D_k]$ (185)

$\mathscr{C}^{\text{Open}}_{X,sid,cid}(\overline{v})[\mathbf{P_A}(v, \underline{v}); \mathbf{P_B} \leftarrow v]$**:**

$P_A \rightarrow P_B :$ (`send-committed-value`, *context*, v) (186)

$P_A, P_B : D'_k \equiv d^{1-c_k}_k : k \in [\ell']$ (187)

For $k \in [\ell'] :$

$\mathscr{C}^{\text{Open}}_{sid,cid,k,c'_k}(\overline{D}'_k)[P_A(D'_k, \underline{D}'_k); P_B \leftarrow D'_k]$ (188)

$P_B : t_k = d_{k,0} \oplus d_{k,1} : k \in [\ell']$ (189)

$P_B : t_k =^{?} \oplus_{i \in [\ell]}(u_{k,i} \cdot s_i) : k \in [\ell']$ (190)

Fig. 14. $\mathscr{C}_X$ from regular bit-commitments (unoptimized). In the context of protocol #1, this X scheme may be applied to a PRG seed selected by P_A. Legend: κ_{PRG} (length of a PRG seed); *context* denotes the message context, including session identifier (*sid*), sub-session identifier (*cfid*) and sender and receiver (e.g., P_A and P_B); σ_X (number of bits needed beyond the original length, in order to allow extractability); ℓ' (*extended length*, i.e., length of the encoding) α (function that determines the additional length, such that there is an overwhelming probability in σ that the boycott by P^*_A of opening certain shares of at most σ checksum-bits (t_k) still allows the simulator to recover the seed-bits; i.e., such that any subset of $\ell' - \sigma$ rows of the matrix has rank ℓ).

- **1.a. Decide encoding.** P_B (the receiver) selects an auxiliary random seed (173), whose PRG expansion will be used to determine the form of certain binary checksums, and sends the seed to P_A(174). The number of checksums is larger than the number of original bits by an additive factor that depends on the statistical parameter and the target length (175) (see discussion below). The overall number of checksums, denoted *extended length*, is determined locally by both parties (176) (or agreed in a setup phase). Both parties locally use the auxiliary seed to pseudo-randomly generate a bit-string of length equal to the number of checksum bits multiplied by the target length (177). The resulting bit-string is then parsed to define, for each checksum bit, a pseudo-random subset of positions of target-bits (i.e., of bits of the seed being committed) (178). P_A uses each of these subsets to compute a checksum as a XOR of the seed-bits whose position is included in the subset (179).
- **1.b. Prepare bit-commitments of bit-shares.** Then, P_A computes a random pair of shares for each checksum bit, such that the XOR of each pair of shares is equal to

the respective checksum bit (180)-(181) and also commits individually to each such share (182).
– **1.c. Challenge-response.** Then, P_B selects a random bit-vector of challenges (183), defining for each pair of shares one of them to be opened, and sends it to P_A (184). P_A opens only the challenged shares (185).

Open phase. P_A (the sender) sends the target value to P_B (186) and opens the complementary shares (187)-(188), i.e., those that remained unopened in the overall *commit* phase. P_B combines each pair of shares into a respective (tentative) checksum bit (189) and then verifies that the encoding of the target seed leads indeed to the obtained checksums (190). P_B accepts the opening only if all verifications are successful.

Initial intuition for extractability. If P_A never aborts, then the simulator ($\mathcal{S}$) extracts the seed of P_A by receiving from P_A, upon successive rewindings, the opening of both shares for several checksum-bits, thus gaining the ability to reconstruct the full sequence of seed-bits (solving a linear equation to find seed-bits from known XOR-relations with the checksum-bits), before reaching the overall *open* phase of the X scheme. If P_A aborts the first time that $\mathcal{S}$ asks for an opening of shares, then $\mathcal{S}$ *emulates an abort*. However, a problem may arise when P_A does not abort in the first time but aborts on subsequent attempts (upon rewinding), namely if P_A has a secret criterion for abort. This problem is solved by selecting enough checksum bits such that the probability of non-abort by P_A^* is: either too small to be meaningful (i.e., negligible in the statistical parameter, namely lower that two to the power of the statistical parameter); or it is sufficiently high to prevent P_A^* from being able to *boycott* the reconstruction of any seed-bit (i.e., because $\mathcal{S}$ across different rewindings is able to obtain both shares for enough cheksum bits). Ensuring this property requires that the encoding matrix u has *rank resilient to σ-erasures*. The simulator is described in the paragraph below. The number of checksum bits and an upper bound on number of rewindings are defined afterward.

Remark on possible optimizations. There are three optimizations that were avoided for the purpose of simplicity of description and parameter definition.

– The encoding matrix (177), which defines how the checksum bits are calculated, could be decided apriori, instead of being derived from a random seed proposed by P_B (173). In theory, both parties could determine an optimal "Maximum Distance Separable" matrix (i.e., defining an optimal erasure code) in the sense of having the minimum size that allows reconstruction of the seed vector, regardless of σ checksums being adaptively erased by a malicious P_A^* (i.e., by boycotting the opening of one share of each of σ checksum bits).[32] Instead, the method described in Fig. 14 uses a (pseudo-)random matrix ((173)-(178)), subject to having a sufficiently large length (as a function of the target length and the statistical parameter) – see further discussion below). This makes the number of checksum-bits be larger than the optimal, but still provides an efficient and feasible X commitment scheme from regular bit-commitments.

[32] At the time of writing this, the author does not know how to efficiently compute such an optimal matrix for a practical target length $\kappa_{\mathrm{PRG}} = 128$ and statistical parameter $\sigma = 40$.

- In the overall *open* phase, P_A does not really need to send the target seed in clear (186), since P_B can reconstruct it from the checksums (and verify consistency with all the redundant information). However, this does not impose a significant communication overhead and prevents P_B from having to invert the encoding matrix.
- The overall communication complexity associated with the several regular bit-commitments can be reduced using a random-seed checking technique, as explicitly described further ahead (§D.3.2).

The extractor-simulator.

- **Initial execution.** $\mathcal{S}$ (in the role of P_B in a simulated execution) interacts with the black-box P_A for a full *commit* phase, namely: receiving the commitment of the shares of checksum bits of P_A (182); sending a random challenge of positions of shares to be opened (184); and finally receiving the respective partial opening (185). If P_A^* aborts without completing the overall *commit* phase, then $\mathcal{S}$ *emulates an abort* and the simulation ends. Otherwise, $\mathcal{S}$ knows one share of each checksum bit.
- **Rewindings to obtain complementary shares.** $\mathcal{S}$ then rewinds until the step of selecting a random challenge of positions of shares (183), and sends a new random challenge to P_A^* (184). $\mathcal{S}$ attempts to proceed with the simulation until the end of the commit phase. If P_A^* does not abort, then $\mathcal{S}$ learns complementary shares for several checksum bits, thus becoming able to learn the value of these bits. If P_A^* has aborted, then $\mathcal{S}$ does not learn any new shares in this attempt. $\mathcal{S}$ proceeds with more rewindings, using new challenges of positions of shares, until either: obtaining enough checksum-bits to reconstruct the target seed of P_A;[33] or reaching an upper bound of number of rewindings (discussed below).

Number of checksum bits. Out of the set of checksum bits, consider a subset whose erasure would prevent the reconstruction of some seed-bit (e.g., the lines of the matrix that have a 1 in each column corresponding to such seed-bit). Then, in the overall *commit* phase, a malicious P_A^* could decide to not-abort only when challenged to *open* a particular share of each of those checksum bits. If these shares would indeed be challenged in the first pass of the simulation, then P_A would initially not abort. However, P_A^* would abort after any rewinding that would challenge a different share of any of those checksum bits. Thus, $\mathcal{S}$ would not in this case be able to reconstruct the mentioned seed-bit, i.e., P_A^* would effectively be able to boycott the reconstruction of a seed-bit. However, if this boycott requires that the number of erasures is larger than σ, then this would imply a probability of not-abort lower than the negligible statistical threshold. In other words, $\mathcal{S}$ could in such case safely abort after an adequate bound of number of rewindings.

To satisfy the above-mentioned condition, there must be an overwhelming probability that the pseudo-randomly generated encoding-matrix u is such that the erasure of any subset of σ rows still leaves the remaining rows forming a matrix with rank κ_{PRG}. Clearly, the number of rows (i.e., the number of checksum bits) must be at least as high as the number of columns (i.e., as the number of seed-bits) plus the number of *erasable* bits, or

[33] Namely when the characteristic vectors corresponding to the obtained checksums form a matrix with rank equal to the target length; i.e., when there are enough linearly independent vectors, modulo 2, to solve the linear equation to obtain the seed-bits from the checksum bits.

otherwise the erasure of any σ rows would lead to a matrix with a number of rows (i.e., equations) smaller than the number of bits to decode. The requirement that the rank is sufficiently high after any the erasure of any subset of σ requires yet additional number of checksum bits (i.e., of rows), namely when $\sigma > 1$, to ensure that enough random rows (i.e., equations) are linearly independent.

An over-simplified estimated approximation to a concrete number of checksum bits. Let *extended length* denote the number ℓ' of rows of the encoding matrix, i.e., the number of checksum bits (191). From the set of available rows, the number of *relevant* subsets of rows from which $\mathrm{P}_{\mathrm{A}}^{*}$ can choose one to erase (i.e., in the context of the above-mentioned erasure-attack) is exponential in the statistical parameter (192). As a simplification, inaccurate but serving the purpose of an approximate estimation of a sufficient number of checksum bits,[34] it is assumed that the erasure of any of these subsets would lead to a new random matrix. Under this simplification, it is pertinent to consider the probability that a random binary matrix with a number of rows equal to the original length plus an added length has maximum rank equal to the number of columns (193). This probability approaches 1 exponentially fast with the number of additional rows (194). Then, based on the (contextualized) simplification that the removal of any subset of σ rows leads to a new random matrix, it is easy to calculate the probability that out of the adequate exponential number of random matrices at least one of them does not have the needed rank – such probability should be lower than a negligible value in the statistical parameter (195). The exponents in the equation can be removed by simple transformations and approximations,[35] until the logarithm of a binomial is compared with a linear function of the parameters that define number of extra rows. The logarithm of the Binomial can also be approximated[36] by an expression based on logarithms of simple quotients of the relevant parameters (196). Taking in consideration that the number of rows after erasure is always larger then the number of erased rows, some logarithms can be further approximated[37] by quotients (197). An approximate estimation of a sufficient *extended length* (i.e., number of rows of the encoding matrix) can thus be numerically calculated as the minimal integer value that satisfies the mentioned equation (198). For example, for a target length of $\ell = 128$ bits and a statistical security of $\sigma = 40$ bits, the resulting (estimated approximation of) *extended length* is $\ell' = 391$, i.e., $\sigma_{\mathrm{X}} = 223$ beyond the seed-length plus the bits of statistical security (199). If the target seed length were $\ell = 256$ bits, the resulting extended length would be $\ell' = 449$, for the same statistical security of $\sigma = 40$ bits.

[34] The number of rows can be lowered, based on a more rigorous analysis of probability, considering the inter-dependencies between the subsets from which $\mathrm{P}_{\mathrm{A}}^{*}$ can choose.

[35] Based on the approximation $\log(1-x) \overset{\text{for } x \approx 0}{\approx} -x$

[36] Stirling's approximation: $\log n! \approx n(\log(n)-1) + \log(2\pi n)/2$

[37] Based on the approximations: $y\log(1+x/y) \overset{\text{for } 0<x<y}{\approx} x - x^2/(2y) + x^3/(3y^2)$ and $\log(1/\sigma + 1/(\kappa+a)) \overset{\text{for } \kappa+a>\sigma}{\approx} \log(1/\sigma) + \sigma/(\kappa+a) - \sigma^2/(2(\kappa+a)^2)$

$$\ell' \equiv \ell + a + \sigma \tag{191}$$

$$\text{Bin}(\ell', \sigma) = \frac{\ell'!}{\sigma!(\ell+a)!} = \frac{1}{\sigma!}\prod_{i\in[\sigma]}(\ell+a+i) \tag{192}$$

$$R(\ell, a) \equiv \text{Prob}\left[u = \left\langle u_k \leftarrow^{\$} \{0,1\}^{\ell} : k \in [\ell+a]\right\rangle ; \text{Rank}(u) =^{?} \ell\right] \tag{193}$$

$$R(\ell, a) \approx 1 - 2^{-a} \tag{194}$$

$$1 - R(\ell, a)^{\text{Bin}(\sigma_{\ell'}, \sigma)} < 2^{-\sigma} \Rightarrow \underbrace{\log(\text{Bin}(\ell', \sigma))}_{\equiv \text{LB}} \geq (a - \sigma)\log(2) \tag{195}$$

$$LB \approx (\kappa + a)\log\left(1 + \frac{s}{\kappa + a}\right) + (s)\log\left(1 + \frac{k+a}{s}\right) + \frac{1}{2}\left(\log\left(\frac{1}{s} + \frac{1}{k+a}\right) - \log(2\pi)\right) \tag{196}$$

$$LB \approx s\left(1 - \frac{s-1}{2(k+a)} + \frac{s(4s-3)}{12(k+a)^2} + \log\left(1 + \frac{k+a}{s}\right)\right) - \frac{1}{2}\log(2\pi s) \tag{197}$$

$$\ell' \equiv \alpha(\ell, \sigma) \equiv \lceil \min(\ell' : \ell' \text{ satisfies (195)}) \rceil \tag{198}$$

$$\sigma_{\text{X}} \equiv \ell' - \ell = a + \sigma \tag{199}$$

As mentioned, an optimal (i.e., minimal) extended length could be achieved by agreeing on an optimal encoding matrix in a setup phase. This would also have the benefit of requiring one less communication step. Clearly, the number of 1's in each column – each column corresponds to a seed-bit – must be at least one plus the number of bits of statistical security, such that the erasure of any subset of rows – each row corresponds to a checksum bit – leaves intact at least one row with a 1 in each column. When $\sigma > 1$, this number of rows is not enough to ensure that the resulting rows contain a subset of enough mutually linearly independent (modulo 2) rows. If one were to assume that the first ℓ rows correspond to the identity matrix (i.e., exactly one 1 in each column), and assuming that the extra rows have approximately half 0's and half 1's, it is suggestible that the number of extra rows must be at least equal to twice the statistical parameter, less one. However, the actual minimum may be larger – this is not further discussed herein and is left as an open problem the determination of optimal matrices with 128 and 256 columns, resilient to arbitrary erasure of $\sigma = 40$ rows.

Number of non-aborting rewindings to extract seed of P_A^*. Hereafter it is considered that P_A^* has not aborted in the first step of the simulation (i.e., before any rewinding). Upon each rewinding, an opening of shares in new challenged positions allows P_B to recover more checksum bits, and thus allows a faster reconstruction of the full seed, even in the case where P_A^* may try to boycott some openings. Thus, there is a tradeoff between the expected number of rewindings and the number ℓ' of checksum bits. Assuming a probability of abort of at most two to the power of minus σ, at least $\ell' - \sigma$ can be opened for both shares (i.e., assuming that P_A^* can at most boycott σ checksum bits). After a first execution where one share is opened for each bit position, it is expected that each subsequent rewinding (with a new pseudo-random challenge of positions of shares) that does not lead to abort will lead to the opening of the complementary shares in about half of the remaining non-boycotted unopened bits. For example, for $\ell = 128$ bits of original target value, and $\ell' = 391$ checksum bits of which at most $\sigma = 40$ are boycotted against extraction, it is expected that about 176 new shares are opened in the first non-abort upon rewinding. In such case, $\mathcal{S}$ could then recover the full 128 bits of the seed of P_A^* if the respective 156 checksums form a matrix with sufficient rank – i.e., with at least

128 linearly independent equations modulo 2. However, if that is not the case, then $\mathcal{S}$ attempts again until obtaining a new opening, from which about 88 new complementary shares are expected to be obtained. In the limit, for a worst case scenario matrix with erasures, $\mathcal{S}$ could have to obtain all $\ell' - \sigma$ checksum bits. Nonetheless, this can be done with overwhelming probability after non-abort rewindings in number logarithmic in the number of rows and linear with the statistical security parameter.

Explicit upper bound for number of rewindings. Overall, if P_A has a probability of non-abort equal to two to the minus 40, the expectation is that about 1 in every 2^{40} attempts will result in a non-abort by the simulator. For higher probabilities of non-abort, the expected number of rewindings until another non-abort is correspondingly inversely proportional. Since for lower probabilities it is not necessary to extract the seed, it is adequate to limit the number of rewinding to *about* two to the minus 40 rewinding attempts per each non-abort case. Thus, overall it is defined that the simulator uses an internal counter, to count the number of rewindings, such that it can limit itself to a number of rewindings that is at most the inverse of the threshold probability (two the additive inverse of the statistical security parameter) times the expected needed number of non-abort rewindings that decreases to a negligible value the probability of not recovering a subset of needed checksum bits (this depends on the matrix, but is at most logarithmic in the number of rows). In this way, and considering that a simulation with initial abort does not require any rewinding, the overall expected number of rewindings is equal to the expected number of non-aborting rewindings.

D.3.2 X commitment scheme – RSC optimization

Intuition for optimization. Using a random-seed-checking (RSC) type of technique, the previous protocol can be adjusted to significantly reduce communication complexity. The initial intuition is that the generation of elements used in the cut-and-choose, namely the bit-commitments (for simplicity assumed hereafter as non-interactive) of bit-shares, can themselves be generated from a single short seed (hereafter denoted *RSC-auxiliary seed*). The complementary intuition is that a CR-hash of several bit-commitments is itself a commitment of all the committed bits. Thus, the early opening of the *check* bit-shares (only one per cheksum bit), still within the overall *commit* phase, can be reduced to revealing other short seeds (themselves generated from the RSC-auxiliary seed, and each of them used to generate the bit-commitment) and then verified against a global hash. Then, the final opening (i.e., in the overall *open* phase) can be achieved by simply revealing the original committed value (a seed, in the current context) and the RSC-auxiliary seed from which everything else (i.e., all bit-commitments and all hashes) can be determined. The optimized scheme is described with succinct notation in Fig. 15 and in textual form in the next paragraphs.

Commit phase.

– **1.a. Decide encoding.** P_A and P_B agree on an encoding matrix (200), in a way that there is an overwhelming probability that it is *rank resilient to σ erasures*. For example, this might be decided as a sufficiently large pseudo-random matrix

$\mathrm{P_A}$ (sender) uses $\mathscr{C}_{\mathrm{X}}$ to commit (10) and open (16) value v (with $\ell \equiv |v|$) to $\mathrm{P_B}$ (receiver).

$\mathscr{C}^{\text{Commit}}_{\mathbf{X},sid,cid}[\mathbf{P_A}(v) \leftarrow (\underline{v}, \overline{v}); \mathbf{P_B} \leftarrow \overline{v}]$:

1.a. Decide encoding.

$\mathrm{P_A} \leftrightarrow \mathrm{P_B} : u$ (decide encoding-matrix) (200)

$\mathrm{P_A} : \sigma_{\mathrm{X}} = \alpha(\ell, \sigma)$ (201)

$\mathrm{P_A} : \ell' = \#\text{ofRows}(u) (= \ell + \alpha(\ell, \sigma))$ (202)

(e.g., using steps (173)-(178))

$\mathrm{P_A} : t_k = \oplus_{i \in [\ell]}(u_{k,i} \cdot s_i) : k \in [\ell']$ (203)

1.b. Prepare help seeds and global hash.

$\mathrm{P_A} : T \leftarrow^{?} \{0,1\}^{\kappa_{\mathrm{PRG}}}$ (RSC-auxiliary seed) (204)

$\mathrm{P_A} : U = \mathrm{PRG}[T]((1 + 2\kappa_{\mathrm{PRG}}) \cdot \ell')$ (205)

$\mathrm{P_A} : \left\langle D \in \{0,1\}^{\ell'}, V \in \{0,1\}^{2\kappa_{\mathrm{PRG}} \cdot \ell'} \right\rangle \overset{\text{Parse}}{\leftarrow} U$ (206)

$\mathrm{P_A} : \langle d_{k,0} : k \in [\ell'] \rangle \overset{\text{Parse}}{\leftarrow} D$ (207)

$\mathrm{P_A} : d_{k,1} = t_k \oplus d_{k,0} : k \in \ell'$ (208)

($d_{k,j}$ are shares of the checksum-bits t_k)

$\mathrm{P_A} : \langle r_{k,j} : k \in [\ell'], j \in \{0,1\} \rangle \overset{\text{Parse}}{\leftarrow} V$ (209)

($r_{k,j}$ are denoted help seeds)

For $k \in [\ell'], j \in \{0,1\} : (\mathrm{P_A})$

$\overline{d}_{k,j} = \mathscr{C}^{\text{Commit}}[r_{k,j}](d_{k,j})$ (210)

($r_{k,j} \equiv \underline{d}_{k,j}$ is used as randomness)

$h = \text{CR-Hash}(\langle \overline{d}_{k,j} : k \in [\ell'], j \in \{0,1\} \rangle)$ (211)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{global-hash}, \textit{context}, h)$ (212)

1.c. Challenge-response.

$\mathrm{P_B} \rightarrow \mathrm{P_A} : c \leftarrow^{\$} \{0,1\}^{\ell'}$ (213)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{challenge-positions}, \textit{context}, c)$ (214)

For $k \in [\ell'] : (\mathrm{P_A} \rightarrow \mathrm{P_B})$

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{bit-share}, \textit{context}, d_{k,c_k})$ (215)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{help-seed}, \textit{context}, r_{k,c_k})$ (216)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{complement-com}, \textit{context}, \overline{d}_{k,c_{1-k}})$ (217)

1.d. Verify responses.

For $k \in [\ell'] : \mathrm{P_B} :$

$\overline{d}_{k,c_k} = \mathscr{C}^{\text{Commit}}[r_{k,c_k}](d_{k,c_k})$ (218)

$\text{CR-Hash}(\langle \overline{d}_{k,j} : k \in [\ell'], j \in \{0,1\} \rangle) =^{?} h$ (219)

$\mathscr{C}^{\text{Open}}_{\mathbf{X},sid,cid}(\overline{v})[\mathbf{P_A}(v, \underline{v}); \mathbf{P_B} \leftarrow v]$:

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{send-auxi-seed}, \textit{context}, T)$ (220)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{send-committed-value}, \textit{context}, v)$ (221)

$\mathrm{P_B}$: Reconstruct h from (s, u, T) (steps (203)-(211)) (222)

$\mathrm{P_B} : h =^{?} h$ (223)

Fig. 15. $\mathscr{C}_{\mathbf{X}}$ from regular bit-commitments (RSC optimized). Legend: legend of Fig. 14 also applies. $\mathscr{C}^{\text{Commit}}[(r_{k,j})](d_{k,j})$ denotes a commitment (locally produced, non-interactively) of share-bit $d_{k,j}$ whenever the secret auxiliary input $r_{k,j}$ is pseudo-random (i.e., $r_{k,j} \equiv \underline{d}_{k,j}$).

(as described for the unoptimized protocol in Fig. 14), or alternatively by simply agreeing on a suitable matrix pre-computed in a setup phase (for the particular target length and statistical security parameter). The encoding matrix is parameterized by an extended length ℓ' (number of rows), which is larger than the number of target bits (number of columns) by a statistical parameter σ_{X} sufficiently larger than the statistical security parameter σ (201), as already discussed §D.3.1. If the matrix is defined deterministically (or in a setup phase) simply based on the target length ℓ and the statistical security parameter σ, i.e., without depending on any randomness from $\mathrm{P_B}$, then it is actually not required that $\mathrm{P_B}$ learns immediately the matrix. Instead, it can locally calculate it later (215) once necessarily learning the extended length of the encoded vector. Nonetheless, for simplicity it is assumed that both parties know in advance the length ℓ of the vector of bits being committed and respectively decide an encoding matrix.

Based on the encoding matrix, $\mathrm{P_A}$ (the sender) computes the checksum bits of the seed being committed, each determined as the inner product (modulo 2) of the seed with the respective row of the encoding matrix (203).

– **1.b. Prepare help seeds and hash.** Then, $\mathrm{P_A}$ locally selects an auxiliary secret and random seed (204) that will be used as generator in the RSC technique. $\mathrm{P_A}$ expands the auxiliary seed (205) to obtain an initial vector D of bits and enough additional bits V for several help seeds (206). The initial vector of pseudo-random bits is

parsed as the shares (index 0) of the checksum bits (207). The complementary shares (index 1) are calculated by XOR with the respective checksum bit (208). Then, the remaining pseudo-random generated bits are parsed into *help seeds* in number equal to twice the number of checksum bits (209). Then, P_A locally produces commitments to all the bit-shares of checksums, using an underlying regular (non-interactive) bit-commitment scheme, using the respective help-seed as the only needed source of randomness (210).[38] It is paramount that these commitments are not yet sent to P_B. P_A then calculates a *global hash* of the concatenation of all bit-commimtents (211), and sends the result to P_B (212).

- **1.c. Challenge-response cut-and-choose.** After receiving the global hash, P_B computes a random vector of challenge bits (213), with one such bit per checksum index, and sends it to P_A (214). Still as part of the commit phase, P_A answers to the cut-and-choose challenges, as follows: For each challenge bit, P_A sends to P_B: the *check* bit-share of the respective checksum bit, but not the complementary share (215); the help seed corresponding to the *check* share, but not the help seed corresponding to the complementary share (216); the commitment of the complementary share, but not of the *check* share (217).
- **1.d. Verify responses.** After receiving the replies from P_A, P_B locally computes, for each challenge bit: the commitment of the revealed share, as would be obtained if using the revealed seed as randomness (218). At this point, P_B has a candidate commitment for each share of each checksum bit. Using these values, P_B computes a candidate global hash (219). If the candidate global hash does not match with the global hash previuosly received from P_A, then P_B rejects the commit phase and aborts. Otherwise, if the global hashes match then P_B accepts the commit phase.

Open phase. To open, P_A (the sender) simply reveals the RSC-auxiliary seed (220) and the committed value (in the current context also a seed) (221). Using the RSC-auxi seed and the committed value, P_B reconstructs the global hash and verifies it against the one calculated in the *commit* phase (222). If the values match, then P_B accepts the opening, other wise it rejects it (223).

Intuition for extractability. Extractability is obtained in the same way as in the unoptimized version of the protocol. Specifically, for each completed commit phase without abort by a possibly malicious P_A^*, the extractor simulator learns one share for each checksum bit. After enough non-aborting attempts, $\mathcal{S}$ is able to recover enough checksum bits to reconstruct the value committed by P_A^*.

Intuition for hiding property. In the *commit* phase, for each checksum bit only a single random share is revealed, which does not contain any information about the committed bit. All other shares are committed, which means the overall hiding is reduced to the hiding property of the underlying regular bit-commitment scheme.

[38] For simplicity, it is here assumed that the underlying regular bit-commitment scheme is non-interactive. Interactive schemes could be considered with a more intricate description.

Table 2. Communication complexity of Q commitment schemes

$\mathscr{C}_{\mathrm{Q}}(1^{\kappa_{\mathrm{Hash}}})$ $\mathrm{P_B}$ as sender	Direction of communication	Commit phase	Open phase
[PW09, §6]	$\mathrm{P_B} \to \mathrm{P_A}$	$(\ell \times \sigma)\left(4\lvert\mathscr{C}^{\mathrm{C}}(1^1)\rvert + 2\lvert\mathscr{C}^{\mathrm{O}}(1^1)\rvert\right)$	$(\ell \times \sigma)\lvert\mathscr{C}^{\mathrm{O}}(1^1)\rvert$
	$\mathrm{P_A} \to \mathrm{P_B}$	$\lvert\mathscr{C}^{\mathrm{C}}(1^\sigma)\rvert + \lvert\mathscr{C}^{\mathrm{O}}(1^\sigma)\rvert^*$	–
This paper (Fig. 16)	$\mathrm{P_B} \to \mathrm{P_A}$	$(\ell + \sigma_{\mathrm{Q}})\left(2\lvert\mathscr{C}^{\mathrm{C}}(1^1)\rvert\right) + \sigma_{\mathrm{Q}} + 2\kappa_{\mathrm{PRG}} + \sigma_{\mathrm{X}}$	$(\ell + \sigma_{\mathrm{Q}})(1 + \lvert\mathscr{C}^{\mathrm{O}}(1^1)\rvert)$
	$\mathrm{P_A} \to \mathrm{P_B}$	$\kappa_{\mathrm{PRG}} + (\kappa_{\mathrm{PRG}} + \sigma_{\mathrm{X}})(2\lvert\mathscr{C}^{\mathrm{C}}(1^1)\rvert + \lvert\mathscr{C}^{\mathrm{O}}(1^1)\rvert)$	$\kappa_{\mathrm{PRG}} + \sigma_{\mathrm{X}}\lvert\mathscr{C}^{\mathrm{O}}(1^1)\rvert$
This paper (RSC) (Fig. 17)	$\mathrm{P_B} \to \mathrm{P_A}$	$\kappa_{\mathrm{Hash}} + 2\kappa_{\mathrm{PRG}} + \sigma_{\mathrm{Q}} + \sigma_{\mathrm{X}}$	$(\ell + \sigma_{\mathrm{Q}})\times$ $\left(1 + \lvert\mathscr{C}^{\mathrm{C}}(1^1)\rvert\right)$
	$\mathrm{P_A} \to \mathrm{P_B}$	$\kappa_{\mathrm{PRG}} + \kappa_{\mathrm{Hash}} + (\kappa_{\mathrm{PRG}} + \sigma_{\mathrm{X}})(1 + \kappa_{\mathrm{PRG}} + \lvert\mathscr{C}^{\mathrm{C}}(1^1)\rvert)$	$2\kappa_{\mathrm{PRG}}$

Legend: legend of Table 1 applies; $\mathscr{C}_{\mathrm{Q}}$ (equivocable commitment scheme); σ_{Q} (statistical term such that the probability of breaking the binding property is less than $2^{-\sigma}$ – see §D.4.1 for details.) The mentioned communication sizes already include instantiating $\mathscr{C}_{\mathrm{X}}$ (used in Fig. 16 and Fig. 17) by the complexities calculated in Table 1 (respectively from Fig. 14 and Fig. 15). *Note: the mentioned communication from $\mathrm{P_A}$ to $\mathrm{P_B}$ for the commit phase of $\mathscr{C}_{\mathrm{Q}}$ [PW09, §6] is assuming that the parallelization across ℓ bits uses the same challenge for all cases).

D.4 Q commitment schemes from regular bit-commitments

This subsection considers Q commitment schemes based on regular bit-commitments. First, it gives a brief description of a method from prior work and describes an intuition for communication improvement. Then it specifies the new protocol, starting with an unoptimized version (§D.4.1) and then introducing a RSC optimization (§D.4.2) based on explicit use of a PRG and CR-Hash (both already required in protocol #1). The comparison in terms of communication is summarized in Table 2. The schemes use a respective (unoptimized or RSC-optimized) underlying X commitment scheme, which can be instantiated by regular bit-commitments as described in the previous subsection.

A prior Q scheme. The Q commitment scheme from Pass and Wee [PW09, §6] (also based on [Kil94, §2.1]) is realized by parallelizing separate Q bit-commitments (*commit* and *open*) of each bit of the target value. Each such Q bit-commitment requires producing regular bit-commitments in number proportional to (namely four times) the statistical parameter. The intuition starts with each target bit being represented as a two-by-two matrix of bits, where the top and bottom rows are equal, and where each row is composed of two random bits (*shares*) that XOR together into the target bit. The sender commits individually to each of the four bits in each representation, and does so for a number of representations equal to four times the statistical parameter, for each *target* bit being committed in a Q manner. Then, the sender is asked to open one random column in each instance, with the column position being chosen by the receiver (who had previously committed to the choice), to show that the column was committing two equal bits. While this does not reveal anything about the committed *target*, the receiver gains statistical confidence that at least one matrix is composed of two identical rows (i.e., that at least

one of the unopened columns is also committing two unknown but equal bits). In contrast, a simulator in the role of sender in a simulated execution would have been able to guess the challenge of the receiver (by means of *non-local* rewinding), and so could have built with two different bits each of the unopened columns. Then, in the overall *open* phase, the sender randomly selects one row-position (top or bottom) and opens the corresponding bit in the unopened column across all instances, thus letting the receiver compute the target bit (equal to the XOR of the two bits in the row). In contrast, the simulator has different bits in each row so it is able to equivocate the opening to any intended bit.

Intuition for a new Q scheme. The above described scheme achieves the intended Q property, but does so at the cost of a quadratic number of bit-commitments, namely four times the target length multiplied by the statistical parameter. The new Q-schemes, devised below, improve communication complexity by making the statistical parameter become additive with (twice) the length of the target value being committed; i.e., bit-commitments are needed only in number bounded by a function linear in the length of the committed value and in the statistical parameter.

Intuitively, equivocability can be considered for each bit in isolation, giving P_B the ability to open each bit as a 0 or a 1. As a starting point, this suggests that P_B (the sender) should produce two commitments for each target bit and later *open* only one. In this approach, a simulator ($\mathcal{S}$) would be able to equivocate by controlling which commitments to open (0 or 1), whereas a real P_B (the sender) would not have control over which commitments are selected for opening and so could not choose a value to equivocate. Still, if a target bit would be undefined, i.e., if a malicious P_B^* would commits to two different values for the same target bit, then the final opening could vary depending on the challenge selected by P_A. In order to ensure binding – also in this flavor of having the committed value defined at the end of the *commit* stage – the actual solution involves also committing to additional check-sum bits that must later allow for a overwhelming probability of error detection in case there is any inconsistency. This is somewhat similar to what was considered for the X schemes, but with the simplification that the checksums do not need to ensure ability to reconstruct the original vector of bits. Consequently, the number of checksums (i.e., besides the original bits) needs to be only linear with the statistical parameter. In the benefit of statistical security, a way if found to have P_A commit to the pairs of additional bits (i.e., the step where it decides about honestly committing to the same bit or maliciously committing to different bits) before the actual checksum-subsets are defined. This is achieved by committing to pairs of help-bits and only later, after learning the checksum subsets but still in the overall commit phase, revealing the respective *differential bits* that transform the help-bits into the checksum-bits. The remaining subsection describes in more detail the unoptimized and RSC-optimized versions of the mention new Q scheme.

D.4.1 Q commitment scheme – unoptimized

The unoptimized Q commitment scheme is described with succinct notation in Fig. 16 and in textual form in the next paragraphs.

Commit phase.

- **Commit target and help bits.** P_B (the sender) defines an adequate *expanded statistical term* σ_Q (224), proportional to the statistical parameter (with multiplicative factor 2.4, as specified ahead). P_B then selects σ_Q *help bits* (225). Then, P_B commits twice to each target bit (226) and twice to each help bit (227).
- **1.b. Decide and commit cheksums.** P_A selects a random PRG seed (228) (denoted *seed-for-subsets*), which will be used to determine how the ckecksum bits will be calculated. P_A sends the seed to P_B (229). Both parties use the seed-for-subsets to generate a pseudo-random bit-string with a number of bits equal to the product of the target length and the expanded statistical term (230). Then, the obtained bit-string is parsed as a sequence of vectors u_k, each having the target length ℓ and being defined as a characteristic vector where a bit 1 corresponds to inclusion in a subset, whereas 0 corresponds to non-inclusion (231). The number of vectors is thus equal to the expanded statistical term σ_Q. Equivalently, this list of vectors corresponds to an encoding matrix (of elements $u_{k,i}$). For each vector, P_B then calculates a XOR of the target bits (of the message being committed) that were included in the respective subset (232), i.e., calculates checksum bits t_k in number equal to the expanded statistical term. Then, P_B calculates the *differential bits* (z_k) that are required to change the previously committed *help bits* (w_k) into the now calculated *checksum bits* (t_k) (233). P_B sends the vector of *differential bits* to P_A (234).
- **1.c Prepare components for later opening.** P_A selects a new random seed (235) and commits to it to P_B, using an extractable commitment scheme (236). It is assumed that this scheme is based on regular bit-commitments, e.g., as defined in the previous section (Fig. 14, Fig. 15).

Open phase.

- **2.a. Finalize decision of positions to open.** P_B (the sender) selects a random binary vector with an *extended length* equal to the number ℓ of target bits plus the number σ_Q of help bits (237) and sends it to P_A (238). P_A then opens to P_B the previously committed seed (239). Locally, each party uses the seed opened by P_A to generate a random bit-string with the extended length (240). Locally, each party XORs the two strings with the extended length (241), obtaining as result a vector specifying for each target bit and each help bit which commitment (one out of two) will be open.
- **2.b. Open selected positions.** P_B proceeds to open one out of two commitments of each target bit (242) and each help bit (243)-(244), in the position specified by the challenge vector determined above. Then, P_A computes the tentative checksums (t_k) as the XOR between the differential bits (z_k previously received) and the respective tentative help bits (w_{i,c_i} just opened by P_B.
- **2.c. Verify checksums.** Finally, P_A computes the adequate checksums of the opened target bits and verifies if they are consistent with the respective tentative checksums obtained in the previous step (246). If all verifications are correct, then P_A accepts the opening of the tentative target bits, otherwise it rejects.

$\mathrm{P_B}$ (sender) uses $\mathscr{C}_{\mathrm{Q}}$ to commit (8) and open (13) value v (with $\ell \equiv |v|$) to $\mathrm{P_A}$ (receiver).

$\mathscr{C}^{\text{Commit}}_{\mathbf{Q},sid,cid}[\mathbf{P_A} \leftarrow \overline{v}; \mathbf{P_B}(v) \leftarrow (\underline{v}, \overline{v})]$:

1.a. Commit target and help bits.

$\mathrm{P_A}, \mathrm{P_B} : \sigma_{\mathrm{Q}} = \beta(\sigma)$ (as defined in (248)) (224)

$\mathrm{P_B} : w \leftarrow^{\$} \{0,1\}^{\sigma_{\mathrm{Q}}}$ (help vector) (225)

For $i \in [\ell], j \in \{0,1\} : (\mathrm{P_B} \rightarrow \mathrm{P_A})$

$\mathscr{C}^{\text{Commit}}_{sid,cid,i,j,1}[\mathrm{P_A} \leftarrow \overline{v}_{i,j}; \mathrm{P_B}(v_i) \leftarrow (\underline{v}_{i,j}, \overline{v}_{i,j})]$ (226)

For $i \in [\sigma_{\mathrm{Q}}], j \in \{0,1\} : (\mathrm{P_B} \rightarrow \mathrm{P_A})$

$\mathscr{C}^{\text{Commit}}_{sid,cid,i,k,2}[\mathrm{P_A} \leftarrow \overline{w}_{i,j}; \mathrm{P_B}(w_i) \leftarrow (\underline{w}_{i,j}, \overline{w}_{i,j})]$ (227)

1.b. Decide and commit checksums.

$\mathrm{P_A} : s_1 \leftarrow^{\$} \{0,1\}^{\kappa_{\mathrm{PRG}}}$ (228)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{seed-subsets}, \textit{context}, s_1)$ (229)

$\mathrm{P_A}, \mathrm{P_B} : u = \mathrm{PRG}[s_1](\ell \times \sigma_{\mathrm{Q}})$ (230)

$\mathrm{P_A}, \mathrm{P_B} : \langle u_{k,i} : k \in [\sigma_{\mathrm{Q}}], i \in [\ell] \rangle \overset{\text{Parse}}{\leftarrow} u$ (231)

$\mathrm{P_B} : t_k = \oplus_{i \in [\ell]} (u_{k,i} \cdot v_i) : k \in [\sigma_{\mathrm{Q}}]$ (232)

$\mathrm{P_B} : z_k = t_k \oplus w_k : k \in [\sigma_{\mathrm{Q}}]$ (233)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{differential-bits}, \textit{context}, z)$ (234)

1.c. Prepare components for later opening.

$\mathrm{P_A} : s_2 \leftarrow^{\$} \{0,1\}^{\kappa_{\mathrm{PRG}}}$ (235)

$\mathscr{C}^{\text{X-Commit}}_{\mathrm{X},sid,cid,0}[\mathrm{P_A}(s_2) \leftarrow (\underline{s}_2, \overline{s}_2); \mathrm{P_B} \leftarrow \overline{s}_2]$ (236)

$\mathscr{C}^{\text{Open}}_{\mathbf{Q},sid,cid}(\overline{v})[\mathbf{P_A} \leftarrow v; \mathbf{P_B}(x) \leftarrow (v, \underline{v})]$:

2.a. Finalize decision of positions to open.

$\mathrm{P_B} : c_B \leftarrow^{\$} \{0,1\}^{\ell+\sigma_{\mathrm{Q}}}$ (237)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{contrib-pos}, \textit{context}, c_B)$ (238)

$\mathscr{C}^{\text{Open}}_{\mathrm{X},sid,cid,0}[\mathrm{P_A}(s_2) \leftarrow (\underline{s}_2, \overline{s}_2); \mathrm{P_B} \leftarrow \overline{s}_2]$ (239)

$\mathrm{P_A}, \mathrm{P_B} : c_A = \mathrm{PRG}[s_2](\ell + \sigma_{\mathrm{Q}})$ (240)

$\mathrm{P_A}, \mathrm{P_B} : c = c_A \oplus c_B$ (241)

2.b. Open selected positions.

For $i \in [\ell] : (\mathrm{P_B} \rightarrow \mathrm{P_A} : \text{open target bits})$

$\mathscr{C}^{\text{Open}}_{sid,cid,i,c_i,1}(\overline{v}_{i,c_i})[\mathrm{P_A} \leftarrow v_i; \mathrm{P_B}(\underline{v}_{i,c_i})]$ (242)

$\mathrm{P_A}, \mathrm{P_B} : c'_k \equiv c_{\ell+k} : k \in [\sigma_{\mathrm{Q}}]$ (243)

For $k \in [\sigma_{\mathrm{Q}}] : (\mathrm{P_B} \rightarrow \mathrm{P_A} : \text{open help bits})$

$\mathscr{C}^{\text{Open}}_{sid,cid,k,c'_k,2}(\overline{w}_{k,c'_k})[\mathrm{P_A} \leftarrow w_k; \mathrm{P_B}(\underline{w}_{k,c'_k})]$ (244)

2.c. Verify checksums.

$\mathrm{P_A} : t_k = z_k \oplus w_{k,c'_k} : k \in [\sigma_{\mathrm{Q}}]$ (245)

$\mathrm{P_A} : \oplus_{i \in [\ell]} (u_{k,i} \cdot v_i) =^{?} t_k : k \in [\sigma_{\mathrm{Q}}]$ (246)

Fig. 16. $\mathscr{C}_{\mathrm{Q}}$ from regular bit-commitments (unoptimized) In the context of protocol #1, this Q scheme may be applied to a CR-Hash output computed by $\mathrm{P_B}$. The sequence of steps (236),(238),(239),(241) corresponds to a coin-flipping, simulatable for the case of corrupted $\mathrm{P_A}$, but not simulatable for the case of a corrupted $\mathrm{P_B}$. Steps (228)-(230) could be replaced by a simple select-and-send of a random bit-string of positions of adequate length, at the expense of a slight increase in communication. Parameter $\sigma_{\mathrm{Q}} \approx 2.4\sigma$ (the number of help/checksum bits, besides the target bits) is chosen to ensure that the committed value is defined at the end of the *commit* phase. The protocol is described based on a X commitment scheme that can be instantiated based on regular bit-commitments (e.g., as shown in Fig. 14 and Fig. 15).

Intuition for equivocability. In the *commit* phase, for each pair of commitments associated with each target bit v_i (226) and with each help bit w_i (227), $\mathcal{S}$ in the role of $\mathrm{P_B}$ in a simulated execution commits to a 0 and a 1, in a randomly permuted order. Later, in the *open* phase, $\mathcal{S}$ will be able to control the result of the embedded coin-flipping ((236),(238),(239),(241)) that determines the components that need to be opened in each position of the extended vector. Since, in this simulation, each bit is committed with two components, one as 0 and another as 1, $\mathcal{S}$ will be able to open any target vector and a respective consistent checksum vector of bits.

Intuition for binding. Once the pairs of bits (i.e., what should be a pair of copies of the same bit) are committed ((226), (227)), even a malicious $\mathrm{P_B^*}$ can later only open a bit that it had committed to ((242),(244)). The malicious $\mathrm{P_B^*}$ may for some pairs commit to two different bit-values, which means the final value is not actually determined at the end of the *commit* phase. These cases are hereafter denoted as *bad* instances. The scheme intends to ensure binding also in the sense that at the end of the commit phase there is

at most one target vector that has noticeable probability of successful opening, despite whatever malicious behavior by P_B^*.

For simplicity, assume that P_B produces only one *bad* target instance, e.g., for the first target bit it commits to one component as 0 and the other one as 1. Then, assume that for the *help bits* it does the same for a particular subset, thus creating a subset of *bad* help-bit instances. Since each diffeerential bit imposes a single mapping between each help-bit componente and a respective checksum-bit component, it follows that the subset of **bad** help-bit instances has a direct correspondence with a subset of *bad* checksum-bits subset. If one of the *good* checksum-bit instances includes the single *bad* target bit in its XOR-subset, then there is at most one target vector that can be successfully opened, i.e., binding is ensured in such case. Thus, in the case of a single *bad* checksum bit, breaking the *binding* property requires that P_B selected *bad* target bits for at least all checksum-bit positions that include the *bad* target bit. However, in the devised protocol the checksum subsets are (on purpose) only defined (231) after P_A has committed to the target bits and the help bits, i.e., at a time where P_B has already determined which ones are good and which ones are bad.

Assume now that all checksum-bits instances that use the *bad* target bit in their XOR-subset are themselves *bad*. For example, P_B^* could ensure this by making *bad* all the help-bit instances, but at the cost of reducing the probability of successful opening to a negligible amount (two to the minus the number of help bits). Each of the *bad* checksum bits contributes with a multiplicative factor 1/2 for the probability of failed opening (hereafter denoted as *error*), because in each such position each possible component has 1/2 probability of being chosen for opening. On the other extreme, if all help-bit instances are good, then binding is never broken. Thus, the optimal behavior for a malicious P_B^* is somewhere in the middle, with P_B^* selecting several good and several bad help-bit instances. Once the number b of bad checksum instances built by P_B^* is fixed, the probability of error (i.e., of successful opening in spite of inconsistent commitments of target-bits or help-bits) is given by (247), where n is the total number of checksum instances.

$$\text{Prob}_{\text{Error}}(n,b) = 2^{-(n+b)} \times \sum_{j=1}^{b} n!/((n-j)!j!) \tag{247}$$

This is the result of multiplying $1/2$ for each bad instance, and then multiply the probability that the b bad instances contain all the checksum instances that include the *bad* target bit in their XOR-subset. The protocol only needs to use the minimal statistical term σ_Q that implies a probability of error detection consistent with the statistical parameter (248).

$$\sigma_Q \equiv \beta(\sigma) \equiv min(n : 2^{-\sigma} > max(\text{Prob}_{\text{Error}}(n,b) : b \in \{1,...,n\})) \tag{248}$$

For example: with $\sigma_Q = 91$ or $\sigma_Q = 301$ checksum instances, out of which P_B^* would optimally produce $b = 31$ or $b = 301$ bad ones, respectively, the statistical security is slightly more than $\sigma = 40$ bits or $\sigma = 128$ bits, respectively. The method could be improved (e.g., selecting the encoding matrix u from a different distribution), but the above analysis is sufficient to show that security can be obtained with a linear number of checksum bits, e.g., with $\sigma_Q = 2.4\sigma$ being enough for all practical parameters

of statistical security σ. The case where P_A produces more than one bad target bit commitment does not favor P_B^*, so the above analysis is enough.

D.4.2 Q commitment scheme – RSC optimization

Intuition. Similarly to what was shown with the X commitment scheme, a RSC approach can also reduce the communication complexity of the Q unoptimized commitment scheme. The intuition is almost the same: in the commit phase, the sender sends a global hash instead of sending a sequence of commitments for each instance of the cut-and-choose; in the open phase, the opening of each instance corresponds to just sending a seed. The bulk of the communication is for the commitment instances that do not need to be opened, as the verification of the global hash requires knowing the actual commitment values for those instances. There is however a difference. For equivocability the simulator needs to essentially "lie" about some commitments, so the final opening cannot be of a single seed for all the instances, but rather of a seed for each instance that needs opening. The RSC-optimized Q commitment scheme is described with succinct notation in Fig. 17 and in textual form in the next paragraphs.

Commit phase.

- **1.a. Prepare seeds, commitments and global hash.** Steps similar to those in the corresponding stage of the unoptimized protocol (Fig. 16), but with the difference that the commitments are produced locally by P_B (the sender), i.e., not sent to P_A. Then, P_B computes the CR-Hash of the concatenation of all commitments (256) and sends it to P_A (257).
- **1.b. Decide and commit checksums** Same as in corresponding stage of the unoptimized protocol (Fig. 16) – P_A sends a random seed to P_B, then P_B uses it to calculate the checksum bits and then sends the respective differential bits to P_A.
- **1.c. Prepare components for later opening** Same as in corresponding stage of the unoptimized protocol (Fig. 16) – P_A commits her contribution to a coin-flipping.

Open phase.

- **2.a. Finalize decision of positions to open** Same as in corresponding stage of the unoptimized protocol (Fig. 16) – both parties learn a binary vector of challenges, specifying for each position which component should have its seed and bit revealed – the complementary component will have its commitment revealed.
- **2.b. Open instances and verify global hash** P_B (the sender) sends the target vector (272) and the help vector (273) to P_A. Then, P_B concatenates the challenged component of all target-bit commitments (274) and of all help-bit commitments (275), including only the challenged components, and sends it to P_A (276). Then, P_B concatenates the random seeds corresponding only the complementary of the challenged compoenent (277) and sends it to P_A.
  P_A is now able to reconstruct the global hash. First, it uses the received targets-bits and respective seeds to recompute the corresponding target-bit commitments (279).

$\mathrm{P_B}$ (sender) uses $\mathscr{C}_{\mathrm{Q}}$ to commit (8) and open (13) value v (with $\ell \equiv |v|$) to $\mathrm{P_A}$ (receiver).

$\mathscr{C}^{\text{Commit}}_{\mathbf{Q},sid,cid}[\mathbf{P_A} \leftarrow \overline{v}; \mathbf{P_B}(v) \leftarrow (\underline{v}, \overline{v})]$:

1.a. Prepare seeds, commitments and global hash.

$\mathrm{P_B} : \sigma_{\mathrm{Q}} = \beta(\sigma)$ (as defined in (248)) (249)

$\mathrm{P_B} : w \leftarrow^{\$} \{0,1\}^{\sigma_{\mathrm{Q}}}$ (help vector) (250)

$\mathrm{P_B} : r_{k,j} \leftarrow^{?} \{0,1\}^{\kappa_{\mathrm{PRG}}} : k \in [\ell + \sigma_{\mathrm{Q}}], j \in \{0,1\}$ (251)

(RSC-auxiliary seeds for commitments of bits)

$\mathrm{P_B} : \overline{v}_{i,j} = \mathscr{C}^{\text{Commit}}[r_{i,j}](v_i) : i \in [\ell], j \in \{0,1\}$ (252)

$\mathrm{P_B} : \overline{V} = \langle \overline{v}_{i,j} : i \in [\ell], j \in \{0,1\} \rangle$ (253)

$\mathrm{P_B} : \overline{w}_{k,j} = \mathscr{C}^{\text{Commit}}[r_{\ell+k,j}](w_k) : k \in [\sigma_{\mathrm{Q}}], j \in \{0,1\}$ (254)

$\mathrm{P_B} : \overline{W} = \langle \overline{w}_{k,j} : k \in [\sigma_{\mathrm{Q}}], j \in \{0,1\} \rangle$ (255)

$\mathrm{P_B} : h = \text{CR-Hash}(\overline{V} || \overline{W})$ (256)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{global-hash}, context, h)$ (257)

1.b. Decide and commit checksums (as in (228)-(234)).

$\mathrm{P_A} : s_1 \leftarrow^{\$} \{0,1\}^{\kappa_{\mathrm{PRG}}}$ (258)

$\mathrm{P_A} \rightarrow \mathrm{P_B} : (\texttt{seed-subsets}, context, s_1)$ (259)

$\mathrm{P_A}, \mathrm{P_B} : u = \mathrm{PRG}[s_1](\ell \times \sigma_{\mathrm{Q}})$ (260)

$\mathrm{P_A}, \mathrm{P_B} : \langle u_{k,i} : k \in [\sigma_{\mathrm{Q}}], i \in [\ell] \rangle \overset{\text{Parse}}{\leftarrow} u$ (261)

$\mathrm{P_B} : t_k = \oplus_{i \in [\ell]} (u_{k,i} \cdot v_i) : k \in [\sigma_{\mathrm{Q}}]$ (262)

$\mathrm{P_B} : z_k = t_k \oplus w_k : k \in [\sigma_{\mathrm{Q}}]$ (263)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{differential-bits}, context, z)$ (264)

1.c. Prepare components for later opening (as in (235)-(236)).

$\mathrm{P_A} : s_2 \leftarrow^{\$} \{0,1\}^{\kappa_{\mathrm{PRG}}}$ (265)

$\mathscr{C}^{\text{X-Commit}}_{\mathrm{X},sid,cid,0}[\mathrm{P_A}(s_2) \leftarrow (\underline{s}_2, \overline{s}_2); \mathrm{P_B} \leftarrow \overline{s}_2]$ (266)

$\mathscr{C}^{\text{Open}}_{\mathbf{Q},sid,cid}(\overline{v})[\mathbf{P_A} \leftarrow v; \mathbf{P_B}(x) \leftarrow (v, \underline{v})]$:

2.a. Finalize decision of positions to open (as in (237)-(241)).

$\mathrm{P_B} : c_B \leftarrow^{\$} \{0,1\}^{\ell + \sigma_{\mathrm{Q}}}$ (267)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{contrib-pos}, context, c_B)$ (268)

$\mathscr{C}^{\text{Open}}_{\mathrm{X},sid,cid,0}[\mathrm{P_A}(s_2) \leftarrow (\underline{s}_2, \overline{s}_2); \mathrm{P_B} \leftarrow \overline{s}_2]$ (269)

$\mathrm{P_A}, \mathrm{P_B} : c_A = \mathrm{PRG}[s_2](\ell + \sigma_{\mathrm{Q}})$ (270)

$\mathrm{P_A}, \mathrm{P_B} : c = c_A \oplus c_B$ (271)

2.b. Open instances and verify global hash.

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{target-bits}, context, v)$ (272)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{help-bits}, context, w)$ (273)

$\mathrm{P_B} : \overline{V}' = \langle \overline{v}_{i,c_i} : i \in \ell \rangle$ (274)

$\mathrm{P_B} : \overline{W}' = \langle \overline{w}_{k,c_k} : k \in \sigma_{\mathrm{Q}} \rangle$ (275)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{half-commitments}, context, (\overline{V}', \overline{W}'))$ (276)

$\mathrm{P_B} : R = \langle r_{k,1-c_k} : k \in [\ell + \sigma_{\mathrm{Q}}] \rangle$ (277)

$\mathrm{P_B} \rightarrow \mathrm{P_A} : (\texttt{half-seeds}, context, R)$ (278)

$\mathrm{P_A} : \overline{v}_{i,1-c_i} = \mathscr{C}^{\text{Commit}}[r_{i,1-c_i}](v_i) : i \in [\ell]$ (279)

$\mathrm{P_A} : k' \equiv \ell + k : k \in [\sigma_{\mathrm{Q}}]$ (280)

$\mathrm{P_A} : \overline{w}_{k,1-c_k} = \mathscr{C}^{\text{Commit}}[r_{k',1-c_{k'}}](w_k) : k \in [\sigma_{\mathrm{Q}}]$ (281)

$\mathrm{P_A} : \overline{V} = \langle \overline{v}_{i,j} : i \in [\ell], j \in \{0,1\} \rangle$ (282)

$\mathrm{P_A} : \overline{W} = \langle \overline{w}_{k,j} : k \in [\sigma_{\mathrm{Q}}], j \in \{0,1\} \rangle$ (283)

$\mathrm{P_B} : \text{CR-Hash}(\overline{V} || \overline{W}) =^{?} h$ (284)

2.c. Verify checksums.

$\mathrm{P_A} : t_k = z_k \oplus w_k : k \in [\sigma_{\mathrm{Q}}]$ (285)

$\mathrm{P_A} : \oplus_{i \in [\ell]} (u_{k,i} \cdot v_i) =^{?} t_k : k \in [\sigma_{\mathrm{Q}}]$ (286)

Fig. 17. $\mathscr{C}_{\mathrm{Q}}$ from regular bit-commitments (RSC-optimized). Legend of Fig. 16 applies.

Then, it uses the received help-bits and respective seeds – the ones in the remaining positions of the vector of seeds (280) – to recompute the corresponding help-bit commitments (281). In possession of bit-commitments corresponding to the two components (i.e., supposedly copies) of each target-bit (282) and help-bit (283), $\mathrm{P_A}$ computes the hash of their concatenation (in the appropriate order) and compares it against the global hash (284) that was received in the commit phase. If the values are not equal, then $\mathrm{P_A}$ rejects the opening phase; otherwise it continues.

– **2.c. Verify checksums.** Finally, $\mathrm{P_A}$ computes the expected checksums as the XOR between the tentative help-bits and the respective differential bits (285), and compares the result to what is obtained by applying the encoding procedure to the respective target bits (286). If all previous comparisons any of the previous comparisons yields inconsistent values, then $\mathrm{P_A}$ rejects the opening. Otherwise it accepts the target vector received from $\mathrm{P_B}$.

E Analysis of UC commitment scheme.

This section analyzes in more detail some aspects of the new UC commitment scheme (X&Q and one-pass simulatable), based on a PRG, a CR-Hash and two underlying schemes for X-commitments and Q-commitments. The scheme is proven to securely

realize the ideal *multiple commitment scheme functionality* in the *universal composability* (UC) framework [Can01], in the hybrid model where the sub-protocols for X-commitments and Q-commitments are replaced by respective ideal commitment functionalities. §E.1 gives the proof of security, by showing suitable simulators, assuming a concrete instantiation of authenticators. §E.2 describes the security and requirements of the authenticator mechanism. §E.3 considers concrete instantiations of authenticators.

E.1 Simulation of protocol #2

Simulation of interaction between environment $\mathcal{Z}$ and real adversary $\mathcal{A}$. The protocol in the ideal world starts when $\mathcal{Z}$ activates the parties to begin the protocol. All messages that $\mathcal{S}$ receives from $\mathcal{Z}$ are relayed to the real adversary $\mathcal{A}$ (a black-box), as if they were coming directly from $\mathcal{Z}$. Correspondingly, every message that $\mathcal{A}$ sends to $\mathcal{Z}$ are intercepted by $\mathcal{S}$ and then relayed to $\mathcal{Z}$ as if originating from $\mathcal{S}$.

E.1.1 Simulation for corrupted $\mathrm{P}_{\mathrm{A}}^{*}$.

The extractor-simulator $\mathcal{S}^{X}$ creates a simulation, accessing $\mathcal{A}$ as a black-box; letting it believe that it is in the real world controlling $\mathrm{P}_{\mathrm{A}}^{*}$.

- **Commit phase.** Once the protocol starts, $\mathcal{S}^{X}$ (in the role of honest P_{B} in the simulated execution) extracts the seeds committed by $\mathrm{P}_{\mathrm{A}}^{*}$ (30) and later receives from $\mathrm{P}_{\mathrm{A}}^{*}$ the maskings of authenticated fragments of the message being committed (42). $\mathcal{S}^{X}$ 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*. (he escurity of authenticators are analyzed in the next subsections.) Then, $\mathcal{S}^{X}$ uses the IDS recovery algorithm to reconstruct the message from a number of *good* fragments equal to the recovery threshold. Interestingly, the IDS *recovery* algorithm only needs to be used by $\mathcal{S}^{X}$, but it is not needed in a real execution. For appropriate cut-and-choose parameters, if $\mathrm{P}_{\mathrm{A}}^{*}$ has produced instances such that it will be able to successfully open a message in the *open* phase, then there is an overwhelming probability (in the number of instances) that the number of good evaluation instances is at least as high as the recovery threshold. In other words, there is an overwhelming probability that $\mathcal{S}^{X}$ is able to extract the correct message, i.e., the only one that $\mathrm{P}_{\mathrm{A}}^{*}$ is later able to open. Finally, in the ideal world, $\mathcal{S}^{X}$ (in the role of the ideal $\widehat{\mathrm{P}}_{\mathrm{A}}^{*}$) sends this message to the ideal functionality $\mathcal{F}_{\mathrm{MCOM}}$, thus committing to it.
- **Open phase.** Once $\mathrm{P}_{\mathrm{A}}^{*}$ opens the message to 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 (53), 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.

E.1.2 Analysis of the simulation with corrupted $\mathrm{P}_{\mathrm{A}}^{*}$ – statistical security.

For the above-defined simulation, the environment only distinguishes real from simulated executions if $\mathcal{S}^{X}$ extracts a message different from the one that $\mathrm{P}_{\mathrm{A}}^{*}$ can successfully open in the *open* phase. This can only occur with negligible probability – the analysis of probabilities depends not only on the number v of check and e of evaluation instances, but also on the *recovery threshold* t of the IDS, which may be agreed by the parties as any number between 1 and the number e of *evaluation* instances, with corresponding variations in statistical security and communication: a lower t correspond to a higher statistical security; a lower proportion e/t correspond to better (i.e., lower) communication complexity. Asymptotically as the message length increases, the parameters can be configured to yield arbitrary high levels of statistical security and at the same time reduce the expansion-rate to values arbitrarily close to 1. Exact probabilities and parameterizations are described in more detail in Appendix F (Table 3 and Table 5). As an example, with $n = 119$ instances, $e = 46$ of which for evaluation, and with an IDS threshold $t = 23$, the scheme achieves 40 bits of statistical security and an asymptotic communication expansion-rate 2 in the *commit* phase, while the open phase has rate 1 (ignoring the opening of several short commitments). Since $\mathcal{S}$ is able to equivocate the final global hash, it does not need to be able to control the cut-and-choose partition; i.e., it is not necessary to use a simulatable coin-flipping protocol to decide the partition). $\mathcal{S}$ simply produces all commitments of seeds and all maskings correctly (for a random value), so that later all check instances are necessarily verified correctly.

E.1.3 Simulation for corrupted $\mathrm{P}_{\mathrm{B}}^{*}$.

The equivocator-simulator $\mathcal{S}^{Q}$ creates a simulation, accessing $\mathcal{A}$ as a black-box; letting it believe that it is in the real world controlling $\mathrm{P}_{\mathrm{B}}^{*}$.

- **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 P_{A}. Then, in the role of P_{A} in the simulated execution, $\mathcal{S}^{Q}$ commits a random message to $\mathrm{P}_{\mathrm{B}}^{*}$, which basically involves keeping state about the respective maskings of authenticated fragments (42), about the Q-commitment to the global hash of masks (33) and the commitments to seeds (30). If $\mathrm{P}_{\mathrm{B}}^{*}$ aborts at any point before the end of the *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.
- **Open phase.** $\mathcal{S}^{Q}$ waits in the ideal world to receive the *opening* of the target message from $\mathcal{F}_{\mathrm{MCOM}}$. Then, $\mathcal{S}^{Q}$ in the role of P_{A} in the simulated execution sends to $\mathrm{P}_{\mathrm{B}}^{*}$ the target message (44), instead of the previously committed random message. 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 $\mathcal{F}_{\mathrm{MCOM}}$. This is done in the exact same way that P_{B} does as receiver: $\mathcal{S}^{Q}$ computes the message fragments (47), then their authenticators (48), and then takes the XOR with the maskings t_j (49) that were transmitted in the commit phase. Finally, $\mathcal{S}^{Q}$ computes the respective global hash (as in (32), but now using the updated masks), and then uses its equivocation power to equivocate said hash (52). This allows $\mathrm{P}_{\mathrm{B}}^{*}$ to perform all verifications as

if $\mathcal{S}^Q$ was in fact a honest P_A. Finally, $\mathcal{S}^Q$ outputs in the ideal world whatever P_B^* outputs in the real world (54).

E.1.4 Analysis of the simulation with corrupted P_B^*.

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, the ability of P_B^* to detect equivocation would require differentiating the random masks from seed-expansions, which is contrary to the pseudo-randomness assumption of the PRG primitive. Thus, it follows that in case of corrupted P_B^* the distributions between real and ideal world are computationally indistinguishable.

E.2 Security of authenticators

This subsection analyzes the authenticator mechanisms referred in protocol #2, where P_A appends an authenticator to each fragment (40), and then masks them both (41), so that the simulator in the role of P_B can verify the correctness of extracted message fragments. The problem characterization considers two types of unmasking that may occur:

- ***Seed-based unmasking*** denotes the result of unmasking a masked-version using as mask the PRG-expansion of the seed committed by P_A. In the *commit* phase, this is temporarily a tentative message that $\mathcal{S}^X$ considers as possibly having been committed by P_A. If the instance is *good*, then the *seed-based unmasking* is the only message that P_A is later able to *open*.
- ***Hash-based unmasking*** denotes the result of unmasking a masked-version using as mask the respective pre-image portion (known by P_A^*) of the global hash committed by P_A^* (33). In particular, this mask corresponds to the value obtained by P_B in the *open* phase (49), calculated as the XOR of tentative authenticated fragments (47)-prot:CTP2c2:PB:eval:authenticator and the respective previously received maskings (41). When clear in the context, the definition can also refer to the pre-image (known by $\mathcal{S}^Q$ in the role of P_A in a simulated execution) of the hash opened with *equivocation* by $\mathcal{S}^Q$.

A collision-resistance flavor. The authenticator must have some collision-resistant type of properties (analyzed ahead), in order to guarantee that if the seed-based unmasking is an authenticated message, and if the hashed (forged-)mask is not the seed-expansion, then the *hash*-based unmasking is not well authenticated. In other words, the role of authenticators is to make infeasible for a malicious P_A^* to produce a masking instance whose two respective unmaskings (*seed-based* and *hash-based*) yield different validly-authenticated messages, thus preventing $\mathcal{S}^X$ from anticipating an incorrect message.

A nonce-based approach. If the authenticator were to depend only on the message value, then necessarily a malicious P_A^* could break the authenticator: it would calculate two authenticated messages, then would obtain the seed-based masking of one of them and then would compute the necessary forged mask corresponding to the other authenticated message. This problem can be avoided by an authenticator mechanism that

requires P_A^* to commit to the seeds and (via a commitment of the global hash) the masks before being able to predict what are the authenticators of two different messages. For example, the authenticator can be made dependent on an unpredictable value (a *nonce*) randomly selected by P_B only after P_A is bound to her choices. Following this logic, the authenticator is thus defined as a function (α) that relates the message m and a *nonce* z in a non-trivial way. Two nuances of authenticator-mechanism are defined:

- **Loose mode.** The authenticator relates the message and a nonce, with the nonce being disclosed to P_A^* after the seeds and global hash are committed, but before P_A^* decides which overall message to commit. (This nuance requires stronger intractability assumptions.)
- **Strict mode.** Same as before, except for the nonce being disclosed to P_A^* only after P_A^* is committed to the target message being (e.g., by an equivocable commitment to a hash of the message).

The analysis ahead builds the intuition to understand the two authenticator nuances. The respective concrete assumptions needed to ensure security are discussed below, and concrete instantiations are proposed and analyzed in §E.3.

Requirements. A path for a solution becomes more clear upon comparing goals, requirements and capabilities between P_A^* and $\mathcal{S}^Q$.

- **The case of a malicious P_A^*.** The goal of a malicious P_A^* is to lead $\mathcal{S}^X$ into anticipating an incorrect message. Conversely, the goal of $\mathcal{S}^X$ (in the role of P_B in a simulated execution) is to be able to detect, when looking in isolation to a single *evaluation* instance, if a malicious P_A^* might have used an arbitrary *bad* mask, different from the PRG-expansion of the extracted seed. If P_A^* has indeed used a mask different from the committed seed, then the *seed*-based unmasking and the *hash*-based unmasking will be different. The envisioned *authenticator* mechanism (an element to append to the message being masked) will imply that at most one of the unmaskings is well authenticated, which means at least one of the unmaskings fails to be authenticated.
- **The case of the equivocator-simulator.** $\mathcal{S}^Q$ in the role of P_A in a simulated execution must be able to equivocate the *opening* of any message to P_B^*. The mask of each instance is already equivocable, because it is a portion of the known pre-image of a hash committed in an equivocable way. Thus, the equivocation of each authenticator is guaranteed if in the *commit* phase the authenticator is also masked. This only requires that each mask is generated, as a PRG-expansion of the respective seed, with an *extended length* that fits the (possibly very large) fragment concatenated with the (short) authenticator. Thus, the mask itself is concealing from P_B until P_A reveals her message.

In summary, in the devised authenticator mechanism there is a sequence of protections: each *authenticator* binds P_A^* to a message fragment; the mask hides the (extractable) fragment and the (extractable) authenticator; the committed hash of the mask binds P_A^* to the extended mask.

In respect to the proof of simulatability of protocol #2, it is only needed to show that the *authenticator* mechanism delivers the promised properties: (i) it prevents P_A^*

from constructing a masking whose seed-based and hash-based unmaskings are different validly authenticated messages, thus allowing $\mathcal{S}^X$ to detect which instances are good; (ii) continues allowing equivocability of the committed message. Property (ii) is trivial, as the authenticator is simply masked by an equivocable mask value. The remainder of this subsection explores what is required to satisfy property (i).

Attack if nonce is known before commitments. It is instructive to notice that P^*_A can successfully forge a mask if knowing the *nonce* z before committing to the seeds and the global hash. In particular, for any pair of fragments m'_1 and m'_2 (with the same length), P^*_A can construct a masked string such that the seed-based unmasking is equal to the first authenticated fragment and the hash-based unmasking is equal to the second authenticated fragment. This can be achieved with the following procedure: First, P^*_A selects a random seed (287) and expands it into an *extended mask* (31), with size equal to the length of the fragment plus the length of the authenticator (288). Then, if in possession of the *nonce* z, P^*_A calculates the authenticator of each fragment (289)-(290). Then, the first (and typically large) part of the forged mask is equal to the XOR of the two fragments and the initial part of the seed-expansion (291). The second part of the forged mask is equal to the XOR of the two authenticators and the extended part of the seed-expansion (292). The complete forged mask whose hash needs to be committed by P^*_A is equal to the concatenation of the first and second parts (293). Finally, the masking (i.e., masked string) that P^*_A sends to P_B is the XOR between the forged mask and the second authenticated fragment (294), which, as intended, is equal to the XOR between the seed-expansion and the first authenticated fragment. Consequently, the commitment of this seed together with the commitment to the global hash (which also incorporates the forged mask in its known pre-image) constitutes a *bad* instance that $\mathcal{S}^X$ would in isolation anticipate as being *good*.

$$\mathrm{P}^*_\mathrm{A} : s \leftarrow^{\$} \{0,1\}^{\kappa_{\mathrm{PRG}}} \quad (287)$$

$$\mathrm{P}^*_\mathrm{A} : s' = s'^{,1} || s'^{,2} \quad (288)$$

$$\mathrm{P}^*_\mathrm{A} : a_1 = \alpha(m'_1, z) \quad (289)$$

$$\mathrm{P}^*_\mathrm{A} : a_2 = \alpha(m'_2, z) \quad (290)$$

$$\mathrm{P}^*_\mathrm{A} : s^{*,\prime,1} = m'_1 \oplus m'_2 \oplus s'^{,1} \quad (291)$$

$$\mathrm{P}^*_\mathrm{A} : s^{*,\prime,2} = a_1 \oplus a_2 \oplus s'^{,2} \quad (292)$$

$$\mathrm{P}^*_\mathrm{A} : s^{*,\prime} = s^{*,\prime,1} || s^{*,\prime,2} \quad (293)$$

$$\mathrm{P}^*_\mathrm{A} : t = s^{*,\prime} \oplus (m'_2 || a_2) = s' \oplus (m'_1 || a_1) \quad (294)$$

Security of* loose *authenticator. If the *nonce* is unpredictable before P^*_A commits to the seeds and hashes, then the above attack cannot take place. Nonetheless, a malicious P^*_A may still produce a *bad* instance: by committing to a *seed* (extractable by the simulator) that determines the value of a seed-expansion, and to a *hash* of a string (denoted *forged mask*) different from the seed expansion. Then, P^*_A receives the nonce and is instructed by the protocol to send a masking (i.e., masked version) of an authenticated fragment (294). From this masking, $\mathcal{S}^X$ can extract a respective tentative fragment and verify whether or not it is well authenticated. The pertinent question is: *after committing to a seed and to a mask, can a computationally bounded malicious P^*_A find a masking for which both the* seed-based unmasking *and the* hash-based unmasking *yield two different correctly authenticated fragments?*

Since the forged mask can be arbitrarily chosen by P^*_A before knowing the nonce, the problem boils down to P^*_A finding two different fragments (with the same length) that satisfy (see (294)) the following two conditions: (i) the XOR of the two fragments

is equal to the XOR combination of the two respective portions of the masks, i.e., a chosen (non-all-zeros) string c_1 – because the value of the forged mask can be chosen arbitrarily (before the nonce is revealed) (295); and (ii) the XOR combination of the respective *authenticators* is equal to the XOR combination of the two respective mask components, i.e., a chosen string c_2 – because the value of the forged mask can be be chosen arbitrarily (before the nonce is revealed) (296).

$$m'_1 \oplus m'_2 = c_1 \equiv s'^{,1} \oplus s^{*\prime,1} \tag{295}$$

$$\alpha(m'_1, z) \oplus \alpha(m'_2, z) = c_2 \equiv s'^{,2} \oplus s^{*\prime,2} \tag{296}$$

Equivalently, the two conditions can be merged into a single equation (297).

$$\alpha(m_1, z) \oplus \alpha(m'_1 \oplus c_1, z) = c_2 \tag{297}$$

In conclusion, *the authenticator scheme is secure if for whatever strategy of P^*_A in choosing two strings (a non-all-zeros* fragment-XOR-offset c_1*, and an* authenticator-offset c_2*) and becoming bound to them, there is a negligible probability (in the number of bits of entropy of the nonce) that, after given a random nonce z, P_A can find a fragment m'_1 for which the respective authenticator XOR'ed with the authenticator of the XOR-offseted fragment is equal to the authenticator-offset.*

Since it has already been shown that P^*_A can win this game if guessing the random nonce (see (287)-(294)), the advantage for successful cheating becomes minimal if it is also infeasible to find a solution that is simultaneously valid for any pair of distinct nonces. In particular, this requires the infeasibility of finding a tuple (m'_1, c_1, c_2, z, z') satisfying the two equations below (298)-(299). Satisfying each equation individually requires finding a triplet composed of a fragment and two distinct nonces such that the two respective authenticators of the fragment are equal.

$$\alpha(m'_1, z) \oplus s'^{,2} = \alpha(m'_1, z') \oplus s'^{,2} \text{ (i.e., } \alpha(m'_1, z) = \alpha(m'_1, z')) \tag{298}$$

$$\alpha(m'_2, z) \oplus s^{*\prime,2} = \alpha(m'_2, z') \oplus s^{*\prime,2} \text{ (i.e., } \alpha(m'_2, z) = \alpha(m'_2, z')) \tag{299}$$

One way to prevent the satisfiability of any of the above equations is to require that the authenticator function is collision-resistant in respect to the second input parameter.[39] Actually, in isolation it would be enough that the authenticator would simply output the nonce, but such solution would not be compatible with the previous assumption – intractability of solving equation (297).

Definition 15 (*alpha-loose* game). *For an implicit function $\alpha : \{0,1\}^* \times Z \to \{0,1\}^{\ell_a}$ (where Z is the nonce space and ℓ_a is the length of an authenticator) and a computational*

[39] Note: while it is assumed that the hash function is pre-image resistant (a.k.a. one-way) because it is compressive and collision-resistant, the authenticator is not necessarily compressive in respect to the second input parameter. Thus, collision resistance of the authenticator does not necessarily imply pre-image resistance. In fact, ahead a secure authenticator solution in the strict mode is defined where the function is invertible with respect to the second parameter.

security parameter 1^κ*, the* alpha-loose *game is defined as follows: (i)* P_A^* *selects and commits to constants* $c_1 \in \{0,1\}^\ell$ *and* $c_2 \in \{0,1\}^{|\ell_a|}$*, with* $c_1 \neq 0^\ell$*; (ii) an oracle sends a uniformly random selected nonce* $z \in Z$ *to* P_A^**; (iii)* P_A *wins if finding a fragment* m'_1 *satisfying* $\alpha(m'_1, z) \oplus \alpha(m'_1 + c_1, z) = c_2$.

Definition 16 (*alpha-loose* assumption). *Let the set of possible nonces* Z *be exponentially large in the computational security parameter. An authenticator function* α*, receiving as first parameter a fragment and as second parameter a nonce, is said to satisfy the* alpha-loose *assumption if: (i) it is collision-resistant with respect to the second parameter (i.e., once fixed a fragment* m' *it is infeasible to find different nonces* z, z' *that lead to equal message authenticators); and (ii) the probability of any computational* P_A^* *winning the* alpha *game is negligibly small in the computational security parameter.*[40]

Security of strict authenticator. If the above assumption is too strong, then it can be weakened by considering the strict mode of authentication. In this case, the condition for successful cheating becomes stricter, because P_A^* has to choose the message (and thus all its fragments) before knowing the nonce. In particular, in the strict mode described in Fig. 3, P_A^* uses the equivocable commitment scheme, before receiving the nonce, to also commit to the hash of the message ($m_2 = m_1 \oplus c_1$) that it wants to be able to open later (35). This means that a malicious P_A^* only learns the nonce z after having fixed the triplet (m'_1, c_1, c_2) (unequivocally defining m'_2, and based on a chosen seed-expansion s and forged mask $s^{*\prime}$). The final masking can be computed only after the nonce is known, but is necessarily done in a deterministic way because there are no more free variables. Thus, successful cheating happens probabilistically only if the chosen triplet satisfies the stated condition (297) depending on a random variable (the nonce).

Definition 17 (*alpha-strict* game). *Same as* alpha *game but* P_A^* *decides the message* m_1 *in step (i).*

Definition 18 (*alpha-strict* assumption). *Same as the* alpha-loose assumption *but based instead on the alpha-strict game.*

The *alpha* assumptions only make sense if Z is of exponential size in the computational security parameter (e.g., $\kappa = 1^{128}$). The assumption below allows a more tight consideration of probabilities, which makes sense even if considering small nonce spaces.

Definition 19 (*alpha*$^+$*-strict* assumption). *An authenticator function* α *is said to satisfy the* alpha$^+$-loose *assumption if: (i) it is collision-resistant with respect to the second argument, i.e., if once fixed the first argument (the fragment* m'*) it is infeasible to find different values for the second argument (i.e., nonces* z, z'*) that lead to equal authenticators; and (ii) the probability of any computational* P_A^* *winning the* alpha-strict *game is negligibly close (in the computational security parameter) to the inverse of the cardinality of the nonce space* Z.

[40] In more rigor, the requirement is that there is a family of α functions corresponding to progressively increasing security parameters, such that the intractabilities hold for any sufficiently large computational security parameter. For simplicity of description this is ignored but remains implicit in all definitions and intractability assumptions.

§E.3 shows an appropriate authenticator function satisfying the *alpha*$^+$-strict assumption, based only on a collision resistant hash function.

E.3 Concrete authenticator instantiations

Strict method – provable security based only on collision resistance. In the *strict* method, a provably secure authenticator function, in the *alpha*$^+$ sense, can be derived based on collision resistance of a (compressive) hash function. The idea is to have the unpredictability of the difference of two authenticators be as unpredictable as the nonce value. This can be achieved by letting the nonce operate as a (non-null) multiplier in an algebraic field, whose multiplicand is the output of a collision-resistant function of the fragment. The image space of the hash function (the set of bit-strings of some fixed length) can be considered as the base set of a Galois field with characteristic 2 (300), modulo some irreducible polynomial of degree equal to the hash length (301). This means that the set Z of possible nonces can be any (large enough) subset of the set of all non-all-zeros bit-strings of hash length (302). The security intuition is that each different nonce leads the same pair of messages with different hash to a different difference (i.e., the result upon subtraction, which in the current field is the XOR operation) (303). In this way, as the nonce varies across all non-null values of the field, any pair of messages with different hash leads the respective authenticators to have a difference that also varies across all non-null elements.

$$\alpha(m', z) = z \times_{GF(2^{|\text{CR-Hash}|})/p(x)} \text{CR-Hash}(m') \tag{300}$$

$$\text{For example, if |CR-Hash|=256: } p(x) = 1 + x^{246} + x^{251} + x^{254} + x^{256} \tag{301}$$

$$Z \subseteq \left(\{0,1\}^{|\text{CR-Hash}|} \setminus 0^{|\text{CR-Hash}|}\right) \tag{302}$$

$$(\forall m_1', m_2' : \text{CR-Hash}(m_1') \neq \text{CR-Hash}(m_2'))\,(\#\left(\{\alpha(m_1', z) \oplus \alpha(m_2', z) : z \in Z\}\right) = \#(Z)) \tag{303}$$

In conclusion, the *alpha*$^+$ assumption holds. If the nonce z is a random variable uniformly selected from a set Z, then the probability of equation (297) being satisfied, i.e., of P_A^* being able to guess the difference c_2 between authenticators of two different messages (with different hash), is equal to the inverse of the cardinality $\#(Z)$ of the nonce space. For simplicity of description the protocol descriptions consider nonces independent from the cut-and-choose partition, but in practice the cut-and-choose partition already provides a source of entropy.

Strict method – a more practical authenticator, based on a stronger assumption. If from an implementation perspective it is preferable not to consider field multiplications, then it is possible to devise an authenticator function that uses only hash computations and XORs, but requiring a stronger assumption. A possibility is to have the authenticator be the hash of the XOR of the nonce and the hash of the fragment (304), with the nonce being selected from the set of all bit-strings with length equal to the hash output. In this case, the authenticator corresponds to the hash of a random input value (of the given length). In order for P_A^* to satisfy the necessary condition of a forged instance (297), it

must chose in advance an offset c_2 as a successful guess of the XOR between a hash with random pre-image r and another hash with chosen offseted pre-image (305). If it is possible to achieve this with noticeable probability, then it is also possible with noticeable probability to find four pre-images whose respective hashes cancel out, i.e., their XOR becomes equal to the all-zeros string (306). This is henceforth denoted as *four-pre-image-canceling*. Even though resistance against *four-pre-image-canceling* is a stronger than needed assumption, it is conceptually simpler to state. It also seems a reasonable assumption, from the perspective of current state-of-the-art constructions, such as SHA-256 or SHA-3 (with appropriate length), for which there is currently no concrete counter-example that breaks the assumption.

$$\alpha(m', z) = \text{Hash}(z \oplus \text{Hash}(m')) \tag{304}$$

$$\alpha(r) \oplus \alpha(r + c) = c_2 \tag{305}$$

$$\alpha(x_1) \oplus \alpha(x_2) \oplus \alpha(x_3) \oplus \alpha(x_4) = 0^{|\text{Hash}|} \tag{306}$$

Loose method. It is more difficult to argue about the security in the *loose* case because the attacker P_A^* becomes flexible in choosing the first parameter m'_1 of input after knowing the second parameter z.

- If using an authenticator function based on field multiplication (300), as described above for the *strict* case, then the scheme is secure in the *loose* setting if the hash function is secure in the sense that there is a negligible probability of winning the following game: (i) P_A^* selects a fragment offset c_1; then (ii) an oracle selects a random value r; then (iii) P_A^* wins the game if finding a fragment m'_1 satisfying $\text{Hash}(m'_1) \oplus \text{Hash}(m'_1 \oplus c_1) = r$. Here, the random value r selected by the oracle corresponds to the multiplicative inverse of the *nonce* multiplied by the constant c_2 selected by P_A^* in the *alpha-loose game*, i.e., $r = z^{-1} \times c_2$.
- If intending to use an authenticator based simply on a collision resistant hash, then different assumptions may be considered. A simple example is an authenticator function defined as the hash of the concatenation of the two input parameters (307). Security requires that it is infeasible to win the *alpha-loose game*. If P_A^* has a noticeable probability of winning this game, when the nonce z is sampled uniformly from an exponentially sized set, then there is also a noticeable probability of winning twice for the same constant c_2. This means breaking resistance against *four-pre-image-canceling*. The only reason to mention this assumption is that it is conceptually easy to state, even though it is stronger than needed. (There may be overlooked solutions, more efficient or requiring weaker assumptions.)

$$\alpha(m', z) = \text{Hash}(m' || z) \tag{307}$$

F Concrete comparisons

This section considers variations and concrete parameterizations of protocol #2. §F.1 makes some remarks about possible adjustments to the protocol. §F.2 analyzes different configurations of threshold recovery, and summarizes in Table 3 aspects of communication and computation in the corresponding application to coin-flipping – also comparing side-by-side with protocol #1 (devised for a setting of simulation with rewinding). §F.3 makes a high-level comparison with the scheme from [GIKW14] and then shows optimal configurations of the cut-and-choose and IDS parameters to achieve 40 bits of statistical security, for different goals of communication rate.

F.1 Remarks on protocol variations.

- **Interactiveness vs. non-interactiveness.** Except for the cut-and-choose and the nonce, the *commit* phase is non-interactive if the base commitment schemes are non-interactive. For large enough number of instances in the cut-and-choose, it is already possible to have the nonce be decided as a pseudo-random value determined from the cut-and-choose partition. The overall scheme can be made non-interactive (i.e., each phase consist of a single message sent from P_A to P_B) if the number of instances is big enough to allow both cut-and-choose and nonce to be decided pseudo-randomly using as seed the concatenation of all previous short commitments (with corresponding adjustments to statistical security).
- **Replacing the Q-scheme by a X-scheme (and some interactiveness).** The use of a Q-scheme with P_A as sender and P_B as receiver could be replaced by a X-scheme with P_B as sender and P_A as receiver. The construction would be somewhat resembling to that shown in the protocols in Fig. 16 and Fig. 17 for a Q commitment scheme in the plain model. Basically it would be used in a way that a simulator in the role of P_A in the simulated execution would be able to determine the outcome of a two party coin-flipping played between P_A and P_B.
- **Number of equivocable hashes.** In the described protocol, there is a single committed hash. A possible variant could use one Q-commitment per hash of each PRG-expansion. Then, each check subset could be verified immediately during the overall commit phase of protocol #2. The advantages and disadvantages of each choice might vary with the implementation scenario. By *opening* the *check* hashes in the *commit* phase, P_B would immediately obtain statistical confidence about correctness, preventing P_A from committing to something for which it could provably not be able to open. In other words, P_B would be able to abort immediately in case any *check* instances is bad, thus avoiding storing the message maskings until an (eventual) *open* phase that will necessarily fail. Conversely, this would be at the cost of requiring several Q-commitments, instead of just one.
  The authenticator mechanisms described in the previous subsections allow $\mathcal{S}^X$ (in the role of P_B in a simulated execution) to anticipate whether or not an instance built by P_A^* is *good*, without comparison with other instances, thus guaranteeing extraction of correct message fragments. This was achieved by adding an *authenticator* element to each instance, without requiring additional primitives.

Table 3. Comparison of coin-flipping protocol variants

Simulation type		Rewinding	One-pass
Protocol description		Fig. 1	Fig. 7 (with $\mathscr{C}_{\text{XQ}}$ instantiated with Fig. 3)
Overall PRG output length	P_A	ℓ	$(n/t)\ell$
	P_B		$(v/t)\ell$
Overall Hash input length	P_A	ℓ	$(n/t)\ell$
	P_B		
Communication*	$\text{P}_\text{A} \to \text{P}_\text{B}$	ℓ	$(e/t+1)\ell$
	$\text{P}_\text{B} \to \text{P}_\text{A}$		ℓ
Commitments	$\mathscr{C}_{\text{Q}}^{\text{Commit}}$	1 hash	1 hash
	$\mathscr{C}_{\text{Q}}^{\text{Open}}$		
	$\mathscr{C}_{\text{X}}^{\text{Commit}}$	1 seed	n seeds
	$\mathscr{C}_{\text{X}}^{\text{Open}}$		v seeds
IDS operations		—	Split
Optimal attack	b	—	$e-t+1$
Error probability		—	$\frac{(n-b)!e!}{(e-b)!n!}$

*Note: the "communication" row ignores the communication relative to the commitments of seeds and hash, the cut-and-choose partition and the authenticators.

– **A X&Q commitment scheme based verifiability mechanism.** Another conceptually simple mechanism providing fragment-verifiability can be implemented based on an X&Q commitment scheme for short strings. In the commit phase, P_A^* would also commit to the hash of each fragment; in the open phase, P_A^* would simply open those hashes and P_B would verify their consistency against the respective opened message; this would be in replacement of appending authenticators (a function of the message and a nonce) to the fragments. With this X&Q-based mechanism there is an obvious condition for verification for $\mathcal{S}^X$ of *good* instances: for each evaluation instance, $\mathcal{S}^X$ extracts the CR-Hash of the message fragment (from the X&Q commitment) and directly compares it against the tentative message fragment that is also obtained after extracting the respective seed and using its PRG-expansion to unmask the respective masking. Equivocability is not affected because the hash of all fragments can still be equivocated by $\mathcal{S}^Q$ in the Q-open phase of a X&Q commitment scheme. While this method would defeat the purpose of building a X&Q commitment scheme from different primitives, it shows another type of UC commitment length extension.

F.2 Protocol variants

The UC commitment scheme devised in §5 is suited for direct plugging into the traditional coin-flipping template. Table 3 summarizes several aspects of computation (i.e., PRG, CR-Hash, and commitment schemes) and communication, and takes the chance to

compare show it side-by-side with the respective complexities of protocol #1 (not UC secure, as the simulation requires rewinding).

If the *recovery threshold* of the IDS is 1, the scheme achieves about 1 bit of statistical security per instance (if there are as many *evaluation* as *check* instances) and requires communication equal to the target length multiplied by the number of evaluation instances added by two. In such case: there is no cost associated with the IDS; each party has to compute one hash per instance; P_A has to compute one PRG per instance; and P_B has to compute one PRG per *check* instance (each hash and each PRG are associated with the target length). As an example, 40 bits of statistical security can be achieved using 44 instances, fixing in advance that 22 are for evaluation. This requires communication of 24 times the target length (22 to commit the contribution of P_A, 1 to reveal the contribution of P_B, 1 to open the contribution of P_A), computation of 44 or 22 PRG expansions, respectively by P_A or P_B, and computation of 44 hashes by each party. A lower number of instances can be achieved by letting the selection (into *check* or *evaluation* type) of each instance be (more) independent of the other instances. For example, 40 bits of statistical security are also achieved using 41 instances and letting the number of *evaluation* instances be variable but limited to 20. Since a variable partition size leads to a variable communication complexity, for simplicity the analysis hereafter will consider only fixed partition-sizes.

If the *recovery threshold* is equal to half the number of evaluation instances, then about 1 bit of statistical security is achieved for every three instances, and the overall communication is improved to require just two times the target length to commit to the contribution of P_A. For example, 40 bits of statistical security can be achieved using 119 instances, with 46 being for evaluation and with an IDS threshold of 23. This requires communication of about four bits per coin (two for the masking of the contribution of P_A, one for the contribution of P_A and one for the contribution of P_B). In terms of computation, the overall PRG output and Hash input for P_A is about 5.17 times the *target length*. For P_B, the hash input is the same, but the PRG output is only concerned with the *check* instances, thus being only about 3.17 times the *target length*. Each party also needs to additionally compute an IDS splitting. This example highlights the tradeoff existing between the computation associated with the IDS and the communication reduction obtained by a dividing factor equal to the IDS threshold, but the concrete computational costs should be better explored in future work. It is interesting to notice that the number of instances in this example is slightly less than the number (123) required in the unoptimized protocol. The reason is that in this example it is enough that half of the *evaluation* instances are *good*, whereas in the unoptimized protocol it was required that less than half of the *evaluation* instances were *bad*.

As an example of an intermediate parametrization, a recovery-threshold equal to one third of the evaluation instances leads the commitment of the contribution of P_A requires communication of three times the length the contributions. For example, a statistical security of 40 bits is achieved using 82 instances, of which 33 are selected for evaluation, and using a threshold equal to 11. In this case the overall communication of coin-flipping becomes about five bits per flipped coin (3 for the commitment the contribution of P_A, 1 for the opening, 1 for the direct sending of the contribution of P_B).

If reducing communication is of utmost importance, then a higher *recovery threshold* can be used in proportion to the number of *evaluation* instances. For example, an expansion factor of 1.5 for commitments (leading to 3.5 bits per flipped coin) can be achieved with 193 instances, 72 of which for evaluation and using a *recovery threshold* equal to 48. As a more extreme example, an expansion factor of 1.1 for commitments (leading to about 3.1 bits per flipped coin) can be achieved with 775 instances, 275 of which for evaluation and using a *recovery threshold* equal to 250. As the number of instances increases, so increases the concrete communication and computational complexity of the base commitment schemes, becoming more relevant if the *target length* remains fixed and not reasonably large.

F.3 Concrete parameters for the UC commitment scheme

For purpose of comparison, the text below describes in high level a recent UC commitment scheme [GIKW14] that can also be parametrized to achieve asymptotic communication rate 1, but with an essential difference of being based on oblivious transfer. Table 4 compares some aspects in high level. Finally, Table 5 enumerates concrete parameterizations for different asymptotic communication rates.

Informal description of the rate-1 UC commitment scheme from [GIKW14]

1. In a *preparation* phase, P_A encodes her message into a vector of *blocks*, using a *multi-secret sharing* encoding scheme [FY92, §3.1]. The encoding scheme, characterized by a *polynomial degree*, a *0-info threshold* and an *error-recovery threshold*, has the following properties: (i) any subset with a number of blocks equal to the *0-info threshold* does not reveal anything about the original message, and from any such subset it is still possible to choose all remaining blocks in a way that the decoding yields any intended message (without need to perform error correction); (ii) a complete vector of blocks has length equal to the polynomial degree plus twice the *error-recovery threshold*, and can be decoded into the original message even if there are erroneous blocks in any number up to the *error-recovery threshold*; and (iii) any two error-free vectors of blocks that decode into different messages are different in a number of blocks that is at least twice the *error-recovery threshold*.
2. In the *commit* phase, P_A independently sends each of the blocks using a δ-Rabin-OT that is parametrized such that P_B receives the block with a probability equal to the inverse of half of the *0-info threshold*, and otherwise does not receive anything.
3. In the *opening* phase, P_A sends all the blocks in clear.
4. In an *acceptance* phase, P_B verifies that the blocks received in the *commit* phase are consistent with the respective ones received in the *opening* phase. P_B then decodes all the blocks into a message, verifying that no block is erroneous. P_B accepts the decoded message if and only if all verifications are successful.

The security of the protocol is proven based on arguments of equivocability and extractability, using a hybrid approach (i.e., assuming an ideal δ-Rabin-OT oracle). The parameters of the encoding scheme are chosen such that the protocol achieves a desired statistical security, namely such that: with overwhelming probability, in the *commit* phase,

Table 4. Comparison of commitment-schemes (primitive and operations)

Aspect	**This work**	**[GIKW14]**
Explicit primitives	PRG, collision-resistant hash, X-commitment, Q-commitment	PRG, δ-Rabin-OT*
IDS type	Erasure code	Multi-secret sharing (error-correcting code)
IDS operations by parties	P_A and P_B encode	P_A encodes, P_B decodes
IDS operations by the extractor-simulator	Decode	Decode using error correction
Message sent by P_A in the *opening* phase	The clear message and Q-*opens* a hash and X-opens the seeds (seeds can optionally be opened before)	The encoding of the message

*Note: δ is the probability with which a message sent is allowed passing through the OT oracle. The authors show how to efficiently instantiate it from 1-out-of-N OT and a PRG, and how to obtain 1-out-of-N from 1-out-of-2 OT.

P_B receives blocks in number not higher than the *0-info threshold*, thus allowing an equivocator-simulator to *open* any intended message; with overwhelming probability, in the *open* phase, P_A cannot send a valid (i.e., error free) vector of blocks that is consistent with the blocks previously received by P_B but encoding a message different from the one that could be decoded by the extractor-simulator (possibly with error correction) from the vector sent in the *commit* phase. Table 5 (column "original") shows concrete parameters allowing a statistical security of 40 bits, for several communication rates. The parameters can be slightly improved (i.e., reduced) with simple adaptations, such as optimizing the parameter of the δ-Rabin-OT (column "optimizing δ"), instead of being fixed to be the proportion of twice the *0-info threshold* over the number of blocks used in the multi-secret sharing scheme. Even better, the parameters (i.e., number of blocks) are reduced by replacing the δ-Rabin-OT with an optimized several-out-of-many OT (the rightmost column).

Table 5. UC X&Q commitment scheme – parameters for 40 bits of statistical security

Maximum allowed rate	**This work**		**[GIKW14]** (original)	Adapting **[GIKW14]**	
	$r = e/t \le r_{\max}$	$r' = n/t \le r_{\max}$	$\delta = t_{\text{0-info}}/(2n')$	Optimal δ	$t_{\text{0-info}}$-out-of-n' OT
$r_{\max} \le 2$	$n = 119$ $v = 73$ $e = 46$ $t = 23$ $r' \approx 5.17$ $r = 2$	$n = 324$ $v = 87$ $e = 237$ $t = 162$ $r' = 2$ $r \approx 1.46$	$n = 826$ $n' = 1652$ $t_{\text{0-info}} = 428$ $t_{\text{error}} = \lfloor 399/2 \rfloor$ $\delta = \frac{107}{826} \approx 0.1295$	$n = 577$ $n' = 1154$ $t_{\text{0-info}} = 339$ $t_{\text{error}} = \lfloor 239/2 \rfloor$ $\delta \approx 0.2064$	$n = 352$ $n' = 704$ $t_{\text{0-info}} = 186$ $t_{\text{error}} = \lfloor 167/2 \rfloor$
$r_{\max} \le 3/2$	$n = 193$ $v = 121$ $e = 72$ $t = 48$ $r' \approx 4.02$ $r = 1.5$	$n = 822$ $v = 144$ $e = 678$ $t = 548$ $r' = 1.5$ $r \approx 1.237$	$n = 2540$ $n' = 3810$ $t_{\text{0-info}} = 650$ $t_{\text{error}} = \lfloor 621/2 \rfloor$ $\delta = \frac{65}{762} \approx 0.0853$	$n = 1706$ $n' = 2559$ $t_{\text{0-info}} = 481$ $t_{\text{error}} = \lfloor 373/2 \rfloor$ $\delta \approx 0.1379$	$n = 1152$ $n' = 1728$ $t_{\text{0-info}} = 296$ $t_{\text{error}} = \lfloor 281/2 \rfloor$
$r_{\max} \le 11/10$	$n = 775$ $v = 500$ $e = 275$ $t = 250$ $r' = 3.1$ $r = 1.1$	$n = 12,793$ $v = 598$ $e = 12,195$ $t = 11,630$ $r' = 1.1$ $r \approx 1.0489$	$n = 48,200$ $n' = 53,020$ $t_{\text{0-info}} = 2424$ $t_{\text{error}} = \lfloor 2397/2 \rfloor$ $\delta = \frac{303}{13255} \approx 0.0229$	$n = 28,740$ $n' = 31,614$ $t_{\text{0-info}} = 1498$ $t_{\text{error}} = \lfloor 1377/2 \rfloor$ $\delta \approx 0.03945$	$n = 23,530$ $n' = 25,883$ $t_{\text{0-info}} = 1185$ $t_{\text{error}} = \lfloor 1169/2 \rfloor$
$r_{\max} \le 101/100$	$n = 7310$ $v = 4684$ $e = 2626$ $t = 2600$ $r' = 2.81$ $r = 1.01$	$n = 1,125,645$ $v = 5631$ $e = 1,120,014$ $t = 1,114,500$ $r' = 1.01$ $r \approx 1.00495$	$n = 4,474,600$ $n' = 4,519,346$ $t_{\text{0-info}} = 22,388$ $t_{\text{error}} = \lfloor 22,359/2 \rfloor$ $\delta = \frac{5597}{2,259,673} \approx 0.00248$	$n = 2,384,200$ $n' = 2,408,042$ $t_{\text{0-info}} = 12,166$ $t_{\text{error}} = \lfloor 11,677/2 \rfloor$ $\delta \approx 0.004737$	$n = 2,231,600$ $n' = 2,253,916$ $t_{\text{0-info}} = 11,166$ $t_{\text{error}} = \lfloor 11,151/2 \rfloor$

– **Legend for columns "This work".** r (communication expansion rate); r' (overall output length produced by the PRG, divided by the target length; also the overall length inputted to the hash function, 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 the leftmost column), as follows: in the right column ("$r = e/t \le r_{\max}$"), the communication expansion-rate is limited to the $r_{\max}$ (in this case the PRG and Hash can be applied to bigger lengths – see r'). in the right column ($t = \lceil n/r \rceil$), the computed output of the PRG and the input of the hash computation are limited to a length which in comparison with the target length has an expansion rate of at most $r_{\max}$ (in this case the overall communication rate is smaller) – after minimizing n, the remaining parameters were chosen to minimize e.

– **Legend for columns "[GIKW14].** n (number of blocks before encoding, i.e., number of symbols in which the message is partitioned); $t_{\text{0-info}}$ (0-info threshold – the original notation is t); 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 $t_{\text{0-info}} = 2\delta n'$); n' (total number of blocks, satisfying $n' = t + n + \Delta - 1$). Parameters were chosen to minimize n; the expansion rate is equal to $r = n'/n$. For the case of the rightmost column, where the equivocator-simulator can always equivocate; statistical security depends only on the probability that a malicious $\mathrm{P}_{\mathrm{A}}^{*}$ can guess $t_{\text{error}} + 1$ positions that will not be selected by the OT. In the original notation, the *polynomial degree* is equal to $d = n + t - 1$ (i.e., $n + t_{\text{0-info}} - 1$), or equivalently $d = n' - \Delta$ (i.e., $n' - 2t_{\text{error}} + 1$).

– **Note:** above, the calculation of communication rate does not take into account the setup phase and the base commitments, whose communication becomes amortized with the increase of the length of the bit-string being committed.

G Lists of Figures and Tables

List of Figures

List of Tables