J. Cryptol. (2009) 22: 530–571
DOI: 10.1007/s00145-009-9045-2

Journal of
CRYPTOLOGY

Efficient Non-malleable Commitment Schemes

Marc Fischlin
Darmstadt University of Technology, Darmstadt, Germany
marc.fischlin@gmail.com
url: http://www.fischlin.de/

Roger Fischlin
BearingPoint GmbH, Frankfurt am Main, Germany
roger.fischlin@bearingpoint.com

Communicated by Cynthia Dworh

Received 16 March 2005 and revised 27 August 2008
Online publication 13 May 2009

Abstract. Non-malleability protects against man-in-the middle attacks on cryptographic protocols. Non-malleable commitment schemes, for example, assure that a commitment of a message does not help to produce a commitment of a related message. Here we present efficient constructions of such commitment schemes in the common reference string model, based on standard assumptions such as RSA, factoring or discrete logarithm. Our protocols require only three rounds and a few modular exponentiations, and provide statistical or even perfect secrecy of committed values.

We also discuss differences between the notion of non-malleable commitment schemes used in previous works by Dolev, Dwork and Naor and by Di Crescenzo, Ishai and Ostrovsky. The former definition requires that it is infeasible to find a commitment such that there exists an encapsulated message which is related to another committed value (non-malleability with respect to commitment). The second approach allows the existence of such messages, but then it is hard to find them and to output them in the opening phase (non-malleability with respect to opening). We note that our solutions are of the second type.

Key words. Chinese remainder theorem, Commitment, Common reference string, Non-malleability, Proof of knowledge, Trapdoor commitment

1. Introduction

Loosely speaking, a commitment scheme is non-malleable if one cannot transform the commitment of another person's secret into one of a related secret. Such non-malleable schemes are, for example, important for auctions over the Internet: It is necessary that one cannot generate a valid commitment of a bid $b + 1$ after seeing the commitment of an unknown bid b of another participant. Unfortunately, this property is not achieved by commitment schemes in general because ordinary schemes are only designated to

hide the secret. Even worse, most known commitment schemes are, in fact, provably malleable.

1.1. *Chronology (Part I)*

The concept of non-malleability has been introduced by Dolev et al. [18]. They present a non-malleable public-key encryption scheme (based on any trapdoor permutation) and a non-malleable commitment scheme with logarithmically many rounds based on any one-way function. Yet, their solutions involve cumbersome non-interactive and interactive zero-knowledge proofs, respectively. Further non-malleable encryption schemes with improved efficiency under various assumptions have appeared since then [3,4,13].

As for commitment protocols, Di Crescenzo et al. [16] present a non-interactive and non-malleable commitment scheme based on any one-way function in the common random string model. Though being non-interactive, their system is rather theoretical as it applies an ordinary commitment scheme many times to non-malleably commit to a single bit. Other non-malleable commitment protocols have been suggested after the proceedings version of our paper [24] had been published; we review these schemes at the end of this introduction.

1.2. *Our Results*

We present efficient perfectly- and statistically-secret non-malleable commitment schemes based on standard assumptions, such as the RSA assumption or the hardness of computing discrete logarithms. Our schemes are designed in the common reference string (CRS) model (aka public parameter or auxiliary string model), where public parameters like a random prime p and generators of some subgroup of $\mathbb{Z}_p^*$ are generated and published by a trusted party. This model relies on a slightly stronger assumption than the common *random* string model where the public data consist simply of a random string. Yet, as in the example of discrete logarithms, the CRS model can sometimes be formally reduced to the common random string model if we let the participants map the random string via standard procedures to a prime and appropriate generators.

In our schemes, the sender commits to his message using an ordinary, possibly malleable discrete-log- or RSA-based commitment scheme and performs an efficient three-round witness-independent proof of knowledge, both times using the CRS. While the straightforward solution of a standard proof of knowledge fails (because the adversary may in addition to the commitment also transform the proof of knowledge), with the help of the CRS we force the adversary to give her "own" proof of knowledge. That is, the adversary cannot adapt the proof of the original sender. Similar ideas have also been used in [18] where independency of the proofs comes from a sophisticated message scheduling of the interactive commitment.

We also present a version of our commitment scheme which is based on the factoring assumption. Yet, while discrete-log and RSA commitments support efficient proofs of knowledge, similar proofs are not known to exist for commitments based on factoring. Fortunately, we do not need proofs of knowledge to the full extent, but it suffices that the proof is verifiable by the receiver only after the sender has decommitted and the witness is revealed. We call such proofs *a posteriori verifiable*.

Based on the Chinese Remainder Theorem we then give efficient instantiations of a posteriori verifiable proofs of knowledge. The resulting commitment scheme based on

factoring thus comes with a potentially milder assumption than in the RSA case, yet at the same time it achieves a technically weaker notion called ϵ-non-malleability. As a positive side effect, our a posteriori verifiable proofs now allow to hash longer messages before committing. In contrast to this, the discrete-log- and RSA-based non-malleable schemes relying on well-known proofs of knowledge do not seem to support the hash-and-commit paradigm in general.

1.3. *On the Definition of Non-malleable Commitments*

We also address definitional issues. According to the definition of Di Crescenzo et al. [16], a scheme is non-malleable if the adversary cannot construct a commitment from a given one, such that, after having seen the opening of the original commitment, the adversary is able to correctly open her commitment with a related message. But the definition of Dolev et al. [18] demands more: If there is a one-to-one correspondence between the commitment and the message (say, if the commitment binds unconditionally), then they define that such a scheme is non-malleable if one cannot even generate a commitment of a related message.

We call schemes having the [18] property *non-malleable with respect to commitment*. For these schemes to contradict non-malleability, it suffices to come up with a commitment such that there exists a related opening. Schemes satisfying the definition of [16] are called *non-malleable with respect to decommitment* or, for the sake of distinctiveness, *with respect to opening*. In this case, the adversary must also be able to open the modified commitment correctly given the decommitment of the original commitment. The scheme in [18] achieves the stronger notion, whereas we do not know if the scheme in [16] is also non-malleable with respect to commitment.

A commitment scheme which is non-malleable in the strong sense is non-malleable with respect to opening, too.[1] We stress that the other direction does not hold, in general. That is, given any statistically-secret commitment scheme which is secure with respect to opening, we can devise a commitment scheme satisfying the weak notion, but not the strong definition. Since our statistically-secret schemes based on standard assumptions like RSA or discrete-log achieve non-malleability with respect to opening, this separates both notions under any of these standard assumptions.

We believe that non-malleability with respect to opening is the appropriate notion for perfectly- and statistically-secret schemes. The reason is that for such schemes virtually any commitment can be opened with any message, in principle. Hence, finding a commitment of a related message to a given commitment is easy: Any valid commitment works with very high probability. Although there is at least one application of non-malleable commitment schemes in the context of authenticated key-exchange where non-malleability with respect to commitment is necessary [28], non-malleability with respect to opening still seems to be adequate for most applications. For instance, recall the example of Internet auctions. The commitments of the bids are collected and then, after a deadline has passed, are requested to be opened. Any secret which is not

[1] Although this seems to follow directly from the requirements, it depends on the subtleties of the definitions. Indeed, compared to [18], we strengthen the requirements for non-malleability with respect to commitment in order to imply the notion of non-malleability with respect to opening. The scheme in [18] also satisfies our more stringent definition.

correctly revealed is banned. Therefore, security with respect to opening suffices in this setting.

1.4. *Chronology (Part II)*

Following the publication of the proceedings version of our work, several other non-malleable commitment schemes have been proposed. Di Crescenzo et al. [17] present more practical variants of the system in [16] relying on the RSA or discrete-log assumption and the CRS model; see also [25] for further improvements of these protocols, resulting in more efficient schemes than the ones here. Focusing on the case of multiple commitments Damgård and Groth [14] derive further efficient non-interactive commitment schemes in the CRS model; see also [26] for constructions of such reusable commitments based on multi-trapdoor commitment schemes. All these protocols are non-malleable with respect to opening.

While our protocols consists of three rounds, the aforementioned schemes all provide efficient non-interactive solutions. Yet, this comes at the price of reduced security. First, all these protocols are not known to preserve non-malleability if the adversary is additionally given some useful side information about the message for which it tries to find a related commitment, e.g., if the message is used in other sub protocol executions. Second, the solutions merely achieve the weaker notion of ϵ-non-malleability. In contrast, our DL-based and RSA-based commitment schemes are not subject to these restrictions but, unlike the non-interactive solutions, do not support hashing of longer messages before committing. Our factoring-based solution lies somewhere in between: It allows upstream hashing, but merely achieves ϵ-non-malleability, yet tolerates a priori knowledge of the adversary about the sender's message.

In [17], it is also pointed out that secure public-key encryption is sufficient for non-malleable commitments. Basically, the CRS contains a public key of a secure encryption scheme and in order to commit the sender encrypts the message and hands it to the receiver. Hence, using potentially stronger assumptions like the decisional Diffie–Hellman assumption and the encryption scheme in [13], or non-standard assumptions like the random oracle methodology, one derives alternatives to the solutions here and in [14,17,25,26]. Yet, the encryption-based approach provides only computational secrecy and the latter may be insufficient in some settings, especially since knowledge of the secret key to the public parameters enables to decrypt the message. Also, using random oracles there is a simpler approach to construct non-malleable commitments. We sketch this solution in Appendix A.

More non-malleable (but less efficient) commitment schemes in the broader context of universally composable commitments have been constructed by Canetti and Fischlin [8] and subsequently by Damgård and Nielsen [15] and Canetti et al. [11]. Also, Prabhakaran and Sahai [44] present quite efficient universally composable commitments based on somewhat non-standard assumptions. Conversely, Liskov et al. [35] derive more efficient commitment schemes by aiming at the weaker notion of mutually independence. MacKenzie and Yang [36] discuss the relationship between non-malleable commitments and so-called simulation-sound trapdoor commitments. Finally, let us remark that, recently, Barak [1] gave the first (rather theoretical) constant-round non-malleable commitment scheme in the plain model. Also, Pass and Rosen [42] recently showed how to build concurrently executable non-malleable commitment schemes.

1.5. *Organization*

The paper is organized as follows. In Sect. 2, we introduce basic notations and definitions of commitment schemes as well as the notions of non-malleability. Section 3 separates the notions of non-malleability with respect to commitment and opening. In Sect. 4, we present efficient schemes in the CRS model based on the discrete-log assumption, and, in Sect. 5, we turn to the RSA case. Finally, in Sect. 6, we show how to use a posteriori verifiable proofs of knowledge to achieve non-malleable commitments under the factoring assumption.

2. Preliminaries

Unless stated otherwise, all parties and algorithms are probabilistic polynomial-time. Throughout this paper, we use the notion of uniform algorithms; all results transfer to the non-uniform model of computation. A function $\delta(n)$ is said to be *negligible* if $\delta(n) < 1/p(n)$ for every polynomial $p(n)$ and sufficiently large n's. A function $\delta(n)$ is called *overwhelming* if $1 - \delta(n)$ is negligible. A function is *noticeable* if it is not negligible.

Two sequences $(X_n)_{n\in\mathbb{N}}$ and $(Y_n)_{n\in\mathbb{N}}$ of random variables are called *computationally indistinguishable* if for any probabilistic polynomial-time algorithm $\mathcal{D}$ the advantage

$$\left|\text{Prob}\left[\mathcal{D}\left(1^n, X_n\right) = 1\right] - \text{Prob}\left[\mathcal{D}\left(1^n, Y_n\right) = 1\right]\right|$$

of $\mathcal{D}$ is negligible, where the probabilities are taken over the coin tosses of $\mathcal{D}$ and the random choice of X_n and Y_n, respectively. The sequences are called *statistically close* or *statistically indistinguishable* if

$$\frac{1}{2} \cdot \sum_{s\in S_n} \left|\text{Prob}\left[X_n = s\right] - \text{Prob}\left[Y_n = s\right]\right|$$

is negligible, where S_n is the union of the supports of X_n and Y_n.

2.1. *Commitment Schemes*

We give a rather informal definition of ordinary commitment schemes and focus on the definition of non-malleability instead. For a formalization of regular commitments we refer the reader to [27]. A commitment scheme is a two-phase interactive protocol between two parties, the sender $\mathcal{S}$ holding a message m and a random string r, and the receiver $\mathcal{R}$.

In the first phase, called the commitment phase, $\mathcal{S}$ gives some information derived from m, r to $\mathcal{R}$ such that, on the one hand, $\mathcal{R}$ does not gain any information about m, and, on the other hand, $\mathcal{S}$ cannot later change his mind about m. We call the whole communication in this phase the commitment of $\mathcal{S}$. Of course, both parties should check (if possible) that the values of the other party satisfy structural properties, e.g., that a value belongs to a subgroup of $\mathbb{Z}_p^*$, and should reject immediately if not. In the following, we do not mention such verification steps explicitly. We say that a commitment, i.e., the

communication, is *valid* if the honest receiver does not reject during the commitment phase.

In the decommitment stage, the sender communicates the message m and some evidence showing the correctness of m to the receiver who verifies that these values match the communication of the first phase. If the sender obeys the protocol description, then the commitment is valid and $\mathcal{R}$ always accepts the decommitment. Usually, the sender's random string itself makes up the decommitment evidence, and for simplicity we adhere to this in the rest of the paper. In particular, we assume that the decommitment phase is non-interactive and consists of a single message from the sender to the receiver, revealing the message and the random coins.

There are two fundamental kinds of commitment schemes:

- A scheme is *statistically-binding* (*and computationally-secret*) if any arbitrarily powerful malicious $\mathcal{S}^*$ cannot open a valid commitment ambiguously except with negligible probability (over the coin tosses of $\mathcal{R}$), and two commitments are computationally indistinguishable for every probabilistic polynomial-time (possibly malicious) $\mathcal{R}^*$. If the binding property holds unconditionally and not only with high probability, then we call the scheme unconditionally-binding.
- A scheme is (*computationally-binding and*) *statistically-secret* if it satisfies the "dual" properties, that is, if the distribution of the commitments are statistically close for any arbitrarily powerful $\mathcal{R}^*$, and yet opening a valid commitment ambiguously contradicts the hardness of some cryptographic assumption. If the distribution of the commitments of any messages are identical, then a statistically-secret schemes is called perfectly-secret.

Technically, there are also commitment schemes where binding and secrecy both hold in a computational sense only. However, since one of the properties is usually attainable in an information-theoretic sense we focus on the aforementioned types only.

2.2. *Non-malleability*

As mentioned in the introduction, different notions of non-malleability have been used implicitly in the literature. To highlight the difference, we give a formal definition of non-malleable commitment schemes, following the approach of [18].

Scenario

For non-interactive commitment schemes, all the adversary attacking the non-malleability property can do is modify a given commitment. In the interactive case, though, the adversary might gain advantage from the interaction. We adopt this worst-case scenario and assume that the adversary interacts with the original sender, while at the same time she is trying to commit to a related message to the original receiver.

A pictorial description of a so-called *person-in-the-middle attack* (PIM attack) on an interactive protocol is given in Fig. 1. The adversary $\mathcal{A}$ intercepts the messages of the sender $\mathcal{S}$. Then $\mathcal{A}$ may modify the messages before passing them to the receiver $\mathcal{R}$ and proceeds accordingly with the answers. In particular, $\mathcal{A}$ decides to whom she sends the next message, i.e., to the sender or to the receiver. This is the setting where $\mathcal{A}$ has full control over the parties $\mathcal{R}_1$ and $\mathcal{S}_2$ in two supposedly independent executions

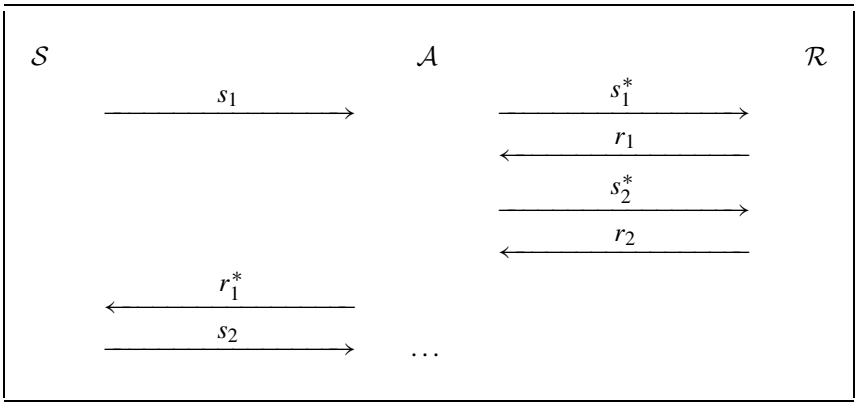

Fig. 1. Person-in-the-middle attack on interactive protocols.

$\langle \mathcal{S}_1, \mathcal{R}_1 \rangle(m)$, $\langle \mathcal{S}_2, \mathcal{R}_2 \rangle(m^*)$ of the same interactive protocol. Here and in the sequel, we usually mark values sent by the adversary with an asterisk.

Apparently, the adversary can always commit to the same message by forwarding the communication. In many applications, this can be prevented by letting the sender append his identity to the committed message. The messages of the sender and the adversary are taken from a space M. Abusing notations, we view M also as an efficiently computable distribution, and write $m \in_R \mathsf{M}$ for a randomly drawn message according to M.

The adversary is deemed to be successful if she commits to a related message, where related messages are identified by the so-called interesting relations: A probabilistic polynomial-time algorithm R taking inputs from $\mathsf{M} \times \mathsf{M}$ and returning a bit is called an *interesting relation* if $\mathsf{R}(m, m) = 0$ with probability 1 for all $m \in \mathsf{M}$ (to exclude copying). Moreover, we let the interesting relation on the second argument accept the undefined symbol $\perp$, capturing the case that the adversary does not produce a valid commitment or decommitment; in this case, we set $m^* = \perp$ and we demand $\mathsf{R}(m, \perp) = 0$ with probability 1.

We assume that M generates the sender's message m and also a value hist_m representing the a priori information the adversary has about m. For instance, hist_m could represent an additional hash value of the sender's message m, or information gathered from other protocol executions where the sender uses m. The value hist_m may not be efficiently samplable, in general. Here, however, we simplify the description and attribute hist_m to the efficient distribution M, admitting an easy way to also include information about the sampling process of m into hist_m. For ease of notation, we then write both $m \in_R \mathsf{M}$ and $(m, \text{hist}_m) \in_R \mathsf{M}$.

Since we work in the CRS model, we extend the input of M and R by adversarial parameters ADVPAR the adversary produces after having learned the parameters CRS (generated before by a trusted third party). The value ADVPAR may, for example, include the public parameters CRS. The motivation for this is that it should be infeasible for the adversary to find a suitable relation or distribution on the messages even if the publicly available parameters are given. For the same reason, we base the relation R

also on the side information hist_m which itself may now depend on ADVPAR through the generation via $\mathsf{M}(\text{ADVPAR})$.[2] In summary, we denote the message space and distribution as $\mathsf{M}(\text{ADVPAR})$ and the relation by $\mathsf{R}(\text{ADVPAR}, \text{hist}_m, \cdot, \cdot)$.

Definition

The definition on non-malleable commitments follows the well-known idea of defining secure encryption, namely, we will demand that, for any adversary $\mathcal{A}$ transforming the sender's commitment successfully, there should be an adversary $\mathcal{A}'$ that finds a commitment to a related message with almost the same probability as $\mathcal{A}$ but without the sender's help. All probabilities below are implicit functions of a security parameter, and the probability space in each case is taken over the randomness of all algorithms.

We describe the attack in detail. First, the public string CRS are generated by a trusted party according to a publicly known, efficiently samplable distribution (if a protocol does not need public information then this step is skipped). On input CRS the adversary $\mathcal{A}$ then picks the adversarial parameters ADVPAR for M and R.

The sender $\mathcal{S}$ is initialized with $m \in_{\mathsf{R}} \mathsf{M}(\text{ADVPAR})$. Now $\mathcal{A}$, given hist_m, mounts a PIM attack with $\mathcal{S}(m)$ and $\mathcal{R}$. Let $\pi_{\mathsf{com}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ denote the probability that, at the end of the commitment phase, the protocol execution between $\mathcal{A}$ and $\mathcal{R}$ constitutes a valid commitment for some message m^* satisfying $\mathsf{R}(\text{ADVPAR}, \text{hist}_m, m, m^*)$. Let $\pi_{\mathsf{open}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ denote the probability that $\mathcal{A}$ is also able to successfully open the commitment after $\mathcal{S}$ has decommitted (where $\mathcal{S}$ does not decommit before the adversary finishes her commitment phase).

In a second experiment, a simulator $\mathcal{A}'$ tries to commit to a related message without the help of the sender. That is, $\mathcal{A}'$ gets as input random parameters CRS, generates adversarial parameters ADVPAR' and then, given hist_m for some $(m, \text{hist}_m) \in_{\mathsf{R}} \mathsf{M}(\text{ADVPAR}')$, it commits to $\mathcal{R}$ without interacting with $\mathcal{S}(m)$. Let $\pi'_{\mathsf{com}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ denote the probability that this is a valid commitment to some related message m' under parameter CRS with respect to relation $\mathsf{R}(\text{ADVPAR}', \text{hist}_m, \cdot, \cdot)$. By $\pi'_{\mathsf{open}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ we denote the probability that $\mathcal{A}'$ additionally reveals a correct decommitment. Equivalently, we may define $\pi'_{\mathsf{open}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ as the probability that $\mathcal{A}'$ simply outputs a related message (without reference to a CRS, commitment and decommitment).

It is now tempting to define non-malleability with respect to commitment and with respect to opening by comparing $\pi_{\mathsf{com}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$, $\pi'_{\mathsf{com}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ as well as $\pi_{\mathsf{open}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$, $\pi'_{\mathsf{open}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ and asking for small differences. In the former case, this would agree with the definition in [18], and in the latter case this would extend it straightforwardly to non-malleability with respect to opening. But, surprisingly at first, for non-malleability with respect to commitment we even oblige the simulator to open its commitment and contrast $\pi_{\mathsf{com}}(\mathcal{A}, \mathsf{M}, \mathsf{R})$ with $\pi'_{\mathsf{open}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$. The are two reasons for this. First, otherwise any statistically-secret commitment protocol would be non-malleable with respect to commitment because if the simulator merely outputs a commitment of some fixed message this is also a commitment of a related message with

[2] We do not include the actual common reference string in the distribution and the relation, as such a definition appears to be unachievable in general (see [23] for a discussion about the case of non-malleable encryption). Testifying to this, our schemes, for example, are not known to satisfy such a stronger notion.

high probability. However, this would certainly contradict the intuition of non-malleable systems.

The other reason is that, even in the case of statistically-binding schemes, we were unable to show that the presumably stronger non-malleability notion à la [18] implies the weaker one. With our approach here this trivially follows from the definition because the requirements for the simulator in both cases are identical, while the adversary trying to refute non-malleability with respect to commitment even faces a simpler task.

For sake of completeness, we include the original definition of Dolev et al. [18] and call this non-malleability with respect to $\text{commitment}_{\text{DDN}}$, whereas we denote the more stringent version by $\text{commitment}_{\text{FF}}$. We remark that the commitment scheme in [18] also satisfies "our" notion of non-malleability with respect to commitment. Unless stated differently, throughout this paper we simply refer to the "FF" version by non-malleability with respect to commitments.

Definition 1. A commitment scheme is called

(a) Non-malleable with respect to $\text{commitment}_{\text{FF}}$ if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for any message space M and any interesting relation R the difference $\pi_{\mathsf{com}}(\mathcal{A}, \mathsf{M}, \mathsf{R}) - \pi'_{\mathsf{open}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ is negligible.[3]

(b) Non-malleable with respect to opening if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for any message space M and any interesting relation R the difference $\pi_{\mathsf{open}}(\mathcal{A}, \mathsf{M}, \mathsf{R}) - \pi'_{\mathsf{open}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ is negligible.

(c) Non-malleable with respect to $\text{commitment}_{\text{DDN}}$ if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for any message space M and any interesting relation R the difference $\pi_{\mathsf{com}}(\mathcal{A}, \mathsf{M}, \mathsf{R}) - \pi'_{\mathsf{com}}(\mathcal{A}', \mathsf{M}, \mathsf{R})$ is negligible.

If M and R are clear from the context we usually abbreviate the success probabilities by $\pi_{\mathsf{com}}(\mathcal{A})$, $\pi'_{\mathsf{com}}(\mathcal{A}')$, $\pi_{\mathsf{open}}(\mathcal{A})$, and $\pi'_{\mathsf{open}}(\mathcal{A}')$, respectively.

Some remarks about the experiment of $\mathcal{A}'$ follow. The simulator $\mathcal{A}'$ does not have the power to choose the string CRS for the commitment to $\mathcal{R}$. This is so because the simulator is obliged to produce a correct commitment to $\mathcal{R}$ under the same honestly chosen public data CRS as the sender and the adversary. This rules out counterintuitive solutions proving obviously transformable commitments non-malleable. For instance, consider (a straightforward non-interactive version of) Naor's bit commitment scheme [39] where the public string consists of the string σ and the sender commits to 0 by transmitting $y = G(r)$ for the pseudorandom generator G, and $y = G(r) \oplus \sigma$ for a 1-commitment. Clearly, this scheme is malleable in an intuitive sense as the adversary can always change y to $y \oplus \sigma$ in order to commit to the flipped bit. If we let the simulator $\mathcal{A}'$ prepare the public string in this case, $\mathcal{A}'$ could set $\sigma = G(r_0) \oplus G(r_1)$ and send $y = G(r_0)$ to the receiver $\mathcal{R}$ and would succeed in committing to a related message without talking to the sender (because y can be opened both as 0 and 1). Hence, formally the scheme would be non-malleable although it allows to transpose commitments.

[3] Here we allow a very liberal definition of negligible functions: The function may also be negative at some value n, in which case it is certainly less than $1/p(n)$ for any strictly positive polynomial $p(\cdot)$. In our case, this means that the simulator does even better than the adversary and thus still reflects our idea of non-malleability.

Still, we allow $\mathcal{A}'$ to pick its own string ADVPAR$'$ in the simulation, not necessarily related to $\mathcal{A}$'s selection ADVPAR. But since the relation R depends on these adversarial parameters ADVPAR and ADVPAR$'$, it is clear that the relation can rule out significantly diverging choices of $\mathcal{A}'$, and hence ADVPAR$'$ is likely to be indistinguishable from ADVPAR.

Slightly relaxing the definition, we admit an *expected* polynomial-time simulator $\mathcal{A}'$. In fact, we are only able to prove our DLog- and RSA-based schemes non-malleable with this deviation. The reason for this is that we apply proofs of knowledge, so in order to make the success probability of $\mathcal{A}'$ negligibly close to the adversary's success probability, we run a knowledge extractor taking expected polynomial-time. Following the terminology in [18], we call such schemes with $\mathcal{A}'$ running in expected polynomial-time *liberal non-malleable* with respect to commitment and opening, respectively.

Alternatively, the authors of [18] also propose a definition of ϵ-non-malleability, which says that for any given ϵ there is a strict polynomial-time simulator (polynomial in the security parameter n and $\epsilon^{-1}(n)$) whose success probability is only ϵ-far from the adversary's probability. Indeed, we will use this definition of ϵ-non-malleability to construct our factoring-based solution with the a posteriori verifiable proofs of knowledge.

Consider a computationally-binding and perfectly-secret commitment scheme. There, every valid commitment is correctly openable with every message (it is, however, infeasible to find different messages that work). Thus, we believe that non-malleability with respect to opening is the interesting property in this case. On the other hand, non-malleability with respect to commitment is also a concern for statistically-binding commitment schemes: with overwhelming probability there do not exist distinct messages that allow to decommit correctly. This holds for *any dishonest* sender and, in particular, for the person-in-the-middle adversary. We can therefore admit this negligible error and still demand non-malleability with respect to commitment.

The Multi-party Setting

Our definition considers the setting of three parties. In the auction case, for instance, usually more parties participate and the adversary's intention may be to overbid only a certain opponent to ensure that this person does not win. Hence, we may let $\mathcal{A}$ talk to several senders $\mathcal{S}_1, \ldots, \mathcal{S}_{\text{poly}}$ with (probably dependent) messages $m_1, \ldots, m_{\text{poly}}$ generated by M(ADVPAR) together with side information hist$(m_1, \ldots, m_{\text{poly}})$. The relation now takes ADVPAR, hist$(m_1, \ldots, m_{\text{poly}})$ and poly$+1$ messages as input and it is required that the (poly$+1$)st message m^* is different from any other message m_i, and that the relation is never satisfied if $m^* = \perp$. We remark that all our protocols remain secure in this multi-sender setting.

A problem occurs if we let the adversary commit in several executions with $\mathcal{R}$ to messages $m_1^*, \ldots, m_{\text{poly}}^*$ and extend the relation accordingly, both in the single- or multi-sender case. Dolev et al. [18] show that this scenario is not reducible to the single case in general and suggest an alternative definition where the adversary is supposed to announce a subset $i_1, \ldots, i_k$ of the executions with the receiver in the commitment phase, inducing a set of messages $m_{i_1}^*, \ldots, m_{i_k}^*$ for which she tries to be successful. In [14], Damgård and Groth also discuss the issue of multiple senders and receivers for non-malleable commitments and present non-interactive schemes for this multi-party

setting. Resembling the "announcement trick" of [18] their result requires that the relation's value does not change if some of the adversary's messages equal $\perp$.

However, the assumptions about the relation's dependencies on dedicated decommitments must be treated with care. In the auction case, for example, the adversary may know beforehand that the sender commits to one out of, say, three values, each bid being equally likely. In order to overbid the sender with the minimal amount, the adversary commits to each of the three values incremented by 1. Later, the adversary only opens the right choice correctly and refuses to decommit to the other two values (which are set to $\perp$). Then the adversary would always overbid the sender easily, but would still not be considered successful according to the formal definition in [14]. Note that this problem is inherent for the commitment scenario and success probabilities depending on the validity of decommitments.

We return to the multi-party case at the end of Sect. 4.3 when discussing this issue for our schemes.

3. On the Relationship of Non-malleability Notions

Clearly, non-malleability with respect to commitment implies non-malleability with respect to opening and with respect to DDN. On the other hand, we show that (under standard cryptographic assumptions) the converse does not hold in the CRS model. To this end, we construct a bit commitment scheme that does not even achieve the DDN notion, but is non-malleable with respect to opening.

To separate the notions we consider once more Naor's bit commitment scheme [39] in the CRS model. Let G be a pseudorandom generator expanding n bits random input to $3n$ bits pseudorandom output. That is, the variables $(X_n)_{n\in\mathbb{N}}$ and $(Y_n)_{n\in\mathbb{N}}$ are computationally indistinguishable, where X_n equals $G(r)$ for a random $r \in \{0,1\}^n$ and Y_n has the uniform distribution on $\{0,1\}^{3n}$.

Let σ be a random $3n$-bit string put into the public parameters. In order to commit to a bit b in Naor's protocol, the sender chooses a random $r \in \{0,1\}^n$ and transmits $y = G(r)$ for $b = 0$ or $y = G(r) \oplus \sigma$ if $b = 1$. The decommitment consists of (b, r). Not only is this scheme computationally secret and statistically binding, it is also *strongly malleable*, i.e., given a commitment y of a bit b one can always derive a commitment of $b \oplus 1$ by sending $y \oplus \sigma$.

Next we construct an assembled commitment scheme (in the CRS model) which consists of a combination of Naor's scheme and an arbitrary statistically-secret system $\mathrm{Com}_{\mathrm{secret}}$ which is non-malleable with respect to opening. To commit to bit b, independently execute the statistically-secret protocol and Naor's scheme on b, either in parallel or sequentially. Opening is done by decommitting for both schemes in parallel.

Obviously, this assembled scheme is computationally-secret and statistically-binding. We show that this scheme only achieves the weaker non-malleability property. The intuition is that the assembled scheme inherits non-malleability with respect to opening from the statistically-secret protocol, and the strong malleability of Naor's scheme (together with the fact that virtually any statistically-secret commitment is, in principle, openable with any value) inhibits non-malleability with respect to commitment.

Theorem 1. *If there is a statistically-secret bit commitment scheme that is non-malleable with respect to opening, then there exists a statistically-binding bit commitment scheme in the CRS model that is non-malleable with respect to opening, but not with respect to commitment$_{FF}$ and not with respect to commitment$_{DDN}$.*

Theorem 1 also holds for liberal non-malleable statistically-secret protocols in the CRS model.

Proof. Since one-way functions exist if commitment schemes exist [32], and one-way functions imply pseudorandom generators [31], Naor's scheme and therefore the assembled system above is realizable given the statistically-secret bit commitment scheme.

We first show that the assembled scheme is *not* non-malleable with respect to commitment$_{\mathrm{DDN}}$ (and therefore not with respect to commitment$_{\mathrm{FF}}$). Define the relation R to consist of the pairs $(b, b \oplus 1)$ and the message space to be the uniform distribution on $\{0, 1\}$, i.e., both M and R are independent of the adversarial parameters. Let hist(b) be empty.

Given access to a sender committing to an unknown random bit $b \in_{\mathrm{R}} \{0, 1\}$ we run a PIM attack and relay all messages between the receiver and the sender for $\mathrm{Com}_{\mathrm{secret}}$. Additionally, we alter Naor's part of the sender's commitment to a commitment of $b^* = b \oplus 1$ by the strong malleability property and forward it to the receiver (Fig. 2).

Since $\mathrm{Com}_{\mathrm{secret}}$ is statistically-secret, with overwhelming probability that part of the sender's commitment can be opened as 0 and 1. Hence, with probability negligibly close to 1 we are able to construct a valid commitment of $b^* = b \oplus 1$ for the assembled scheme and to satisfy the relation R. On the other hand, any simulator not seeing the commitment of the random bit b cannot output a commitment of $b' = b \oplus 1$ with probability exceeding $1/2$ by more than a negligible amount (this negligible amount is due to the binding error of Naor's protocol). Thus, the assembled scheme is *not* non-malleable with respect to commitment$_{\mathrm{DDN}}$.

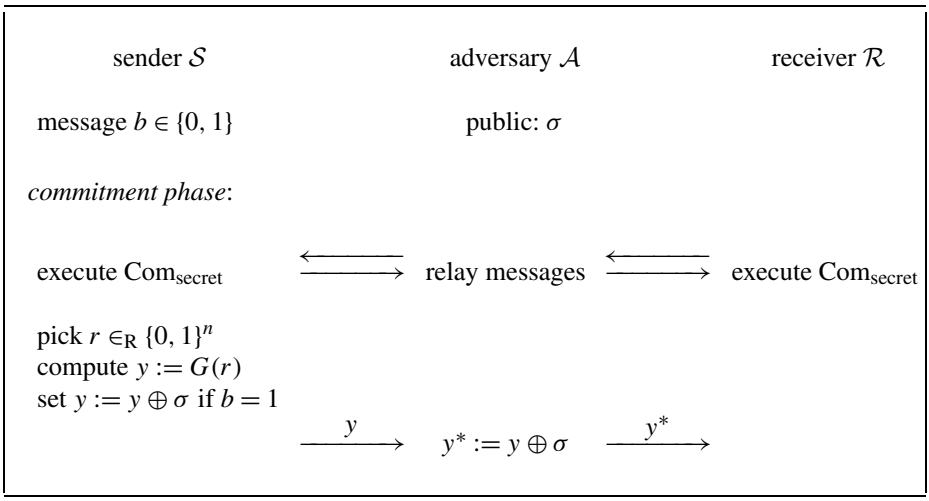

Fig. 2. Malleability with respect to commitment of assembled scheme.

The fact that the combined scheme is non-malleable with respect to opening follows from the non-malleability of the statistically-secret system. Specifically, let $\mathcal{A}$ be an adversary attacking the assembled system. We have to present a simulator that—"out of the blue"—outputs a related message with essentially the same probability $\pi_{\mathsf{open}}(\mathcal{A})$ as $\mathcal{A}$ for all M, R. In an intermediate step, we construct an adversary $\mathcal{A}_{\text{secret}}$ from $\mathcal{A}$ such that $\mathcal{A}_{\text{secret}}$ attacks the non-malleability property of $\text{Com}_{\text{secret}}$.

Define the adversary $\mathcal{A}_{\text{secret}}$ that commits and decommits to a related message for the protocol $\text{Com}_{\text{secret}}$ as follows. $\mathcal{A}_{\text{secret}}$ mounts a PIM attack interacting with the sender $\mathcal{S}_{\text{secret}}$ and receiver $\mathcal{R}_{\text{secret}}$ of $\text{Com}_{\text{secret}}$ on (possibly empty) parameters $\text{CRS}_{\text{secret}}$. $\mathcal{A}_{\text{secret}}$ also runs a virtual copy of $\mathcal{A}$ attacking the assembled scheme. Basically, $\mathcal{A}_{\text{secret}}$ uses $\mathcal{A}$ to generate a related commitment and opening for $\text{Com}_{\text{secret}}$ by adding the steps of Naor's scheme. For this, $\mathcal{A}_{\text{secret}}$ exploits the equivocal version of Naor's scheme presented in [16]. Informally, such an equivocal commitment enables the sender to prepare a dummy commitment which can be later opened with any value, yet this process is indistinguishable from a true execution. This means, instead of letting σ be a random string, we choose σ as $G(r_0) \oplus G(r_1)$ for random $r_0, r_1 \in \{0,1\}^n$. Then, to commit to a dummy value, send $y = G(r_0)$; to open it with 0 reveals r_0 or transmits r_1 for a decommitment to 1.

$\mathcal{A}_{\text{secret}}$ emulates $\mathcal{A}$ by choosing $\sigma = G(r_0) \oplus G(r_1)$ and passing $(\text{CRS}_{\text{secret}}, \sigma)$ to $\mathcal{A}$. Adversary $\mathcal{A}$ returns parameters ADVPAR which $\mathcal{A}_{\text{secret}}$ uses in her attack on $\text{Com}_{\text{secret}}$, too. This defines a distribution $\mathsf{M}(\text{ADVPAR})$ on $\{0,1\}$ as well as a relation $\mathsf{R}(\text{ADVPAR}, \cdot, \cdot, \cdot)$ for both $\mathcal{A}$'s and $\mathcal{A}_{\text{secret}}$'s attack. $\mathcal{A}_{\text{secret}}$ next feeds all messages of $\mathcal{S}_{\text{secret}}$ and $\mathcal{R}_{\text{secret}}$ of the execution of $\text{Com}_{\text{secret}}$ into $\mathcal{A}$ and also forwards all replies of $\mathcal{A}$. Additionally, $\mathcal{A}_{\text{secret}}$ submits a dummy commitment $y = G(r_0)$ on behalf of the sender to $\mathcal{A}$ in the simulation. Later, when $\mathcal{A}_{\text{secret}}$ learns $\mathcal{S}_{\text{secret}}$'s decommitment of bit b it forwards this decommitment to $\mathcal{A}$ and opens the dummy commitment y in $\mathcal{A}$'s simulation accordingly. Output the part of $\mathcal{A}$'s opening for $\text{Com}_{\text{secret}}$ and stop. See Fig. 3.

As for the analysis, first note that $\mathcal{A}_{\text{secret}}$'s success probability producing a valid commitment and decommitment of a related messages is negligibly close to $\pi_{\mathsf{open}}(\mathcal{A})$ for any M, R. This follows from the fact that a fake σ is indistinguishable from a honestly chosen one, i.e., otherwise it would be easy to derive a successful distinguisher contradicting the pseudorandomness of G's output.

More formally, assume that $\mathcal{A}$'s success probability drops noticeably when run on a fake string in the simulation (for some M, R). Then we construct a distinguisher for the pseudorandom generator G as follows. We are given 1^n and $z \in \{0,1\}^{3n}$ and are supposed to tell whether z is truly random or has been derived by running G. Pick random $r \in \{0,1\}^n$ and set $\sigma = G(r) \oplus z$. Next, start $\mathcal{A}$'s attack on the assembled scheme by presenting $(\text{CRS}_{\text{secret}}, \sigma)$. Sample $(b, \text{hist}(b))$ according to the distribution $\mathsf{M}(\text{ADVPAR})$ and continue $\mathcal{A}$'s attack by impersonating the honest parties in the execution of $\text{Com}_{\text{secret}}$. Also, let the simulated sender commit in Naor's protocol execution by sending $y = G(r)$ if $b = 0$ and z if $b = 1$. In the opening phase, decommit to this part by revealing (b, r). Output 1 exactly if $\mathcal{A}$ succeeds, that is, if $\mathsf{R}(\text{ADVPAR}, \text{hist}(b), b, b^*) = 1$ for a valid opening of $\mathcal{A}$ to b^*.

Observe that if z is really random we output 1 with probability $\pi_{\mathsf{open}}(\mathcal{A})$, because the distribution of the data in the simulation is the same as in an actual attack on the assembled scheme. If z is pseudorandom then we output 1 with the probability that $\mathcal{A}_{\text{secret}}$ is

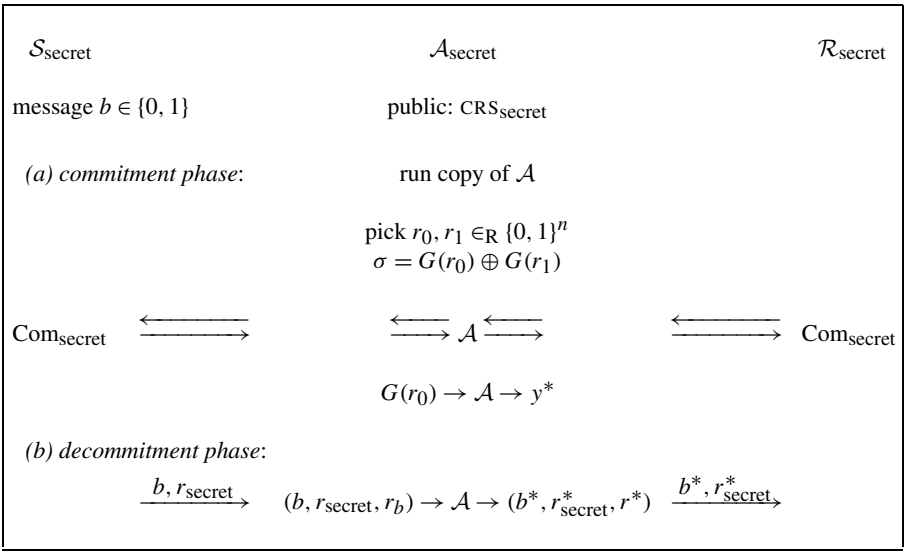

Fig. 3. Non-malleability with respect to opening of assembled scheme.

victorious. By assumption, this is noticeably smaller than $\pi_{\mathsf{open}}(\mathcal{A})$, and therefore we distinguish random and pseudorandom inputs with noticeable advantage. This, however, refutes the pseudorandomness of G.

Altogether, we have started with an arbitrary adversary $\mathcal{A}$ attacking the assembled scheme, and derived an adversary $\mathcal{A}_{\text{secret}}$ that succeeds in attacking $\text{Com}_{\text{secret}}$ for parameters $\text{CRS}_{\text{secret}}$ virtually with the same probability that $\mathcal{A}$ succeeds in attacking the assembled scheme on $\text{CRS}_{\text{secret}}$ and truly random σ. By assumption about the non-malleability of $\text{Com}_{\text{secret}}$, for $\mathcal{A}_{\text{secret}}$ there is a simulator $\mathcal{A}'_{\text{secret}}$ succeeding in outputting a related message essentially with the same probability as $\mathcal{A}_{\text{secret}}$. But then this algorithm $\mathcal{A}'_{\text{secret}}$ is also an appropriate simulator for adversary $\mathcal{A}$ attacking the assembled scheme. □

Applying our constructions in this paper we conclude:

Corollary 1. *Under the discrete-log or RSA assumption, there is an interactive bit commitment scheme in the CRS model that is liberal non-malleable with respect to opening, but not with respect to commitment$_{FF}$ and not with respect to commitment$_{DDN}$.*

We finally remark that our separation shows that schemes which are non-malleable with respect to opening do not necessarily fulfill the stronger definition per se. Yet, there may still be general transformations lifting such schemes to the higher security level. For example, a trivial transformation is to neglect the original scheme entirely and to run the DDN protocol from scratch instead.

4. Discrete-Log-Based Non-malleable Commitments

In this section, we introduce our discrete-log based commitment schemes which are non-malleable with respect to opening; the RSA and factoring case are discussed in Sects. 5 and 6, respectively.

In Sect. 4.1, we start with an instructive attempt to achieve non-malleability by standard proof-of-knowledge techniques. We show that this approach yields a scheme which is only non-malleable with respect to opening in the presence of static adversaries, i.e., adversaries that try to find a commitment after passively observing a commitment between the original sender and receiver. In Sect. 4.2, we develop out of this our scheme which is non-malleable against the stronger PIM adversaries. The formal proof of non-malleability appears in Sect. 4.3.

4.1. *Non-malleability with Respect to Static Adversaries*

Consider Pedersen's well-known discrete-log-based perfectly-secret scheme [43]. Let G_q be a cyclic group of prime order q and g_0, h_0 two random generators of G_q. Assume that computing the discrete logarithm $\log_{g_0} h_0$ is intractable (e.g., if G_q is an appropriate elliptic curve or subgroup of $\mathbb{Z}_p^*$). To commit to a message $m \in \mathbb{Z}_q$, choose $r \in_R \mathbb{Z}_q$ and set $M := g_0^m h_0^r$. To open this commitment, reveal m and r. Obviously, the scheme is perfectly-secret as M is uniformly distributed in G_q, independently of the message. It is computationally-binding because opening a commitment with distinct messages requires computing $\log_{g_0} h_0$.

Unfortunately, Pedersen's scheme is malleable: Given a commitment M of some message m, an adversary obtains a commitment for $m + 1 \bmod q$ by multiplying M with g. Later, the adversary reveals $m + 1 \bmod q$ and r after learning the original decommitment m, r. This holds even for static adversaries. Such adversaries do not try to inject messages in executions, but rather learn a protocol execution of $\mathcal{S}$ and $\mathcal{R}$—which they cannot influence—and afterwards try to commit to a related message to $\mathcal{R}$. In the case of non-malleability with respect to opening, the adversary must also be able to open the commitment after the sender has decommitted.

A possible fix that might come to one's mind are proofs of knowledge showing that the sender actually knows the message encapsulated in the commitment. For the discrete-log case, such a proof of knowledge consists of the following steps [41]: The sender transmits a commitment $S := g_0^s h_0^t$ of a random value $s \in_R \mathbb{Z}_q$ under randomness $t \in_R \mathbb{Z}_q$, the receiver replies with a random challenge $c \in_R \mathbb{Z}_q$ and the sender answers with $y := s + cm \bmod q$ and $z := t + cr \bmod q$. The receiver finally checks that $SM^c = g_0^y h_0^z$.

If we add a proof of knowledge to Pedersen's scheme we obtain a protocol which is non-malleable with respect to opening against static adversaries. This follows from the fact that any static adversary merely sees a commitment of an unknown message before trying to find an appropriate commitment of a related message. Since the proof of knowledge between $\mathcal{S}$ and $\mathcal{R}$ is already finished at this point, the static adversary cannot rely on the help of $\mathcal{S}$ and transfer the proof of knowledge. We leave further details to the reader and address the non-malleable protocol against PIM adversaries in the next section instead.

4.2. *Non-malleability with Respect to PIM Adversaries*

The technique of assimilating a proof of knowledge as in the previous section does not thwart PIM attacks. Consider again the PIM adversary committing to $m+1 \bmod q$ by multiplying M with g. First, this adversary forwards the sender's commitment S for the proof of knowledge to the receiver and hands the challenge c of the receiver to the sender. Conclusively, she modifies the answer y, z of the sender to $y^* := y + c \bmod q$ and $z^* := z$. See Fig. 4. Clearly, this is a valid proof of knowledge for $m+1 \bmod q$ and this PIM adversary successfully commits and later decommits to a related message.

Coin-flipping comes to the rescue. In a coin flipping protocol, one party commits to a random value a, then the other party publishes a random value b, and finally, the first party decommits to a. The result of this coin flipping protocol is set to $c := a \oplus b$ or, in our case, to $c := a + b \bmod q$ for $a, b \in \mathbb{Z}_q$. If at least one party is honest, then the outcome c is uniformly distributed (if the commitment scheme is binding and secret).

The idea is now to let the challenge in our proof of knowledge be determined by such a coin-flipping protocol. But if we also use Pedersen's commitment scheme with the public generators g_0, h_0 to commit to value a in this coin-flipping protocol, we do not

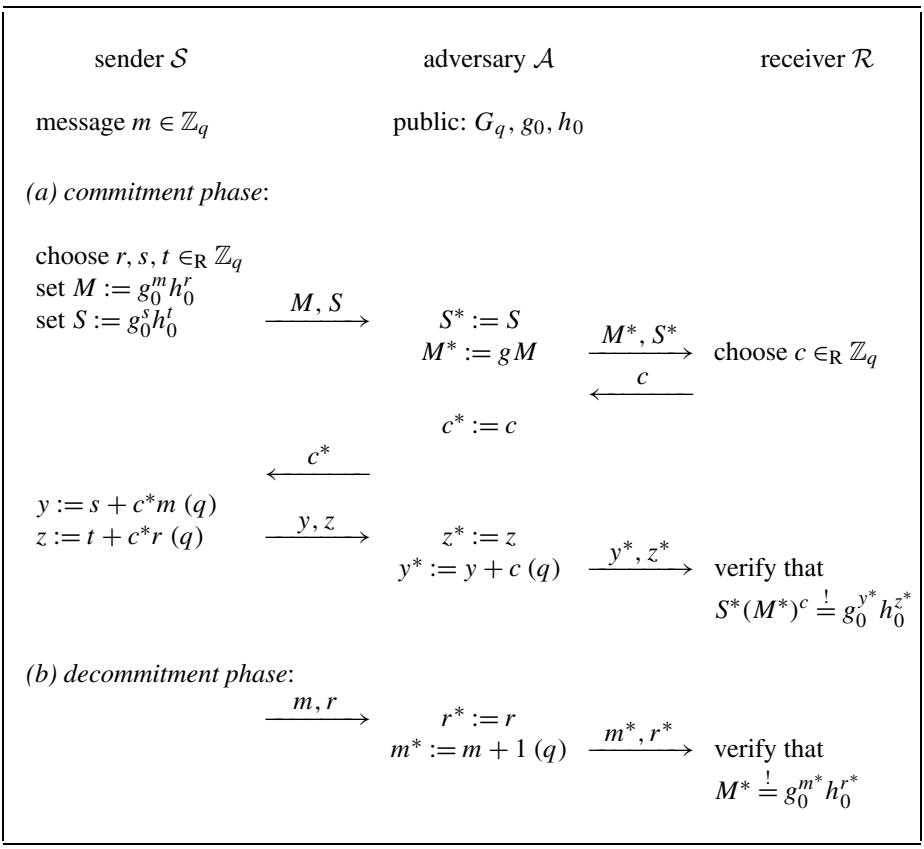

Fig. 4. PIM attack on Pedersen's commitment scheme with proof of knowledge.

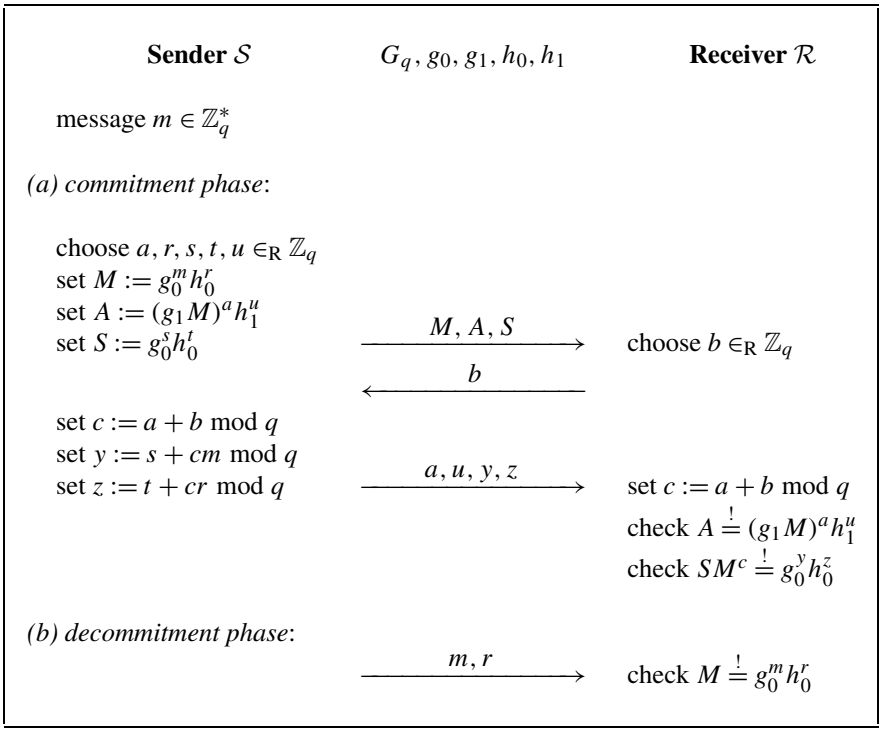

Fig. 5. Discrete-log-based non-malleable commitment scheme.

achieve any progress: The adversary might be able to commit to a related a^* and thus bias the outcome of the coin-flipping to a suitable challenge c^*.

The solution is to apply Pedersen's scheme in this sub-protocol with the commitment M as one of the generators, together with an independent generator h_1 instead of g_0, h_0; for technical reasons, we rather use $(g_1 M)$ and h_1 for another generator g_1. As we will show, the coin-flipping in the proof of knowledge between $\mathcal{A}$ and $\mathcal{R}$ is based on generators $g_1 M^*$ and h_1 instead of $g_1 M, h_1$ as in the sender's proof of knowledge. Because the adversary's commitment M^*, even though possibly being related to M, is most likely different from M (or else the adversary's decommitment m^*, r^* for M^* together with the original decommitment m, r for M would allow to compute the discrete logarithm of h_0 to g_0), this prevents the adversary from adapting the sender's and receiver's values and therefore from transferring the proof of knowledge. Details follow.

We describe the protocol given in Fig. 5 which combines the aforementioned ideas. The public parameters are (a description of) a cyclic group G_q of prime order q and four random generators g_0, g_1, h_0, h_1 of G_q. Basically, the sender $\mathcal{S}$ commits to his message $m \in \mathbb{Z}_q^*$ with Pedersen's scheme[4] by computing $M = g_0^m h_0^r$ and proves by a proof of knowledge (values S, c, y, z in Fig. 5) that she is aware of a valid opening of

[4] Note that, as opposed to Pedersen's scheme, we require that $m \neq 0$; the technical reason is that in the security proof we need to invert the message modulo q.

the commitment. The challenge c in this proof of knowledge is determined by a coin-flipping protocol with values A, a, u, b.

It is clear that our protocol is computationally-binding under the discrete-log assumption, and perfectly-secret as the additional proof of knowledge for m is *witness independent* (aka perfectly witness indistinguishable) [22], i.e., for any challenge c, the transmitted values S, y, z are distributed independently of the actual message [41].

Proposition 1. *The commitment scheme in Fig. 5 is perfectly-secret and, under the discrete-log assumption, computationally-binding.*

In the next section, we strictly prove that our scheme is indeed non-malleable. By now, we already remarked that the non-malleability property of our scheme also relies on the hardness of computing discrete logarithms. This dependency is not surprising. After all, any adversary being able to compute discrete logarithms with noticeable probability also refutes the binding property of Pedersen's scheme and can thus decommit for any related message with this probability.

A rough idea why our protocol is non-malleable can be described as follows. Given a commitment M of some unknown message m (together with a witness-independent proof of knowledge described by S, c, y, z) with respect to parameters p, q, g_0, h_0, we show how to employ the PIM adversary $\mathcal{A}$ to derive some information about m. Namely, if we are able to learn the related message m^* of the adversary by extracting it via her "self-employed" proof of knowledge, then we know that m is related to m^* for the relation R. This, of course, contradicts the perfect-secrecy of the commitment M. We remark that the formal proof of non-malleability requires to come up with a simulator generating a related message without the help of the sender. However, as we will show, the essential part of the simulator is made out of such an extraction procedure.

For details and further discussion we refer to the next section.

Theorem 2. *Under the discrete-logarithm assumption, the scheme in Fig. 5 is a perfectly-secret commitment scheme which is liberal non-malleable with respect to opening.*

It is worthwhile to point out that we cannot hash longer messages to $\mathbb{Z}_q^*$ before applying our non-malleable commitment scheme because then we extract the hash value and not the message m^* itself. But this could be insufficient, since it might be impossible to deduce anything about m via $\mathsf{R}(\text{ADVPAR}, \text{hist}_m, m, m^*)$ given solely the hash value of m^*. The same disadvantage occurs in the RSA case. We stress that the schemes in Sect. 6 with the a posteriori verifiable proofs of knowledge do not suffer from this problem. There, one can first hash the message as the proof of knowledge operates on the original message instead of the hash value.

4.3. *Formal Proof of Non-malleability*

We present the proof of non-malleability of the protocol in the previous section first from a bird's eye view and progressively fill in more details. The main part of the proof consists of the construction of an extraction procedure that enables us to extract the

adversary's message related to the original message. We start with an outline of this procedure, then analyze it with respect to restricted attacks and, subsequently, supplement the remaining steps for full-fledged attacks. Finally, we discuss that the required non-malleability simulator can be derived from the extraction procedure. At the end of this section, we address the multi-party setting.

Outline of Extraction Procedure

In this outline here, we make some simplifications concerning the adversary: First, we assume that the PIM adversary always catches up concerning the order of the transmissions, i.e., sends her first message after learning the first message of $\mathcal{S}$ and answers to $\mathcal{S}$ after having seen $\mathcal{R}$'s response, etc. Second, let the adversary *always* successfully commit and decommit to a related message, rather than with, say, small probability. Third, we presume that M is independent of the adversarial parameters. All restrictions will be removed in subsequent sections.

To learn the adversary's message m^* in the simplified case we use the proof of knowledge in our commitment protocol. Intuitively, a proof of knowledge guarantees that the prover knows the message, i.e., one can extract the message by running experiments with the prover. Specifically, we inject values $p, q, g_0, h_0, M, S, c, y, z$ into a simulated PIM attack with $\mathcal{A}$ and impersonate $\mathcal{S}$ and $\mathcal{R}$. Additionally, we choose g_1 at random and set $h_1 := (g_1 M)^w$ for a random $w \in_R \mathbb{Z}_q^*$. We also compute random $a_0, u_0 \in_R \mathbb{Z}_q$ and insert g_1, h_1 and $A := (g_1 M)^{a_0} h_1^{u_0}$ into the experiment with $\mathcal{A}$. We start with the extraction procedure by committing to m, s, a_0 via M, S, A on behalf of the sender. Then, by the presumption about the order of the transmissions, the adversary sends M^*, S^*, A^* (possibly by changing M, S, A and without knowing explicitly the corresponding values m^*, r^* etc.). See Fig. 6 on p. 551 for a pictorial description.

We play the rest of the commitment phase twice by rewinding it to the step where the receiver chooses b and sends it to the adversary $\mathcal{A}$. To distinguish the values in both repetitions we append the number of the loop as subscript and write a_1, a_1^*, a_2, a_2^*, etc.

The first time, the adversary upon receiving b_1 passes some b_1^* to the (simulated) sender $\mathcal{S}$, and expects $\mathcal{S}$ to open the commitment for a and supplement the proof of knowledge for M with respect to the challenge $a_1 + b_1^* \bmod q$. By the trapdoor property of Pedersen's commitment scheme [6], we are able to open A with any value for a_1 since we know $\log_{(g_1 M)} h_1$. That is, to decommit A with some a_1 reveals a_1 and $u_1 = u_0 + (a_0 - a_1)/\log_{(g_1 M)} h_1 \bmod q$; it is easy to verify that indeed $A = (g_1 M)^{a_1} h_1^{u_1}$. In particular, we choose a_1 such that $a_1 + b_1^* \bmod q$ equals the given value c. Hence, y and z are proper values to complement the proof of knowledge for M. Finally, the adversary answers with the decommitment a_1^*, u_1^* for A^* and the rest of the proof of knowledge for M^* with respect to challenge $a_1^* + b_1 \bmod q$.

Now we rewind the execution and select another random challenge b_2. The adversary then decides upon her value b_2^* (possibly different from her previous choice b_1^*) and hands it to $\mathcal{S}$. Again, we open A with a_2 such that $c = a_2 + b_2^* \bmod q$. The adversary finishes her commitment with a_2^*, u_2^* as opening for A^* and the missing values for the proof of knowledge.

The fundamental proof-of-knowledge paradigm [2,19,20] (together with the so-called special soundness of Okamoto's protocol [41]) says that we can extract the message

m^* if we learn two valid executions between $\mathcal{A}$ and $\mathcal{R}$ with the same commitment M^*, S^*, A^* but different challenges. Hence, if the adversary's decommitments satisfy $a_1^* = a_2^*$ and we have $b_1 \neq b_2$ (which happens with probability $1 - 1/q$), then this yields different challenges $a_1^* + b_1, a_2^* + b_2$ in the executions between $\mathcal{A}$ and $\mathcal{R}$ and we get to know the message m^*.

We are therefore interested in the event that the adversary is able to "cheat" by presenting different openings $a_1^* \neq a_2^*$ during the extraction procedure. Below, we prove that the adversary cannot find different openings for commitment A^* too often, else we would derive a contradiction to the intractability of the discrete-log problem (Lemma 1). Hence, under the discrete-log assumption this event hardly occurs, and we extract m^* with sufficiently high probability. To be precise, we extract *some* message m' and have to show that this extracted message m' is almost as likely related to the original message as m^* is. This again follows form the hardness of computing discrete logarithms (Lemma 2).

Extraction with Respect to Restricted Attacks

We address a more formal approach to the extraction procedure, still considering a slightly restricted attack. Namely, as in the outline, we too adopt the convention that the adversary $\mathcal{A}$ does not "mix" the order of messages but rather catches up. We also presume for simplicity that the message space M is independent of the adversarial parameters. Call this a *restricted* attack. We afterwards explain how to deal with *full-fledged* attacks.

Before we jump into restricted attacks, we first remark that the history value hist_m can be neglected for the analysis of the extraction procedure for both restricted and full-fledged attacks. We omit mentioning it since we use only black-box simulations to extract the adversary's message in the commitment phase, hence, any value hist_m given to $\mathcal{A}'$ is simply forwarded to $\mathcal{A}$ in order to run the black-box simulation. Only the conclusive construction of the non-malleability simulator from the extraction procedure requires a more careful look at the history value.

Our aim is to extract the adversary's message from her commitment within a negligibly close bound to the adversary's success probability $\pi_{\mathsf{open}}(\mathcal{A})$. To this end, we repeat some basic facts about proofs of knowledge and knowledge extractors [2,19,20]; we discuss them for the example of Okamoto's discrete-log-based proof of knowledge (see [41] or Sect. 4.1) for a given $M = g_0^m h_0^r$.

The knowledge extractor interacting with the prover works in two phases. Namely, it first generates a random conversation S, c, y, z by running the prover to obtain S, by selecting c and by letting the prover answer with y, z to S, c. If this communication in the initial run is invalid, then the extractor aborts. Otherwise it tries to extract at all costs. That is, the extractor fixes this communication up to the challenge, and then loops (till success) to seek another accepting conversation with the same communication prefix S. This is done by rewinding the execution to the choice of the challenge and reselecting other random challenges. Once the extractor has found another accepting execution for challenge c', it can extract if $c \neq c'$, and otherwise it stops with a failure message.

We claim that the extractor runs in expected polynomial time and outputs a representation of M with respect to g_0, h_0 with probability at least $\pi - 1/q$. Here, π denotes the probability that the prover makes the verifier accept, and $1/q$ is called the error of the

protocol. This can been seen as follows. Fix an arbitrary pair M and S and the prover's coin tosses, and condition all subsequent probabilities on these data. Let p denote the conditional probability (over the choice of the challenge) that the verifier accepts. Then we enter the loop phase with probability at most p and then need at most $\frac{1}{p}$ repetitions on average to find another accepting execution, resulting in an expected polynomial running time. Since M, S and the coin tosses are arbitrary, this also holds for randomly chosen values. As for the success probability note that the extractor fails if the initial run is invalid (with probability at most $1 - \pi$) or the second successful execution is for the same challenge (with probability $1/q$). Hence, the extractor succeeds with probability at least $\pi - 1/q$.

Assume that we communicate with some party $\mathcal{C}$ which is going to commit to an unknown message $m \in_R \mathsf{M}$ in Pedersen's commitment scheme, augmented by a proof of knowledge. Recall that our goal is to show that we can break the secrecy of this commitment scheme with the help of the attacker on the non-malleability. We choose a group G_q and two generators g_0, h_0 and send them to $\mathcal{C}$. Party $\mathcal{C}$ selects $r, s, t \in_R \mathbb{Z}_q$ and sends $M := g_0^m h_0^r$, $S := g_0^s h_0^t$. We answer with a random challenge $c \in_R \mathbb{Z}_q$ and $\mathcal{C}$ returns $y := s + cm, z := t + cr \bmod q$. Finally, we check the correctness. Put differently, we perform all the steps of the sender in our protocol except for the coin flipping.

The aim of our extraction procedure now is to get the message m^* of the PIM adversary when the adversary faces $\mathcal{C}$'s commitment. For this, the extractor chooses additional generators g_1, h_1 by setting $g_1 := g_0^v$ and $h_1 := (g_1 M)^w$ for random $v, w \in_R \mathbb{Z}_q^*$, and computes $A := (g_1 M)^{a_0} h_1^{u_0}$ according to the protocol description for random $a_0, u_0 \in_R \mathbb{Z}_q$.[5] Then the extractor starts to emulate the PIM attack by pretending to be $\mathcal{S}$ and $\mathcal{R}$ and with values $G_q, g_0, g_1, h_0, h_1, M, S, A$. Figure 6 shows a description.

Because of the assumption about the order of messages, the adversary commits to M^*, S^*, A^* after seeing M, S, A. Next, we use the same stop-or-extract technique as in [2,19]. In our case, the rewind point (if we do rewind) is the step where the receiver sends b. In each repetition, we send a random value $b_i \in_R \mathbb{Z}_q$—the subscript denotes the number $i = 1, 2, \ldots$ of the loop—on behalf of the receiver and the adversary hands some value b_i^* to the simulated sender. Knowing the trapdoor $w = \log_{(g_1 M)} h_1$ we open A with $a_i, u_i = u_0 + (a_0 - a_i)/w \bmod q$ such that $a_i + b_i^*$ equals the given value c, and send the valid answer y, z to the challenge c in the proof of knowledge for M. The adversary replies with $a_i^*, u_i^*, y_i^*, z_i^*$ to the receiver. Again, see Fig. 6.

An important modification of the knowledge extractor in comparison to the one in [2,19] is that, once having entered the loop phase, not only does our extractor stop in case of success, it also aborts with no output if in some repetitions i, j the adversary both times successfully finishes the commitment phase—which includes a correct decommitment to the "coin-flipping commitment" A^*—but opens A^* with distinct values $a_i^* \neq a_j^*$. We say that $\mathcal{A}$ *counterfeits a coin* if this happens. In this case, the extractor fails to extract a message. We remark that we are only interested in the case that $\mathcal{A}$ sends distinct openings of A^* in *accepting* executions because the extractor only relies on such executions. We call the derived procedure EXTRACT.

[5] Clearly, the choice of the generators requires that M and therefore m and M are determined *before* the adversary is presented CRS and selects ADVPAR.

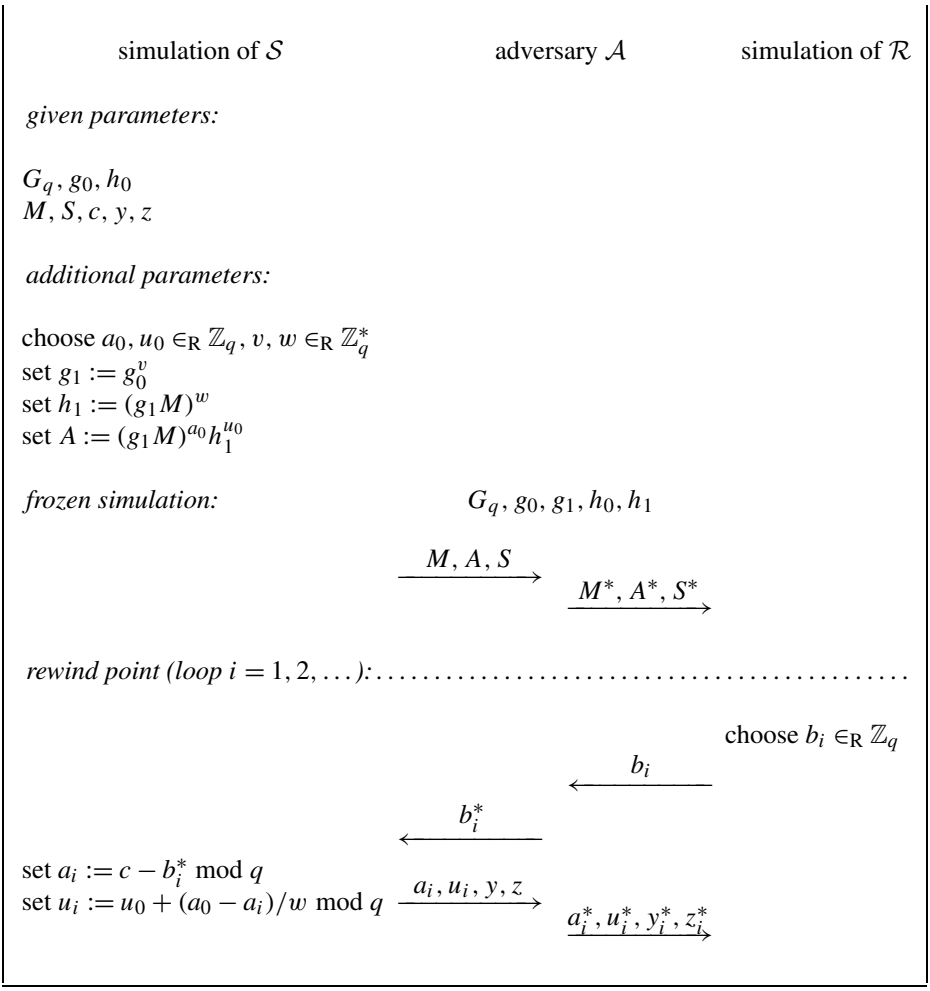

Fig. 6. Knowledge extraction.

Our first observation is that our knowledge extractor stops (either with success or aborting prematurely) in expected polynomial-time. This follows as in [2,19] since our extractor even has an additional abort requirement.

To analyze the success probability of our extractor, let π denote the probability of $\mathcal{A}$ completing the commitment phase with $\mathcal{R}$ successfully in procedure EXTRACT. The basic extraction paradigm says that we are able extract with probability $\pi - 1/q - \epsilon(n)$, where $\epsilon(n)$ denotes the probability that $\mathcal{A}$ counterfeits a coin (n is the security parameter). The reason for this is that, given $\mathcal{A}$ does not counterfeit, the adversary's openings $a_{i_1}^* = a_{i_2}^* = \cdots$ in the valid commitment conversations are all equal. But then the values $b_{i_j} + a_{i_j}^* \bmod q$ for $j = 1, 2, \ldots$ of challenges in the proof of knowledge between $\mathcal{A}$ and $\mathcal{R}$ are independently distributed. Analogously to [2,19], it follows that the extractor finds a message with probability $\pi - 1/q - \epsilon(n)$ in this case.

Recall that we would like to guarantee that we extract with probability approximately $\pi_{\mathsf{open}}(\mathcal{A})$. Obviously, π upper bounds $\pi_{\mathsf{open}}(\mathcal{A})$, and it would thus suffice to show that $\epsilon(n)$ roughly equals $\pi - \pi_{\mathsf{open}}(\mathcal{A})$ or, put differently, that $\delta(n) := \epsilon(n) - (\pi - \pi_{\mathsf{open}}(\mathcal{A}))$ is negligible. One may think of the difference $\pi - \pi_{\mathsf{open}}(\mathcal{A})$ describing the probability of executions in which $\mathcal{A}$ successfully commits but never finds a related, valid opening (e.g., if $\mathcal{A}$ simply duplicates all messages of $\mathcal{S}$ in the commitment phase).

It remains to bound the probability $\delta(n)$. We will prove that $\delta(n)$ is negligible under the discrete-log assumption.

Lemma 1. *The probability that $\mathcal{A}$ counterfeits a coin in procedure* EXTRACT *is negligibly close to $\pi - \pi_{\mathsf{open}}(\mathcal{A})$.*

We remark that the proof of this lemma makes use of two important aspects. On the one hand, we exploit that the message space is fixed before the adversarial parameters are chosen. On the other hand, we apply the fact that we merely demand non-malleability with respect to opening, i.e., that $\mathcal{A}$ also reveals a valid decommitment.

Proof. We show that if Lemma 1 does not hold this contradicts the intractability of the discrete-log problem. We are given a group G_q, a generator g, and a random value $X \in G_q$ for which we are supposed to compute $\log_g X$. We show how to use $\mathcal{A}$ to do so (in expected polynomial-time with noticeable probability, yielding a strict polynomial-time algorithm with noticeable probability by standard truncation techniques).

Instead of using the commitment M of the third party $\mathcal{C}$, we modify procedure EXTRACT into a procedure COUNTERFEIT. In this modified procedure COUNTERFEIT, we instead run the knowledge extraction procedure incorporating the given values G_q, g, X, but generate the same distribution as the extractor. That is, select a message $m \in_{\mathsf{R}} \mathsf{M}$ and $v, w \in_{\mathsf{R}} \mathbb{Z}_q^*$, then set

$$g_0 := g^{-1/m} X, \qquad g_1 := g, \qquad h_0 := X^v, \qquad h_1 := X^w,$$

and compute M, A, S, c, y, z according to the protocol description. W.l.o.g., assume that $X \neq 1$ and $X^m \neq g$, else we already know the discrete-log of X. Then g_0, g_1, h_0 and h_1 are random generators of the subgroup G_q. Furthermore, $g_1 M = g g_0^m h_0^r = X^{m+rv}$ and thus $\log_{(g_1 M)} h_1 = (m + rv)/w \bmod q$.

Next we emulate $\mathcal{A}$ on values G_q, g_0, g_1, h_0, h_1 and M, A, S by running the extraction procedure above—with the exception that this time we enter the rewind phase only if the adversary successfully commits *and* also reveals a valid decommitment (m^*, r^*) to a related message after learning our decommitment (m, r) in the initial execution.

Once we have entered the rewind phase, whenever the extractor is supposed to open A to determine the challenge c in the loop, we also open the commitment so that the coin flipping protocol always yields the same value c. This is possible as we know $\log_{(g_1 M)} h_1$ and are therefore able to open A ambiguously.

Unlike in the case of an actual extraction process, here we sometimes suspend before looping although the adversary's initial commitment is accepted (because we also stop if the adversary's decommitment in the initial execution is invalid or unrelated). This restriction decreases the probability of $\mathcal{A}$ counterfeiting a coin at most by $\pi - \pi_{\mathsf{open}}(\mathcal{A})$. We call runs in which $\mathcal{A}$ also opens correctly in the initial execution *good*.

From $\mathcal{A}$'s point of view, the communication in the commitment phase in procedure COUNTERFEIT for a good run is identically distributed to the one in the original procedure EXTRACT because the data $G_q, g_0, g_1, h_0, h_1, M, A, S, c, y, z$ and the a_i, u_i, b_i's have the same distribution in both cases (i.e., the generators are random in both cases, the values M, A, S, c, y, z have been computed according to the protocol description in both cases, the b_i's are random both times, and the values a_i, u_i are determined by the other values). Hence, given that $\mathcal{A}$ counterfeits with probability $\epsilon(n) = \pi - \pi_{\mathsf{open}}(\mathcal{A}) + \delta(n)$ in the actual extraction procedure EXTRACT, $\mathcal{A}$ finds some $a_i^* \neq a_j^*$ for two accepting executions i, j with probability at least $\delta(n)$ in a good run in procedure COUNTERFEIT. By assumption, $\delta(n)$ is noticeable, so it suffices to prove that if $\mathcal{A}$ counterfeits in a good run in COUNTERFEIT then we can compute the discrete logarithm of X.

Let u_i^*, u_j^* denote the corresponding portions of the decommitment to a_i^* and a_j^* for A^* in loops i and j in COUNTERFEIT. In a good run, we have obtained some m^*, r^* satisfying the verification equation $M^* = g_0^{m^*} h_0^{r^*}$ from the adversary by revealing m, r in place of the sender in the initial execution. Particularly, we have:

$$(g_1 M^*)^{a_i^*} h_1^{u_i^*} = A^* = (g_1 M^*)^{a_j^*} h_1^{u_j^*},$$

and therefore

$$h_1^{(u_i^* - u_j^*)/(a_j^* - a_i^*)} = g_1 M^* = g_1 g_0^{m^*} h_0^{r^*} = g^{1 - m^*/m} X^{m^* + r^* v}.$$

Since $h_1 = X^w$ we can transform this into

$$g^{1 - m^*/m} = X^{\Delta} \quad \text{for } \Delta = w(u_i^* - u_j^*)/(a_j^* - a_i^*) - (m^* + r^* v) \bmod q.$$

Observe that Δ is computable from the data that we have gathered so far. From $m^* \neq m$ we conclude that $1 - m^*/m \neq 0 \bmod q$, and therefore $\Delta \neq 0 \bmod q$ has an inverse modulo q. Thus the discrete logarithm of X to base g equals $(1 - m^*/m)/\Delta \bmod q$. □

In summary, with probability $\pi_{\mathsf{open}}(\mathcal{A}) - 1/q - \delta(n)$—which is negligibly close to the adversary's success probability—we extract some message m' through procedure EXTRACT. The final step is to show that indeed m' equals the adversary's decommitment m^* except with negligible probability (or, more precisely, that m' is at least an appropriate substitution for m^* insofar as it also satisfies R often enough). Denote by $\pi_{\mathsf{open}}(\mathcal{E})$ the probability that the extraction procedure EXTRACT returns m' that is related to m under R.

Lemma 2. *The probabilities $\pi_{\mathsf{open}}(\mathcal{A}) - 1/q - \delta(n)$ and $\pi_{\mathsf{open}}(\mathcal{E})$ are negligibly close.*

Again, this lemma relies on the fact that the message space is independent of the adversarial parameters.

Proof. Similarly to Lemma 1, if this were not the case we could compute the discrete logarithm of X to g in group G_q. Namely, modify procedure EXTRACT into procedure

AMBOPEN by letting $g_0 := g$ and $h_0 := X$ and running the extraction procedure as before, only this time compute M, S, c, y, z for yourself, in particular, sample $m \in_R \mathsf{M}$, $r \in_R \mathbb{Z}_q$ and set $M := g_0^m h_0^r$, and choose g_1 at random and set $h_1 := (g_1 M)^w$ for a random $w \in_R \mathbb{Z}_q^*$.

In the initial run of the extraction procedure, if the adversary has finished the commitment phase successfully, hand the decommitment of M to the adversary and try to elicit the opening m^*, r^* of M^*. If the adversary refuses to decommit to M^* correctly, then stop; else continue the extraction. According to Lemma 1, the extraction yields a representation m', r' of M^* with probability $\pi_{\mathsf{open}}(\mathcal{A}) - 1/q - \delta(n)$. We are interested in the probability that m' also satisfies the relation.

Suppose that $\pi_{\mathsf{open}}(\mathcal{A}) - 1/q - \delta(n)$ and $\pi_{\mathsf{open}}(\mathcal{E})$ have noticeable difference. In particular, we conclude that $m' \neq m^*$ with noticeable probability in procedure AMBOPEN. But this implies that sufficiently often we obtain distinct representations (m^*, r^*), (m', r') of M^*. We are thus able to compute the discrete logarithm of $h_0 = X$ to base $g_0 = g$ with noticeable probability. Hence, under the discrete logarithm assumption, the probability that the extraction procedure returns m' that stands in relation to the sender's message is negligibly close to $\pi_{\mathsf{open}}(\mathcal{A}) - 1/q - \delta(n)$. □

Thwarting Full-Fledged Attacks

Our first observation is that the order of the messages in the PIM attack does not violate any of the discussions above. This is quite easy to see since any message on the sender's side can be predetermined at the outset of the knowledge extraction procedure (in terms of a function over the protocol communication so far).

So the final step is to remove the assumption about the message space. We have used three times the fact that M can be determined before the adversarial parameters are presented to the adversary. First, we have set h_1 equal to $g_1 M$, i.e., generated h_1 after seeing the commitment of $m \in_R \mathsf{M}$ in the extraction procedure. Second, in the proof of Lemma 1, we have sampled $m \in_R \mathsf{M}$ and then incorporated it into the generators. Third, Lemma 2 also requires to choose M before the adversary generates ADVPAR. In any case, this boils down to selecting the parameters CRS before sampling m because ADVPAR is a random variable depending on CRS only. Note that we do not change our protocol, but only the extraction and simulation procedures.

In the knowledge extraction procedure EXTRACT, recall that we copy the commitment M, S, c, y, z of party $\mathcal{C}$ into the extraction procedure and then set $h_1 := (g_1 M)^w$ for random w. To remove the dependency of a preselected message space, we modify M, S before using it in the proof of knowledge. That is, one first selects a group G_q and $M_{\mathsf{fake}} \in_R G_q$. Then we present G_q, g_0, h_0, g_1, h_1 to the adversary, where g_0, h_0, g_1 are random generators and $h_1 := (g_1 M_{\mathsf{fake}})^w$. This also determines $\mathsf{M} = \mathsf{M}(\text{ADVPAR})$ and we invoke $\mathcal{C}$ on G_q, g_0, h_0 and M to obtain M, S, c, y, z. Instead of using M, S in the extraction procedure, we run the knowledge extractor with M_{fake} and $S_{\mathsf{fake}} := S(M M_{\mathsf{fake}}^{-1})^c$ as well as c, y, z. Clearly, these values satisfy the verification equation $S_{\mathsf{fake}} M_{\mathsf{fake}}^c = g_0^y h_0^z$. Moreover, they are identically distributed to honestly generated ones, and hence the extractor achieves the same success probability. It is instructive to think of M_{fake} and S_{fake} as re-randomized versions of M, S.

The solution for the problem in Lemma 1 (procedure COUNTERFEIT) is similar to the previous case. There, we have chosen a group G_q and $g_0 := g^{-1/m} X$, $g_1 := g$, $h_0 :=$

X^v and $h_1 := X^w$. By this, we have possessed the discrete logarithm of $h_1 = X^w$ to base $g_1 M = g_1 g_0^m h_0^r = X^{(m+rv)}$. Instead, we now select G_q, choose a dummy message $m_0 \in_R \mathbb{Z}_q^*$ and set $g_0 := g^{-1/m_0} X$, $g_1 := g$, $h_0 := (g^{-1/m_0} X)^v$ and $h_1 := X^w$ and $M := g_0^{m_0}$. The values G_q, g_0, g_1, h_0, h_1 fix $\mathsf{M} = \mathsf{M}(\text{ADVPAR})$ and enable us to choose now the genuine message $m \in_R \mathsf{M}$. Since we know $v = \log_{g_0} h_0$ we can find an appropriate r with $m + vr = m_0$. Thus, $g_1 M = g_1 g_0^m h_0^r = X^{m+rv}$ and, again, $\log_{(g_1 M)} h_1 = (m + rv)/w$. Except for the case that $m + rv = 0$ in Lemma 1, which happens with probability $1/q$, this way of selecting the CRS is identical to the generation there.

We discuss that the proof carries over to the modification for Lemma 1. In the proof of Lemma 1, we finally find Δ with $g^{1-m^*/m} = X^\Delta$ and are able to compute the discrete logarithm of X to g since $m^* \neq m$. Here, we obtain the equation $g^{1-(m^*+vr^*)/(m+vr)} = X^\Delta$. If we had $m^* + vr^* = m + vr$ with noticeable probability, then from $m^* \neq m$ it would follow that the adversary finds a different representation m^*, r^* of $M = g_0^m h_0^r$ to base g_0, h_0 with noticeable probability. Specifically, defining another procedure, given $G_q, g_0 := g, h_0 := X$ select random g_1, h_1 and then sample a message $m \in_R \mathsf{M}(\text{CRS})$. Compute the commitment $M = g_0^m h_0^r$ for a random r as well as the values S, A for the proof of knowledge. Run only the initial commitment and decommitment phase of Lemma 1. If the adversary sends b^* for the coin-flipping sub-protocol in this initial run, then open the commitment for A with the previously selected values a, u and evaluate y, z for the proof of knowledge for $S, c = a \oplus b^*$. Finally, reveal m, r to the adversary to obtain m^*, r^*.

Note that we do not need to know the discrete logarithm of h_1 to $g_1 M$ here, since we do not loop, but merely run the initial phase. By assumption, $m^* + r^* \log_{g_0} h_0 = m + r \log_{g_0} h_0$ with noticeable probability. This, in turn, yields the discrete logarithm of $h_0 = X$ to $g_0 = g$. Hence, under the discrete logarithm assumption, this happens with negligible probability only, and, by analogy with Lemma 1, we therefore derive that the probability of $\mathcal{A}$ counterfeiting a coin does not exceed $\pi - \pi_{\mathsf{open}}(\mathcal{A})$ noticeably.

Finally, to adapt Lemma 2, we need to show that extracting m' different than m^* is infeasible, even if we have to publish the CRS ahead of the choice of M. Remember that in Lemma 2 we have used the adversary to find distinct representations $(m^*, r^*), (m', r')$ of M^* and to compute the discrete logarithm of $h_0 = X$ to $g_0 = g$ in G_q. Here, given G_q, g, X we make the following selection for random $r, v, w \in \mathbb{Z}_q$:

$$g_0 := g, \qquad g_1 := g_0^v h_0^{-r}, \qquad h_0 := X, \qquad h_1 := g_0^w.$$

These parameters pin down $\mathsf{M} = \mathsf{M}(\text{ADVPAR})$. We sample $m \in_R \mathsf{M}$ and let $M := g_0^m h_0^r$ for the preselected value r; the values of the proof of knowledge are computed honestly. It is easy to see that all values have the correct distribution (unless $g_1 = 1$ or $h_1 = 1$, in which case we simply abort). Furthermore, we know the discrete logarithm $w/(v + m)$ of h_1 with respect to $g_1 M$.

This completes the analysis of the extraction procedure with respect to full-fledged attacks.

Extraction Implies Non-malleability

A general construction of a non-malleability simulator $\mathcal{A}'$ from an extraction procedure has already appeared in [18] (for the plain model, but it is straightforward to adapt it to the CRS model, as done below). We briefly review the construction of $\mathcal{A}'$ for our case.

The non-malleability simulator $\mathcal{A}'$ prepares the CRS as required for the extraction procedure, invokes the adversary $\mathcal{A}$ to obtain ADVPAR and sets ADVPAR$' :=$ ADVPAR. Then the honest sender $\mathcal{S}$ is given a secret message $m \in_R \mathsf{M}(\text{ADVPAR}')$ and $\mathcal{A}'$ receives hist_m (which is forwarded to $\mathcal{A}$ for the black-box simulation).

For the extraction procedure, $\mathcal{A}'$ also has to prepare a commitment M of m together with a proof of knowledge S, c, y, z, but without actually knowing the secret message m of the sender. We let $\mathcal{A}'$ simply take an arbitrary message $m_0 \in \mathsf{M}(\text{ADVPAR}')$ and compute M, S, c, y, z from this message m_0 instead. Since the commitment M is perfectly secret and S, c, y, z are distributed independently of m_0, these values are equivalent to genuine values. This holds even if m_0 does not match the a priori information hist_m the adversary has about the sender's message.[6]

Finally, the simulator $\mathcal{A}'$ outputs the message it extracts from the PIM adversary. The results about the extraction procedure in the previous sections show that the success probability of $\mathcal{A}'$ is at most negligibly smaller than the probability of the PIM adversary. This completes the proof.

The Multi-party Case

It is not hard to see that non-malleability in the multiple-sender scenario follows from the single-sender case for our protocols. Nevertheless, if we grant the adversary the possibility to commit in several executions then we are not aware if our proof technique still works. For a *weaker* security notion we use the proposal from [18] that the adversary announces some subset of indices $i_1, \ldots, i_k$ in the commitment phase. The adversary is then called successful if she finds valid openings for these commitments and if $m^*_{i_1}, \ldots, m^*_{i_k}$ stand in relation to m. That is, we can view R as a restricted relation $\mathsf{R}(\text{ADVPAR}, \text{hist}_m, m, m^*_{i_1}, \ldots, m^*_{i_k})$. It follows straightforwardly that, if we let the adversary in our case announce the subset after having sent all the commitments $M^*_1, \ldots, M^*_{\text{poly}}$, then our scheme becomes liberal non-malleable with respect to opening in the multi-sender/multi-receiver setting.

Similarly, we can achieve non-malleability if we adopt the viewpoint of [14]. Suppose $\mathsf{R}(\text{ADVPAR}, \text{hist}_m, m_1, \ldots, m_{\text{poly}}, m^*_1, \ldots, m^*_{\text{poly}^*})$ is 0 whenever some $m^*_i = \perp$, and presume further that the adversary first has to commit to each value by sending all $M^*_1, \ldots, M^*_{\text{poly}^*}$ before seeing any of the receiver's challenges. Then our scheme is also non-malleable with respect to opening in the multi-sender/multi-receiver setting.

5. Non-malleable Commitments Based on RSA

In this section, we present the protocols based on RSA. The basic ideas remain: Add a proof of knowledge to a commitment of the message, where the challenge is determined by a coin-flip sub protocol which involves the commitment of the message. Some slight adjustments have to been done, though.

[6] In fact, a slightly more sophisticated argument shows that it would even be imperceptible for the adversary if the commitment scheme was only computationally-secret [18].

5.1. *RSA-Based Scheme*

Let N be an RSA modulus, i.e., the product of two large primes. An RSA exponent for N is an integer e which is co-prime to the Euler totient function $\varphi(N)$ and satisfies $e \neq 1 \bmod \varphi(N)$. The RSA assumption says that computing $g^{1/e} \bmod N$ for a random $g \in_R \mathbb{Z}_N^*$ is intractable.

The RSA-based non-malleable commitment scheme is built on the function $(m, r) \mapsto g^m r^e \bmod N$ for $m \in \mathbb{Z}_e, r \in \mathbb{Z}_N^*$ and e prime [41]. A commitment of $m \in \mathbb{Z}_e$ is given by $M := g^m r^e \bmod N$ for a random $r \in_R \mathbb{Z}_N^*$. This commitment scheme is perfectly secret (as taking eth powers is a permutation on $\mathbb{Z}_N^*$) and computationally-binding, and it supports an efficient three-round witness-independent proof of knowledge similar to the discrete-log case. Furthermore, it also gives rise to a trapdoor property. If (and only if) one knows the trapdoor $g^{1/e} \bmod N$, then one can open the commitment with arbitrary messages. Finally, we notice that one can efficiently compute an eth root of h from k, h, Δ, N, e satisfying the equation $h^k = \Delta^e \bmod N$ for $k \neq 0 \bmod e$.

For our protocol we also require a family of universal one-way hash functions [40]. This is a sequence $H = (H_n)_{n \in \mathbb{N}}$ of function sets $H_n := \{H_{k,n} \mid k\}$, where each $H_{k,n}$ maps elements from the common domain D_n to a common range R_n. Additionally, the family is target-resistant, i.e., for any probabilistic polynomial-time algorithm $\mathcal{A}$ the probability that $\mathcal{A}(1^n)$ generates some $x \in D_n$ and, after some function $H_{k,n}$ has been chosen uniformly from H_n and has been presented to $\mathcal{A}$, then $\mathcal{A}$ returns $x' \neq x$ with $H_{k,n}(x) = H_{k,n}(x')$, is negligible. In particular, every collision-intractable hash function is also universal one-way. In the following, we usually refer to an instance $H_{k,n}$ simply as H.

We describe our non-malleable commitment in Fig. 7. The CRS consists of a random RSA instance N, e and four random elements $g, G, h_0, h_1 \in_R \mathbb{Z}_N^*$ together with a universal one-way hash function $H : \mathbb{Z}_N^* \to \mathbb{Z}_e$. To commit to $m \in \mathbb{Z}_e$, choose $r \in_R \mathbb{Z}_N^*$ and set $M := g^m r^e$. Furthermore, compute $x := H(G^m R^e)$ for random $R \in_R \mathbb{Z}_N^*$ and select $a \in_R \mathbb{Z}_e, r, u \in_R \mathbb{Z}_N^*$ to calculate $A := (h_0^x h_1)^a u^e$ for the coin-flipping protocol. We remark that, in contrast to the discrete-log case where $A = (g_1 M)^a h_1^u$, here a commitment of the message enters A vicariously by means of h_0^x for the hash value x of yet another commitment $G^m R^e$ of the message. In addition to the computations above, execute the proof of knowledge protocol given in [41] for M. Clearly, the derived scheme is computationally-binding and perfectly-secret.

In comparison to the discrete-log case, we have to perform some extra work. Namely, we give two commitments of m and we use a universal one-way hash function H. The reason for this basically stems from the lack of symmetry: In the discrete-log case we use two generators and two exponents, whereas here the party selects one exponent and a single value raised to the eth power. Indeed, the second commitment $G^m R^e$ is only necessary if the message space depends on the adversarial parameters. Otherwise one could hash M to x and still achieve non-malleability with respect to such an "independent" message space.

5.2. *Proof of Non-malleability*

It remains to prove non-malleability. The proof is very similar to the one of the discrete-log case, so we only sketch the necessary adaptations of the main steps. We again begin

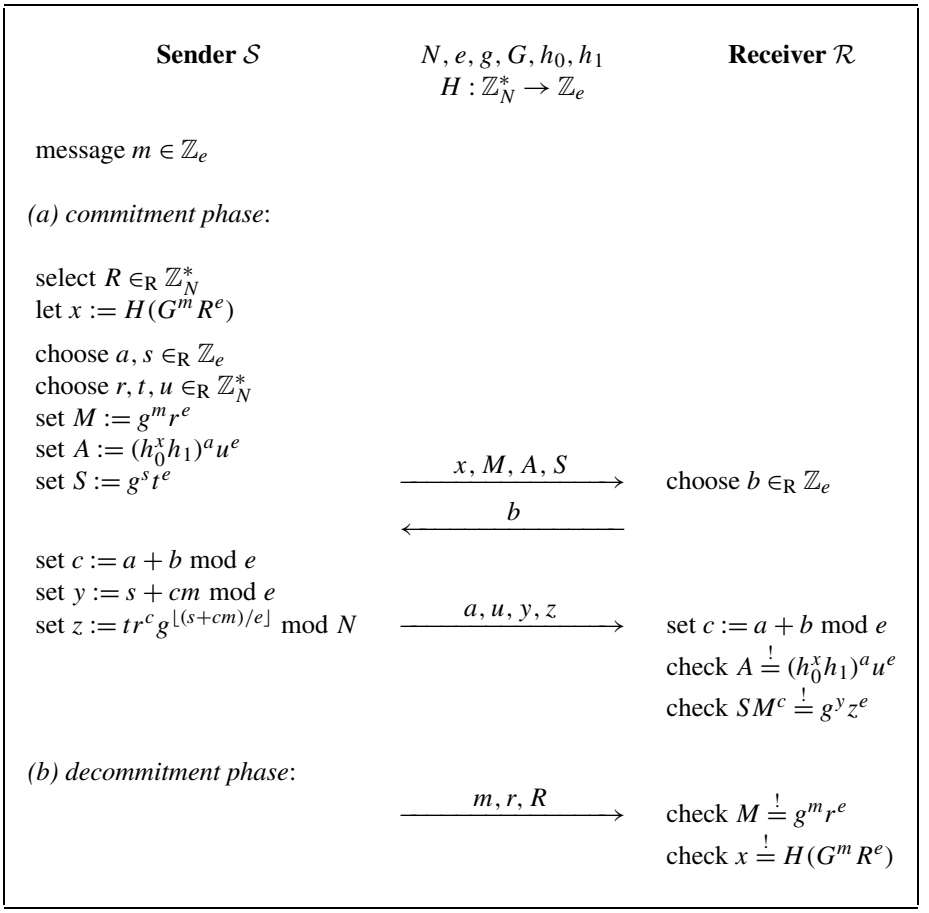

Fig. 7. RSA-based non-malleable commitment scheme.

with the extraction procedure with respect to restricted attacks where the message space is independent of the adversarial parameters and then lift it to full-fledged attacks. Once more, the order of the messages in the executions between the sender and the adversary, and the adversary and the receiver is irrelevant to the discussion. Also, the construction of the non-malleability simulator from the extraction procedure is identical to the discrete-log case and we do not address this part of the proof here.

Restricted Attacks

We first describe the extraction procedure in the RSA case. Given N, e, g and a commitment $M = g^m r^e$ for an unknown messages $m \in \mathsf{M}$ together with a proof of knowledge, select $v \in_R \mathbb{Z}_N^*$ and set $G := v^e \bmod N$. Also, let $x := H(R^e)$ for random $R \in \mathbb{Z}_N^*$ and define $h_0 \in_R \mathbb{Z}_N^*$ as well as $h_1 := h_0^{-x} w^e$ for $w \in_R \mathbb{Z}_N^*$. With these choices the eth root of $h_0^x h_1$ equals w, hence the coin-flip commitment $A := (h_0^x h_1)^a u^e$ is openable with any value a, and the extraction process is therefore identical to the discrete-log case.

The extraction works as long as the adversary does not find ambiguous decommitments for the commitment A^*. In the discrete-log case, in Lemma 1, it is shown that this probability is negligibly close to $\pi - \pi_{\mathsf{open}}(\mathcal{A})$ under the discrete-log assumption. Basically, the technique there was to choose appropriate parameters to be able to mimic the extraction procedure and to use the ambiguous opening to A^* to compute the discrete logarithm of X with respect to g in group G_q.

Here, in an intermediate step, we first show that we can essentially restrict ourselves to the case that the adversary sends a different hash value $x^* \neq x$. If the adversary were able to find a related opening with noticeable probability for $x^* = x$, this would contradict either the one-wayness of H or the RSA assumption. Namely, given N, e and a random $X \in \mathbb{Z}_N^*$ let $G := X$ and compute the other public parameters correctly, and sample $m \in_R \mathsf{M}$ and compute $M := g^m r^e$ and $G^m R^e$. Then, given the universal one-way hash function $H(\cdot)$, compute $x := H(G^m R^e)$ and run the adversary on these parameters. If the adversary chooses $x^* = x$ and later reveals a correct decommitment m^*, r^*, R^* after learning m, r, R, we either have $G^m R^e = G^{m^*}(R^*)^e$ from which we can compute the eth root of $G = X$, or we have $G^m R^e \neq G^{m^*}(R^*)^e$ yielding a collision $H(G^m R^e) = x = x^* = H(G^{m^*}(R^*)^e)$ for $H(\cdot)$. Hence, the adversary succeeds for $x = x^*$ only with negligible probability. Observe that this argument even holds if the message space depends on the adversarial parameters.

From now on, we condition on the event that the adversary always selects $x^* \neq x$. Transferring Lemma 1 means that we are given N, e, X and try to compute the eth root of the random value $X \in \mathbb{Z}_N^*$. For this, we copy N, e, sample $m \in_R \mathsf{M}$, compute $M := g^m r^e$ and $x := H(G^m R^e)$ for $r, g, G, R \in_R \mathbb{Z}_N^*$ and, again, set $h_1 := h_0^{-x} w^e$ for random $w \in_R \mathbb{Z}_N^*$ and $h_0 := X$. Analogously to Lemma 1, we run the extraction procedure (with the opening step in the initial execution to obtain m^*, r^*, R^*). Under this assumption, and following the approach in Lemma 1, from an ambiguous decommitment for a^* for the values chosen above, we finally derive the equation

$$h_0^{(x^*-x)(a_i^*-a_j^*)} = \Delta^e \bmod N$$

for known $\Delta, x^* \neq x, a_i^* \neq a_j^*$. Since $(x^* - x)$, $(a_i^* - a_j^*) \neq 0 \bmod e$ we can compute an eth root of $h_0 = X$. Hence, under the RSA assumption the extraction procedure succeeds with probability $\pi_{\mathsf{open}}(\mathcal{A}) - 1/e - \delta(n)$, where $\delta(n)$ is negligible.

The final step in the proof of the discrete-log case is Lemma 2, where we show that the extracted messages m' is (at least) a suitable replacement for m^*. In that lemma, we prove that if this were not true, then we could compute discrete logarithms. The analogous proof here is the same as in the adaption of Lemma 1: Given N, e, X, choose $m \in_R \mathsf{M}$, set $g := X$ and compute $M := g^m r^e$ as well as $x := H(G^m R^e)$ for random $G, R \in_R \mathbb{Z}_N^*$. Moreover, let $h_0 \in_R \mathbb{Z}_N^*$ and $h_1 := h_0^{-x} w^e$. Run the extraction procedure (with an initial decommitment step to get m^*, r^*, R^*) to obtain m', r', R' with $M^* = g^{m^*}(r^*)^e = g^{m'}(r')^e$; since $m^* \neq m'$ this yields the eth root of $g = X$.

Full-Fledged Attacks

Here, the message space is not independent of the adversarial data anymore. Similar to the discrete-log case we have to ensure that we are able to produce an appropriate CRS before we get to know the message space.

For the extraction procedure we choose the same re-randomizing technique as in the discrete-log case. To adapt the modification of Lemma 1, we select $G := v^e \bmod N$ for a random $v \in_R \mathbb{Z}_N^*$ and precompute $x := H(R_0^e)$ for $R_0 \in_R \mathbb{Z}_N^*$; since we know the eth root of G, it is easy to find an appropriate value R matching x for the afterwards chosen message $m \in_R \mathsf{M}(\text{ADVPAR})$. Choose the parameters $g, h_0 \in_R \mathbb{Z}_N^*$ honestly, and set $h_1 := h_0^{-x} w^e \bmod N$ for random $w \in_R \mathbb{Z}_N^*$. Conditioning again on the adversary sending $x^* \neq x$, the proof goes through in this case as well.

Finally, we have to prove that the extracted message equals the adversary's one (or, more precisely, satisfies the relation). Similarly as in the previous step, we select G as $G := v^e \bmod N$ such that we are able to preselect the value x. The rest of the proof is as before, i.e., we finally derive different openings of M yielding the eth root of g.

Theorem 3. *If the RSA assumption holds and if H is a family of universal one-way hash functions, then the protocol in Fig. 7 is a perfectly-secret commitment scheme which is liberal non-malleable with respect to opening.*

Although without practical significance, one can, in principle, construct collision-intractable hash functions and thus universal one-way hash functions under the RSA assumption. We may therefore reduce the prerequisite of the theorem to the RSA assumption only.

6. Factoring-Based Non-malleable Commitments

The DLog-based scheme and the RSA-based one use Okamoto's witness-independent proof of knowledge for the corresponding representation problem [41]. Although there are similar protocols for the factoring representation problem, such protocols are usually not known to be proofs of knowledge, i.e., a simulator for extracting the message is missing. However, our protocols heavily rely on this extraction property.

6.1. *Preliminaries*

We overcome the problem of not having a proof of knowledge by observing that, since we are actually interested in non-malleability with respect to opening, the proof of knowledge need not be verifiable immediately in the commitment phase. It suffices that the sender convinces the receiver of the proof's validity in the decommitment stage. To refute non-malleability, the adversary must open her commitment correctly, and particularly, the proof must be shown to be right then. Therefore, the simulator can already in the commitment phase assume that the proof is indeed valid. We call such a proof of knowledge *a posteriori verifiable*. In fact, with this technique we are able to derive a non-malleable commitment scheme similar to the RSA-based one, but relying on the intractability of factoring large numbers.

Factoring Representation

We briefly recall the factoring representation. For a thorough treatment of this problem see [25]. Given an n-bit RSA modulus $N = pq$ let τ denote an integer such that $2^{\tau+1}$

neither divides $p-1$ nor $q-1$, e.g., let τ be the smallest integer with this property. Now, squaring permutes the set of $2^{\tau+1}$th powers modulo N. Let HQR_N denote this subgroup of higher quadratic residues which is efficiently samplable by raising a random element from $\mathbb{Z}_N^*$ to its 2^nth power.

Our factoring-based non-malleable commitment scheme utilizes the analogue to the RSA function $(m,r) \mapsto g^m r^e \bmod N$, namely, the mapping $(m,r) \mapsto g^m r^{2^{\tau+t}} \bmod N$ for $m \in \mathbb{Z}_{2^t}$ for parameter $t \geq 1$ and random $g \in_{\mathrm{R}} \mathrm{HQR}_N$. Then the ability to find $(m,r), (m',r')$ with distinct $m, m' \in \mathbb{Z}_{2^t}$ such that $g^m r^{2^{\tau+t}} = g^{m'}(r')^{2^{\tau+t}} \bmod N$ enables to efficiently compute a $2^{\tau+1}$th root of g, which is as hard as factoring N. On the other hand, given a $2^{\tau+t}$th root of g, one may open a given commitment with appropriate values later (trapdoor property). Hence, for appropriate choices of the parameters we benefit from the same properties as for RSA. Unfortunately, unlike in the DLog or RSA case, no efficient proof of knowledge is known for this type of representation problem. As explained above, we therefore switch to a posteriori verifiable proofs.

Outline of CRT Extraction Procedure

Using the Chinese Remainder Theorem, we present a fast a posteriori verifiable proof of knowledge and thus a non-malleable commitment scheme based on the factoring representation problem.

We first explain the underlying idea of our construction by a simpler, yet insecure version of our proof of knowledge. Recall that, given a sequence of values $y_{p_i} = m \bmod p_i$ for distinct primes $p_1, \ldots, p_k$ such that $\prod_i p_i \geq m$, one can efficiently reconstruct the original number m (over the integers) by applying the Chinese Remainder Theorem. To derive an a posteriori verifiable proof of knowledge from this, we basically let the sender of the commitment to m also supply $y = m \bmod p$ for a small random prime p (determined again by a coin-flipping protocol). The receiver is then able to verify this value y later with help of the known prime p, after having received the original message m in the decommitment phase. The value y essentially serves as a proof of knowledge as we can derive m by rewinding the protocol and finishing it with different primes until the original message is recoverable according to the Chinese Remainder Theorem.

The basic construction suffers from two fundamental problems. First, $y = m \bmod p$ leaks some information about the message m, even if p is small. Second, we have to ensure that the extraction works even if some malicious prover sometimes supplies incorrect values y. In particular, the extractor should be able to detect if the recovered message is correct or not. Next, we describe how to solve these problems, yielding our a posteriori verifiable proof of knowledge.

To ensure secrecy of the message, we first demand that the actual message length is only $t-2d$ instead of t, where d is an appropriate parameter. The slightly shorter message is then padded with random bits $s \in_{\mathrm{R}} \mathbb{Z}_{2^{2d}}$ and the sender $\mathcal{S}$ now commits to the message $m_{\mathsf{pad}} := s + 2^{2d} m \in \mathbb{Z}_{2^t}$ instead of m (using randomness r to compute the commitment). Then both parties agree on a d-bit prime number p as the challenge and $\mathcal{S}$ reveals the residue $y_p := m_{\mathsf{pad}} + 2^t r \bmod p$, i.e., the sender also includes the randomness r of the commitment into the computation of the residue (this is to ensure verifiability for the extractor, as we will see below). Because of the $2d$ random bits s in the padded message, the distribution of y_p is almost independent of the message.

Specifically, for superlogarithmic d the distribution of y_p is statistically close to the uniform distribution on $\mathbb{Z}_p$, and therefore y_p does not leak any essential information about m.

To extract the message we repeat the challenge/response step several times with different challenges i.e., independent prime numbers. As explained above, we could easily retrieve the value (m_{pad}, r) and thus the message m by applying the Chinese Remainder Theorem *if all replies were correct*. But this assumption is too optimistic. The situation is even worse due to the a posteriori verifiability. For common proofs of knowledge, the extractor identifies the successful responses by simply checking the verifier's control equation. But our new approach lacks such a possibility as the proof would only be verifiable afterwards in the decommitment phase.

Luckily, since the adversary for non-malleability with respect to opening must also open her commitment successfully—which includes a successful check of the previously sent y_p—repeating the challenge/response step still guarantees that a small set of correct residues can be found among such repetitions. And although we may not be able to distinguish good from bad values immediately, such a list of residues allows identifying the right pair (m_{pad}, r), in principle, by checking which of the possible CRT solutions matches the given commitment for m_{pad} under randomness r.[7]

But how fast can we reconstruct the "right" pair (m_{pad}, r) from a list including presumably invalid residues? This problem has already been solved in the context of codes, namely, transmitting the residues of a message modulo different primes can be viewed a linear error-correcting code. The so-called list decoding algorithms generate a list of all possible messages which match the received codeword up to a given number of errors. In our context, we ask for the message/randomness pair which fulfills the codeword $(y_{p_1}, y_{p_2}, \ldots, y_{p_k})$ for at least K out of k primes (where we choose the number of repetitions k in dependence of the adversary's success probability). We then apply the following result [5, Corollary 2.1]:

Lemma 3 (Boneh 2000). *Given primes $p_1 < p_2 < \cdots < p_k$, integers $y_{p_1}, y_{p_2}, \ldots, y_{p_k} \in \mathbb{Z}$ and a threshold B, a list of all positive integers $m < B$ with $m = y_{p_i} \bmod p_i$ for at least K of the k congruences can be computed in polynomial-time if $\frac{K}{k} \geq \frac{\log_2 p_k}{\log_2 p_1} \cdot \sqrt{\frac{\log_2 4B}{\log_2 P}}$ where $P := \prod_{i=1}^{k} p_i$.*

See also [29] for a slightly better bound on K/k. Yet, in our setting the algorithm of [29] only improves the value $\log p_k / \log p_1 \leq 2$ to an arbitrarily small constant (whose inverse affects the running time of the reconstruction algorithm).

ϵ-Non-malleability

Extracting via the CRT list decoding algorithm requires collecting a sufficiently large number k of congruences, among which there must be K good ones. To guarantee that the extractor works in polynomial-time, we use the ϵ-variant of non-malleability. That is, the simulator using the extractor is given a parameter $\epsilon = \epsilon(n)$ and has to compute a related message with the same probability as the adversary, up to an error ϵ (plus a negligible function). The running time of the simulator may now also depend on $\epsilon^{-1}(n)$.

[7] Here we exploit the fact that the proof of knowledge comprises the message and the randomness.

For our approach we further require that ϵ is not too small. Specifically, we assume that any algorithm running in time $\text{poly}(n) \cdot \epsilon^{-2}(n)$ for some polynomial $\text{poly}(n)$ factors a random modulus with negligible probability only. For instance, take the subexponential running time $L[n] = e^{O(n^{1/3}(\log n)^{2/3})}$ of the currently best factorization algorithm for moduli of bit size n (see [34] for a discussion about the estimated security of such moduli), and let $\epsilon^{-1}(n)$ be bounded by, say, the time $L[n/8]$ to factor moduli of length $n/8$. Similar assumptions about the subexponential hardness of cryptographic problems have been used in other settings, e.g., [10,35,38].

6.2. *Factoring-Based Scheme*

The factoring-based non-malleable commitment scheme is given in Fig. 8. The public parameters consist of a random instance N, τ, t, g of the factoring representation problem and three random elements $G, h_0, h_1 \in_R \text{HQR}_N$ together with a universal one-way hash function $H : \mathbb{Z}_N^* \to \mathbb{Z}_{2^t}$. We assume that $N \leq 2^n$ such that values from $\mathbb{Z}_N^*$ can be described with n bits.

To commit to $m \in \mathbb{Z}_{2^{t-2d}}$, choose $s \in_R \mathbb{Z}_{2^{2d}}$, $r \in_R \mathbb{Z}_N^*$ and set $m_{\mathsf{pad}} := s + 2^{2d}m$, $M := g^{m_{\mathsf{pad}}} r^{2^{\tau+t}}$. Furthermore, compute $x := H(G^m R^{2^{\tau+t}})$ for random $R \in_R \mathbb{Z}_N^*$ and select $a \in_R \mathbb{Z}_{2^t}$, $r, u \in_R \mathbb{Z}_N^*$ to calculate $A := (h_0^x h_1)^a u^{2^{\tau+2t}}$ for the coin-flipping protocol. Note, that for technical reasons the exponent for the commitment A is $2^{\tau+2t}$ rather than $2^{\tau+t}$.

To derive the challenge prime p, both parties run the coin-flipping protocol agreeing on a random $c \in \mathbb{Z}_{2^t}$. Then they map c to a d-bit prime $p := \mathsf{PrimeGen}_d(c)$ via function $\mathsf{PrimeGen}_d$. The sender finally answers with the residue of $y_p = m_{\mathsf{pad}} + 2^t r \mod p$.

The function $\mathsf{PrimeGen}_d$ is assumed to be collision-intractable in the sense that two distinct random values $c, c' \in \mathbb{Z}_{2^t}$ are hardly ever sent to the same prime. The function may also fail to generate a truly prime number with some very small probability. We specify the exact requirement below. The function $\mathsf{PrimeGen}_d$ can be built, for instance, based on ideas developed by Maurer [37] or Cramer and Shoup [12]. Indeed, generating a random prime slows down the protocol significantly, and we therefore later explore alternatives based on relatively prime polynomials.

As for the technical parameters, let $t > 2d \geq 8$ and suppose d and $\frac{t+n}{d-1}$ are superlogarithmic but polynomial in n. Assume further that any algorithm running in time $\text{poly}(n) \cdot \epsilon^{-2}(n)$ for some polynomial $\text{poly}(n)$ factors a random modulus with negligible probability only. Also, let the probability of $\mathsf{PrimeGen}_d$ generating a collision or producing a composite number in $\text{poly}(n) \cdot \epsilon^{-2}$ independent executions be negligible. We call such parameters t, d, n and ϵ *admissible*.

6.3. *Security Proof*

Again, first we show that the factoring-based non-malleable commitment scheme in Fig. 8 is non-malleable for restricted attacks. Namely, $\mathcal{A}$ does not "mix" the order of messages and the messages space M is independent of the adversarial parameters. The step from such restricted attacks to full-fledged attacks follows straightforwardly from the RSA setting and is omitted.

As in the RSA case for restricted attacks, we first prove that we can condition on the adversary sending a different hash value $x^* \neq x$. Suppose towards contradiction

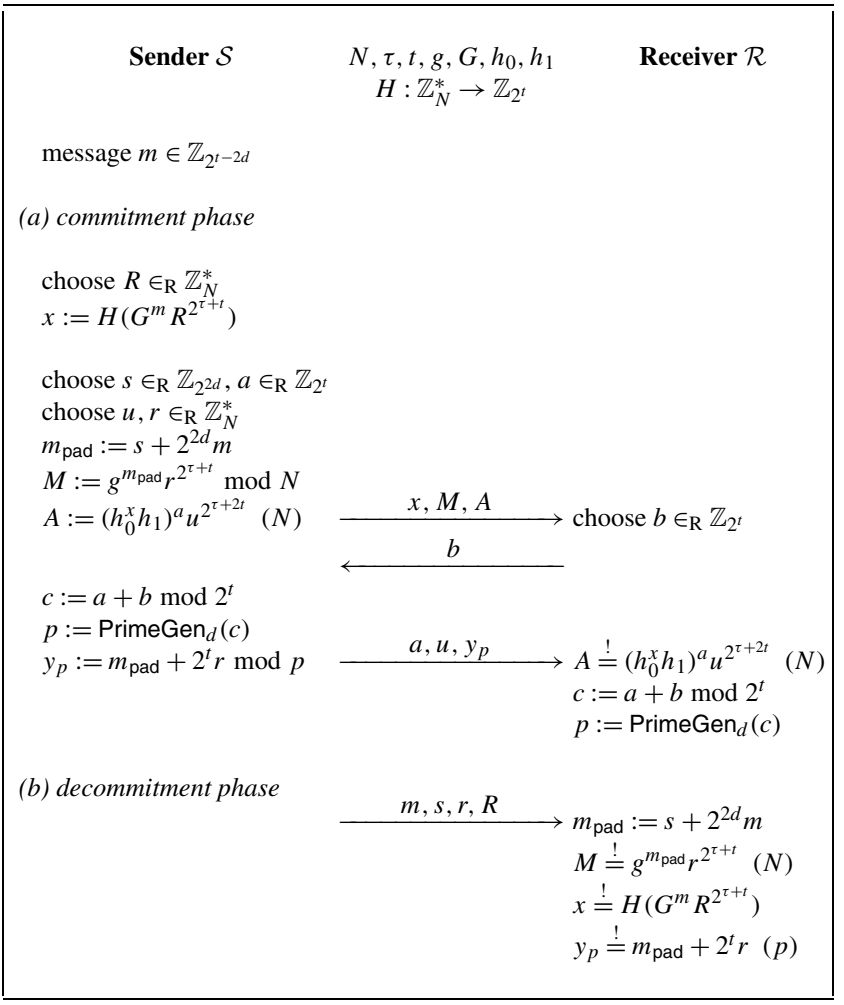

Fig. 8. Factoring based non-malleable commitment scheme.

that the adversary is able to find a related opening with noticeable probability for such $x^* = x$. Then we claim that this contradicts either the one-wayness of H or the factoring assumption. To see this, assume we are given N and a random $X \in \mathbb{Z}_N^*$. Let $G := X$ and compute the other public parameters correctly. Sample $m \in_R \mathsf{M}$ and set $M := g^{m_{\mathsf{pad}}} r^{2^{\tau+t}}$ and $G^m R^{2^{\tau+t}}$. Then, after receiving the universal one-way hash function $H(\cdot)$, compute $x := H(G^m R^{2^{\tau+t}})$ and run the adversary on these data. Suppose the adversary selects $x^* = x$ and, after seeing the opening m, r, R, later decommits correctly to m^*, r^*, R^*. Then, if $G^m R^{2^{\tau+t}} = G^{m^*}(R^*)^{2^{\tau+t}}$ we derive the $2^{\tau+1}$th root of $G = X$, and if $G^m R^{2^{\tau+t}} \neq G^{m^*}(R^*)^{2^{\tau+t}}$ we get a collision $H(G^m R^{2^{\tau+t}}) = x = x^* = H(G^{m^*}(R^*)^{2^{\tau+t}})$ for the function $H(\cdot)$. It follows that the adversary succeeds for $x = x^*$

with negligible probability only. We finally remark that the argument remains valid if the message space depends on the adversarial parameters.

In the following, we can now assume that the adversary always selects $x^* \neq x$. Next we address the CRT extractor. Given N, τ, t, g and a commitment $M := g^{m_{\mathsf{pad}}} r^{2^{\tau+t}}$ for an unknown messages $m \in \mathsf{M}$, select $v \in_{\mathrm{R}} \mathbb{Z}_N^*$ and set $G := v^{2^{\tau+t}} \bmod N$. Also, let $x := H(R^{2^{\tau+t}})$ for random $R \in \mathbb{Z}_N^*$ and define $h_0 \in_{\mathrm{R}} \mathbb{Z}_N^*$ as well as $h_1 := h_0^{-x} w^{2^{\tau+2t}}$ for $w \in_{\mathrm{R}} \mathbb{Z}_N^*$. With these choices the $2^{\tau+2t}$th root of $h_0^x h_1$ equals w, hence the coin-flipping commitment $A := (h_0^x h_1)^a u^{2^{\tau+2t}}$ is openable with any value a_i:

$$A = \left(h_0^x h_1\right)^a u^{2^{\tau+2t}} = \left(h_0^x h_1\right)^{a_i} \left(w^{a_i - a} u\right)^{2^{\tau+2t}}.$$

Therefore, we are in the analogous situation to the RSA case, being able to open A ambiguously.

The extractor next starts to emulate the PIM attack by pretending to be $\mathcal{S}$ and $\mathcal{R}$ using the given values and the constructed CRS. It suspends the simulation right before the receiver sends b. Next, the extractor repeats the challenge/response step

$$k := \frac{32(t+n+2)\epsilon^{-2}}{d-1}$$

times. It finally runs Boneh's algorithm on the data, trying to obtain a decommitment m'_{pad}, r' of the adversary's commitment M^*. If the algorithm returns such a pair the extractor outputs the unpadded message m'. If it returns more than one pair then we take the first one which is a valid decommitment to M^*.

For the analysis of the extractor's success probability, we first prove an analogous statement to Lemma 1 about the adversary being essentially unable to open her "coin" commitment ambiguously during the extraction procedure (i.e., counterfeits a coin). Here, this should hold under the factoring assumption in the sense that, given N, τ, t, g, X, we should not be able to efficiently compute the $2^{\tau+1}$th root of the random value $X \in \mathbb{Z}_N^*$. To show that we can use an allegedly successful adversary to refute this, copy N, τ, t, sample $m \in_{\mathrm{R}} \mathsf{M}$, compute $M := g^{m_{\mathsf{pad}}} r^{2^{\tau+t}}$ and $x := H(G^m R^{2^{\tau+t}})$ for $G \in_{\mathrm{R}} \mathrm{HQR}_N$ and $r, R \in_{\mathrm{R}} \mathbb{Z}_N^*$. Also, set $h_1 := h_0^{-x} w^{2^{\tau+2t}}$ for random $w \in_{\mathrm{R}} \mathbb{Z}_N^*$ and $h_0 := X$.

As in the proof for Lemma 1, we run our extraction procedure here, including an opening step in the initial execution to obtain m^*, r^*, R^*. From an ambiguous decommitment a_i^*, a_j^* of the adversary to A^* for the values chosen above, we finally derive the equation

$$h_0^{(x^*-x)(a_i^*-a_j^*)} = \Delta^{2^{\tau+2t}} \bmod N$$

for known $\Delta, x^* \neq x, a_i^* \neq a_j^*$. Since $0 < |x^* - x|$, $|a_i^* - a_j^*| < 2^t$ one has $0 < |x^* - x| \cdot |a_i^* - a_j^*| < 2^{2t}$. Hence, we can compute an $2^{\tau+1}$th root of $h_0 = X$. The running time of this algorithm is clearly bounded by $\mathrm{poly}(n) \cdot \epsilon^{-2}$, and we would thus be able to factor the modulus N in this time with noticeable probability. Altogether,

since the parameters are admissible, under the factoring assumption the probability that the adversary ambiguously decommitments to A^* during the k repetitions is negligible.

Next, note that the probability of generating collisions among the primes or to derive a composite number by running $\mathsf{PrimeGen}_d$ during the k repetitions is negligible by assumption. Hence, from now on we condition on the adversary opening her commitment A^* unambiguously and having k distinct primes output by $\mathsf{PrimeGen}_d$. By this, we only lose a negligible probability in total.

Now fix the random bits ω of the adversary. The tuple describing the execution up to the rewind point,

$$\mathsf{ExSoFar} = (N, \tau, t, g, G, h_0, h_1, w, \omega, x, M, A),$$

is called *good*, if the PIM adversary completes this commitment phase after receiving b and later reveals a valid decommitment (event Succ) with probability at least $\frac{1}{2}\epsilon(n)$. In other words, the adversary's choice y_p^* corresponds to some related message which the adversary would later open. Since all other parameters are fixed, the probability space now merely consists of the choice of b.

We claim that the probability of $\mathsf{ExSoFar}$ being good is at least $\frac{1}{2}\epsilon(n)$. This can been seen as follows. We can assume without loss of generality that $\pi_{\mathsf{open}}(\mathcal{A})$ is larger than $\epsilon(n)$ for the specific parameter n; else, even if the simulator completely fails to produce a related output, the simulator's success probability would be $\epsilon(n)$-close. But if we had $\mathrm{Prob}\left[\mathsf{ExSoFar}\text{ good}\right] < \frac{1}{2}\epsilon(n)$ then

$$\begin{aligned}\pi_{\mathsf{open}}(\mathcal{A}) &\leq \mathrm{Prob}\left[\mathsf{Succ} \mid \mathsf{ExSoFar}\text{ not good}\right] + \mathrm{Prob}\left[\mathsf{ExSoFar}\text{ good}\right] \\ &< \frac{1}{2}\epsilon(n) + \frac{1}{2}\epsilon(n) = \epsilon(n).\end{aligned}$$

This would contradict $\pi_{\mathsf{open}}(\mathcal{A}) > \epsilon(n)$, of course.

Given the prefix $\mathsf{ExSoFar}$ is good, $\mathcal{A}$ reveals a residue which matches a successful opening with probability of at least $\frac{1}{2}\epsilon$. For the number K of correct residues y_p^* among the k repetitions, it therefore holds

$$K \geq \frac{8(t+n+2)\epsilon^{-1}}{d-1},$$

except with probability $e^{-2\frac{(t+n+2)\epsilon^{-2}}{d-1}}$ by the Chernoff bound. Since the parameters are admissible, this probability is negligible.

Note that the correctness of a residue y_p^* merely assures that this value matches *some* valid decommitment of the adversary. In principle, all K correct residues could correspond to different adversarial decommitments. However, using an approach similar to the one above showing that the adversary essentially cannot open the commitment for A^* ambiguously under the factoring assumption, we can also prove that the adversary cannot open her commitment M^* ambiguously, except with negligible probability. For this execute the same protocol as above, but this time we also open the sender's commitment to see the adversarial decommitment to M^* in each repetition; if we find distinct valid decommitments we can compute a $2^{\tau+1}$th root of g and therefore factor the modulus. Hence, except with negligible probability, all K good residues correspond to a

single decommitment, and there must be at least one entry in the list output by Boneh's algorithm which is also a valid decommitment to M^*.

We have $\log_2 4B \leq t + n + 2$ for the $(t + n)$-bit number $(m^*_{\mathsf{pad}}, r^*)$, $\log_2 P \geq k(d-1) = 32(t+n+2)\epsilon^{-2}$ for the product of k distinct d-bit primes and $\frac{\log_2 p_k}{\log_2 p_1} \leq \frac{d}{d-1} \leq \sqrt{2}$ for $d \geq 4$. Hence, the prerequisites to apply Boneh's algorithm (Lemma 3) are satisfied:

$$\frac{K}{k} \geq \frac{\epsilon}{4} = \sqrt{\frac{2}{32\epsilon^{-2}}} \geq \sqrt{\frac{2 \cdot (t+n+2)}{32(t+n+2)\epsilon^{-2}}} \geq \frac{\log_2 p_k}{\log_2 p_1} \cdot \sqrt{\frac{\log_2 4B}{\log_2 P}}.$$

Overall, the extractor thus retrieves a decommitment m' except with probability negligibly close to $\frac{1}{2}\epsilon$. Taking an analogous approach as in the RSA case, one easily proves that the extracted message m' is also a suitable replacement for m^*. This too implies that, if Boneh's algorithm returns more than a single valid decommitment m'_{pad}, r' to M^*, then simply taking the first one is an admissible strategy.

Theorem 4. *Let t, n, d, ϵ be admissible parameters. If the factoring assumption holds and if H is a family of universal one-way hash functions, then the protocol in Figure 8 is a statistically-secret commitment scheme which is ϵ-non-malleable with respect to opening.*

Hash-and-Commit Paradigm

The a posteriori proof of knowledge may also be applied to the DLog and RSA based scheme. It allows to hash longer messages before committing. In this case, the sender $\mathcal{S}$ pads the ℓ-bit long message m by $2d$ random bits s. Then, $\mathcal{S}$ hashes $m_{\mathsf{pad},h} := h(m_{\mathsf{pad}}) := h(s + 2^{2d}m)$ using some collision-resistant function $h : \mathbb{Z}_{2^{\ell+2d}} \to \mathbb{Z}_{2^t}$ and commits to $M := g^{m_{\mathsf{pad},h}} r^{2^{\tau+t}}$. The response corresponds to the actual message m_{pad} rather than its hash value $y_p := m_{\mathsf{pad}} + 2^{2d+\ell} r \bmod p$. One can easily identify a right message by applying the hash function and seeing if the result matches the given commitment M.

6.4. *Construction Based on Polynomials*

A disadvantage of our a posteriori proof of knowledge is the generation of prime numbers. A faster approach can be based on polynomials over a finite fields. For this let q be a prime of bit size $\sqrt{d}$. For simplicity, assume $D := \sqrt{d}$ is an integer. Use $\psi_q(z) := \sum_{i \geq 0} z_i \chi^i \in \mathbb{Z}_q[\chi]$ to embed a non-negative integer $z = \sum_{i \geq 0} z_i q^i$ to the polynomial ring $\mathbb{Z}_q[\chi]$ over the field $\mathbb{Z}_q$.

For the proof of knowledge, both parties as before agree on a uniformly distributed $c \in \mathbb{Z}_{q^D}$ via the coin-flipping protocol. Then they use the polynomial

$$p(\chi) := \mathsf{PolyGen}_{q,D}(c) := \chi^D + \psi_q(c)$$

as the challenge for the proof of knowledge. The sender replies with

$$y_p(\chi) := \psi_q(m_{\mathsf{pad}} + 2^t r) \bmod p(\chi),$$

where $m_{\mathsf{pad}} = s + q^D m$ and $s \in_{\mathrm{R}} \mathbb{Z}_{q^D}$.

First, observe that $\mathsf{PolyGen}_{q,D}(c)$ is an injective mapping. Then, we remark that the Chinese Remainder Theorem holds for polynomial rings over fields, too (see Knuth [33, Sect. 4.6.2, Example 3]): Given $y_p(\chi)$ for pairwise co-prime polynomials $p(\chi)$, one can efficiently construct $\psi_q(m_{\mathsf{pad}} + 2^t r)$. Note that $\psi_q(s)$ (which is part of $\psi_q(m_{\mathsf{pad}} + 2^t r)$) is uniformly distributed in $\mathbb{Z}_q[\chi]/(p)$. Therefore, it serves as a real one-time pad, and as a result the commitment scheme achieves even perfect secrecy.

The main advantage of using polynomials is that random elements are more likely to be relatively prime than random integers: Two randomly chosen monic polynomials of the same degree over $\mathbb{Z}_q$ are relatively prime with overwhelming probability $1 - \frac{1}{q}$ (see [33, 4.6.5, Example 5]). Hence, even generating a large number of random polynomials is unlikely to result in polynomials with a non-trivial gcd.

On the downside, to best of our knowledge, CRT list-decoding algorithms do not work for polynomials in place of integers. Ideas how one could derive such a procedure for polynomials are described in a more general framework by Sudan [45]. Here we restrict the number of correct solutions to be collected to a constant such that searching all possible subsets of congruences can be done in polynomial-time.

We call parameters t, n, D, ϵ admissible if the all of the following properties hold. Suppose that $t \geq 2d$, that $\frac{t+n}{D}$ is constant and that $\frac{t+n}{D}\epsilon^{-1}$ is superlogarithmic. Assume further that any algorithm running in time $\mathrm{poly}(n) \cdot \epsilon^{-1}(n)$ for some polynomial $\mathrm{poly}(n)$ factors a random modulus with negligible probability only. Moreover, let the probability of $\mathsf{PolyGen}_{q,D}$ generating two polynomials which are not relatively prime in $\mathrm{poly}(n) \cdot \epsilon^{-1}$ independent executions be negligible. Then, we have

Theorem 5. *Let t, n, D, ϵ be admissible. If the factoring assumption holds and if H is a family of universal one-way hash functions, then the version of the protocol in Fig. 8 based on polynomials is a perfectly-secret commitment scheme which is ϵ-non-malleable with respect to opening.*

The proof is almost identical to the prime generation case. Only this time we let the extractor make $k := 4\frac{t+n}{D}\epsilon^{-1}(n)$ repetitions to get $K \geq \frac{t+n}{D}$ correct solutions, except with negligible probability $\exp(-2\frac{t+n}{D}\epsilon^{-1}(n))$. Then, we run Knuth's polynomial-based CRT algorithm for each subset of $\frac{t+n}{D}$ values among the k answers; we check each possible solution against the given commitment. This gives us the right pair in polynomial time.

Acknowledgements

We are indebted to Cynthia Dwork for discussions about non-malleability. We also thank the participants of the Luminy 1999 crypto workshop for stimulating discussions, as well as the Crypto 2000 reviewers and program committee, especially Shai Halevi. We are also grateful to Yehuda Lindell and to Jonathan Katz for informing us about their works. Finally, we thank Rosario Gennaro, Tal Rabin and Alon Rosen for discussions and pointing out a gap in the main proof, and Claus Schnorr for drawing our attention to problems with the previously given definitions of non-malleability in the proceedings version of our paper.

Most of this work was done while both authors were at the Johann Wolfgang Goethe-University Frankfurt. Part of the work of the first author supported by the Emmy Noether Program Fi 940/1-1 and Fi 940/2-1 of the German Research Foundation (DFG).

Appendix A. Non-malleable Commitments via Random Oracles

The random oracle methodology [3,21] exploits the very strong assumption that a hash function behaves like a truly random function. In this model, where all parties have oracle access to such a random function H, we devise non-interactive non-malleable commitments in the plain model. However, we remark that the random oracle heuristic provides only some evidence that the scheme is indeed secure if one uses appropriate instantiations for H. It might well be that there is no secure implementation in practice at all [9].

The definition of non-malleability transfers to the random oracle model if we augment each party $\mathcal{S}, \mathcal{R}, \mathcal{A}$ and $\mathcal{A}'$ as well as M and R with the same oracle H representing a random function with infinite domain and fixed output length. The probability that $\mathcal{A}$ and $\mathcal{A}'$, respectively, succeed is then taken over the random choice of H, too.

Let $\mathrm{Com}_{\mathrm{secret}}$ be the non-interactive statistically-secret commitment scheme described in [7,30,40]. The protocol goes like this: First, the sender hashes the message m to a short string m_h with some collision-intractable hash function. Then the sender picks a pairwise independent function h and a value x such that $h(x) = m_h$. It computes the hash value y of x under the collision-intractable hash function and sends (y, h) to $\mathcal{R}$. To decommit the sender reveals m and x.

Since the protocol $\mathrm{Com}_{\mathrm{secret}}$ merely requires a collision-intractable hash function and random oracles have this property by construction, we may use H as the collision-intractable hash function in the scheme. Then $\mathrm{Com}^{H}_{\mathrm{secret}}$ is indeed non-interactive and still provides statistical secrecy as well as computational unambiguity. We claim that this scheme is even non-malleable with respect to opening in the random oracle model.

Basically, the protocol is non-malleable because any adversary $\mathcal{A}$ sending a commitment (y^*, h^*) and later a correct decommitment (m^*, x^*), each time after having seen the sender's values (y, h) and (m, x), must have obtained the answers $m^*_h = H(m^*)$ and $y^* = H(x^*)$ from the oracle queries to H. Otherwise, the probability that $\mathcal{A}$ finds a good decommitment is negligible because predicting H on a new value is infeasible. But then $\mathcal{A}$ already "knows" a related message m^* to m in the commitment phase, contradicting the secrecy.

It is now easy to formalize the intuition and define the simulator. $\mathcal{A}'$ first sends a dummy commitment (y, h) on behalf of the sender to $\mathcal{A}$, say, by committing to the all-zero string. Then it records all queries of $\mathcal{A}$ to oracle H and the answers—this is possible as $\mathcal{A}'$ simulates $\mathcal{A}$ and sees all queries of $\mathcal{A}$ before forwarding it to H. Since all hash values of H are distinct with overwhelming probability, we may assume that every image has a unique pre-image in the list of recorded pairs. If finally $\mathcal{A}$ sends some commitment (y^*, h^*) then the simulator looks up y^* in the list and obtains the corresponding query x^* yielding y^*. This gives the unique $m^*_h = h^*(x^*)$ and another search reveals the pre-image m^* of m^*_h under H. Let $\mathcal{A}$' output $m' := m^*$. Clearly, the probability that $\mathcal{A}'$ returns a related message is negligibly close to $\mathcal{A}$'s success probability.

References

[1] B. Barak, Constant-round coin-tossing with a man in the middle or realizing the shared random string model, in *Proceedings of 43rd IEEE Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Los Alamitos, 2002)

[2] M. Bellare, O. Goldreich, On defining proofs of knowledge, in *Advances in Cryptology, Proceedings Crypto '92*. Lecture Notes in Computer Science, vol. 740 (Springer, Berlin, 1993), pp. 390–420

[3] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *First ACM Conference on Computer and Communication Security* (ACM, New York, 1993), pp. 62–73

[4] M. Bellare, P. Rogaway, Optimal asymmetric encryption, in *Advances in Cryptology, Proceedings Eurocrypt '94*. Lecture Notes in Computer Science, vol. 950 (Springer, Berlin, 1993), pp. 92–111

[5] D. Boneh, Finding smooth integers using CRT decoding, in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 2000)

[6] G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.* **37**(2), 156–189 (1988)

[7] I. Damgård, T. Pedersen, B. Pfitzmann, On the existence of statistically hiding bit commitment schemes and fail-stop signatures, in *Advances in Cryptology, Proceedings Crypto '93*. Lecture Notes in Computer Science, vol. 773 (Springer, Berlin, 1993), pp. 250–265

[8] R. Canetti, M. Fischlin, Universally composable commitments, in *Advances in Cryptology, Proceedings Crypto 2001*. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 19–40

[9] R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited, in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 1998), pp. 209–218

[10] R. Canetti, O. Goldreich, S. Goldwasser, S. Micali, Resettable zero-knowledge, in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 2000)

[11] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally composable two-party and multi-party secure computation, in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 2002), pp. 459–503

[12] R. Cramer, V. Shoup, Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur. (ACM TISSEC)* **3**(3), 161–185 (2000)

[13] R. Cramer, V. Shoup, Design and analysis of practical public key cryptosystem provable secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**, 167–226 (2003)

[14] I. Damgård, J. Groth, Non-interactive and reusable non-malleable commitment schemes, in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 2003), pp. 426–437

[15] I. Damgård, J. Nielsen, Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor, in *Advances in Cryptology, Crypto 2002*. Lecture Notes in Computer Science, vol. 2442 (Springer, Berlin, 2002), pp. 581–596

[16] G. Di Crescenzo, Y. Ishai, R. Ostrovsky, Non-interactive and non-malleable commitment, in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)* (ACM, New York, 1998), pp. 141–150

[17] G. Di Crescenzo, J. Katz, R. Ostrovsky, A. Smith, Efficient and non-interactive non-malleable commitment, in *Advances in Cryptology, Proceedings Eurocrypt 2001*. Lecture Notes in Computer Science, vol. 2045 (Springer, Berlin, 2001), pp. 40–59

[18] D. Dolev, C. Dwork, M. Naor, Non-malleable cryptography. *SIAM J. Comput.* **30**(2), 391–437 (2000)

[19] U. Feige, A. Shamir, Zero-knowledge proofs in two rounds, in *Advances in Cryptology, Proceedings of Crypto '89*. Lecture Notes in Computer Science, vol. 435 (Springer, Berlin, 1990), pp. 526–544

[20] U. Feige, A. Fiat, A. Shamir, Zero-knowledge proofs of identity. *J. Cryptol.* **1**(2), 77–94 (1988)

[21] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature schemes, in *Advances in Cryptology, Crypto '86*. Lecture Notes in Computer Science, vol. 263 (Springer, Berlin, 1986), pp. 186–194

[22] A. Fiat, A. Shamir, Witness indistinguishable and witness hiding protocols, in *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing (STOC)* (ACM, New York, 1990), pp. 416–426

[23] M. Fischlin, Completely non-malleable schemes, in *ICALP 2005*. Lecture Notes in Computer Science, vol. 3580 (Springer, Berlin, 2005), pp. 779–790

[24] M. Fischlin, R. Fischlin, Efficient non-malleable commitment schemes, in *Advances in Cryptology, Crypto 2000*. Lecture Notes in Computer Science, vol. 1880 (Springer, Berlin, 2000), pp. 414–432
[25] M. Fischlin, R. Fischlin, The representation problem based on factoring, in *RSA Cryptographer's Track 2002*. Lecture Notes in Computer Science, vol. 2271 (Springer, Berlin, 2002), pp. 96–113
[26] R. Gennaro, Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks, in *Advances in Cryptology, Proceedings Crypto 2004*. Lecture Notes in Computer Science, vol. 3152 (Springer, Berlin, 2004), pp. 220–236
[27] O. Goldreich, Foundations of Cryptography, Fragments of a Book, Version 2.03 (1998)
[28] O. Goldreich, Y. Lindell, Session-key generation using human passwords only, in *Advances in Cryptology, Proceedings Crypto 2001*. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 408–432
[29] V. Guruswami, A. Sahai, M. Sudhan, "Soft-decision" decoding of Chinese remainder theorem, in *Proceedings of 41st IEEE Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Los Alamitos, 2000)
[30] S. Halevi, S. Micali, Practical and provably-secure commitment schemes from collision-free hashing, in *Advances in Cryptology, Proceedings Crypto '96*. Lecture Notes in Computer Science, vol. 1109 (Springer, Berlin, 1996), pp. 201–215
[31] J. Håstad, R. Impagliazzo, L.A. Levin, M. Luby, A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)
[32] R. Impagliazzo, M. Luby, One-way functions are essential for complexity based cryptography, in *Proceedings of 30th IEEE Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Los Alamitos, 1989), pp. 230–235
[33] D.E. Knuth, *Seminumerical Algorithms*, 3rd edn. The Art of Computer Programming, vol. 2 (Addison–Wesley, Reading, 1998)
[34] A. Lenstra, E. Verheul, Selecting cryptographic key sizes. *J. Cryptol.* **14**, 255–293 (2001)
[35] M. Liskov, A. Lysyanskaya, S. Micali, L. Reyzin, A. Smith, Mutually independent commitments, in *Advances in Cryptology, Asiacrypt 2001*. Lecture Notes in Computer Science, vol. 2248 (Springer, Berlin, 2001), pp. 385–401
[36] P. MacKenzie, K. Yang, On simulation-sound trapdoor commitments, in *Advances in Cryptology, Proceedings Eurocrypt 2004*. Lecture Notes in Computer Science, vol. 3027 (Springer, Berlin, 2004), pp. 382–400
[37] U. Maurer, Fast generation of prime numbers and secure public-key cryptographic parameters. *J. Cryptol.* **8**, 123–155 (1995)
[38] S. Micali, M. Rabin, S. Vadhan, Verifiable random functions, in *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Los Alamitos, 1999), pp. 120–130
[39] M. Naor, Bit commitment using pseudo-randomness. *J. Cryptol.* **4**, 151–158 (1991)
[40] M. Naor, M. Yung, Universal one-way hash functions and their cryptographic applications, in *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing (STOC)* (1989), pp. 33–43
[41] T. Okamoto, Provable secure and practical identification schemes and corresponding signature schemes, in *Advances in Cryptology, Proceedings Crypto '92*. Lecture Notes in Computer Science, vol. 740 (Springer, Berlin, 1993), pp. 31–53
[42] R. Pass, A. Rosen, Concurrent non-malleable commitments, in *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society Press, Los Alamitos, 2005), pp. 563–572
[43] T.P. Pedersen, Non-interactive and information-theoretical secure verifiable secret sharing, in *Crypto '91*. Lecture Notes in Computer Science, vol. 576 (Springer, Berlin, 1991), pp. 129–140
[44] M. Prabhakaran, A. Sahai, New notions of security: achieving universal composability without trusted setup, in *Proceedings of the Annual ACM Symposium on the Theory of Computing (STOC)* (2004), pp. 242–251
[45] M. Sudan, Ideal error-correcting codes: unifying algebraic and number-theoretic algorithms, in *Proceedings of the 14th Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting (AAECC-14)*. Lecture Notes in Computer Science, vol. 2227 (Springer, Berlin, 2001), pp. 36–45