Integer Commitments, Old and New Tools

Iftach Haitner[*] Yehuda Lindell [†] Nikolaos Makriyannis[‡]

January 19, 2025

Abstract

This self-contained and detailed tutorial covers RSA-based integer commitments and related protocols. It also presents a new, highly efficient setup protocol for sampling commitment parameters.

Contents

1 Introduction **3**

2 Preliminaries **4**

2.1 Notation . . . 4

2.2 Sigma Protocols . . . 5

2.3 Quadratic Residuosity for Primes and Biprimes . . . 7

2.3.1 Basic Notation . . . 7

2.3.2 Jacobi Symbol . . . 7

2.3.3 The Group $\mathcal{QR}_n$ . . . 8

2.3.4 Structure of $\mathbb{Z}_n^*$ . . . 9

2.4 The Strong RSA Assumption . . . 10

2.4.1 Two Hard Games . . . 10

3 The Integer Commitment Scheme **13**

3.1 Security . . . 14

3.2 Parameters Generation . . . 15

4 Range Proof based on Strong RSA **16**

5 Equality Proof **19**

5.1 Equality with Group Commitments in Rough Groups . . . 20

5.2 Equality with Pedersen Commitments in Known-Order Rough Groups . . . 21

5.3 Equality with Paillier Encryption . . . 22

5.3.1 Paillier Ciphertexts Divisors . . . 23

[*]Stellar Development Foundation and Tel Aviv University. E-mail: `iftachh@gmail.com`.. Part of this work was conducted while at Coinbase.

[†]Coinbase. E-mail: `yehuda.lindell@coinbase.com`..

[‡]Fireblocks. E-mail: `nikos@fireblocks.com`.

5.3.2 Proving Theorem 5.8 . . . 24
5.4 Applications . . . 25
5.4.1 Range Proof for Rough Groups . . . 25
5.4.2 Equality Proof between Rough Groups . . . 25
5.4.3 Commit-and-Prove . . . 25

6 Lightweight Leaky Well-Formedness Proof **26**
6.1 Commitments over Rough Integer Rings . . . 27

1 Introduction

(RSA-based) Integer commitments, introduced by Fujisaki and Okamoto [FO97a] and Damgård and Fujisaki [DF02], are Pedersen-like additively homomorphic commitments that are binding over the integers (as opposed to Pedersen commitments, which are only binding over the underlying finite ring or field). Integer commitments also have a highly efficient *range proof* protocol—a protocol for proving the knowledge of an opening of the commitment for a value within a specified range. Having an efficient range proof is what makes integer commitments so useful. Specifically, they are used for implementing efficient range proofs for other types of additively homomorphic commitments: Pedersen and ElGamal commitments over known-order groups, and the commitment induced by the Paillier public-key encryption scheme.

A downside of using integer commitments is that they are parameterized by values that have to be sampled properly. Specifically, integer commitments are parametrized by a tuple (n, g, h), where n denotes a suitable RSA modulus and g, h are carefully chosen elements generating the same subgroup in $\mathbb{Z}_n^*$. To commit to an integer integer $m \in \mathbb{Z}$, one computes $c = g^m h^r \bmod n$ where r is a random value chosen from a suitable domain. To decommit, it suffices to reveal m and r. For the binding property to hold, the parameters must not be chosen by the committer,[1] while for the hiding property, the parameters must be well-formed so that g lies in the multiplicative group generated by h. Unfortunately, no highly efficient protocol is known for proving well-formedness. As a result, when using integer commitments, one must either rely on a trusted setup phase or use a rather inefficient protocol (where communication and computation scales linearly with the computational security parameter).

In this work, we give an overview with detailed, self-contained proofs of the basic facts about integer commitments and of a range proof protocol. We then highlight the usefulness of integer commitments—and in particular, of having an efficient range proof protocol—by discussing some standard applications. In addition, we introduce what we believe is the first highly efficient parameter well-formedness protocol. The resulting protocol guarantees that the quotient group $\langle g, h\rangle / \langle h\rangle$ is small.[2] Although it does not guarantee perfect hiding, the resulting hiding property is sufficient for many applications.

What this paper does not cover. This paper does not aim to cover every known aspect of integer commitments or their related protocols and extensions. Rather, this paper is intended as a detailed and self-contained tutorial for using RSA-based integer commitments and to showcase a novel parameter well-formdness proof. In particular, we do not discuss commitments based on non-RSA assumptions (e.g., class groups [CKLR21]) or formally present the four-square decomposition technique [Lip03] for *tight* range proofs. The reader is referred to [CBCMRW24] for a more detailed survey on range proofs. We will conclude with further discussion on these and related range-proof protocols.

Related work

As we shall see in the subsequent technical sections, the range proofs [FO97a; Bou00; DF02] presented herein exhibit some *slackness*. In detail, when proving that a committed value m lies in

[1] If the committer knows the factorization of n or a discrete-log relation between g and h, the binding property can be easily broken.

[2] The key takeaway is that g can be written as $\sigma h'$, where $h' \in \langle h\rangle$ and σ has small order.

some range $[-a, a]$, it is possible for a cheating prover to make the verifier accept witnesses from a slightly larger set $[-a \cdot 2^{\varepsilon}, a \cdot 2^{\varepsilon}]$ where ε is some value known to all. This slackness does not affect many applications of interest, e.g., threshold ECDSA [Lin17; LN18; GG18; CGGMP20] (and removing it incurs substantial overhead, as discussed below), so we treat it as a quirk of the proofs.

The four-square decomposition technique. Starting with Lipmaa [Lip03] (and later refined by Groth [Gro05] and Couteau, Peters, and Pointcheval [CPP17]), the four-square decomposition technique for proving $m \in [a, b]$ involves showing that both $m - a$ and $-m + b$ are nonnegative. Specifically, by Lagrange's theorem, any nonnegative integer can be written as a sum of four squares, and there are known efficient algorithms for finding this decomposition. The last step involves proving, for example, that $m - a = a_0^2 + a_1^2 + a_2^2 + a_3^2$ by committing to each a_i separately and using Schnorr-style protocols. However, the overhead of finding the square decomposition and making these additional commitments can be quite costly.

The bit-decomposition technique. Alternatively, one can avoid integer commitments entirely by committing to the bit (or u-ary) decomposition of m, i.e., $m = \sum_j m_j 2^j$, and using a bit (or u-ary) commitment scheme [BCDv88; CCs08; Gro11]. General zero-knowledge techniques—such as bulletproofs [BBBPWM18]—are then used to show that the committed integer lies within the desired range. However, this approach inherently results in super-constant proof sizes (which scale linearly with the witness size) or incurs the overhead of using general-purpose SNARKs like Groth [Gro16].

On [CKLR21] and SHARP [CGKR22]. In conclusion, we mention the work of Couteau, Klooß, Lin, and Reichle [CKLR21] and its follow-up [CGKR22], which introduce so-called bounded integer commitments that behave like integer commitments but operate on rationals rather than integers. They can be instantiated in various groups (including known-order groups) in combination with the square-decomposition technique to obtain tight range proofs. In [CGKR22], the authors show that these proofs achieve performance comparable to the bulletproofs approach.

Paper Organization

A rather detailed preliminaries section is provided in Section 2. The integer commitment scheme is described and analyzed in Section 3, and the strong-RSA-based range proof is described and analyzed in Section 4. In Section 5, we use the range proof protocol for constructing *equality-proof* protocols for many additively homomorphic commitment schemes of interest. We also present some applications of these proofs. Our new lightweight setup protocol and its security guarantees are given in Section 6.

2 Preliminaries

2.1 Notation

We use calligraphic letters to denote sets, uppercase for random variables, and lowercase for integers and functions. All logarithms considered here are base 2. Let $\mathbb{N}$ denote the set of natural numbers. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$ and $(n) := \{0, \ldots, n\}$. For $n \in \mathbb{N}$, let $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ denote the ring of integers modulo n.

All distributions considered are finite. For a set $\mathcal{X}$, let $\sim \mathcal{X}$ denote the uniform distribution over $\mathcal{X}$, and let $x \overset{\mathrm{R}}{\leftarrow} \mathcal{X}$ stand for $x \overset{\mathrm{R}}{\leftarrow} \sim \mathcal{X}$, i.e., the process of sampling x uniformly over $\mathcal{X}$. The support of a distribution X over a discrete set $\mathcal{X}$, is defined by $\mathrm{Supp}(X) := \{x \in \mathcal{X} : X(x) > 0\}$. For two distributions/random variables A, B, let $\mathrm{SD}(A, B)$ denote their *statistical distance.* We let $\mathrm{neg}(\kappa)$ denote an arbitrary function in κ_{c}.

Security parameters. Throughout, we use two security parameters: κ_{s} used for quantities (functions of the security parameter) that are robust to the adversary running time, e.g., statistical distance, and κ_{c} used for quantities that are sensitive to the adversary running time, e.g., computational distance. In practice, this distinction allows the protocols to use much smaller value for κ_{s} (e.g., 64) than for κ_{c}, e.g., 128, leading to more efficient protocols.

To avoid notational clutter, when defining languages and promise problems, defined further below, we consider a single global security parameter κ. (Of course, all these definitions can be parameterized by different values of the security parameter.)

2.2 Sigma Protocols

The zero-knowledge protocols presented in this paper are variants of *Sigma protocols.*[3] We start with the basic definition due to Damgard [Dam10].

Definition 2.1 (Sigma protocols). *A* Sigma protocol (P, V) *for an* NP *relation* $\mathcal{R}$*, is a three-message, public-coin, perfectly correct, honest-verifier statistical zero-knowledge protocol with special soundness for* $\mathcal{R}$*. (The statistical zero-knowledge error is a function of the security parameter.)*

Correctness: *For any* $\kappa \in \mathbb{N}$ *:* (P, V) *is a perfectly correct protocol for* $\mathcal{R}$*.*

Zero knowldege: *There exist a an efficient* Sim *such that for any* $\kappa \in \mathbb{N}$*:* (P, V) *is honest-verifier statistical zero-knowledge protocol for* $\mathcal{R}$*, and it is realized by* Sim*.*

Special soudness. *There exist an efficient* Ext *such that for any* $\kappa \in \mathbb{N}$ *and any* $\widehat{\mathsf{P}}$*:*

$$\Pr_{(c,\alpha,\beta_0,\beta_1,\gamma_0,\gamma_1) \overset{R}{\leftarrow} \widehat{\mathsf{P}}}\left[\begin{array}{c}\beta_0 \neq \beta_1 \\ \forall j \in \{0,1\}\colon \mathsf{V}(t_j \leftarrow (c,\alpha,\beta_j,\gamma_j)) = \mathrm{true} \\ (c, \mathsf{Ext}(t_0,t_1)) \notin \mathcal{R}\end{array}\right] \leq \mathrm{neg}(\kappa),$$

An extended Sigma protocol *has a* preamble stage *that is independent of the parties' input. The special soundness extractor is given the prover's view in this stage.*

Computational variant. The special soundness property of Sigma protocols is naturally relaxed to the computational settings, referred to as computational special soundness, in which it is only required to hold for efficiently generated transcripts.

Definition 2.2 (Computational Sigma protocols). *A* computational Sigma protocol (P, V) *for* $\mathcal{R}$ *is a Sigma protocol for* $\mathcal{R}$ *but with the special soundness requirement replaced with* computational special soundness*: there exists an efficient* Ext *such that for any efficient* $\widehat{\mathsf{P}}$*:*

$$\Pr_{(c,\alpha,\beta_0,\beta_1,\gamma_0,\gamma_1) \overset{R}{\leftarrow} \widehat{\mathsf{P}}}\left[\begin{array}{c}\beta_0 \neq \beta_1 \\ \forall j \in \{0,1\}\colon \mathsf{V}(t_j \leftarrow (c,\alpha,\beta_j,\gamma_j)) = \mathrm{true} \\ (c, \mathsf{Ext}(t_0,t_1)) \notin \mathcal{R}\end{array}\right] \leq \mathrm{neg}(\kappa),$$

[3]We use the sometimes non-standard Sigma protocol formulations given below, since it is more informative than just stating the security of the different protocols are proof-of-knowledge, honest-verifier statistical zero knowledge.

all above algorithms are getting 1^κ as a parameter. If (P, V) *is an extended Sigma protocol, then* $\widehat{\mathsf{P}}$ *and* V *interact in the preamble stage before* $\widehat{\mathsf{P}}$ *outputs* $(c, \alpha, \beta_0, \beta_1, \gamma_0, \gamma_1)$*.*

Computational special soundness immediately implies a proof of knowledge extractor: after the first message is sent, use rewinding to generate two accepting transcripts and apply the computational special soundness extractor.

Distributional variant. In some cases we do not know how to construct a (information theoretic or computational) Sigma protocol that is sound for *any* instance. Rather, we relax the special soundness property to hold only for "typical" instances. For concreteness, we only focus on the computational variant.

Definition 2.3 (Computational Sigma protocols with respect to distributions)**.** *A three-message, public-coin protocol* (P, V) *is a* computational Sigma protocol with respect to a family of relations $\{\mathcal{R}_p\}_p$ *and distribution ensemble* $\{P_\kappa\}_\kappa$*, if the following hold:*

Correctness: *For any* $\kappa \in \mathbb{N}$ *and* $p \in \mathrm{Supp}(P_\kappa)$*:* $(\mathsf{P}_p, \mathsf{V}_p)$*, i.e., p is given as an additional parameter to the parties, is perfectly correct protocol for* $\mathcal{R}_p$*.*

Zero knowldege: *There exist a an efficient* Sim *such that for any* $\kappa \in \mathbb{N}$ *and* $p \in \mathrm{Supp}(P_\kappa)$*:* $(\mathsf{P}_p, \mathsf{V}_p)$ *is honest-verifier statistical zero-knowledge protocol for* $\mathcal{R}_p$*, and it is realized by* Sim_p*.*

Computational special soudness. *There exist a an efficient* Ext *such that for any* $\kappa \in \mathbb{N}$*, with save but negligible probability over* $p \overset{R}{\leftarrow} P_\kappa$*:* $(\mathsf{P}_p, \mathsf{V}_p)$ *has computational special soundness, and it is realized by* Ext_p*.*

Promise problems. We consider Sigma protocols for *promise problems.* A promise problem is a union of two distinct sets, the "Yes instances" and the "No instances". Throughout, we define a promise problem $\Pi = \Pi^{\mathsf{Yes}} \cup \Pi^{\mathsf{No}}$ by defining Π^{Yes} and a "Slack subset" $\Pi^{\mathsf{Slack}} \supseteq \Pi^{\mathsf{Yes}}$. The No instances Π^{No} are then implicitly defined by $\neg\Pi^{\mathsf{Slack}}$. For instance, the Yes instance can be $\Pi^{\mathsf{Yes}} = \{(A, a) \colon A = a \cdot G \colon a \in [100]\}$, where G is an (additive) group generator, and the Slack instances can be $\Pi^{\mathsf{Slack}} = \{(A, a) \colon A = a \cdot G \colon a \in [1000]\}$.

Sigma protocols are naturally generalized to promise problems as follows.

Definition 2.4 (Sigma protocols for promise problems)**.** A Sigma protocol for a promise problem $\Pi = \Pi^{\mathsf{Yes}} \cup \Pi^{\mathsf{No}}$ *is a three-message, public-coin, where*

1. *Completeness and honest-verifier zero knowledge only guaranteed to hold for* $x \in \Pi^{\mathsf{Yes}}$*.*
2. *The knowledge extractor is guaranteed to output w with* $(c, w) \notin \Pi^{\mathsf{No}}$*.*

Computational Sigma protocols for promise problems, also with respect to distributions, are naturally defined.

For instance, given a Sigma protocol for the promise problem Π we considered above, we are guaranteed that the extractor outputs a "not too large" discrete log of A. While this is a weaker security grantees, this slackness sometimes allows simpler/more efficient protocols. A second advantage of using promise problems is a not handling of bad "parameters", e.g., weak public keys. See Sections 4 and 5.3.

2.3 Quadratic Residuosity for Primes and Biprimes

2.3.1 Basic Notation

Let $\mathcal{P}$ be the set of all primes. A prime number $p \in \mathrm{P}$ is *safe* if $p = 2p' + 1$ for some prime $p' > 2$. (Note that if p is a safe prime then $p = 3 \bmod 4$, since if $p = 1 \bmod 4$ then $p = 4k + 1$ and so $p' = 2k$ which is impossible). A biprime is *safe*, or a *strong RSA modulus*, if it is the product of two distinct safe primes.

Let $(x)_p := x \bmod p$. Note that $(x \cdot y)_p = (x)_p \cdot (y)_p \bmod p$. For a biprime $n = pq$, we use the "CRT function" $\mathsf{crt}\colon \mathbb{Z}_n \mapsto \mathbb{Z}_p \times \mathbb{Z}_q$ defined by $\mathsf{crt}(x) = ((x)_p, (x)_q)$. It is well known that crt is a bijection from $\mathbb{Z}_n^*$ to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ (and also from $\mathbb{Z}_n$ to $\mathbb{Z}_p \times \mathbb{Z}_q$). A value $x \in \mathbb{N}$ is a quadratic residue modulo p if there exists a $y \in \mathbb{N}$ such that $x = y^2 \bmod p$. We denote the set of quadratic residues modulo p by $\mathcal{QR}_p$. That is, $\mathcal{QR}_n := \left\{y \in \mathbb{Z}_n^*\colon \exists x \in \mathbb{Z}_n^* \text{ s.t. } x^2 = y \bmod n\right\}$. We denote $-\mathcal{QR}_n = \{-x\colon x \in \mathcal{QR}_n\}$ and $\pm\mathcal{QR}_n = \mathcal{QR}_n \cup -\mathcal{QR}_n$.

2.3.2 Jacobi Symbol

The Jacobi symbol for primes and biprimes is defined as follows.

Definition 2.5 (Jacobi symbol)**.** *The Jacobi symbol of x modulo $p \in \mathrm{P}$ is defined by*

$$\mathcal{J}_p(x) := \begin{cases} +1 & \textit{if } x \textit{ is a quadratic residue modulo } p \\ -1 & \textit{if } x \textit{ is not a quadratic residue modulo } p. \end{cases}$$

The Jaccobi symbol modulo a biprime $n = p \cdot q$, is defined by $\mathcal{J}_n(x) = \mathcal{J}_p(x) \cdot \mathcal{J}_q(x)$.

It is well-known that for any prime p: $\mathcal{J}_p(x) = 1$ iff $x \in \mathcal{QR} + p$. But modulo a composite, the Jacobi symbol is *not* an indication of quadratic residuosity: it is true that every quadratic residue has Jacobi symbol 1, but there are also an equal number of non-quadratic residues with Jacobi symbol 1. Yet, the following facts hold:

Proposition 2.6 (Properties of the Jacobi symbol for biprimes)**.** *For any biprime $n = p \cdot q$ with $p, q > 2$:*

1. $\mathcal{J}_n(x \cdot y) = \mathcal{J}_n(x) \cdot \mathcal{J}_n(y)$.
2. $\Pr_{x \xleftarrow{R} \mathbb{Z}_p^*}[\mathcal{J}_p(x) = 1] = 1/2$.
3. *If $p = q = 3 \bmod 4$, then for any $x \in \mathbb{N}$:*
   (a) $\mathcal{J}_p(x) = -\mathcal{J}_p(-x)$ *and* $\mathcal{J}_q(x) = -\mathcal{J}_q(-x)$.
   (b) $\mathcal{J}_n(x) = \mathcal{J}_n(-x)$.
4. *If $p = q = 3 \bmod 4$, then for $u, v \in \mathbb{Z}_n^*$ with $u^2 = v^2 \bmod n$ and $\mathcal{J}_n(u) \neq \mathcal{J}_n(v)$, it holds that* $\gcd((u + v) \cdot (u - v), n) > 1$.

Proof.

1. Since $\mathcal{J}_p(x) = x^{\frac{p-1}{2}}$ it follows that $\mathcal{J}_p(x \cdot y) = \mathcal{J}_p(x) \cdot \mathcal{J}_p(y)$. By definition, $\mathcal{J}_n(x \cdot y) = \mathcal{J}_p(x \cdot y) \cdot \mathcal{J}_q(x \cdot y)$. Thus, $\mathcal{J}_n(x \cdot y) = \mathcal{J}_p(x) \cdot \mathcal{J}_p(y) \cdot \mathcal{J}_q(x) \cdot \mathcal{J}_q(y) = (\mathcal{J}_p(x) \cdot \mathcal{J}_q(x)) \cdot (\mathcal{J}_p(y) \cdot \mathcal{J}_q(y)) = \mathcal{J}_n(x) \cdot \mathcal{J}_n(y)$.

2. This holds if half of the elements of $\mathbb{Z}_p^*$ are quadratic residues. Since $\mathbb{Z}_p^*$ is a cyclic group, it has a generator g. Thus, $1, g^2, g^4, \ldots, g^{p-1}$ are all quadratic residues, implying that there are at least $\frac{p-1}{2}$ quadratic residues. Define the polynomial $f(x) = x^{\frac{p-1}{2}} - 1 \bmod p$. Each root of the polynomial f is a quadratic residue (since a root x is such that $x^{\frac{p-1}{2}} = 1 \bmod p$ and by Proposition 2.7 this implies that it is a quadratic residue). Now, a degree $\frac{p-1}{2}$ polynomial has at most $\frac{p-1}{2}$ roots. We conclude that there are *exactly* $\frac{p-1}{2}$ quadratic residues, implying that exactly half the elements of $\mathbb{Z}_p^*$ are quadratic residues.

3.

   (a) Since $p = 3 \bmod 4$, there exists a value $k \in \mathbb{N}$ such that $p = 4k+3$. Thus $(-1)^{\frac{p-1}{2}} = (-1)^{2k+1} = -1$. By Proposition 2.7, $\mathcal{J}_p(-x) = (-x)^{\frac{p-1}{2}} = (-1)^{\frac{p-1}{2}} \cdot x^{\frac{p-1}{2}} = -\left(x^{\frac{p-1}{2}}\right) = -\mathcal{J}_p(x)$.

   (b) Compute $\mathcal{J}_n(x) = \mathcal{J}_p(x) \cdot \mathcal{J}_q(x) = \mathcal{J}_p(-x) \cdot \mathcal{J}_q(-x) = \mathcal{J}_n(-x)$.

4. $u^2 - v^2 = (u+v) \cdot (u-v) = 0 \bmod n$. By Item 3, $u \neq -v$ (since if $u = -v \bmod n$ then it would follow that $\mathcal{J}_n(u) = \mathcal{J}_n(v)$). Now, $N \mid (u+v) \cdot (u-v)$ and thus $p \mid (u+v) \cdot (u-v)$. Since p is prime, $p \mid (u+v)$ or $p \mid (u-v)$; assume $p \mid (u+v)$; the proof for the other case is similar. If $q \mid (u+v)$ as well, then $N \mid (u+v)$, but this implies that $u+v = 0 \bmod n$ and so $u = -v \bmod n$. Thus, $p \mid (u+v)$ and $q \not| (u+v)$ implying that $\gcd(N, u+v) = p > 1$.

□

The Jacobi symbol can be easily computed.

Proposition 2.7 (Computing the Jacobi symbol). *Let $p > 2$ be a prime. Then, $\mathcal{J}_p(x) = x^{\frac{p-1}{2}} \bmod p$.*

Proof. Since p is a prime, it follows that $\mathbb{Z}_p^*$ is a cyclic group; let g be a generator of the group. If x is a quadratic residue, then $x = g^i$ for some even $i = 2j$. Using the fact that $z^{p-1} \bmod p = 1$ for every z, we have

$$x^{\frac{p-1}{2}} = \left(g^{2j}\right)^{\frac{p-1}{2}} = g^{(p-1)\cdot j} = 1^j = 1 \bmod p.$$

If x is not a quadratic residue, then $x = g^i$ for some odd $i = 2j+1$. Thus,

$$x^{\frac{p-1}{2}} = \left(g^{2j+1}\right)^{\frac{p-1}{2}} = \left(g^{2j}\right)^{\frac{p-1}{2}} \cdot g^{\frac{p-1}{2}} = g^{\frac{p-1}{2}} \bmod p.$$

Since $\left(g^{\frac{p-1}{2}}\right)^2 = g^{p-1} = 1 \bmod p$ we have that $g^{\frac{p-1}{2}}$ is a square root of 1. The only square roots of 1 are $1, -1$. Now, since g is a generator, it follows that $g^{\frac{p-1}{2}} \neq 1 \bmod p$ and thus $x^{\frac{p-1}{2}} = g^{\frac{p-1}{2}} = -1$.

□

2.3.3 The Group $\mathcal{QR}_n$

It is easy to verify that $\mathcal{QR}_n$ is a multiplicative group, since the product of two quadratic residues is a quadratic residue. We make use of the following observation that an element x of $\mathbb{Z}_n^*$ is a quadratic residue relative to n if and only if x is a quadratic residue relative to both p and q.

Proposition 2.8 (Structure of $\mathcal{QR}_n$)**.** *For any biprime $n = p \cdot q$ and $x \in \mathbb{Z}_n^*$: $x \in \mathcal{QR}_n$ iff $\mathsf{crt}(x) \in \mathcal{QR}_p \times \mathcal{QR}_q$.*

Proof. Let $x_p = x \bmod p$ and $x_q = x \bmod q$.

Assume that $x \in \mathcal{QR}_n$. Let $y \in \mathbb{Z}_n^*$ be such that $x = y^2 \bmod n$. Since $n \mid x - y^2$ it holds that both $p \mid x - y^2$ and $q \mid x - y^2$ and $x = y^2 \bmod p$ and $x = y^2 \bmod q$, implying that $x_p = {y_p}^2 \bmod p$ and $x_q = {y_q}^2 \bmod p$.

Assume $x_p \in \mathcal{QR}_p$ and $x_q \in \mathcal{QR}_q$. Let y_p and y_q be such that $x = {y_p}^2 \bmod p$ and $x = {y_q}^2 \bmod q$. By the CRT, there exists a unique $y \in \mathbb{Z}_n$ such that $y = y_p \bmod p$ and $y = y_q \bmod q$. Since $y^2 = x \bmod p$ and, $y^2 = x \bmod q$ and since p, q are prime (and so relatively prime), it follows that $y^2 = x \bmod n$, as required. □

Proposition 2.9. *For every $x \in \mathcal{QR}_n$ there are four square roots, two of which have Jacobi symbol 1 and two of which have Jacobi symbol -1.*

Proof. By Proposition 2.8, it holds that if $x \in \mathcal{QR}_n$ then $x_p \in \mathcal{QR}_p$ and $x_q \in \mathcal{QR}_q$, where $x_p = x \bmod p$ and $x_q = x \bmod q$. Let $a, -a$ be the square roots of x in $\mathbb{Z}_p^*$, and let $b, -b$ be the square roots of x in $\mathbb{Z}_p^*$. By the Chinese remainder theorem, the combinations $(a, b), (-a, -b), (a, -b), (-a, b)$ map to 4 different elements in $\mathbb{Z}_n$; denote them y_1, y_2, y_3, y_4. Furthermore, by Proposition 2.8, each element is a square root and it holds that $(a, b) = \mathsf{crt}(y_1)$, $(-a, -b) = \mathsf{crt}(y_2)$, $(a, -b) = \mathsf{crt}(y_3)$, and $(-a, b) = \mathsf{crt}(y_4)$. Clearly, $y_2 = -y_1 \bmod n$ and $y_4 = -y_3 \bmod n$ since if $a = y_1 \bmod p$ then $-a = -y_1 \bmod p$, and likewise for all other values. By Proposition 2.6(3), $\mathcal{J}_p(a) = -\mathcal{J}_p(-a)$ and $\mathcal{J}_q(b) = -\mathcal{J}_q(-b)$, and by definition $\mathcal{J}_n(c) = \mathcal{J}_p(c) \cdot \mathcal{J}_q(c)$. We therefore conclude that if $\mathcal{J}_p(a) = \mathcal{J}_q(b) = 1$, then $\mathcal{J}_n(y_1) = \mathcal{J}_n(y_2) = 1$ and $\mathcal{J}_n(y_3) = \mathcal{J}_n(y_4) = -1$; likewise, for all combinations of values of $\mathcal{J}_p(a)$ and $\mathcal{J}_q(b)$. □

Proposition 2.10. *$-1 \notin \mathcal{QR}_n$ for any biprime $n = p \cdot q$ with $p = q = 3 \bmod 4$.*

Proof. Since $p = 3 \bmod 4$, $\mathcal{J}_p(-1) = -1 \bmod p$ and therefore $-1 \notin \mathcal{QR}_p$. Thus, the proof follows by Proposition 2.8. □

Proposition 2.11. *For every $y \in \mathbb{Z}_n^*$, $\mathcal{J}_n(y) = 1$ if and only if $y \in \mathcal{QR}_n$ or $-y \in \mathcal{QR}_n$.*

Proof. If $y \in \mathcal{QR}_n$ then $\mathcal{J}_n(y) = 1$ (by Proposition 2.8). Furthermore, by Proposition 2.6(3), we have that $\mathcal{J}_n(y) = \mathcal{J}_n(-y)$. Thus, it also holds that if $-y \in \mathcal{QR}_n$ then $\mathcal{J}_n(y) = 1$. For the other direction, if $\mathcal{J}_n(y) = 1$ then either $\mathcal{J}_p(y) = \mathcal{J}_q(y) = 1$ or $\mathcal{J}_p(y) = \mathcal{J}_q(y) = -1$ (since $\mathcal{J}_n(y) = \mathcal{J}_p(y) \cdot \mathcal{J}_q(y)$). In the former case, this implies that $y \in \mathcal{QR}_n$ (by Proposition 2.8). In the latter case, observe that $\mathcal{J}_p(-y) = \mathcal{J}_q(-y) = 1$ (since $\mathcal{J}_p(-1) = \mathcal{J}_q(-1) = -1$) and so $-y \in \mathcal{QR}_n$. □

2.3.4 Structure of $\mathbb{Z}_n^*$

Proposition 2.12 (Structure of $\mathbb{Z}_n^*$)**.** *For any safe biprime $n = p \cdot q$ with $p = 2p' + 1$ and $q = 2q' + 1$, it holds that $\mathbb{Z}_n^*$ is isomorphic to $\mathbb{Z}_2 \times \pm\mathcal{QR}_n$. Specifically, for any $\alpha \in \mathbb{Z}_n^* \setminus \pm\mathcal{QR}_n$, it holds that $\phi(b, r) = \alpha^b \cdot r \bmod n$ is an isomorphism from $\mathbb{Z}_2 \times \pm\mathcal{QR}_n$ to $\mathbb{Z}_n^*$.*

Proof. First we prove that ϕ is a bijection. Fix $\alpha \in \mathbb{Z}_n^* \setminus \mathcal{QR}_N$, and consider the inverse mapping

$$\phi^{-1}(x) := \begin{cases} (0, x) & x \in \pm\mathcal{QR}_n \\ (1, \alpha^{-1} \cdot x \bmod n) & \text{otherwise} \end{cases}.$$

It is clear that $\phi(\phi^{-1}(x)) = x$ for any $x \in \mathbb{Z}_n^*$: for $x \in \pm\mathcal{QR}_n$, $\phi^{-1}(x) = (0, x)$ and $\phi(0, x) = \alpha^0 \cdot x = x \bmod n$; for $x \notin \pm\mathcal{QR}_n$, $\phi^{-1}(x) = (1, \alpha^{-1} \cdot x)$ and $\phi(1, \alpha^{-1} \cdot x) = \alpha^1 \cdot \alpha^{-1} \cdot x = x \bmod n$. Thus, ϕ^{-1} is a 1–1 function.

We now prove that it maps all of $\mathbb{Z}_n^*$ onto $\mathbb{Z}_2 \times \pm\mathcal{QR}_n$. First, for any $x \in \pm\mathcal{QR}_n$ we have that $\phi^{-1}(x) = (0, x)$. Thus, clearly $\left\{\phi^{-1}(x) \mid x \in \pm\mathcal{QR}_n\right\} = \{0\} \times \pm\mathcal{QR}_n$. Next, for $x \notin \pm\mathcal{QR}_n$ we have that $\phi^{-1}(x) = (1, \alpha^{-1} \cdot x)$. Thus, it suffices to show that $\{\alpha^{-1} \cdot x \mid x \in \pm\mathcal{QR}_n\} = \pm\mathcal{QR}_n$, since that implies that $\left\{\phi^{-1}(x) \mid x \notin \pm\mathcal{QR}_n\right\} = \{1\} \times \pm\mathcal{QR}_n$. Defining $f(x) = \alpha^{-1} \cdot x \bmod n$, it remains to show that f is a bijection over $\pm\mathcal{QR}_n$.

The fact that f is invertible is immediate. We therefore just need to show that for any $\alpha, x \in \mathbb{Z}_n^* \setminus \pm\mathcal{QR}_n$ it holds that $\alpha^{-1} \cdot x \in \pm\mathcal{QR}_n$. By Proposition 2.11, $\mathcal{J}_n(\alpha) = \mathcal{J}_n(x) = -1$ (since neither are in $\pm\mathcal{QR}_n$). Furthermore, $\mathcal{J}_n(\alpha^{-1}) = \mathcal{J}_n(\alpha)$ (since $\alpha \cdot \alpha^{-1} = 1$ and $\mathcal{J}_n(1) = 1$). Thus,

$$\mathcal{J}_n(\alpha^{-1} \cdot x) = \mathcal{J}_n(\alpha^{-1}) \cdot \mathcal{J}_n(x) = -1 \cdot -1 = 1$$

and so $\alpha^{-1} \cdot x \in \pm\mathcal{QR}_n$, as required. □

2.4 The Strong RSA Assumption

We use the following algorithm for sampling RSA moduli:

Algorithm 2.13 (StModuliGen)**.**

Input: 1^{κ_c}, $s \in \mathbb{N}$. *The default value of the parameter* s*, if not set, is* 0.

Operation:

1. *Sample uniformly two distinct* $\max\{\kappa_c, \lfloor \log(s)/2 + 1 \rfloor\}$*-bit safe-primes* p, q.
2. *Return* $n \leftarrow p \cdot q$

That is, StModuliGen returns a strong RSA modulus n with the additional property that $n \geq s$.

The strong RSA assumption states that given a modulus n generated by StModuliGen and a random element $y \in \mathbb{Z}_n^*$, it is hard to find a pair (e, x) such that $x^e = y \bmod n$ for $e \notin \{1, -1\}$. The difference between this and the standard RSA assumption is that e is not fixed and can be chosen by the adversary.

Assumption 2.14 (Strong RSA)**.** *For any* PPT *algorithm* A*:*

$$\Pr_{n \leftarrow \mathsf{StModuliGen}(1^{\kappa_c});\, y \xleftarrow{R} \mathbb{Z}_n^*}\left[(e, x) \leftarrow \mathsf{A}(1^{\kappa}, n, y)\colon e \in \mathbb{Z}_n^* \setminus \{-1, 1\} \wedge x^e = y \bmod n\right] \leq \mathrm{neg}(\kappa).$$

2.4.1 Two Hard Games

In this section, we consider two games whose hardness follows from strong RSA (Assumption 2.14). Recall that an integer commitment to x using randomness r and parameters $n, g, h \in \mathbb{N}$ has the form $g^x \cdot h^r \bmod n$. (See details in Section 3.)

Generating non-trivial commitments. In this game, the player is given properly chosen parameters (n, g, h), and his goal is to find c, α, β, γ such that $c^\gamma = g^\alpha \cdot h^\beta \bmod n$, and γ divides neither α nor β. Clearly, A can find c, α, β where $c = g^\alpha \cdot h^\beta \bmod n$ by just choosing α, β and computing them (this is just computing an integer commitment). Furthermore, it can easily choose α', β' and γ and can compute $\alpha \leftarrow \alpha' \cdot \gamma$ and $\beta \leftarrow \beta'\gamma$. Then, by setting $c \leftarrow g^{\alpha'} \cdot h^{\beta'}$, it holds that $c^\gamma = g^\alpha \cdot h^\beta$. In this case, however, γ divides α and β. Thus, the hard problem is finding these values where γ does not divide α and β. The following proposition, stating the hardness of this game, was proven in [FO97b; MR04; CMP20].[4]

Proposition 2.15 (Hardness of generating non-trivial commitments)**.** *Assume Assumption 2.14 holds and let $s\colon \mathbb{N} \mapsto \mathbb{N}$ be a non-decreasing function. Then for any* PPT A*:*

$$\Pr_{\substack{n \leftarrow \mathsf{StModuliGen}(1^\kappa), h \xleftarrow{R} \mathcal{QR}_n \\ \ell \xleftarrow{R} [s(n)], g \leftarrow h^\ell \bmod n}} \left[\substack{(c,\alpha,\beta,\gamma) \leftarrow \mathsf{A}(1^\kappa, n, g, h): \\ (c \in \mathbb{Z}_n^*) \wedge (g^\alpha \cdot h^\beta = c^\gamma \bmod n) \wedge (\gamma \nmid \alpha \vee \gamma \nmid \beta)} \right] \le \operatorname{neg}(\kappa).$$

Proof. Let A be an algorithm that succeeds in the above with probability $\varepsilon(\kappa)$. We construct an algorithm A' of essentially the same running time that solves the strong RSA problem with probability at least $\varepsilon(\kappa)/10 - \operatorname{neg}(\kappa)$. Recall that algorithm A' receives $(1^\kappa, n, y)$. We assume that $y \in \mathcal{QR}_n$ which happens with probability $1/4$ (by Proposition 2.8 together with the fact that each of $\mathcal{QR}_p$ and $\mathcal{QR}_q$ are half the size of $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ as shown in Proposition 2.6(2)). Note that in this case y is uniform in $\mathcal{QR}_n$. Now, A' sets $h = y$, samples $g \leftarrow h^\ell$ for $\ell \xleftarrow{R} [s]$ and invokes $\mathrm{A}(1^\kappa, n, g, h)$ to obtain its output $(c, \alpha, \beta, \gamma)$. If $c \notin \mathbb{Z}_n^*$ or $c^\gamma \neq g^\alpha \cdot h^\beta \bmod n$ or $(\gamma \mid \alpha$ and $\gamma \mid \beta)$, then A' halts. Else, A has "succeeded" in its task and A' proceeds.

Since $g = h^\ell \bmod n$ and $c^\gamma = g^\alpha \cdot h^\beta \bmod n$, it holds that $c^\gamma = h^{\ell \cdot \alpha + \beta} \bmod n$. Let $t = \ell \cdot \alpha + \beta$ (note that A' can compute t). Let ℓ^* be the smallest value in $[n]$ such that $h^{\ell^*} = g \bmod n$. Note that given (h, g), the value ℓ chosen by A' when generating g is uniformly distributed in $\mathcal{L} \leftarrow \{\ell^*, \ell^* + |\langle h \rangle|, \ell^* + 2|\langle h \rangle|, \ldots, \}$. By Proposition 2.12, the set $\mathcal{QR}_n$ is $1/4$ of the size of $\mathbb{Z}_n^*$, and $h \in \mathcal{QR}_n$. Thus, $|\langle h \rangle| \le \frac{n}{4}$. Since $g^{\ell^*} = g^{\ell^* + |\langle h \rangle|} \bmod n$ and ℓ is uniform in $[s] \ge [n]$, we conclude that $|\mathcal{L}| \ge 4$.

If $\gamma \nmid \alpha$, if $\gamma \mid (\ell' \cdot \alpha + \beta)$ for some ℓ', then $\gamma \nmid ((\ell' + |\langle h \rangle|) \cdot \alpha + \beta)$. Otherwise, γ divides both $\ell' \cdot \alpha + \beta$ and $(\ell' + |\langle h \rangle|) \cdot \alpha + \beta$, and thus it divides $|\langle h \rangle| \cdot \alpha$. We claim that $\gcd(\gamma, |\langle h \rangle|) = 1$ since $h \in \mathcal{QR}_n$ and $|\mathcal{QR}_n| = \frac{|\mathbb{Z}_n^*|}{4} = \frac{\phi(n)}{4} = p' \cdot q'$ and so $|\langle h \rangle|$ divides $p' \cdot q'$ implying that it equals either $1, p', q'$ or $p' \cdot q'$. Now, if $\gcd(\gamma, |\langle h \rangle|) \neq 1$ then $\gcd(\gamma, |\langle h \rangle|) \in \{p', q', p' \cdot q'\}$ and thus given γ it is possible to factor n, as follows. This is not immediate since we do not know $|\langle h \rangle|)$ and so cannot compute $\gcd(\gamma, |\langle h \rangle|)$. However, we do know that if $n \mid a \cdot b$ then $\frac{n}{\gcd(n,a)} \mid b$. Therefore, $\frac{\gamma}{\gcd(\alpha,\gamma)} \mid |\langle h \rangle|$. Since $\frac{\gamma}{\gcd(\alpha,\gamma)} > 1$ (because $\gamma \nmid \alpha$), we have that $\frac{\gamma}{\gcd(\alpha,\gamma)} \in \{p', q', p' \cdot q'\}$ and so it is possible to factor, in contradiction with the strong RSA assumption (given p' or q' factoring is immediate, and given $p' \cdot q'$ one can compute $\phi(n) = 4 \cdot p' \cdot q'$ which enables factoring n). Let $\mathcal{T} \leftarrow \{\ell' \cdot \alpha + \beta\}_{\ell' \in \mathcal{L}}$. From the above, $|\mathcal{T}| \ge 4$ and γ can divide at most $\left\lceil \frac{|\mathcal{T}|}{2} \right\rceil$ of the elements in the set, and therefore does not divide at least $\frac{2|\mathcal{T}|}{5}$ of the elements of $\mathcal{T}$ ($2|\mathcal{T}|/5$ happens for example if there are five elements and γ divides the first, third and fifth). Thus, if $\gamma \nmid \alpha$, then $\gamma \nmid t$ with probability at least $\frac{2}{5}$.

[4][FO97b; MR04; CMP20] require $s(n) \ge n \cdot 2^\kappa$, whereas below we only require $s(n) \ge n$.

If $\gamma \mid \alpha$ but $\gamma \not| \beta$ (recall that all we know is that $\gamma \not| \alpha \vee \gamma \not| \beta$), then we have that $\gamma \mid \ell \cdot \alpha$ and $\gamma \not| \beta$, implying that $\gamma \not| t = \ell \cdot \alpha + \beta$. Thus, from here on we assume that $\gamma \not| t$.

Let $\delta \leftarrow \gcd(\gamma, t)$, $\delta_t \leftarrow t/\delta$, $\delta_\gamma \leftarrow \gamma/\delta$, and let v_t, v_γ be such that $v_t \cdot \delta_t + v_\gamma \cdot \delta_\gamma = 1$ (v_t, v_δ can be found efficiently using the extended Euclid algorithm since δ_t, δ_γ are relatively prime). It follows that

$$(c^{v_t} \cdot h^{v_\gamma})^{\delta_\gamma} = c^{v_t \cdot \gamma/\delta} \cdot h^{v_\gamma \cdot \delta_\gamma} = h^{v_t \cdot \delta_t} \cdot h^{v_\gamma \cdot \delta_\gamma} = h^{v_t \cdot \delta_t + v_\gamma \cdot \delta_\gamma} = h \bmod n,$$

where the second equality holds because $c^{v_t \cdot \gamma/\delta} = (c^\gamma)^{v_t/\delta} = \left(h^{\ell \cdot \alpha + \beta}\right)^{v_t/\delta} = \left(h^t\right)^{v_t/\delta} = h^{v_t \cdot t/\delta} = h^{v_t \cdot \delta_t}$. Thus, A' outputs $y = c^{v_t} \cdot h^{v_\gamma}$ and $e = \delta_\gamma$ as the solution to the strong RSA game (since $y^e = h \bmod n$). If $\delta_\gamma = \pm 1$ then $\gamma = \pm\delta$ and so $\gamma \mid t$, in contradiction with our assumption. Thus,

$$\mathsf{A}'$$

succeeds in the strong RSA game with probability $\varepsilon(\kappa)/10 - \mathrm{neg}(\kappa)$.[5] By the strong RSA assumption, this implies that $\varepsilon(\kappa)$ is a negligible function, as required. $\square$

Range proof game. The following game captures the security of the range proof protocol presented in Section 3. The proof of the following proposition is implicit in [CMP20, Sec 6.1].

Proposition 2.16 (Hardness of range proof game)**.** *Let $s \geq n$. If Assumption 2.14 holds, then for any* PPT *algorithm* A *there exists a negligible function* neg *such that for every $\kappa \in \mathbb{N}$,*

$$\Pr_{\substack{n \leftarrow \mathsf{StModuliGen}(1^\kappa), h \xleftarrow{R} \mathcal{QR}_n \\ \ell \xleftarrow{R} [s], g \leftarrow h^\ell \bmod n}} \left[(c, \alpha, \beta, \gamma) \leftarrow \mathsf{A}(1^\kappa, n, g, h) \colon f_{n,g,h}(c, \alpha, \beta, \gamma)\right] \leq \mathrm{neg}(\kappa)$$

where $f_{n,g,h}(c, \alpha, \beta, \gamma)$ is true if and only if

1. *$c \in \mathbb{Z}_n^*$, and*
2. *$\gamma < 2^{\kappa - 1}$, and*
3. *$g^\alpha \cdot h^\beta = c^\gamma \bmod n$, and*
4. *$\gamma \not| \alpha$ or $\gamma \not| \beta$ or $g^{\alpha/\gamma} \cdot h^{\beta/\gamma} \neq \pm c \bmod n$*

Proof. Let $(c, \alpha, \beta, \gamma)$ be the output by A when given input $(1^\kappa, n, g, h)$ such that $f_{n,g,h}(c, \alpha, \beta, \gamma) = 1$. By Proposition 2.15, the probability that conditions (1) and (3) hold and ($\gamma \not| \alpha$ or $\gamma \not| \beta$) is at most negligible. Thus, assume that $\gamma \mid \alpha$ and $\gamma \mid \beta$, and let $\alpha' \leftarrow \alpha/\gamma$ and $\beta' \leftarrow \beta/\gamma$. Since by assumption $f_{n,g,h}(c, \alpha, \beta, \gamma) = 1$ it follows that $\gcd(\gamma, \phi(n)) \neq 1$, since otherwise it holds that $\gamma^{-1} \bmod \phi(n)$ is well defined, and so

$$g^{\alpha'} \cdot h^{\beta'} = g^{\alpha \cdot \gamma^{-1}} \cdot g^{\beta \cdot \gamma^{-1}} = c^{\gamma \cdot \gamma^{-1}} = c \bmod n, \tag{1}$$

[5]This probability comes from the fact that if A succeeds and $y \in \mathcal{QR}_n$ (with probability $\frac{1}{4}$) and $\gamma \not| t$ (with probability $\frac{2}{5}$) and A doesn't factor n (negligible probability), then

$$\mathsf{A}'$$

solves the strong RSA assumption.

contradicting condition (4).

Next, recall that $n = p \cdot q = (2p' + 1) \cdot (2p' + 1)$ for some κ-bit primes p', q', and so $\phi(n) = 4 \cdot p' \cdot q'$. Since p', q' are both primes greater than $2^{\kappa-1}$ and since $\gamma < 2^{\kappa-1}$ by condition (2), we deduce that $\gcd(\gamma, p' \cdot q') = 1$. Therefore, $\gamma^{-1} \bmod p' \cdot q'$ exists and γ is *even* (this holds because $\gcd(\gamma, \phi(n)) \neq 1$ and so it must be that γ has a common factor with 2). By Proposition 2.12, we can write $c = r^b \cdot c' \bmod n$ for any $r \in \mathbb{Z}_n^* \setminus \pm\mathcal{QR}_n$, with $b \in \{0, 1\}$ and $c' \in \pm\mathcal{QR}_n$. In particular, we can take $r \in \mathbb{Z}_n^* \setminus \pm\mathcal{QR}_n$ to be a square root of 1 in $\mathbb{Z}_n^*$ (there are four square roots of 1; two of them in $\pm\mathcal{QR}_n$ and two not in $\pm\mathcal{QR}_n$; see Proposition 2.9). Recall that γ is even, and therefore $r^\gamma = 1 \bmod n$. Thus,

$$g^\alpha \cdot h^\beta = c^\gamma = (r^b)^\gamma \cdot c'^\gamma = (r^\gamma)^b \cdot c'^\gamma = 1 \cdot c'^\gamma = c'^\gamma \bmod n$$

and so similarly to Equation (1), it holds that

$$g^{\alpha'} \cdot h^{\beta'} = c' \bmod n.$$

This implies that $c' \neq \pm c \bmod n$, since otherwise $f_{n,g,h}(c, \alpha, \beta, \gamma) = 0$. Let $r = c' \cdot c^{-1} \bmod n$ and note that $g^{\alpha'} \cdot h^{\beta'} = r \cdot c \bmod n$ where $r^\gamma = 1$. This means that the order of r divides γ. Since $\gamma < 2^{\kappa-1}$, by considering the possible divisors of $\phi(n)$, it follows that the order of r is in $\{1, 2, 4\}$. Since there are no elements of order 4 in $\mathbb{Z}_n^*$, and, by assumption, $r \neq \pm 1$, it means that r is a non-trivial root of 1 which allows factoring.

□

3 The Integer Commitment Scheme

In this section, we define the integer commitment scheme and prove its basic properties. In the following let $\mathsf{IntCom}_{n,g,h}(x, \rho) := g^x \cdot h^\rho \bmod n$.

Definition 3.1 (Integer commitment). *The* integer commitment scheme $\mathsf{IntComSc} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Verify})$ *is defined as follows:*

Gen: *On input* $1^{\kappa_s}, 1^{\kappa_c}, s$:

1. *Sample* $h \overset{R}{\leftarrow} \mathcal{QR}_n$ *and* $\alpha \overset{R}{\leftarrow} [n \cdot 2^{\kappa_s}]$.
2. *Let* $g \leftarrow h^\alpha \bmod n$.
3. *Output* $\mathsf{pp} \leftarrow (\kappa \leftarrow (\kappa_s, \kappa_c), n, g, h)$.

The default value of the parameter s*, if not set, is* 0*. Let* $\mathsf{ExtdGen}$ *be the variant of* Gen *that also outputs* α.

Commit: *On input* $\mathsf{pp} = (\kappa, n, g, h)$ *and* $m \in \mathbb{Z}$:

1. $\rho \overset{R}{\leftarrow} [n \cdot 2^{\kappa_s}]$.
2. *Output* $\mathsf{IntCom}_{n,g,h}(m; \rho)$.

Verify: *On input* $\mathsf{pp} = (\kappa, n, g, h)$, $c \in \mathbb{Z}_n$ *and* $m, \rho \in \mathbb{Z}$: *Output* true *iff* $\mathsf{IntCom}_{n,g,h}(m; \rho) = \pm c \bmod n$.

When clear from the context, we typically omit κ from pp.

Remark 3.2 (Negative commitments). Rather than defining the Verify procedure to (also) accept "negative commitments", i.e., $\mathsf{IntCom}_{\mathsf{pp}}(x;\rho) = -c \bmod n$, we could have defined $\mathsf{IntCom}_{n,g,h}(x,\rho,b) := g^x \cdot h^\rho \cdot (-1)^b \bmod n$, where $b \in \{0,1\}$ is uniformly sampled by the committer, and change Verify so that $\mathsf{IntCom}_{\mathsf{pp}}(x;\rho,b) = c \bmod n$. It is easy to see that all claims below hold with respect to the modified definition. This would make working with integer commitments less confusing (e.g., does the homomorphism work with negative commitments?), but at the price of deviating from the standard notation.

3.1 Security

In this section, we prove the basic facts about the above scheme.

Claim 3.3 (Homomorphism)**.** *For any* $\mathsf{pp} = (n,g,h)$ *with* $n,g,h \in \mathbb{N}$ *and* $x,x',\rho,\rho' \in \mathbb{Z}$*:* $\mathsf{IntCom}_{\mathsf{pp}}(x;\rho) \cdot \mathsf{IntCom}_{\mathsf{pp}}(x';\rho') = \mathsf{IntCom}_{\mathsf{pp}}(x+x';\rho+\rho) \bmod n$*.*

Proof. Immediate. □

Claim 3.4 (Correctness)**.** *The scheme* $\mathsf{IntComSc}$ *is perfectly correct.*

Proof. Immediate. □

The hiding property of $\mathsf{IntComSc}$ holds even if the parameters were not chosen as in Gen (only requires $g \in \langle h \rangle$).

Claim 3.5 (Statistically hiding)**.** *For any* $\mathsf{pp} = ((\kappa_{\mathsf{s}},\cdot),n,g,h)$ *with* $n \in \mathbb{N}$*,* $h \in \mathbb{Z}_n$ *and* $g \in \langle h \rangle$*, it holds that* $\mathsf{Commit}_{\mathsf{pp}}$ *is* $2^{-\kappa_{\mathsf{s}}}$ *hiding.*

Proof. Let α be such that $g = h^\alpha \bmod n$. For any $x,x',\rho \in \mathbb{Z}$:

$$\mathsf{IntCom}_{\mathsf{pp}}(x;\rho) = g^x \cdot h^\rho = h^{\rho+\alpha x} = h^{\rho+\alpha(x-x')+\alpha x'} = g^{x'} \cdot h^{\rho+\alpha(x-x')} = \mathsf{IntCom}_{\mathsf{pp}}(x';\rho') \bmod n$$

for $\rho' = \rho + \alpha \cdot (x-x') \bmod |\langle h \rangle|$. Since the randomness for the commitment is uniform in $2^{\kappa_{\mathsf{s}}} \cdot n$ and $|\langle h \rangle| < n$, we have that both ρ and ρ' are $2^{-\kappa_{\mathsf{s}}}$ close to uniform in $[|\langle h \rangle|]$. Thus, the distributions over commitments to x and to x' are $2^{-\kappa_{\mathsf{s}}}$ close. □

The binding property of $\mathsf{IntComSc}$ holds assuming factoring is hard (over the distribution of biprimes induced by Gen).

Claim 3.6 (Computational binding)**.** *If factoring is hard with respect to the distribution of biprimes induced by* Gen*, then* $\mathsf{IntComSc}$ *is computationally binding.*

The following proof shows that an adversary who can output a commitment and two different openings, can factor n.

Proof. Let $(n,g,h) \leftarrow \mathsf{Gen}(\kappa_{\mathsf{c}},\kappa_{\mathsf{s}})$ and let $n = (2p'+1)\cdot(2q'+1)$. Assume there exists a PPT algorithm A that on input $\mathsf{pp} = (n,g,h)$ outputs two valid openings (x_0,ρ_0) and (x_1,ρ_1) with $x_0 \neq x_1$ of the same commitment c with probability $\varepsilon(\kappa_{\mathsf{c}})$. We construct an algorithm A$'$ for factoring. A$'$ receives a safe biprime n, chooses $h \overset{\text{R}}{\leftarrow} \mathcal{QR}_n$ by choosing a random $w \in_R \mathbb{Z}_n$ and computing $h = w^2 \bmod n$. Then, A$'$ chooses $\alpha \overset{\text{R}}{\leftarrow} [n \cdot 2^{\kappa_{\mathsf{s}}}]$ and computes $g \leftarrow h^\alpha \bmod n$, as in Gen.

Next, A$'$ invokes A upon (n,g,h). Let (x_0,ρ_0,x_1,ρ_1) be A's output. If $\mathsf{IntComSc}_{n,g,h}(x_0;\rho_0) \neq \pm\mathsf{IntComSc}_{n,g,h}(x_1;\rho_1) \bmod n$ or $x_0 = x_1$, then A$'$ halts.

Else, it holds that $g^{x_0} \cdot h^{\rho_0} = \pm g^{x_1} \cdot h^{\rho_1} \bmod n$ and therefore $g^{x_0 - x_1} = \pm h^{\rho_1 - \rho_0} \bmod n$. Since $g, h \in \mathcal{QR}_n$ and, by Proposition 2.10, $-1 \notin \mathcal{QR}_n$, it holds that $g^{x_0-x_1} = h^{\rho_1-\rho_0} \bmod n$. Let $x = x_0 - x_1$ and $\rho = \rho_0 - \rho_1$ (over the integers) and so $g^x = h^\rho \bmod n$. Recalling that $g = h^\alpha \bmod n$, we conclude that $h^{\rho - \alpha x} = 1 \bmod n$. Hence, for $m = \rho - \alpha x$ and $\mathsf{ord} = |\langle h \rangle|$, it holds that

$$m = 0 \bmod \mathsf{ord} \tag{2}$$

Since $\alpha \in [n \cdot 2^{\kappa_\mathsf{s}}]$ was chosen uniformly, it follows that for any given (n,g,h) there are at least 2^{κ_s} possible values of α such that $g = h^\alpha \bmod n$. Thus, the probability that $\rho = \alpha \cdot x$ with equality over the integers (for the specific α chosen by A) is at most $2^{-\kappa_\mathsf{s}}$ (observe that A$'$'s view is identical for all $\alpha' = \alpha \bmod |\langle h \rangle|$). (This holds unless $x = \rho = 0$ but that implies that $x_0 = x_1$, in contradiction with the assumption.) Thus, it follows that $m \neq 0$ (over the integers), except with probability $2^{-\kappa_\mathsf{s}}$. Assume $\gcd(h-1, n) = 1$ (otherwise, we found a root), then m is a multiple of $\phi(n)/4$ which allows factoring.[6]

□

3.2 Parameters Generation

The above hiding and binding properties hold under certain guarantees regarding the commitment parameters. These guarantees clearly hold if the parameters are *honestly generated* in a trusted setup (using Gen). In practice, however, such a strong trust assumption may be considered unreasonable. Instead, it suffices for the receiver to generate the parameters (n,g,h) and provide a zero-knowledge proof that $g \in \langle h \rangle$. To understand why this is sufficient, consider the two corruption cases:

Committer is corrupt. In this case, the receiver is assumed to be honest (or else, there is no security required). As a result, the parameters are honestly generated and the proof of binding holds as proven.

Receiver is corrupt. In this case, the security property required is that of hiding (binding is needed for a corrupted committer, but here the committer is honest). Observe that the proof of hiding holds for *all* integers n, and only requires that $h \in \mathbb{Z}_n^*$ and $g \in \langle h \rangle$. As such, if the receiver provides a zero-knowledge proof that $g \in \langle h \rangle$ then as long as the committer verifies that $h \in \mathbb{Z}_n^*$ and the ZK proof that $g \in \langle h \rangle$ then hiding is guaranteed.

Unfortunately, there is no highly efficient ZK proof for the verifier to use for proving that $g \in \langle h \rangle$. In Section 6, we address this issue by presenting a highly efficient protocol for proving that g is "not far" from $\langle h \rangle$. While the resulting protocol does not guarantee full hiding, its hiding property suffices for many applications.

[6] This is a standard fact. It is straightforward that $a^{m'} \bmod n$ is a non trivial root of 1 for m' being the largest odd factor of m and arbitrary a of Jacobi symbol -1.

4 Range Proof based on Strong RSA

The main reason for using integer commitments is that they have highly efficient *range proof* protocols. In a range proof protocol, the prover convinces the verifier it knows an opening (a, ρ) of a commitment $\widehat{A}$ with $a \in [d]$. Ideally, we would like to have a sigma protocol for the relation, quantified by a range parameter $d_v \in \mathbb{N}$ and public key $\mathsf{pp} = (n, g, h) \in \mathbb{N}^3$:

$$\mathcal{R}^{\mathsf{Rng}}_{\mathsf{pp}} := \{(c, (a, \rho) \colon \mathsf{IntCom}_{\mathsf{pp}}(a; \rho) = c \wedge a \in [d_v]\}$$

For the sake of this survey, we focus on a sigma protocol for a relaxation of the above relation formulated as the following promise problem.

Definition 4.1 (Π^{Rng})**.** *For* $\mathsf{s}, d_v, d_r \in \mathbb{N}$ *and* $\mathsf{pp} = (n, g, h) \in \mathbb{N}^3$*, let*

$$\Pi^{\mathsf{Rng}}_{\mathsf{s},d_v,d_r,\mathsf{pp}} := \begin{cases} \text{Yes :} & \{(c, (a, \rho) \colon \mathsf{IntCom}_{\mathsf{pp}}(a; \rho) = c \wedge a \in [d_v], \rho \in [d_r]\} \\ \text{Slack :} & \{(c, (a, \rho) \colon \mathsf{IntCom}_{\mathsf{pp}}(a; \rho) = \pm c \wedge a \in [\pm \mathsf{s} \cdot d_v])\} \end{cases}$$

That is, the slack instances enable the input to be larger than the designated range $[d_v]$ of the yes instances (in particular, $x \in [\pm \mathsf{s} \cdot d_v]$). Unfortunately, we are not aware of a sigma protocol (or any highly efficient zero-knowledge POK protocol) for the above problem. For instance, it is not clear how to extract when auxiliary information about the parameters (e.g., factoring of n) is known to the prover. Yet, the following Schnorr-like proof has soundness against uniformly sampled pp.

Protocol 4.2 (Range proof for integer commitments)**.**

Parties: P, V.

Parameters: $1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}}$, $d_v, d_r \in \mathbb{N}$, $\mathsf{pp} = (n, g, h) \in \mathbb{N}^3$.

Common input: $c \in \mathbb{N}$.

P's private input: $a, \rho \in \mathbb{N}$.

Operation:

1. P:
   (a) Sample $a' \xleftarrow{\text{R}} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_v]$ and $\rho' \xleftarrow{\text{R}} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot \max\{n, d_r\}]$.
   (b) Send $c' \leftarrow \mathsf{IntCom}_{\mathsf{pp}}(a'; \rho')$ to V.
2. V: Send $e \xleftarrow{\text{R}} [2^{\kappa_{\mathsf{c}}}]$ to P.
3. P:
   (a) $a'' \leftarrow e \cdot a + a'$.
   (b) $\rho'' \leftarrow e \cdot r + \rho'$.
   (c) Send (a'', ρ'') to V.
4. V: Verify:

(a) $c, c' \in \mathbb{Z}_n^*$.[7]

(b) $a'' \in [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}+1} \cdot d_v]$.

(c) $\mathsf{IntCom}_{\mathsf{pp}}(a''; \rho'') = c^e \cdot c' \bmod n$.

Note that unlike Schnorr-like proof over known-order groups, since it does not know the order of $\mathcal{QR}_n$, the prover masks its witness (a, ρ) using much larger (a', ρ').

The security of Protocol 4.2, implicit in [CMP20], is stated in the following theorem.

Theorem 4.3 (Security of Protocol 4.2)**.** *If Assumption 2.14 holds, then Protocol 4.2 is computational Sigma protocol for* $\left\{\Pi^{\mathsf{Rng}}_{\mathsf{s},d_v,d_r,\mathsf{pp}}\right\}$ *with respect to the distribution ensemble* $\left\{P_{\kappa=(\kappa_{\mathsf{s}},\kappa_{\mathsf{c}})}\right\}$ *that outputs* $\mathsf{s} \leftarrow 2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}+2}$*, arbitrary* $d_v, d_r \in \mathbb{N}$*, and* $\mathsf{pp} \overset{R}{\leftarrow} \mathsf{Gen}(1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}})$.

That is, assuming strong RSA, with save but negligible probability over pp, Protocol 4.2 is a computational Sigma protocol for $\Pi^{\mathsf{Rng}}_{\mathsf{s},d_v,d_r,\mathsf{pp}}$. We prove Theorem 4.3 by proving that Protocol 4.2 is a (non-computational) Sigma protocol for the following promise problem:

Definition 4.4 (Π^{Rng})**.** *For* $\kappa_{\mathsf{c}}, \mathsf{s}, d_v, d_r \in \mathbb{N}$*, let*

$$\Pi^{\mathsf{Rng}}_{\kappa_{\mathsf{c}},\mathsf{s},d_v,d_r} := \begin{cases} For & z = ((n,g,h),c),(x,r,\gamma)) \\ \text{Yes :} & \{z \colon \mathsf{IntCom}_{\mathsf{pp}}(x;r) = c \wedge x \in [d_v], r \in [d_r]\} \\ \text{Slack :} & \{z \colon (\mathsf{IntCom}_{\mathsf{pp}}(x;r) = \pm c \wedge x \in [\pm \mathsf{s} \cdot d_v]) \\ & \quad \vee \gcd(x,n) \in (1, 2^{\kappa_{\mathsf{c}}+1}] \\ & \quad \vee \left(c \in \mathbb{Z}_n^*, \gamma \le 2^{\kappa_{\mathsf{c}}-1}\right) \wedge (g^x \cdot h^r = c^\gamma \bmod n) \wedge \\ & \qquad (\gamma \nmid x \vee \gamma \nmid r \vee (g^{x/\gamma} \cdot h^{r/\gamma} \neq \pm c \bmod n))\} \end{cases}.$$

That is, the slack instances not only allow the input to be larger than the designated range, but also enables the extractor to output (c, x, r, γ) that sets f to be true in Proposition 2.16. We sometimes refer to the first condition of the slack set, i.e., $\mathsf{IntCom}_{\mathsf{pp}}(x;r) = \pm c \wedge x \in [\pm d_v \cdot \mathsf{s}]$, as the *meaningful slack instances.* We prove the following result.

Theorem 4.5. *There exists an efficient algorithms* Sim *and* Ext *such that the following holds for any* $p = (\kappa_{\mathsf{c}}, \mathsf{s}, d_v, d_r) \in \mathbb{N}^3$*: Protocol 4.2 is a Sigma-protocol for* Π^{Rng}_p*, with simulator* Sim_p *and POK extractor* Ext_p.

Theorem 4.5 is proved below, but we first use it for proving Theorem 4.3.

Proof of Theorem 4.3. Completeness and zero-knowledge readily follow from Theorem 4.5, which states that these properties hols for *any* any choice of pp. For the POK part, we prove that the extractor Ext guaranteed by Theorem 4.5 is a good extractor. Fix parameters $p \leftarrow (1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}}, d_v, d_r, \mathsf{pp})$ and two transcripts $\{t_j \leftarrow (c, c', e_j, \gamma_j)\}_{j \in \{0,1\}}$ that make V accept. Let $\mathsf{s} \leftarrow 2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}+2}$ and $(x, \rho, \gamma) \leftarrow \mathsf{Ext}_{\mathsf{s},p}(c, c', e_0, e_1, \gamma_0, e_1)$. Note that if $\gcd(x,n) \in (1, \ldots, 2^{\kappa_{\mathsf{c}}+1}]$, then x is a non-trivial

[7] Assuming factoring n is hard, in a setting where the factorization of n is not known to P it suffices to check that $c, c' \bmod n \neq 0$.

factorization of n. Since we assume Assumption 2.14 holds, this happens only with negligible probability. Similarly, by Proposition 2.16, it holds that that $\left(c \in \mathbb{Z}_n^*, \gamma \le 2^{\kappa_{\mathsf{c}}-1}\right) \wedge (g^x \cdot h^r = c^\gamma \bmod n)$ which only happens with negligible probability over the choice of pp. So with save but negligible probability over the choice of pp, it holds that $\mathsf{IntCom}_{\mathsf{pp}}(x;\rho) = \pm c \wedge x \in [d_v \cdot \mathsf{s}]$, concluding the proof. ☐

Proof of Theorem 4.5. Fix $p = (\kappa_{\mathsf{c}}, \mathsf{s}, d_v, d_r) \in \mathbb{N}^3$. The proof follows by Claims 4.6, 4.8 and 4.10 stated below. ☐

Claim 4.6 (Correctness)**.** *Protocol 4.2 is perfectly correct.*

Proof. Immediate. ☐

Zero knowledge. Consider the following simulator.

Algorithm 4.7 (Sim)**.**

Paramters: p.

Input: $c \in \mathbb{N}$.

Operation:

1. *Sample*
   (a) $a'' \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_v]$.
   (b) $\rho'' \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_r]$.
   (c) $e \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{c}}-1}]$.
2. *Let* $c' \leftarrow \mathsf{IntCom}_{\mathsf{pp}}(a'';\rho'') \cdot (c^e)^{-1} \bmod n$.[8]
3. *Output* (c', e, a'', ρ'').

Claim 4.8 (Zero knowledge)**.** *For any* $((n,g,h),c),(a,r,\gamma)) \in \mathrm{Yes}_p^{\mathsf{Rng}}$*, the output of* $\mathsf{Sim}_p(c)$ *is* $2^{1-\kappa_{\mathsf{s}}}$*-close to the semi-honest execution on parameter* p *and common input* c*.*

Proof. Let $a' \leftarrow a'' - a \cdot e$ and $\rho' \leftarrow \rho'' - \rho \cdot e$. For c' computed by by Sim it holds that

$$c' = g^{a''} \cdot h^{\rho''} \cdot g^{-a\cdot e} \cdot h^{-\rho\cdot e} = g^{a''-a\cdot e} \cdot h^{\rho''-\rho\cdot e} = g^{a'} \cdot h^{\rho'} \bmod n$$

Hence, the value of c' in the emulated execution and in the real execution is the same function of (c, e, a'', ρ''). Since, the e's in both executions are identically distributed, it is left to bound the statistical distance between (a'', ρ'') given c, e.

In the emulated execution, $a'' \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_v]$ and $\rho'' \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_r]$. In contrast, in the real execution, $a'' = e \cdot a + a'$ and $\rho'' = e \cdot \rho + \rho'$, for $a' \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_v]$ and $\rho' \overset{R}{\leftarrow} [2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_r]$. It follows that the statistical distance between the value of a'''s is at most $ea/2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_v \le 2^{-\kappa_{\mathsf{s}}}$ and between the value of ρ'''s is at most $e\rho/2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}} \cdot d_r \le 2^{-\kappa_{\mathsf{s}}}$. All in all, the statistical distance is at most $2 \cdot 2^{-\kappa_{\mathsf{s}}}$. ☐

[8] c^e is guaranteed to have an inverse since $c \in \mathbb{Z}_n^*$ for any correct c in the language, since $c = g^a \cdot h^\rho \bmod n$ where $g, h \in \mathbb{Z}_n^*$.

Special soundness. Consider the following knowledge extractor.

Algorithm 4.9 (Ext)**.**

Paramter: p.

Input: $c \in \mathbb{N}$ *and two transcripts* $(c', e_0, a''_0, \rho''_0)$ *and* $(c', e_1, a''_1, \rho''_1)$.

Operation:

```
1. Let $\alpha \leftarrow a''_0 - a''_1$, $\beta \leftarrow \rho''_0 - \rho''_1$ and $\gamma \leftarrow e_0 - e_1$.
2. If the event $E := (\gamma \mid \alpha) \wedge (\gamma \mid \beta) \wedge (g^{\alpha/\gamma} \cdot h^{\beta/\gamma} = \pm c \bmod n)$ is true, output
       $(\alpha' \leftarrow \alpha/\gamma, \beta' \leftarrow \beta/\gamma)$.
3. Else, output $(\alpha, \beta, \gamma)$.
```

Claim 4.10 (Special soundness)**.** *For any input* $c \in \mathbb{N}$ *and two transcript* $t_0 = (c', e_0, a''_0, \rho''_0)$ *and* $t_1 = (c', e_1, a''_1, \rho''_1)$ *with* $e_0 > e_1$ *and* $\mathsf{V}_p(c, t_j) = 1$ *for both* $j \in \{0, 1\}$*, it holds that* $(c, \mathsf{Ext}_p(c, t_0, t_1)) \in \text{Slack}_p^{\mathsf{Rng}}$.

Proof. Since both t_0 and t_1 are accepting, for both $j \in \{0, 1\}$:

1. $a''_j \in [2^{\kappa_s + \kappa_c + 1} \cdot d_v]$.
2. $g^{a''_j} \cdot h^{\rho''_j} = c^{e_j} \cdot c' \bmod n$.

By the second item, we deduce that $g^{\alpha} \cdot h^{\beta} = c^{\gamma} \bmod n$. Now if $E = \text{true}$, then $\mathsf{IntComSc}_{\mathsf{pp}}(\alpha'; \beta') = g^{\alpha'} \cdot h^{\beta'} = \pm c \bmod n$ and $\alpha \in [\pm d \cdot 2^{\kappa_s + \kappa_c + 1}]$. So indeed $(c, \mathsf{Ext}_p(c, t_0, t_1)) \in \text{Slack}_p^{\mathsf{Rng}}$.

Else, observe that $\gamma < 2^{\kappa_c - 1}$ (since each of $e_0, e_1 \in [2^{\kappa_c - 1}]$), $g^{\alpha} \cdot h^{\beta} = c^{\gamma} \bmod n$ (by how they are defined above), and at least one of the following holds: $\gamma \nmid \alpha$, $\gamma \nmid \beta$ or $g^{\alpha/\gamma} \cdot h^{\beta/\gamma} \neq \pm c \bmod n$. Hence, $(c, \mathsf{Ext}_p(c, t_0, t_1)) \in \text{Slack}_p^{\mathsf{Rng}}$ also in this case. □

5 Equality Proof

In this section, we consider equality proofs, which are the main application of integer commitments. Equality proofs allow us to prove that two additively homomorphic commitments open to the same value. The two commitments can be any combination of integer commitments, Pedersen or ElGamal commitments, or the Paillier additively homomorphic public-key encryption scheme (naturally viewed as a commitment).

We start by presenting an equality proof between integer commitments and the additively homomorphic "group commitments", naturally defined by exponentiation over prime-order groups, i.e., $\mathsf{Com}_{\mathbb{G},G}(x) := x \cdot G$.[9] In Section 5.2, we give an equality proof between integer commitments and Pedersen commitments over prime-order groups, and in Section 5.3, we do the same for Paillier encryption. In Section 5.4, we explain, among other applications, how to use the above protocols to obtain equality proofs between any type of commitment (group, Pedersen, or Paillier). To make the text more readable, in the following we denote integer commitments using an overhead tilde, e.g., $\tilde{A} \leftarrow \mathsf{IntCom}_{\mathsf{pp}}(a; \rho)$.

[9]Obviously, being deterministic, group commitments are not hiding (at least not in the usual sense), but it is simpler to explain equality proofs using this simple example.

5.1 Equality with Group Commitments in Rough Groups

Definition 5.1. *We say that a group $\mathbb{G}$ is ℓ-rough, for $\ell \in \mathbb{N}$, if $\ell' \nmid |\mathbb{G}|$, for every $\ell' < \ell$.*

Equality proof for integer and group commitments is a protocol for the following promise problem:

$$\Pi^{\mathsf{EqGrp}}_{\mathsf{s},d_v,d_r,\mathsf{pp},G} := \begin{cases} \text{Yes :} & \left\{((\widetilde{A}, A), (a, \rho)) \colon \mathsf{IntCom}_{\mathsf{pp}}(a;\rho) = \widetilde{A} \wedge a \cdot G = A \wedge a \in [d_v], \rho \in [d_r]\right\} \\ \text{Slack :} & \left\{((\widetilde{A}, A), (a, \rho)) \colon \mathsf{IntCom}_{\mathsf{pp}}(a;\rho) = \pm\widetilde{A} \wedge a \cdot G = A \wedge a \in [\pm \mathsf{s} \cdot d_v]\right\} \end{cases}$$

Note that description of the generator G of the additive group is part of the above problem, but as we shall see later, G can be a fixed generator of a "large enough" group. Also note that the above promise problem not only states that a is the opening of the two commitments, but also that it is small. Our protocol critically relies on this bound to guarantee zero-knowledge. Furthermore, this bound is critical for our applications of equality proof. See Section 5.4.

Protocol 5.2 (Equality proof with a group commitment)**.**

Parameters: $1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}}$, $d_v, d_r \in \mathbb{N}$, $\mathsf{pp} = (n, g, h) \in \mathbb{N}^3$, G.

Common input: $\widetilde{A}, A$.

P's private input: $a, \rho \in \mathbb{N}$.

Operation: The protocol follows the same lines as Protocol 4.2, with the following additions:

Step 1. P (also) sends $A' \leftarrow a' \cdot G$ to V.

Step 4. V (also) verifies that $a'' \cdot G = e \cdot A + A'$.

. . . .

Theorem 5.3 (Security of Protocol 5.2)**.** *If Assumption 2.14 holds, then Protocol 5.2 is computational Sigma protocol for $\left\{\Pi^{\mathsf{EqGrp}}_{\mathsf{s},d_v,d_r,\mathsf{pp}}\right\}$ with respect to the distribution ensemble $\left\{P_{\kappa=(\kappa_{\mathsf{s}},\kappa_{\mathsf{c}})}\right\}$ that outputs $\mathsf{s} \leftarrow 2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}+2}$, arbitrary $d_v, d_r \in \mathbb{N}$, generator G of a $2^{\kappa_{\mathsf{c}}}$-rough group, and $\mathsf{pp} \overset{R}{\leftarrow} \mathsf{Gen}(1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}})$.*

Proof.

Correctness. Clear.

Zero knowledge. Consider the simulator Sim' that acts like Algorithm 4.7 but in Step 3, also output $A' \leftarrow a'' \cdot G - e \cdot A$. It is clear that the added A' part in the view is perfectly emulated by Sim'. Hence, the zero-knowledge property follows from the same lines as that of Protocol 4.2.

Special soundness. Fix two accepting transcripts $\left\{(\widetilde{A}', A', e_j, a'''_j, \rho''_j)\right\}_{j\in\{0,1\}}$ with $e_0 > e_1$, and let $\alpha \leftarrow a'''_0 - a'''_1$, $\beta \leftarrow \rho''_0 - \rho''_1$ and $\gamma \leftarrow e_0 - e_1$. The proof of Theorem 4.3 yields that with save but negligible probability

$$(\gamma \mid \alpha) \wedge (\gamma \mid \beta) \wedge (g^{\alpha/\gamma} \cdot h^{\beta/\gamma} = \pm c \bmod n) \tag{3}$$

We instruct the extractor to output $(\alpha' \leftarrow \alpha/\gamma, \beta' \leftarrow \beta/\gamma)$. Since α' is of the right size, it is left to argue that $\alpha' \cdot G = A$. Since both transcripts are accepting, it holds that $x''_j \cdot G = e_j \cdot A + A'$ for both $j \in \{0,1\}$, and therefore $\gamma \cdot A = \alpha \cdot G$. Let q be the not-necessarily-prime order of the group generated by G. Next, we will be using the following fact.

Fact 5.4. *Let $\ell, q \in \mathbb{N}$ such that $\ell' \not| \, q$ for all $\ell' < \ell$. Then, $\gcd(\ell', q) = 1$, for all $\ell' < \ell$.*

Proof. Fix $\ell' < \ell$ and let $p_1^{a_1} \cdots p_k^{a_k}$ denote the prime factorization of ℓ'. By assumption, for all $i \in [k]$, $p_i \not| \, q$, and thus $\gcd(p_i, q) = 1$, for all $i \in [k]$. Consequently, $\gcd(\ell', q) = 1$. □

Since $\mathbb{G}$ is 2^{κ_c}-rough and $\gamma \in [2^{\kappa_c} - 1]$, it follows by Fact 5.4 that $\gcd(\gamma, q) = 1$ and therefore $A = \gamma^{-1} \cdot \alpha \cdot G$. Writing $\alpha = \alpha' \cdot \gamma$, we deduce that $\gamma^{-1} \cdot \alpha = \alpha' \bmod q$, and therefore $A = \alpha' \cdot G$. □

5.2 Equality with Pedersen Commitments in Known-Order Rough Groups

In this section, we present an equality proof protocol between integer commitments and the (additively homomorphic) Pedersen commitments over known-order rough groups, e.g., elliptic curve groups. The following text readily extends to *ElGamal* commitments over the same types of groups.

For a generator G of an additive group $\mathbb{G}$, the Pedersen commitment with respect to public key $E \in \mathbb{G}$, input $m \in \mathbb{Z}_q$, and randomness $\rho \in \mathbb{Z}_q$, is defined by $\mathsf{Ped}_{G,E}(m;\rho) := \rho \cdot E + m \cdot G$. That is, the Pedersen commitment is just an integer commitment written here in additive form, where the operations are performed in the group. Consider the following promise problem:

$$\Pi^{\mathsf{EqPed}}_{\mathsf{s},d_v,d_r,\mathsf{pp},G,E} := \begin{cases} \text{Yes}: & \left\{((\widetilde{A}, \widehat{A}), (a, \rho, \widehat{\rho}))\colon \mathsf{IntCom}_{\mathsf{pp}}(a;\rho) = \widetilde{A} \wedge \mathsf{Ped}_E(a;\widehat{\rho}) = \widehat{A} \wedge a \in [d_v], \rho \in [d_r]\right\} \\ \text{Slack}: & \left\{((\widetilde{A}, \widehat{A}), (a, \rho, \widehat{\rho}))\colon \mathsf{IntCom}_{\mathsf{pp}}(a;\rho) = \pm\widetilde{A} \wedge \mathsf{Ped}_{G,E}(a;\widehat{\rho}) = \widehat{A} \wedge a \in [\pm \mathsf{s} \cdot d_v]\right\} \end{cases}$$

Namely, a Yes instance is a pair of Integer and Pedersen commitments to same (not too large) value.

Protocol 5.5 (Equality proof with Pedersen Commitments).

Parameters: $1^{\kappa_s}, 1^{\kappa_c}$, $d_v, d_r \in \mathbb{N}$, $\mathsf{pp} = (n, g, h) \in \mathbb{N}^3, q, G, E$.

Common input: $\widetilde{A}, \widehat{A}$.

P's private input: $a, \rho, \widehat{\rho} \in \mathbb{N}$.

Operation: The protocol follows the same lines as Protocol 4.2, with the following additions:

Step 1. P (also) sends $\widehat{A}' \leftarrow \mathsf{Ped}_E(a';\widehat{\rho}')$ to V, for $\widehat{\rho}' \xleftarrow{\text{R}} \mathbb{Z}_q$.

Step 3. P (also) sends $\widehat{\rho}'' \leftarrow e \cdot \widehat{\rho} + \widehat{\rho}' \bmod q$ to V.

Step 4. V (also) verifies that $\mathsf{Ped}_E(a'';\widehat{\rho}'') = e \cdot \widehat{A} + \widehat{A}'$.

Theorem 5.6 (Security of Protocol 5.5)**.** *Fix two accepting transcripts* $(\widetilde{A}', \widehat{A}', e_0, a_0''', \rho_0'',)$ *and* $(\widetilde{A}', A', e_1, a_1''', \rho_1'')$ *with* $e_0 > e_1$*, and let* $\alpha \leftarrow a_0''' - a_1'''$*,* $\beta \leftarrow \rho_0'' - \rho_1''$ *and* $\gamma \leftarrow e_0 - e_1$*. The proof of Theorem 4.3 yields that with save but negligible probability*

$$(\gamma \mid \alpha) \wedge (\gamma \mid \beta) \wedge (g^{\alpha/\gamma} \cdot h^{\beta/\gamma} = \pm c \bmod n) \tag{4}$$

We instruct the extractor to output $(\alpha' \leftarrow \alpha/\gamma, \beta' \leftarrow \beta/\gamma)$*, and so is left to argue that* $\alpha' \cdot G = A$*.*

If Assumption 2.14 holds, then Protocol 5.5 is computational Sigma protocol for $\left\{\Pi^{\mathsf{EqPed}}_{\mathsf{s},d_v,d_r,\mathsf{pp},E}\right\}$ *with respect to the distribution ensemble* $\left\{P_{\kappa=(\kappa_{\mathsf{s}},\kappa_{\mathsf{c}})}\right\}$ *that outputs* $\mathsf{s} \leftarrow 2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}+2}$*, arbitrary* $d_v, d_r \in \mathbb{N}$*,* G *that is a generator of an additive* q*-order* $2^{\kappa_{\mathsf{c}}}$*-rough group* $\mathbb{G}$*,* $E \in \mathbb{G}$*, and* $\mathsf{pp} \overset{R}{\leftarrow} \mathsf{Gen}(1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}})$*.*

Proof.

Correctness. Clear.

Zero knowledge. Consider the simulator Sim' that acts like Algorithm 4.7 but adds the following values to the generated view:

- $\widehat{\rho}'' \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$.
- $\widehat{A}' \leftarrow \mathsf{Ped}_{G,E}(a''; \widehat{\rho}'') - e \cdot \widehat{A}$.

It is clear that the added $(\widehat{A}', \widehat{\rho}'')$ part in the view, is perfectly emulated by Sim'. Hence, the zero-knowledge property follows from the same lines as that of Protocol 4.2.

Special soundness. Fix two accepting transcripts $\left\{(\widetilde{A}', \widehat{A}', e_j, a_j''', \rho_j'', \widehat{\rho}_j'')\right\}_{j\in\{0,1\}}$ with $e_0 > e_1$, and let $\alpha \leftarrow a_0''' - a_1'''$, $\beta \leftarrow \rho_0'' - \rho_1''$, $\hat{\beta} \leftarrow \widehat{\rho}_0'' - \widehat{\rho}_1''$ and $\gamma \leftarrow e_0 - e_1$. The proof of Theorem 4.3 yields that with save but negligible probability

$$(\gamma \mid \alpha) \wedge (\gamma \mid \beta) \wedge (g^{\alpha/\gamma} \cdot h^{\beta/\gamma} = \pm c \bmod n) \tag{5}$$

Note that since $\gamma \in [2^{\kappa_{\mathsf{c}}} - 1]$, $\gcd(\gamma, q) = 1$. We instruct the extractor to output $(\alpha' \leftarrow \alpha/\gamma, \beta' \leftarrow \beta/\gamma, \hat{\beta}' \leftarrow \hat{\beta} \cdot \gamma^{-1} \bmod q)$. It thus left to prove that

$$\widehat{A} = \mathsf{Ped}_{G,E}(\alpha'; \gamma^{-1} \cdot \hat{\beta})$$

Since both transcripts are accepting, we get that $\widehat{A} = \mathsf{Ped}_{G,E}(\gamma^{-1} \cdot \alpha \bmod q; \gamma^{-1} \cdot \hat{\beta})$. This concludes the proof since, as in the proof of Theorem 5.3 we have that $\gamma^{-1} \cdot \alpha = \alpha' \bmod q$. □

5.3 Equality with Paillier Encryption

In this section, we present an equality proof protocol between integer commitments and the additively homomorphic Paillier encryption scheme introduced by Paillier [Pai99]. Recall that for a public key $n \in \mathbb{N}$, the plaintext domain is $\mathbb{Z}^*_{n^2}$, and the randomness domain is $\mathbb{Z}^*_n$. Given $m \in \mathbb{Z}^*_{n^2}$ and randomness $\rho \in \mathbb{Z}^*_n$, the Paillier encryption with respect to public key n is defined by $\mathsf{PEnc}_n(m; \rho) := (n+1)^m \cdot \rho^n \bmod n^2$. Consider the following promise problem.

$$\Pi^{\mathsf{EqPal}}\mathsf{s}, d_v, d_r, \mathsf{pp}, \mathsf{pk} := \begin{cases} \text{Yes :} & \{((\widetilde{A}, \widehat{A}), (a, \rho)) : \\ & \quad \mathsf{IntCom}_{\mathsf{pp}}(a;\rho) = \widetilde{A} \wedge \mathsf{PEnc}_{\mathsf{pk}}(a;\widehat{\rho}) = \widehat{A} \wedge a \in [d_v], \rho \in [d_r]\} \\ \text{Slack :} & \{((\widetilde{A}, \widehat{A}), (a, \rho, \widehat{\rho})) : \\ & \quad \mathsf{IntCom}_{\mathsf{pp}}(a;\rho) = \pm\widetilde{A} \wedge \mathsf{PEnc}_{\mathsf{pk}}(a;\widehat{\rho}) \wedge a \in [\pm \mathsf{s} \cdot d_v]\} \end{cases}$$

Namely, a Yes instance is a pair of Integer commitment and a Paillier encryption of the same (not too large) value.

Protocol 5.7 (Equality proof with Paillier encryption)**.**

Parameters: $1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}}$, d_v, d_r, $\mathsf{pp} = (n, g, h) \in \mathbb{N}^3, \mathsf{pk} \in \mathbb{N}$.

Common input: $\widetilde{A}, \widehat{A}$.

P's private input: $a, \rho, \widehat{\rho} \in \mathbb{N}$.

Operation: The protocol follows the same lines as Protocol 4.2, with the following additions:

Step 1. P (also) sends $\widehat{A}' \leftarrow \mathsf{PEnc}_{\mathsf{pk}}(a';\widehat{\rho}')$ to V, for $\widehat{\rho}' \xleftarrow{R} \mathbb{Z}^*_{\mathsf{pk}}$.

Step 3. P (also) sends $\widehat{\rho}'' \leftarrow \widehat{\rho}^e \cdot \widehat{\rho}' \bmod \mathsf{pk}$ to V.

Step 4. V (also) verifies that $\mathsf{PEnc}_{\mathsf{pk}}(a'';\widehat{\rho}'') = e \cdot \widetilde{A} + \widetilde{A}'$ and that $\widehat{\rho}'' \in \mathbb{Z}^*_{\mathsf{pk}}$.

. . . .

Let $\mathsf{PKeyGen}$ be any algorithm that on input $1^{\kappa_{\mathsf{c}}}$ outputs $\mathsf{pk} \in \mathbb{N}$ whose prime factors are of size at least $2^{\kappa_{\mathsf{c}}}$.[10]

Theorem 5.8 (Security of Protocol 5.7)**.** *If Assumption 2.14 holds and factoring is hard with respect to* $\mathsf{PKeyGen}$*, then Protocol 5.5 is computational Sigma protocol for* $\left\{\Pi^{\mathsf{EqPed}}_{\mathsf{s},d_v,d_r,\mathsf{pp},E}\right\}$ *with respect to the distribution ensemble* $\left\{P_{\kappa=(\kappa_{\mathsf{s}},\kappa_{\mathsf{c}})}\right\}$ *that outputs* $\mathsf{s} \leftarrow 2^{\kappa_{\mathsf{s}}+\kappa_{\mathsf{c}}+2}$*, arbitrary* $d_v, d_r \in \mathbb{N}$*,* $\mathsf{pp} \xleftarrow{R} \mathsf{Gen}(1^{\kappa_{\mathsf{s}}}, 1^{\kappa_{\mathsf{c}}})$ *and* $\mathsf{pk} \xleftarrow{R} \mathsf{PKeyGen}(1^{\kappa_{\mathsf{c}}})$*.*

The proof of Theorem 5.8, given in Section 5.3.2, uses a fact about the "Paillier modulus" that is stated and proven in Section 5.3.1.

5.3.1 Paillier Ciphertexts Divisors

Proposition 5.9 (Paillier ciphertexts divisors)**.** *There exists an efficient algorithm that given as input* $n \in \mathbb{N}$*,* $x \in \mathbb{Z}_n$*,* $z, \rho \in \mathbb{Z}^*_n$ *and* $\widetilde{A} \in \mathbb{Z}^*_{n^2}$ *such that*

$$\mathsf{PEnc}_n(z \cdot x;\rho) = z \cdot \widetilde{A},$$

it outputs $r \in \mathbb{Z}^*_n$ *with* $\mathsf{PEnc}_n(x;r) = \widetilde{A}$*.*

Proposition 5.9 is proved using the following fact.

[10]This formulation allows the prover to sample pk by itself, and prove that it is of the right form. Note that we do not require $\gcd(\mathsf{pk}, \phi(n)) = 1$, which is not needed for the range proof.

Proposition 5.10. *There exists an efficient algorithm that given as input $n \in \mathbb{N}$ and $\rho, y, z \in \mathbb{N}$ such that*

1. *ρ, z are relatively prime to n, and*
2. *$\rho^n = y^z \bmod n^2$,*

it outputs $r \in \mathbb{Z}_n^*$ with $r^n = y \bmod n^2$.

Proof. Using the extended GCD algorithm, we find the Bezout coefficients $u, v \in \mathbb{Z}$ such that $u \cdot z + v \cdot n^2 = 1$. Let $r \leftarrow \rho^u \cdot y^{nv} \bmod n^2$. It holds that

$$r^n = \rho^{un} \cdot y^{vn^2} = y^{zu} \cdot y^{vn^2} = y \bmod n^2.$$

Note that the same holds for $\rho' \leftarrow r \bmod n$ (powering by n removes any additive terms that are divisible by n). Finally, since ρ and y are relatively prime to n, so are r and ρ'. □

Proof of Proposition 5.9. For $\widehat{Y} \leftarrow \widetilde{A} - \mathsf{PEnc}_n(x; 1)$, it holds that $z \cdot \widehat{Y} = \mathsf{PEnc}_n(0; \rho)$. Thus, for $y \leftarrow \widehat{Y}$ (y is viewed as integer), it holds that $y^z = \rho^n \bmod n^2$. By Proposition 5.10, we can efficiently compute $r \in \mathbb{Z}_n^*$ such that $r^n = y \bmod n^2$. It follows that $\mathsf{PEnc}_n(0; r) = \widehat{Y}$, and therefore $\mathsf{PEnc}_{\mathsf{pk}}(x; r) = \widetilde{A}$. □

5.3.2 Proving Theorem 5.8

Using Proposition 5.9, we prove Theorem 5.8 as follows.

Proof of Theorem 5.8.

Correctness. Clear.

Zero knowledge. Consider the simulator Sim' that acts like Algorithm 4.7 but adds the following values to the generated view:

- $\widehat{\rho}'' \xleftarrow{R} \mathbb{Z}_{\mathsf{pk}}^*$.
- $\widehat{A}' \leftarrow \mathsf{PEnc}_{\mathsf{pk}}(a''; \widehat{\rho}'') - e \cdot \widehat{A}$.

It is clear that the added $(\widehat{A}', \widehat{\rho}'')$ part in the view, is perfectly emulated by Sim'. Hence, the zero-knowledge property follows from the same lines as that of Protocol 4.2.

Special soundness. Fix two accepting transcripts $\left\{(\widetilde{A}', \widehat{A}', e_j, a_j''', \rho_j'', \widehat{\rho}_j'')\right\}_{j \in \{0,1\}}$ with $e_0 > e_1$, and let $\alpha \leftarrow a_0''' - a_1'''$, $\beta \leftarrow \rho_0'' - \rho_1''$, $\hat{\beta} \leftarrow \widehat{\rho}_0'' \cdot (\widehat{\rho}_1'')^{-1} \bmod \mathsf{pk}$ (note that $\widehat{\rho}_1'' \in \mathbb{Z}_{\mathsf{pk}}^*$), and $\gamma \leftarrow e_0 - e_1$. The proof of Theorem 4.3 yields that with save but negligible probability

$$(\gamma \mid \alpha) \wedge (\gamma \mid \beta) \wedge (g^{\alpha/\gamma} \cdot h^{\beta/\gamma} = \pm c \bmod n) \tag{6}$$

Since both transcripts are accepting, it holds that

$$\mathsf{PEnc}_{\mathsf{pk}}(\alpha; \hat{\beta}) = \gamma \cdot \widetilde{A} \tag{7}$$

Since $\gamma \in [2^{\kappa_c} - 1]$, it holds that $\gcd(\gamma, \mathsf{pk}) = 1$. Thus by Proposition 5.9 (letting $n \leftarrow \mathsf{pk}$, $x \leftarrow \alpha'$, $z \leftarrow \gamma$, and $\rho \leftarrow \hat{\beta}$), we can efficiently compute $\hat{\beta}' \in \mathbb{Z}_{\mathsf{pk}}^*$ such that $\mathsf{PEnc}_{\mathsf{pk}}(\alpha'; \hat{\beta}') = \widetilde{A}$. Outputting $(\alpha', \beta', \hat{\beta}')$ concludes the proof. □

5.4 Applications

In this section we present several application of the above protocols. For readability, we focus on group commitments, but all the following readily generalized to Pedersen Commitments and Paillier encryption.

5.4.1 Range Proof for Rough Groups

An immediate application of Protocol 5.2 is a range proof protocol for arbitrary rough groups. Namely, an extended computational Sigma protocol for the promise problem

$$\Pi^{\mathsf{Rng}}_{\mathsf{s},d_v,G} = \begin{cases} \text{Yes}: & \{(A,a)\colon a \cdot G = A \,\wedge\, a \in [d_v]\} \\ \text{Slack}: & (A,a)\colon a \cdot G = A \,\wedge\, a \in [\mathsf{s} \cdot d_v] \end{cases}$$

The protocol is straightforward. In the preamble stage, the verifier samples pp using Gen and prove the verifier it is well formed (see Section 3.2). Then the prover sends $\widetilde{A} \leftarrow \mathsf{IntCom}_{\mathsf{pp}}(a;\rho)$ to the verifier, and the parties continue according to Protocol 5.2.

5.4.2 Equality Proof between Rough Groups

Similarly to the above, Protocol 5.2 yields a computational, extended, Sigma protocol for the following promise problem:

$$\Pi^{\mathsf{EqGrp}}_{\mathsf{s},d_v,G,\widehat{G}} := \begin{cases} \text{Yes}: & \left\{((A,\widehat{A}),a)\colon a \cdot G = A \,\wedge\, a \cdot \widehat{G} = \widehat{A} \wedge a \in [d_v]\right\} \\ \text{Slack}: & \left\{((A,\widehat{A}),a)\colon a \cdot G = A \,\wedge\, a \cdot \widehat{G} = \widehat{A} \wedge a \in [\pm\mathsf{s} \cdot d_v]\right\} \end{cases}$$

where G and $\widehat{G}$ are generators of two suitable rough groups.

Note that without the bound on a, the above promise problem is meaningless: let q and $\widehat{q}$ be the orders of the prime order groups generated by G and $\widehat{G}$, respectively, and assume $\gcd(q,\widehat{q}) = 1$. By CRT, for any $a \in \mathbb{Z}_q$ and $\widehat{a} \in \mathbb{Z}_{\widehat{q}}$ there exists $b \in \mathbb{Z}_{q\cdot\widehat{q}}$ such that $b = a \bmod q$ and $b = \widehat{a} \bmod \widehat{q}$. So *any* $(A,\widehat{A})$ is a Yes instance for this variant. In typical applications, we set the bound d_v so that $q > \mathsf{s} \cdot d_v$. Hence, $(A,\widehat{A})$ is a Yes instance only if the *unique* discrete log $a \in [q-1]$ of A, is a discrete log of $\widehat{A}$.

The protocol is again straightforward. In the preamble stage, the verifier samples pp using Gen and proves to the prover that it is well formed. Then the prover sends $\widetilde{A} \leftarrow \mathsf{IntCom}_{\mathsf{pp}}(a;\rho)$ to the verifier, and the parties interact in two parallel executions of Protocol 5.2: one with input $(\widetilde{A},A)$ with respect to the group $\mathbb{G}$, and the second with input $(\widetilde{A},\widehat{A})$ with respect to the group $\widehat{\mathbb{G}}$. (Actually, it is easy to see that it suffices for the verifier to send a single challenge e, to be used for both executions.)

5.4.3 Commit-and-Prove

Another immediate application of Protocol 5.2 is when $\mathbb{G}$ is taken as the additive group $\mathbb{Z}_q$, i.e., the additive group generated by 1 modulo q, assuming q does not have small factors. In this case, Protocol 5.2 allows you to commit to a bounded integer a using an integer commitment, and later expose the value of $a \bmod q$ for any suitable q. Thus, it implements the so-called commit-and-prove functionality, presented below.

Functionality 5.11 (Cmt&Prv)**.**

Parameter: $\mathsf{s} \in \mathbb{N}$.

Parties: S and R.

Commit:

Common input: $d_a \in \mathbb{N}$.

S's input: $a \in [d_a]$.

Operation: Store a.

Prove:

Common input: $q \in \mathbb{N}$, $a' \in \mathbb{Z}_q$.

Operation: Abort if $a \notin (d_a)$ or if $a' \neq a \bmod q$. A malicious S can instruct to accept $a \in \pm(\mathsf{s}{\cdot}d_a)$.[11]

6 Lightweight Leaky Well-Formedness Proof

A main obstacle in using integer commitments is the choice of the parameters $\mathsf{pp} = (n, g, h)$. Recall that for the soundness of the range and equality proofs, we require pp to be uniformly sampled without leaking the randomness used to sample it to the prover. For the hiding property to hold, however, we need $g \in \langle h \rangle_n$. We can let the verifier sample pp and prove that $g \in \langle h \rangle_n$, but unfortunately, no highly efficient proof is known for this task. While the following sigma protocol does not enforce $g \in \langle h \rangle_n$, it does guarantee that any leakage induced by committing with parameters that have passed verification is limited. In many settings, like those in this paper, this leakage can be overcome.

Protocol 6.1 (Parameters well-formedness)**.**

Parties: P, V.

Parameters: 1^{κ_s}.

Common input: $n, g, h \in \mathbb{N}$.

P's private input: $\alpha \in \mathbb{N}$.

Operation:

1. P:
   (a) Sample $\beta \overset{\mathrm{R}}{\leftarrow} [n \cdot 2^{2\kappa_\mathsf{s}}]$.
   (b) Send $c \leftarrow h^\beta \mod n$ to V.
2. V: Send $e \overset{\mathrm{R}}{\leftarrow} [2^{\kappa_\mathsf{s}}]$ to P.
3. P: Send $x \leftarrow e \cdot \alpha + \beta \bmod \varphi$ to V.
4. V: Verify $h^x = c \cdot g^e \mod n$.

[11]Namely, correctness and zero-knowledge are only guaranteed to hold for inputs in (d_a), but a malicious sender might cause the receiver to accept values in $(\mathsf{s} \cdot d_a)$.

It is clear that Protocol 6.1 is correct and zero-knowledge for the relation

$$\mathcal{R}^{\mathsf{WF}} := \{((n, g, h), \alpha) \colon g = h^{\alpha} \bmod n\}.$$

The soundness of the protocol is given in the following claim. Let $\mathsf{ord}_n(a) = |\langle a \rangle_n|$.

Definition 6.2. *For $n, h, g \in \mathbb{N}$, let $m_{n,h}(g) := \operatorname{argmin}\{\mathit{ord}_n(a) \colon a \in \mathbb{Z}_n^* \wedge a \cdot g \in \langle h \rangle_n\}$, and let $\delta_{n,h}(g) := \mathit{ord}_n(m_{n,h}(g))$.*

That is, $m_{n,h}(g)$ is the minimal element in $\mathbb{Z}_n^*$ (i.e., of smallest order) that maps g into $\langle h \rangle_n$.

Claim 6.3. *On common input $n, g, h \in \mathbb{N}$, V accepts in Protocol 6.1 with probability at most $2^{-\kappa_{\mathsf{s}}} + 1/\delta_{n,h}(g)$.*

Specifically, if V accepts with probability at least $2^{1-\kappa_{\mathsf{s}}}$, then there exist $a \in \mathbb{Z}_n^*$ of order at most $2^{\kappa_{\mathsf{s}}}$ such that $a \cdot g \in \langle h \rangle_n$. The following claims yields that if $\delta_{n,h}(g)$ is small (and then V might accept), using $\mathsf{pp} = (n, g, h)$ still yields a not-too-leaky commitment.

Claim 6.4. *Let $\mathsf{pp} = (1^{\kappa_{\mathsf{s}}}, n, h, g)$ and let $a \in \mathbb{Z}_n^*$ be such that $a \cdot g \in \langle h \rangle_n$. Then*

$$\mathrm{SD}(\mathsf{IntCmt}_{\mathsf{pp}}(1^{\kappa_{\mathsf{s}}}, x_0), \mathsf{IntCmt}_{\mathsf{pp}}(1^{\kappa_{\mathsf{s}}}, x_1)) \leq 2^{-\kappa_{\mathsf{s}}}$$

for any $x, x' \in \mathbb{Z}$ with $x = x' \bmod \mathit{ord}_n(a)$.

Namely, a commitment to x only leaks $x \bmod \delta_{n,h}(g)$.

Proof. Let $\widehat{g} \leftarrow a \cdot g$, and let $\mathsf{pp}' = (n, h, \widehat{g})$. It holds that

$$\mathsf{IntCom}_{\mathsf{pp}}(x; \rho) = h^{\rho} g^{x} = h^{\rho} \cdot \widehat{g}^{x} \cdot a^{-x} = \mathsf{IntCom}_{\mathsf{pp}'}(x; \rho) \cdot a^{-x} \bmod n.$$

Since $\widehat{g} \in \langle h \rangle_n$, it holds that $\mathsf{IntCmt}_{\mathsf{pp}'}$ is of hiding. And since $a^{-x} = a^{-x'} \bmod n$, we conclude that $\mathsf{IntCmt}_{\mathsf{pp}}(1^{\kappa_{\mathsf{s}}}, x)$ is of statistical distance at most $2^{-\kappa_{\mathsf{s}}}$ from $\mathsf{IntCmt}_{\mathsf{pp}}(x')$. □

6.1 Commitments over Rough Integer Rings

When the value to commit comes from $\mathbb{Z}_q$ for some $q \in \mathbb{N}$ that does not admit small factors, Protocol 6.1 can be used to obtain non-leaky a commitment; after using the light-setup protocol (yielding parameter pp), commit to $x \in \mathbb{Z}_q$ as follows:

Algorithm 6.5 (IntCom'**).**

Paramters: $\mathsf{pp} = (1^{\kappa_{\mathsf{s}}}, n, g, h), q$.

Input: x.

Operation:

1. *Sample $x' \overset{R}{\leftarrow} [q \cdot 2^{2\kappa_{\mathsf{s}}}]$ conditioned on $x' = x \bmod q$.*

2. *Output to $\mathsf{IntCom}_{\mathsf{pp}}(x', \rho)$ for $\rho \overset{R}{\leftarrow} [n \cdot 2^{\kappa_{\mathsf{s}}}]$.*

Claim 6.6. *Let* $\mathsf{pp} = (1^{\kappa_\mathsf{s}}, n, g, h)$ *such that Assume* $\delta_{n,h}(g) \leq 2^{\kappa_\mathsf{s}}$*. Then for any* $x_0, x_1 \in \mathbb{Z}_q$*, the statistical distance between* $\mathsf{IntCom}'(x_1)$ *and* $\mathsf{IntCom}'(x_1)$ *is at most* $2^{-\kappa_\mathsf{s}}$*.*

That is, either the light setup protocol aborts with probability $1-2^{-\kappa_\mathsf{s}}$, or IntCom' is statistically hiding.

Proof. Take the smallest integer $s' > s \leftarrow n \cdot 2^{\kappa_\mathsf{s}}$ divided by $\delta \leftarrow \delta_{n,h}(g)$. Assuming IntCom' would sample $x' \overset{\text{R}}{\leftarrow} [s']$. Since $\gcd(\delta \leftarrow \delta, q) = 1$, by CRT $\mathsf{IntCom}'(x_1)$ and $\mathsf{IntCom}'(x_1)$ are identically distributed (given pp). It follows that in the real algorithm, $\mathsf{IntCom}'(x_1)$ and $\mathsf{IntCom}'(x_1)$ are at distance at most $(s' - s)/s' \leq \delta/s \leq 2^{-\kappa_\mathsf{s}}$. □

Applications. Recall, see Section 3.2, that the zero knowledge property of all the integer commitment Sigma protocols considered in this paper hold for *any* choice (well formed or not) of the parameter pp. Hence, for values in $\mathbb{Z}_q$ (for any suitable q), using Protocol 6.1 yields a highly efficient well-formedness proof for integer commitment-based protocols.

References

[BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020 (cit. on p. 4).

[BCDv88] Ernest F. Brickell, David Chaum, Ivan Damgård, and Jeroen van de Graaf. "Gradual and Verifiable Release of a Secret". In: 1988, pp. 156–166. DOI: 10.1007/3-540-48184-2_11 (cit. on p. 4).

[Bou00] Fabrice Boudot. "Efficient Proofs that a Committed Number Lies in an Interval". In: 2000, pp. 431–444. DOI: 10.1007/3-540-45539-6_31 (cit. on p. 3).

[CBCMRW24] Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang. *SoK: Zero-Knowledge Range Proofs*. Cryptology ePrint Archive, Report 2024/430. 2024. URL: https://eprint.iacr.org/2024/430 (cit. on p. 3).

[CCs08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. "Efficient Protocols for Set Membership and Range Proofs". In: 2008, pp. 234–252. DOI: 10.1007/978-3-540-89255-7_15 (cit. on p. 4).

[CGGMP20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. "UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts". In: 2020, pp. 1769–1787. DOI: 10.1145/3372297.3423367 (cit. on p. 4).

[CGKR22] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. "Sharp: Short Relaxed Range Proofs". In: 2022, pp. 609–622. DOI: 10.1145/3548606.3560628 (cit. on p. 4).

[CKLR21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. "Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments". In: 2021, pp. 247–277. DOI: 10.1007/978-3-030-77883-5_9 (cit. on pp. 3, 4).

[CMP20] Ran Canetti, Nikolaos Makriyannis, and Udi Peled. *UC Non-Interactive, Proactive, Threshold Ecdsa.* Tech. rep. 2020/492. Cryptology ePrint Archive, 2020 (cit. on pp. 11, 12, 17).

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Removing the Strong RSA Assumption from Arguments over the Integers". In: 2017, pp. 321–350. DOI: 10.1007/978-3-319-56614-6_11 (cit. on p. 4).

[Dam10] Ivan Damgard. *On Sigma-protocols.* Manuscript. 2010. URL: https://www.cs.au.dk/ivan/Sigma.pdf (cit. on p. 5).

[DF02] Ivan Damgård and Eiichiro Fujisaki. "A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order". In: 2002, pp. 125–142. DOI: 10.1007/3-540-36178-2_8 (cit. on p. 3).

[FO97a] Eiichiro Fujisaki and Tatsuaki Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". In: 1997, pp. 16–30. DOI: 10.1007/BFb0052225 (cit. on p. 3).

[FO97b] Eiichiro Fujisaki and Tatsuaki Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". In: *Annual International Cryptology Conference (Crypto).* Ed. by Burton S. Kaliski Jr. 1997, pp. 16–30 (cit. on p. 11).

[GG18] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: 2018, pp. 1179–1194. DOI: 10.1145/3243734.3243859 (cit. on p. 4).

[Gro05] Jens Groth. "Non-interactive Zero-Knowledge Arguments for Voting". In: 2005, pp. 467–482. DOI: 10.1007/11496137_32 (cit. on p. 4).

[Gro11] Jens Groth. "Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments". In: 2011, pp. 431–448. DOI: 10.1007/978-3-642-25385-0_23 (cit. on p. 4).

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11 (cit. on p. 4).

[Lin17] Yehuda Lindell. "Fast Secure Two-Party ECDSA Signing". In: 2017, pp. 613–644. DOI: 10.1007/978-3-319-63715-0_21 (cit. on p. 4).

[Lip03] Helger Lipmaa. "On Diophantine Complexity and Statistical Zero-Knowledge Arguments". In: 2003, pp. 398–415. DOI: 10.1007/978-3-540-40061-5_26 (cit. on pp. 3, 4).

[LN18] Yehuda Lindell and Ariel Nof. "Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody". In: 2018, pp. 1837–1854. DOI: 10.1145/3243734.3243788 (cit. on p. 4).

[MR04] Philip D. MacKenzie and Michael K. Reiter. "Two-party generation of DSA signatures". In: *International Journal of Information Security* 2.3-4 (2004), pp. 218–239 (cit. on p. 11).

[Pai99] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *International conference on the theory and applications of cryptographic techniques.* 1999, pp. 223–238 (cit. on p. 22).