Efficient CCA Timed Commitments in Class Groups

Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, Giulio Malavolta

Efficient CCA Timed Commitments in Class Groups

Sri Aravinda Krishnan Thyagarajan
Friedrich Alexander Universität Erlangen-Nürnberg
Nürnberg, Germany
t.srikrishnan@gmail.com

Guilhem Castagnos
Université de Bordeaux, INRIA, CNRS
Talence, France
guilhem.castagnos@math.u-bordeaux.fr

Fabien Laguillaumie
LIRMM, Univ Montpellier, CNRS
Montpellier, France
Fabien.Laguillaumie@lirmm.fr

Giulio Malavolta
Max Planck Institute for Security and Privacy
Bochum, Germany
giulio.malavolta@hotmail.it

ABSTRACT

Timed commitments [Boneh and Naor, CRYPTO 2000] are the timed analogue of standard commitments, where the commitment can be non-interactively opened after a pre-specified amount of time passes. Timed commitments have a large spectrum of applications, such as sealed bid auctions, fair contract signing, fair multi-party computation, and cryptocurrency payments. Unfortunately, all practical constructions rely on a (private-coin) trusted setup and do not scale well with the number of participants. These are two severe limiting factors that have hindered the widespread adoption of this primitive.

In this work, we set out to resolve these two issues and propose an efficient timed commitment scheme that also satisfies the strong notion of CCA-security. Specifically, our scheme has a transparent (i.e. public-coin) one-time setup and the amount of sequential computation is essentially independent of the number of participants. As a key technical ingredient, we propose the first (linearly) homomorphic time-lock puzzle with a transparent setup, from class groups of imaginary quadratic order. To demonstrate the applicability of our scheme, we use it to construct a new distributed randomness generation protocol, where n parties jointly sample a random string. Our protocol is the first to simultaneously achieve (1) high scalability in the number of participants, (2) transparent one-time setup, (3) lightning speed in the optimistic case where all parties are honest, and (4) ensure that the output random string is *unpredictable* and *unbiased*, even when the adversary corrupts $n-1$ parties. To substantiate the practicality of our approach, we implemented our protocol and our experimental evaluation shows that it is fast enough to be used in practice. We also evaluated a heuristic version of the protocol that is at least 3 orders of magnitude more efficient both in terms of communication size and computation time. This makes the protocol suitable for supporting hundreds of participants.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484773

CCS CONCEPTS

• **Security and privacy** → **Cryptography**.

KEYWORDS

Timed commitments; Distributed randomness generation

ACM Reference Format:
Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. 2021. Efficient CCA Timed Commitments in Class Groups. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3460120.3484773

1 INTRODUCTION

Timed commitments [17, 66] allow one to hide a message m into a commitment c, for a pre-specified amount of time $\mathbf{T}$. Anyone can recover the committed message m by performing a long sequential computation, which terminates (approximately) after time $\mathbf{T}$. The security property of interest is that no amount of parallel computation can give a significant advantage in opening the commitment earlier than scheduled. Specifically, we are interested in the strong notion of *chosen commitment attack (CCA)* security [49]: The message m inside the commitment c must be kept hidden until time $\mathbf{T}$, even if the adversary has access to an oracle that instantly force-opens all commitments (except for the challenge commitment c).

Timed commitments [8, 19, 25, 45, 49, 57] and timed-based cryptography [15, 40, 63, 74] have seen a recent surge of popularity in the research community, due to their wide array of applications. Examples of interest include protocols for distributed randomness generation [68], contract signing [17], e-voting [57], multi-signature transactions in cryptocurrencies [72], zero-knowledge arguments [44], and non-malleable commitments [54], among many others.

Yet, there are pertinent questions about this time-based cryptographic primitive that have remain unanswered, and consequently affected its large scale adoption. We identify two such questions that have been the major obstacles for the usage of timed commitments (and more in general timed cryptography) in real-life applications.

(1) Scalability of Sequential Computation. The major limiting factor of the timed commitment construction is the computational cost of forcefully opening the commitments. This mechanism is inherited from solving the underlying time-lock puzzles [66]. This

drawback is significantly amplified in protocols where many users participate, each with their own commitment. In these situations, the computational effort needed to learn the output and terminate the protocol might also grow with the total number of commitments. We stress that this is not only an environmental concern, but also constitutes an attack vector for *denial-of-service*: An attacker can prolong or even prevent the termination of a protocol by flooding the network with a massive amount of commitments, that all users need to forcefully open.

This issue has motivated the development of homomorphic time-lock puzzles [57] and delay encryption [25], which use additional structural properties to solve this problem: At any point in time, only one puzzle/ciphertext needs to be solved, regardless on the number of participants. However, all of these works consider only a *weak* notion of security (analogous to the CPA-security of encryption schemes) and thus are not sufficient for many applications where the stronger notion of CCA-security is required (more on this later).

(2) Necessity of Trusted Setup. Virtually all efficient timed commitment schemes rely on the sequentiality of computing squarings over RSA groups. I.e. they assume that given a group element G, the fastest algorithm to compute

$$G \xrightarrow{\text{Squaring}} G^{2^{\mathbf{T}}} \mod N$$

takes (approximately) $\mathbf{T}$ steps given only G and N, where $N = pq$ is the product of two large primes. However note that if the factorization of N is known, then this assumption is simply false. Thus many of these schemes [17, 57] are forced to rely on a trusted party to sample the RSA modulus N and not reveal its trapdoor (i.e. the prime factors p and q) to anyone. In practice, such trusted party can be substituted by an execution of a multi-party computation (MPC) protocol where a set of mutually distrustful jointly generate the RSA modulus. Efficient protocols for this task exist [34, 41], but their adoption is cumbersome and error prone. Furthermore, one needs to assume that at least one of the parties involved in this MPC protocol is behaving honestly.

A more elegant solution is to design schemes where the setup is *transparent* (a.k.a. public-coin): The random coins of the setup algorithm are not required to be kept hidden. This prevents catastrophic failures of the system, since no one knows the trapdoor and it is hard to compute a trapdoor even given the random coins. This is not a concern unique to the timed commitment settings: A large body of literature on succinct non-interactive arguments (SNARGs) aims at designing efficient protocols with the same guarantees [9, 22, 73] and there has been a strong push especially from the cryptocurrency community advocating for the usage of transparent protocols [1, 2]. Surprisingly, for the case of timed commitments, constructing an efficient scheme with a transparent setup is a largely unexplored territory, even given their wide range of applications.

In summary, the large applicability of timed commitment schemes calls for a *scalable* solution that satisfies *strong security definitions* under *minimal trust assumptions*.

1.1 Our Contribution

Our main result is a new efficient construction of CCA timed commitments with transparent setup, where the amount of sequential computation does not scale with the number of users. We then show how this scheme immediately implies an efficient and scalable distributed randomness generation protocol. We discuss our contributions in more details below.

Efficient CCA Timed Commitments. We present a concretely efficient construction of CCA timed commitments with a transparent setup algorithm. The scheme is equipped with a homomorphic evaluation algorithm that allows us to avoid the computational blowup in the number of users (the relation between CCA security and homomorphic evaluation is discussed in details in Section 1.2). Our construction can be conceptually broken down into three main steps:

(1) *Homomorphic Time-Lock Puzzle*: We build a linearly homomorphic time-lock puzzle scheme over $\mathbb{Z}_q$, for some prime q, with transparent (public-coin) setup from class groups of imaginary quadratic order [20].
(2) *Efficient Simulation-Extractable NIZKs*: To make the scheme CCA secure, we follow the Naor-Yung paradigm [59] and equip the time-lock puzzle with a non-interactive zero-knowledge (NIZK) proof π certifying the well-formedness of the puzzle. We then show how to build this proof π that is concretely efficient, has a transparent setup, and satisfies the strong notion of straight-line simulation extractability.
(3) *Cross-Group DLog Equality*: In the process of instantiating the efficient NIZK scheme we develop new techniques to efficiently prove the equality of discrete logarithm between class groups (of unknown order) and standard prime order groups, which might be of independent interest.

One caveat of relying on class groups, as opposed to RSA groups, is that the time needed to compute the setup is proportional to the time parameter $\mathbf{T}$ (although the size of the public parameters is independent of $\mathbf{T}$). This seems inevitable since, as opposed to RSA group, a trapdoor for the scheme is hard to compute even given the random coins of the setup. This means that one needs to run a one-time pre-processing phase (that lasts approximately $\mathbf{T}$ steps) to compute the public parameters. As we will discuss later, this is perfectly acceptable in many applications of interest.

The security of our time-lock puzzle (and consequently of our CCA timed commitment) relies on the sequential squaring problem over class groups. Although somewhat less studied than the sequential squaring over RSA groups, this problem has recently received a lot of attention [3, 74] and even implemented for usage in the Chia network [4].

Distributed Randomness Generation. We demonstrate the utility of our timed commitment construction by presenting a distributed randomness generation protocol among n parties, where the parties jointly generate a 256-bit random string. Our protocol is the first to simultaneously satisfying all of the following desirable properties.

(1) *All-but-one Corruption*: The output random string is unpredictable and unbiased even against an attacker that corrupts any set of $n-1$ parties. The CCA security of our timed commitments is crucial to achieve this guarantee.
(2) *Optimistic Efficiency*: In the optimistic case where all of the parties behave honestly, the protocol is extremely efficient and no sequential computation is done *at all*. In practice, the sequential

computation aspect would function as a deterrent to misbehave and we expect most of the executions to terminate without the need of force-open the commitments.

(3) *Scalability*: Even in the case where some party misbehaves, the amount of sequential computation needed to terminate the protocol is always independent of the number of parties n. Thus, increasing the number of parties does not significantly impact the performance of our protocol, due to the homomorphic properties of our CCA timed commitment scheme.

(4) *Transparent Setup*: The protocol has a one-time transparent (a.k.a. public-coin) setup and it requires otherwise minimal interaction among parties.

To the best of our knowledge, no prior distributed randomness generation protocol (even among the less practical ones) satisfied all the above properties. We discuss these prior works in more detail in Section 7.

Implementation and Experiments. We implement each of the cryptographic techniques used in this work and our results show that our CCA timed commitments are indeed practical. Due to the strong security requirements, our NIZK proof involves a large number of exponentiations which results in running times of several minutes on a single thread. This might be acceptable in applications where users have a long time to commit. Moreover these exponentiations can be parallelized which results in running times that are under a minute. Finally, we implemented a heuristic approach using a sigma protocol that we assume is simulation-extractable when transformed into a NIZK proof. We gain significant efficiency improvements both in terms of bandwidth and running time, suitable for any application.

1.2 Homomorphism vs CCA Security

An astute reader might wonder why the homomorphic property of the commitments is not at odds with the CCA-security of the primitive. It is well-known that fully-homomorphic encryption cannot be CCA secure since one can simply evaluate some trivial function (e.g. the identity) homomorphically over the challenge ciphertext and query the resulting ciphertext to the decryption oracle. However, for the case of commitments there is a subtle aspect that one needs to consider: Our CCA timed commitments come with a proof π, which guarantees that the commitment is well-formed. On the other hand, the homomorphic evaluation algorithm operates only on the commitments

$$(c_1, \pi_i, \ldots, c_n, \pi_n) \xrightarrow{\mathsf{Eval}(f,\cdot)} \tilde{c}$$

and in particular does *not* produce a validity proof $\tilde{\pi}$ for the evaluated commitment $\tilde{c}$. This immediately counters the attack outlined above: Due to the missing proof, the open/decryption oracle will refuse to open the commitment.

This however does not contradict the usefulness of the homomorphic evaluation procedure: Instead of force-opening all commitments, we can compute the function of interest homomorphically and then force-open (in time $\mathbf{T}$) the *single* resulting commitment $\tilde{c}$ that contains the function output. Depending on the number n of input commitments, the savings can be substantial. Note that this is a purely efficiency-related consideration and does not affect security, since all commitments can anyway be force-opened in (parallel) time $\mathbf{T}$.

2 TECHNICAL OVERVIEW

In this section we give a brief outline of the techniques that we develop in this work. Our technical contributions can be conceptually split into three main steps:

Step I: We construct a homomorphic time-lock puzzle from class groups of imaginary quadratic order. The scheme has a transparent setup and supports homomorphic evaluations of linear functions over $\mathbb{Z}_q$, for some prime q.

Step II: We turn our time-lock puzzle into a CCA timed commitment by augmenting it with a simulation-extractable NIZK. We then propose a new special-purpose efficient NIZK scheme with a transparent setup.

Step III: We show how our CCA timed commitments give raise to a distributed randomness generation protocol that is concretely efficient and satisfies many desirable properties.

2.1 Homomorphic Time-Lock Puzzles from Class Groups

Known constructions of linearly homomorphic time-lock puzzles (HTLP) [57] are very close to construction of linearly homomorphic encryption schemes [60]. A natural approach is thus to adapt the linearly homomorphic CL encryption scheme [31], based on class groups, and more precisely the so-called faster variant of this scheme. In a nutshell, the CL cryptosystem uses the relations between two class groups, one related to a negative number (a discriminant) $\Delta_q = -pq^3$, $Cl(\Delta_q)$, and the other one related to the square-free (a fundamental discriminant) $\Delta_K = -pq$, $Cl(\Delta_K)$. This makes it possible to build a subgroup of order q generated by an element F where the discrete logarithm problem is easy, a situation similar to the Paillier cryptosystem [60] which uses $\mathbb{Z}^*_{N^2}$ and $\mathbb{Z}^*_N$ and a subgroup of order N. Then a plaintext m is encrypted as in "lifted" Elgamal denoted by $(G^r, pk^r \cdot F^m)$ defined in a cyclic subgroup of $Cl(\Delta_q)$. The faster variant of the CL scheme works by defining G and pk in a cyclic subgroup $\mathbb{G} \subset Cl(\Delta_K)$, and encrypting m as $(G^r, \psi_q(pk^r) \cdot F^m)$ where ψ_q lifts the element to $Cl(\Delta_q)$ where F is defined. This is more suitable for our context: all the NIZK proofs that we need will be defined in $Cl(\Delta_K)$ resulting in more efficient implementation as computation in this group is faster. Furthermore, this makes it possible to use a sequential squaring assumption in $Cl(\Delta_K)$ where Δ_K is fundamental, a setting similar to verifiable delay functions (VDF) [74] based on class groups. However the security of this fast variant was not really analysed in [31]. We revisit this scheme and show that one can build a HTLP scheme (with linear homomorphism) from it, by setting the puzzle

$$Z := (G^r, \psi_q(H^r) \cdot F^m)$$

where $H := G^{2^{\mathbf{T}}}$ is output by the puzzle setup. The solving procedure simply computes H^r via repeated squaring of G^r and obtain F^m, from which it is easy to extract m since the discrete logarithm problem is easy in this subgroup. The security of our HTLP scheme relies on the HSM_{CL} assumption (introduced later in [32]), an adaptation of Paillier's DCR assumption in class groups, and a decisional

variant of the sequential squaring assumption. A technical point is the fact that one can efficiently compute square roots in $Cl(\Delta_K)$. However we observe that this has only a marginal impact on the parameters. We also show that our HTLP setup in the CL framework is compatible with a transparent setup as other cryptographic schemes based on class group. As a side contribution, we prove that the CL fast variant is IND-CPA under the HSM_{CL} assumption, which might be of independent interest.

2.2 CCA Timed Commitments

The notion of CCA security for timed commitments is analogous to the one for encryption schemes: The committed message m in c is required to be hidden (until time $\mathbf{T}$) even if the distinguisher has access to an oracle that *instantly* force-opens any commitment $c' \neq c$. This models the fact that the adversary cannot maul a commitment c to produce a valid commitment c' for a related message. One canonical approach to lift schemes to CCA secure one is the Naor-Yung paradigm [59]. Rephrased for timed commitments, the idea is to augment a commitment $\mathsf{TCom}(m)$ with

$$(\mathsf{TCom}(m), \mathsf{Enc}(pk, m), \pi)$$

where Enc is the encryption algorithm of a standard semantically secure encryption scheme, pk is a public key placed in the common public parameters, and π is a proof that certifies that the commitment and the encryption contain the same message. In the proof, one can simulate the force-opening oracle by using the secret key sk to recover m from $\mathsf{Enc}(pk, m)$. Clearly in the actual scheme, the public key pk must be sampled uniformly and without the knowledge of the sk.

How to Sample pk? The immediate first attempt to implement the above paradigm would be to sample the public key $pk := K \in \mathbb{G}$ as an element of the class group $Cl(\Delta_K)$ and use it to compute and Elgamal encryption of m as $\mathsf{Enc}(pk, m) := (G^s, K^s \cdot G^m)$. Unfortunately this simple attempt runs into an immediate barrier: There is no known algorithm to obliviously sample a well-formed public key K. In other words, the only efficient method to sample an element K (public key) in the cyclic subgroup $\mathbb{G}$ of the class group uniformly at random is to first sample an integer k (the secret key) and set $K := G^k$ where G is the generator of the group. This however requires a fully trusted (private-coin) setup, which contradicts our goal of having a transparent (public-coin) setup.

This difficulty seems to be curtailed to the class group settings, as for standard prime-order groups $\tilde{\mathbb{G}}$ we know of efficient algorithms to sample a uniform $\tilde{pk}$ without knowing the corresponding secret key. With this observation in mind, we can implement the above paradigm bridging both groups $\mathbb{G}$ and $\tilde{\mathbb{G}}$. However, this needs to be done with care, due to the structural differences among these (e.g. the groups have different orders, and the order of $\mathbb{G}$ is unknown). Recall that our time-lock puzzle is of the form

$$(Z_1, Z_2) = (G^r, \psi_q(H^r) \cdot F^m) \tag{1}$$

where (G, F) are the generators of the respective subgroups and $H := G^{2^{\mathbf{T}}}$. As discussed above, the common random string is augmented with a uniformly sampled public key $\tilde{K} \in \tilde{\mathbb{G}}$, where $\tilde{\mathbb{G}}$ is a group of prime order $\tilde{q}$. Then the commitment is augmented with a set of ciphertexts

$$\left\{(\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i} \cdot \tilde{G}^{r_i})\right\}_{i \in [\alpha]} \tag{2}$$

where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$ is the bit-length of $\tilde{q}$, along with a proof π that certifies that $(\tilde{c}_{i,0}, \tilde{c}_{i,1})$ is indeed a "lifted" Elgamal encryption of r_i (the i-th bit of r), under the public key $\tilde{K}$. Note that this is in some sense equivalent to giving an encryption of m: In particular, in the proof, the knowledge of the secret key $\mathsf{DLog}_{\tilde{G}}(\tilde{K}) = \tilde{k}$, allows the simulator to reconstruct r and consequently recover $\psi_q(H^r)$, which in turn reveals m. Another subtlety to take into account is that we assumed that the randomness space of the time-lock puzzles matches exactly the order $\tilde{q}$. For the sake of this overview we are going to ignore these subtleties and we refer the reader to the technical sections for a precise choice of the parameters.

Efficient NIZK for Cross-Group Relations. In principle, this solution works and the security analysis can be carried out with minor modifications to the argument. However, this solution requires an efficient NIZK to prove relations across two groups of different order, one of which is unknown (the class group). Concrete efficiency for this class of statements seems to be out of reach of generic NIZK systems, let alone the ones with a transparent setup. A recent work by Alamati et al. [**?**] dealt with NIZK proofs over cross-group relations (involving RSA groups) in the standard model. On the contrary, we deal with Class groups and use the random oracle model to gain practical efficiency. Intuitively, we would like to reduce the language that we want to prove to a bunch of logical combination of discrete-logarithm equality proofs *within the same group*, for which efficient sigma protocols exist [70].

To do this, we circle back to our original idea, except that now we let the committer sample the public key K in the class group, instead of placing it in the common reference string. This way, we can use the trivial algorithm that samples an integer k and sets $K := G^k$. We also further augment the commitment with an bitwise encryption of the randomness r (as defined above), except that these ciphertext are computed in the class group, under the newly sampled key K. More concretely, our timed commitment consists of the puzzle Z from (1), prime-order group ciphertexts in (2) and

$$K, \left\{(c_{i,0}, c_{i,1}) = (G^{s_i}, K^{s_i} \cdot G^{r_i})\right\}_{i \in [\alpha]}$$

where r_i is the i-th bit of r. Our NIZK proof must now certify that the following conditions are met:

1) The public key K is correctly sampled from the class group.
2) The class group ciphertexts $\{c_{i,0}, c_{i,1}\}_{i \in [\alpha]}$ encrypt the bit decomposition of the randomness r used in Z_1.
3) Both i-th ciphertexts $(\tilde{c}_{i,0}, \tilde{c}_{i,1})$ and $(c_{i,0}, c_{i,1})$ either encrypt 0 or 1.

Statement 1) is a standard proof of knowledge of discrete logarithm (over the class group), whereas 2) can be reduced to a proof discrete logarithm equality (again over the class group) by running the linear reconstruction in the exponent. Thus the only statement that concerns two different groups is 3). Fortunately, we can split the statement as

$$(\tilde{c}_{i,0}, \tilde{c}_{i,1}) \text{ AND } (c_{i,0}, c_{i,1}) \text{ Encrypt } 0$$
$$\text{OR}$$
$$(\tilde{c}_{i,0}, \tilde{c}_{i,1}) \text{ AND } (c_{i,0}, c_{i,1}) \text{ Encrypt } 1.$$

where each clause individually can again be reduced to a standard proof of discrete logarithm equality (over the respective group). The clauses are then combined via standard OR and AND composition of sigma protocols. In some sense, the bridging across the two groups is delegated to the AND composition of sigma protocols, which can be implemented very efficiently (e.g. simply use the same challenge in the sigma protocol). Finally, statements 1), 2), and 3) are again stitched together via AND composition of sigma protocols.

2.3 Distributed Randomness Generation

We show the applicability of our CCA timed commitment by constructing a distributed randomness generation protocol: The protocol is run among n parties $(P_1, \ldots, P_n)$, and the objective is to compute a 256-bit random string r^*, that is, *unbiased*, and *unpredictable* even in the presence of $n-1$ corrupted parties, that cooperate to bias the distribution of r^*. Our protocol proceeds in three phases.

Commitment Phase: Each party P_i locally samples a random integer $x_i \leftarrow \mathbb{Z}_q$ (where $\mathbb{Z}_q$ is the message space of the timed commitments) and generates a timed commitment (c_i, π_i) to such an integer. All the timed commitments are generated with respect to a fixed time parameter $\mathbf{T}$, which conservatively bounds the duration of the commitment phase. Parties broadcast their timed commitments to other parties which locally verify the validity of each individual commitment.

(Fast Termination) Opening Phase: Each party P_i then reveals x_i along with the random coins used in generating their timed commitment (c_i, π_i). Parties can locally check if the opening is valid, by recomputing the timed commitments of other parties themselves. The final random value is computed as $r^* := \sum_{i\in[n]} x_i$ by each party. This *optimistic case*, where everyone reveals their valid openings does not require force opening of any of the timed commitments and the final value r^* is generated without the need to perform any sequential computation.

(Slow Termination) Force-Opening Phase: In the event that one or more parties do not reveal a valid opening for their commitments, the other parties need to force-open their commitments to compute r^*. To avoid a computational blowup in the number of aborting parties, this is done by evaluating the addition *homomorphically* over the commitments

$$(c_1, \ldots, c_n) \xrightarrow{\mathsf{Eval}(\mathsf{sum},\cdot)} \tilde{c}$$

where $\tilde{c}$ contains the output $\sum_{i\in[n]} x_i = r^*$. Thus the output of the protocol can be obtained by simply force-opening $\tilde{c}$. This phase can be further optimized by letting a single designated party (say P_j) compute the force-opening algorithm along with succinct proof of correctness [63, 74]. The end result is that the revealing of commitments is *publicly verifiable* even if one or many parties do not reveal their valid openings.

Analysis. The resulting output r^* is both unbiased and unpredictable. In our analysis, we show a *stronger* statement: for an adversary running in time at most $\mathbf{T}$, r^* is computationally indistinguishable from a value sampled uniformly from $\mathbb{Z}_q$. For this analysis to go through, the CCA security of the timed commitment is crucial: Intuitively, it prevents the adversary from mauling honestly generated commitments and choosing its own x_i as a function of the honestly committed values, which would ultimately result in a biased distribution.

3 PRELIMINARIES

We denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\mathsf{in}; r)$ the output of the algorithm $\mathcal{A}$ on input in using $r \leftarrow \{0,1\}^*$ as its randomness. We often omit this randomness and only mention it explicitly when required. The notation $[n]$ denotes a set $\{1, \ldots, n\}$ and $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$. We model *non-uniform probabilistic polynomial time* (PPT) adversaries as families of circuits $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ of size $\lambda^{O(1)}$ with $\lambda^{O(1)}$ input and output bits. We also consider the parallel running time of (PRAM) adversaries that we also model as circuits. The parallel time is determined by the depth of the circuit and the total running time is determined by the total size of the circuit.

Non-Interactive Zero-Knowledge Proofs. We make use of *non-interactive zero-knowledge (NIZK)* proof [14] for a language $\mathcal{L}$ that allows a prover to convince a verifier about the validity of a certain statement $stmt \in \mathcal{L}$ without revealing any other information. We require a NIZK proof to satisfy the properties of zero-knowledge, and simulation soundness [67]. We recall the formal definitions in Appendix A.

Homomorphic Time-Lock Puzzles. Time-lock puzzles [66] allow one to hide a secret for a certain amount of time $\mathbf{T}$. A homomorphic time-lock puzzle additionally offers homomorphic evaluation of several puzzles to generate a single puzzle. The notion was proposed by Malavolta and Thyagarajan [57]. It consists of a setup algorithm (PSetup), that takes as input a time hardness parameter $\mathbf{T}$ and outputs public parameters of the system pp, a puzzle generation algorithm (PGen) that, on input the public parameter and a message, generates the corresponding puzzle. One can then evaluate homomorphically functions over encrypted messages (PEval) and solve the resulting puzzle in time $\mathbf{T}$ (Solve). The security requirement is that for every PRAM adversary $\mathcal{A}$ of running time $\leq \mathbf{T}^\epsilon(\lambda)$ the messages encrypted are computationally hidden. They also propose efficient constructions for linear and multiplicative homomorphism based on the sequential squaring assumption in the RSA group. Below we recall the formal definitions from [57]. We recall the formal definitions in Appendix A.

Class Groups. Given a non square integer $\Delta < 0$, called discriminant, the imaginary quadratic order of discriminant Δ, denoted $\mathcal{O}_\Delta$ is the ring $\mathbb{Z}[(\Delta + \sqrt{\Delta})/2]$. The associated class group $Cl(\Delta)$ is defined as the quotient of the group of (invertible fractionnal) ideals of $\mathcal{O}_\Delta$ quotiented by the subgroup of principal ideals. This a finite abelian group, with an efficiently computable group law and a compact representation of elements. Basically, elements are classes of ideals, with a unique reduced representative, which can be represented by (a, b), where $a, b \in \mathbb{N}$ are smaller that $\sqrt{|\Delta|}$, so using $\log_2(|\Delta|)$ bits. For background on this algebraic object, see [20].

Cryptography based on class groups was introduced by Buchmann and Williams in [21] using the hardness of the discrete logarithm problem in $Cl(\Delta)$. Another feature of class groups is that given Δ, the order of $Cl(\Delta)$ (called the *class number*) is only known to be computable in sub-exponential time. Consequently, these groups are good candidates to implement protocols based

on groups of unknown order. This fact has lead to a revival of class groups based cryptography this last decade. First, class groups have been used for decentralised protocols without trusted setup (e.g., accumulators [56], Verifiable Delay Functions [74], Succinct Non-Interactive Argument of Knowledge [16, 23, 52]). Furthermore, a linearly homomorphic encryption scheme modulo a prime was proposed in [31] using these groups.

4 HOMOMORPHIC TIME-LOCK PUZZLE FROM CLASS GROUPS

We first revisit the setup algorithm of the so-called faster variant of the CL linearly homomorphic encryption scheme introduced by Castagnos and Laguillaumie in [31]. To start with, q is a λ-bit prime describing the message space $\mathbb{Z}_q$, and we consider a fundamental discriminant $\Delta_K = -pq$ whose size $\eta(\lambda)$ is chosen such that best algorithm to compute the class number takes $O(2^\lambda)$ time. The CL setting considers another discriminant $\Delta_q = q^2\Delta_K$ and relies on the relations between the class group $Cl(\Delta_q)$ and the class group $Cl(\Delta_K)$.

More precisely, two maps are crucial in the design of our time-lock puzzle: First the injective map $\psi_q : Cl(\Delta_K) \to C(\Delta_q)$ that maps a class $a \in Cl(\Delta_K)$ to b^q where $b \in Cl(\Delta_q)$ is the class of the ideal $I \cap \mathcal{O}_{\Delta_q}$ where I is a representative ideal (prime to q) of the class a. The other one is the surjective map $\varphi_q : Cl(\Delta_q) \to C(\Delta_K)$ which maps the class $a \in Cl(\Delta_q)$ to the class of $I\mathcal{O}_{\Delta_K}$ where I is a representative ideal (prime to q) of the class a. Note the important properties: for all $a \in Cl(\Delta_K)$, $\varphi_q(\psi_q(a)) = a^q$ and for all $a \in Cl(\Delta_q)$, $\psi_q(\varphi_q(a)) = a^q$. See [31] for details and algorithms to compute these maps.

The kernel of φ_q is a subgroup of $Cl(\Delta_q)$ of order q where the discrete logarithm problem is *easy*. A canonical generator of this subgroup is the class $F \in Cl(\Delta_q)$ represented by (q^2, q). We denote $\mathsf{Solve}_{\mathsf{DL}}$ the polynomial time algorithm that computes discrete logarithms in basis F (cf. [31, Fig. 2]).

Castagnos and Laguillaumie choose the primes p and q such that $pq \equiv -1 \mod 4$ and q is not a square modulo p (in other words, the Legendre symbol $(q/p) = -1$). This ensures that the subgroup of squares of $Cl(\Delta_K)$ has odd (unknown) order $\tilde{s}$ which is half the class number. An upper bound B on $\tilde{s}$ can be computed using the class number formula (cf [58]). It will be needed to sample exponents uniformly. More precisely, exponents will be sampled in $\mathbb{Z}_{\tilde{q}}$ where $\tilde{q} > 2^\lambda B$ is a prime. The reason why we need to use a prime number will become clear in our construction of CCA timed commitments.

We will work with a cyclic subgroup $\mathbb{G} \subset Cl(\Delta_K)$, generated by a random square G of $Cl(\Delta_K)$ (such an element can be efficiently generated by generating an ideal of $\mathcal{O}_{\Delta_K}$ above a random splitting prime, cf [48, Subsection 3.1]). We denote s the (unknown) order of $\mathbb{G}$ which is a divisor of $\tilde{s}$. For large q we can assume that $\gcd(s, q) = 1$.

In $Cl(\Delta_q)$, we will work with a cyclic subgroup Γ of the squares of $Cl(\Delta_q)$ generated by γ where $\gamma := \gamma_q F$ with $\gamma_q = \psi_q(G)$. We thus have that Γ is of order qs and satisfy $\Gamma \simeq \Gamma^q \times \langle F \rangle$, where $\Gamma^q = \langle \gamma_q \rangle$ is the subgroup of q−th powers, $\Gamma^q := \{a^q, a \in \Gamma\}$. The HSM_{CL} assumption (introduced in [32], cf Definition B.3) states that given an element δ of Γ it is hard to tell if $\delta \in \Gamma^q$ or not. It can be seen as an adaptation of Paillier's DCR assumption in the CL setting.

The class group generator CGGen depicted in Fig. 1 outputs all these parameters: $\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}$. Note that this is a public coin setup: the seeds of the probabilistic prime generators to compute $p, \tilde{q}$ and G are published.

The fast variant of the CL encryption scheme that we consider is described in Appendix C. As a side contribution, we prove in Theorem C.1 that with a slight modification of the definition of G, the indistinguishability of the scheme holds under the HSM_{CL} assumption (the security of this variant was not really analysed in [31]).

Our new homomorphic time-lock puzzle from class group is depicted in Figure 2. The setup consists in running CGGen and computing $H := G^{2^\mathrm{T}}$. Then the puzzle is

$$Z := (Z_1, Z_2) := (G^r, \psi_q(H^r) \cdot F^m)$$

Note that $Z_2 \in \Gamma$ and $\psi_q(H^r)$ is an element of Γ^q. As a result, retrieving F^m from Z_2 corresponds to solving the computational subgroup decomposition problem $\Gamma \simeq \Gamma^q \times \langle F \rangle$ associated to the HSM_{CL} assumption. This is done when solving the puzzle, by computing $\psi_q(H^r)$ as $\psi_q((G^r)^{2^\mathrm{T}})$, and then retrieving m from F^m using the $\mathsf{Solve}_{\mathsf{DL}}$ algorithm. The homomorphic property of the scheme follows from the Elgamal structure of the puzzle and the fact that ψ_q is an homomorphism.

$\underline{\mathrm{CGGen}(1^\lambda, q)}$: On input the security parameter 1^λ, and a λ bits prime q do the following:

- Let μ be the bit size of q. Pick p a $\eta(\lambda) - \mu$ bits prime such that $pq \equiv -1 \pmod 4$ and $(q/p) = -1$
- $\Delta_K := -pq$, $\Delta_q := q^2\Delta_K$
- Compute B an upper bound on the order of $Cl(\Delta_K)$
- Pick a random prime $\tilde{q} \in [2^\lambda B, 2^{\lambda+1}B]$
- Generate a random square $G \in Cl(\Delta_K)$
- Compute $\gamma_q = \psi_q(G)$
- Set F the class (q^2, q) in $Cl(\Delta_q)$
- Set $\gamma := \gamma_q \cdot F$ and $\mathbb{G} = \langle G \rangle$
- Output $pp := (\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q})$

Figure 1: Class Group Generator

Analysis. Intuitively, from the structure of Z_2, the puzzle hides m under the HSM_{CL} assumption for adversaries that cannot distinguish $\psi_q((Z_1)^{2^\mathrm{T}})$ from random. In the following we thus recall a definition of a strong sequential squaring assumption, which states that knowing of the group structure does not help to break the sequentially of the squaring operation, and analyse it in the context of class groups.

Definition 4.1 (Strong Sequential Squaring Assumption ([57])). *Let $\lambda \in \mathbb{N}$, q be a λ-bit prime, and $(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q})$ the output of $\mathrm{CGGen}(1^\lambda, q)$ and $\mathbf{T}(\cdot)$ be a polynomial. Then there exists some $0 < \varepsilon < 1$ such that for every polynomial-size adversary $(\mathcal{S}_1, \mathcal{S}_2) = \{(\mathcal{S}_1, \mathcal{S}_2)_\lambda\}_{\lambda \in \mathbb{N}}$ where the depth of $\mathcal{S}_2$ is bounded from above by $\mathbf{T}^\varepsilon(\lambda)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that*

PSetup$(1^\lambda, 1^{\mathbf{T}}, q)$: On input the security parameter 1^λ, the time parameter $1^{\mathbf{T}}$, and a λ bits prime q do the following:

- Run CGGen$(1^\lambda, q)$ to get $(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q})$
- Set $H := G^{2^{\mathbf{T}}}$
- Output $pp := (G, H, F, \tilde{q})$

PGen(pp, m): On input public parameters $pp = (G, H, F, \tilde{q})$ and a message m, do the following:

- Sample $r \leftarrow \mathbb{Z}_{\tilde{q}}$
- Output $Z := (G^r, \psi_q(H^r) \cdot F^m)$

PEval$(\{Z_1, \ldots, Z_n\})$: On input $Z_i = (U_i, V_i)$ for all i, do the following:

- Compute $\tilde{U} := \prod_{i=1}^n U_i$ and $\tilde{V} = \prod_{i=1}^n V_i$
- Output $Z^* := (\tilde{U}, \tilde{V})$

Solve(Z): On input $Z = (U, V)$, do the following:

- Compute $\omega := U^{2^{\mathbf{T}}}$
- Output $\mathsf{Solve}_{\mathsf{DL}}(V \cdot \psi_q(\omega)^{-1})$

Figure 2: Homomorphic Time-Lock Puzzle from Class Group

$$\Pr\left[b' = b \middle| \begin{array}{c} \tau \leftarrow \mathcal{S}_1(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}, \mathbf{T}(\lambda)) \\ x \leftarrow \mathbb{Z}_{\tilde{q}}; X := G^x \\ H_0 := X^{2^{\mathbf{T}(\lambda)}} \\ y \leftarrow \mathbb{Z}_{\tilde{q}}; H_1 := G^y; \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{S}_2(X, H_b, \tau) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Computational versions of the sequential squaring assumption have been used within class groups in the context of VDF [74]. In such a setting, the factorization of the discriminant Δ_K is usually public. As a consequence one can efficiently compute square roots in $\mathbb{G} \subset Cl(\Delta_K)$ using an algorithm from Lagarias [51], while it is not possible in $\mathbb{Z}/n\mathbb{Z}$ when n is an RSA modulus of unknown factorization. From [36, Prop. 3.11], there are two elements of order dividing 2 in $Cl(\Delta_K)$, one is the neutral element, and the other one, ϵ has order exactly 2 and it is not a square in our setting where the subgroup of squares has odd order. As the order s of $\mathbb{G}$ is odd, each element of $\mathbb{G}$ has at least a square root $S \in \mathbb{G}$. The other one, $S \cdot \epsilon$ is not a square so is not in $\mathbb{G}$. In other word, each element of $\mathbb{G}$ as a unique square root in $\mathbb{G}$.

For our decisional problem, we can use these facts to implement a meet in the middle attack. Namely, from a challenge, X, H_b, we can iteratively compute $X_0 := X, X_1 := X^2, X_2 := X^4, \ldots$, and in parallel $X_t := H_b$, X_{t-1} the unique square root of X_t in $\mathbb{G}$, and so on. If both ends meet, it means that H_b is equal to $X^{2^{\mathbf{T}}(\lambda)}$. In practice, this is not a huge improvement, because of the complexity of the square root algorithm, which involves computations of square roots modulo the prime factors of the discriminant and a reduction procedure of ternary quadratic forms due to Gauss. This is far more expensive than squaring in $\mathbb{G}$, and in practice our implementation using the setting of Section 8 suggests that we gain only a 5% time improvement using this strategy, we means that one has to increase $\mathbf{T}$ by 5%.

Recent improvements have been obtained to partially parallelize squarings in class groups with dedicated hardware in [75], which result in a speedup by a factor 2 compared to a standard CPU. Again, computing square roots in class group is far more intricate, but similar techniques might apply to a certain extent.

To conclude, the fact that one can compute square roots only affects marginally the time parameter of the scheme. We now state the theorem that ensures the security of our HTLP.

Theorem 4.2. *If the strong sequential squaring and HSM_{CL} assumption hold for the output of the* CGGen *generator, then the homomorphic time-lock puzzle from Figure 2 is secure.*

Proof. Let's consider a sequence of hybrid games.

Hyb_0 It is the original game.

Hyb_1 In this hybrid game, the only change is the second component of the time-lock puzzle challenge $Z^\star = (Z_1^\star, Z_2^\star)$ which is replaced by $Z_2^\star = \psi_q(H^\star) \cdot F^{s_b}$ where $H^\star = G^y$ for y picked uniformly at random in $\mathbb{Z}_{\tilde{q}}$.

Hyb_2 Again, the second component of the time-lock puzzle challenge $Z^\star = (Z_1^\star, Z_2^\star)$ is modified. It is computed as $Z_2^\star := \gamma^r \cdot F^{s_b}$ for r uniformly at random in $\mathbb{Z}_{q\tilde{q}}$.

We now analyse the transitions:

$\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$ A distinguisher between the two hybrids breaks the strong sequential squaring assumption. Indeed, let's construct $(\mathcal{S}_1, \mathcal{S}_2)$ as follows: $\mathcal{S}_1$ receives $(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}, \mathbf{T}(\lambda))$ and computes $H = G^{2^{\mathbf{T}(\lambda)}}$ (which he can do since he is not bounded by $\mathbf{T}$), feeds $\mathcal{A}_1$ with $(G, H, F, \tilde{q})$ and waits for its output (τ, s_0, s_1), which he forwards as his output for his own challenger. Then $\mathcal{S}_2$ receives $(X^\star, H^\star, \tau)$. Now $\mathcal{S}_2$ runs $\mathcal{A}_2$ with $Z^\star = (X^\star, \psi_q(H^\star) \cdot F^{s_b})$ for a random bit b. When $\mathcal{A}_2$ outputs his bit b', $\mathcal{S}_2$ outputs the bit $b == b'$. Let us now analyse the situation:

- Either $H^\star$ equals to $X^{\star 2^{\mathbf{T}(\lambda)}}$ and in this case the challenge $Z^\star = (G^x, \psi_q(H^x) \cdot F^{s_b})$ is distributed as in Hyb_0;
- or $H^\star$ is random in $\mathbb{G}$ and in this case, $Z^\star$ is distributed as in Hyb_1.

It means that any distinguisher between Hyb_0 and Hyb_1 will translate into an adversary against the strong sequential squaring assumption.

$\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ A distinguisher between Hyb_1 and Hyb_2 can be turned into an attacker against HSM_{CL}. Let us construct such an attacker $\mathcal{D}$: he takes as input $(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q})$ and δ. He computes $H = G^{2^{\mathbf{T}(\lambda)}}$ and feeds $\mathcal{A}_1$ with $(G, H, F, \tilde{q})$ and waits for its output (τ, s_0, s_1). Now $\mathcal{D}$ runs $\mathcal{A}_2$ with $Z^\star = (G^r, \delta \cdot F^{s_b})$ for a random bit b and a random r in $\mathbb{Z}_{\tilde{q}}$. When $\mathcal{A}_2$ outputs his bit b', $\mathcal{S}_2$ outputs the bit $b == b'$. We now have:

- Either $\delta = \gamma_q^x$ with $x \leftarrow \mathbb{Z}_{\tilde{q}}$. In this case, $\delta = \psi_q(G^x)$ so $Z^\star$ is distributed as in Hyb_1;
- Or $\delta = \gamma^y$ with $y \leftarrow \mathbb{Z}_{q\tilde{q}}$ so $\mathbb{Z}^\star$ is distributed as in Hyb_2.

It means that any distinguisher between Hyb_1 and Hyb_2 will translate into an attacker against the HSM_{CL} assumption, which concludes the proof. □

Expanding Message Space. Our HTLP can be generalized into a scheme with message space $\mathbb{Z}_{q^t}$ using Damgård-Jurik's ideas [37] for Paillier's encryption. This generalization was shown in [72] to be useful in constructing efficient *verifiable timed signatures* which has applications in privacy preserving timed payments in cryptocurrencies.

Indeed, starting from a discriminant $\Delta_q = q^2\delta_K$, it is possible to extend the parameters into a scheme with message space $\mathbb{Z}_{q^t}$ by considering the new discriminant $\Delta_{q_t} = q^{2^t}\Delta_k$, and $F_t := (q^{2t}, q)$ in $Cl(\Delta_{q_t})$ which is now of order q^t. The puzzle becomes $Z := (G^r, \psi_{q^t}(H^r) \cdot {F_t}^m)$ (using the appropriate mappings between class groups) and it remains to adapt the $\mathsf{Solve}_{\mathsf{DL}}$ à la Pohlig-Hellman as suggested in [31] and analysed in [38]. In this setting, a message is of size $t\log(q)$ bits whereas the ciphertext is of size $2\log(p) + (2t + 2)\log(q)$ so that the expansion factor tends to 2 when t grows to infinity.

5 CCA TIMED COMMITMENTS

In the following we introduce the notion of CCA timed commitments and we propose a new construction.

5.1 Definitions

We recall the definition of CCA timed commitment, an object recently introduced by Katz et al. [49]. Our syntax heavily borrows from their definitional framework.

Definition 5.1 (Timed Commitments). *A timed commitment scheme consists of PPT algorithms* (TSetup, TCom, TVfy, TForceOp) *that are defined below:*

$\underline{\mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}})}$*: the setup algorithm on input the security parameter* 1^λ *and the time parameter* $1^{\mathbf{T}}$ *returns a common reference string crs.*

$\underline{\mathsf{TCom}(crs, m)}$*: the commitment algorithm takes as input a common reference string crs and a message m and returns a commitment c and a proof* π*.*

$\underline{\mathsf{TVfy}(crs, c, \pi)}$*: the verification algorithm takes as input a common reference string crs, a commitment c, and a proof* π*, and returns a bit* $b \in \{0, 1\}$*.*

$\underline{\mathsf{TForceOp}(crs, c)}$*: the force opening algorithm on input a common reference string crs and a commitment c, returns a message m.*

As a note on efficiency, all algorithms should run in time polynomial in the security parameter and poly-logarithmic in $\mathbf{T}$, except for the TForceOp and (possibly) the TSetup algorithms. For correctness, we require that for all $\lambda \in \mathbb{N}$, all time parameters $\mathbf{T} \in \mathbb{N}$, and all messages m it holds that

$$\Pr[\mathsf{TForceOp}(crs, c) = m] = 1 \text{ and } \Pr[\mathsf{TVfy}(crs, c, \pi) = 1] = 1$$

where $crs \leftarrow \mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}})$ and $(c, \pi) \leftarrow \mathsf{TCom}(crs, m)$. We define the properties perfect binding, verifiability and CCA security in the following.

Definition 5.2 (Perfect Binding). *A timed commitment* (TSetup, TCom, TVfy, TForceOp) *is perfectly binding if for all* $\lambda \in \mathbb{N}$*, all time parameters* $\mathbf{T} \in \mathbb{N}$*, all* (m_0, m_1) *such that* $m_0 \neq m_1$*, and all* $(r_0, r_1) \in \{0, 1\}^{2\lambda}$ *it holds that*

$$\mathsf{TCom}(crs, m_0; r_0) \neq \mathsf{TCom}(crs, m_1; r_1)$$

where $crs \leftarrow \mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}})$.

Definition 5.3 (Verifiability). *A timed commitment* (TSetup, TCom, TVfy, TForceOp) *is verifiable if there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, all time parameters* $\mathbf{T} \in \mathbb{N}$*, and all PPT algorithms* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{TVfy}(crs, c, \pi) & crs \leftarrow \mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}}) \\ \wedge\ c \notin \mathsf{TCom}(crs, m) & (c, \pi) \leftarrow \mathcal{A}(crs) \\ & m \leftarrow \mathsf{TForceOp}(crs, c) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 5.4 (CCA Security). *A timed commitment* (TSetup, TCom, TVfy, TForceOp) *is CCA secure with gap* $\epsilon < 1$ *if there exists a negligible function* $\mathsf{negl}(\cdot)$*, a polynomial* $\tilde{\mathbf{T}}$ *such that for all polynomials* $\mathbf{T} > \tilde{\mathbf{T}}$ *and all* $\lambda \in \mathbb{N}$*, all PRAM algorithms* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *where* $\mathcal{A}_2$*'s parallel running time is bounded by* $\mathbf{T}^\epsilon$*, it holds that*

$$\Pr\left[\begin{array}{l|l} b = b' & crs \leftarrow \mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}}) \\ \wedge\ c \notin Q & (m_0, m_1) \leftarrow \mathcal{A}_1^{\mathcal{O}}(crs) \\ & b \leftarrow \{0, 1\} \\ & (c, \pi) \leftarrow \mathsf{TCom}(crs, m_b) \\ & b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(c, \pi) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ *is an oracle to which the adversary can query with* (c, π) *and if* $\mathsf{TVfy}(crs, c, \pi) = 1$*, the oracle uses* $\mathsf{TForceOp}(crs, c)$ *and returns the output. Here* Q *denotes the set of commitments queried by* $\mathcal{A}$ *to the oracle* $\mathcal{O}$*.*

Homomorphic Evaluation. We define an additional homomorphic evaluation algorithm that is going to be useful for our main scheme.

Definition 5.5 (Homomorphic Evluation). *A homomorphic evaluation algorithm* TEval *for a function family* $\mathcal{F}$ *is defined as follows.*

$\underline{\mathsf{TEval}(crs, f, (c_1, \ldots, c_n))}$*: On input a common reference string crs, a function* $f \in \mathcal{F}$*, and a set of commitments* $(c_1, \ldots, c_n)$*, the evaluation algorithm returns a new commitment* $\tilde{c}$*.*

We only require the following notion of correctness. For all $\lambda \in \mathbb{N}$, all time parameters $\mathbf{T} \in \mathbb{N}$, all functions $f \in \mathcal{F}$, and all messages $(m_1, \ldots, m_n)$ it holds that

$$\Pr[\mathsf{TForceOp}(crs, c^*) = f(m_1, \ldots, m_n)] = 1$$

where $c^* := \mathsf{TEval}(crs, f, (c_1, \ldots, c_n))$, $crs \leftarrow \mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}})$ and $(c_i, \pi_i) \leftarrow \mathsf{TCom}(crs, m_i)$.

5.2 Construction

In the following we present our efficient scheme for CCA timed commitments. We assume the existence of a homomorphic time-lock puzzle (PSetup, PGen, PEval, Solve) over $\mathbb{Z}_q$ from class groups (cf. Section 4) defined over some DDH-hard group $\mathbb{G}$ (cf. Theorem B.2) and a DDH-hard prime-order group (cf. Theorem B.1) generation algorithm $(\tilde{\mathbb{G}}, \tilde{G}, \tilde{K}) \leftarrow \mathsf{GGen}(1^\lambda, \tilde{q})$ that, on input the security parameter 1^λ and a uniformly sampled prime $\tilde{q}$ (of λ bits) returns a group description $\tilde{\mathbb{G}}$ and two generators $(\tilde{G}, \tilde{K})$. We let $\alpha := \lfloor \log \tilde{q} \rfloor + 1$.

In addition, we assume the existence of a simulation sound NIZK (Appendix E) proof system (Setup, Prv, Vfy) for each of the following languages. For groups of unknown order where computing square roots is easy, the languages we are able to prove are slightly different from those in groups of known order. In particular, we cannot prove the knowledge of the integer value of an exponent x, but we can prove the knowledge of two integers k and ρ such that $x = k \cdot 2^{-\rho}$ modulo the unknown order, which is sufficient for our applications. An honest prover always sets $\rho := 0$ when running the proving algorithm for all of the following languages.

- Language $\mathcal{L}_1$ contains all statements (G, K) such that K is generated by G, defined as

$$\mathcal{L}_1 := \left\{ (G, K) \;\middle|\; \exists\, k, \rho, \text{ s.t. } K = G^{k \cdot 2^{-\rho}} \right\}.$$

- Language $\mathcal{L}_2$ contains all the DDH-tuples, defined as

$$\mathcal{L}_2 := \left\{ (G_0, G_1, H_0, H_1) \;\middle|\; \exists\, s, \rho \text{ s.t. } H_0 = G_0^{s \cdot 2^{-\rho}} \text{ AND } H_1 = G_1^{s \cdot 2^{-\rho}} \right\}$$

- Language $\mathcal{L}_3$ contains pairs of ciphertexts encrypting the same bit, defined as

$$\mathcal{L}_3 := \left\{ \begin{array}{l|l} & \exists\, \{s_i, \tilde{s}_i, \rho_i\}_{i \in [\alpha]} \text{ s.t.} \\ & (c_{i,0}, c_{i,1}) = (G^{s_i \cdot 2^{-\rho_i}}, K^{s_i \cdot 2^{-\rho_i}}) \\ (G, K, \tilde{G}, \tilde{K}) & \quad \text{AND} \\ \{c_{i,0}, c_{i,1}\}_{i \in [\alpha]} & (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i}) \\ \{\tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i \in [\alpha]} & \quad \text{OR} \\ & (c_{i,0}, c_{i,1}) = (G^{s_i \cdot 2^{-\rho_i}}, K^{s_i \cdot 2^{-\rho_i}} \cdot G) \\ & \quad \text{AND} \\ & (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i} \cdot \tilde{G}) \end{array} \right\}$$

Efficiency and Instantiations. The setup algorithm internally runs the setup of the homomorphic time-lock puzzles, the setup of the NIZK proof system, and the group generation algorithm for the DDH-hard prime order group of order $\tilde{q}$. All of the above three algorithms are public coin algorithms (cf. Section 4 and Appendix E) and therefore our CCA timed commitment scheme also has a public coin setup. As for the efficiency, the running time of the setup is proportional to λ and $\mathbf{T}$ (the latter dependency is due to $\mathsf{PSetup}(1^\lambda, 1^{\mathbf{T}})$), however the size of the public parameters is a fixed polynomial in λ and it is in particular independent of $\mathbf{T}$.

The commitment algorithm runs the puzzle generation of the homomorphic time-lock puzzle and generates ElGamal-like ciphertexts (that encrypt single bits) both in the class group and in the prime order group. In total, we have $[\alpha]$ (where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$) ciphertexts in both groups. The algorithm also computes a NIZK proof of well-formedness (cf. Appendix E). The running time of the commitment algorithm is bounded by a fixed polynomial in λ and in particular is independent of $\mathbf{T}$. The verification algorithm simply runs the verifier routine of the NIZK proof system. The force open algorithm solves the time-lock puzzle that takes $\mathbf{T}$ sequential computational steps and the evaluation algorithm runs the homomorphic evaluation algorithm of the time-lock puzzle.

Analysis. It is easy to show that the scheme satisfies perfect correctness and perfect binding. In the below theorem we formally state the security of our CCA timed commitment construction, and defer the proof to Appendix D.

Theorem 5.6. *Let* (PSetup, PGen, PEval, Solve) *be a secure time-lock puzzle over a DDH-hard group* $\mathbb{G}$, GGen *be a DDH-hard group generator, and* (Setup, Prv, Vfy) *be simulation-sound NIZK proof system. Then the timed commitment construction from Figure 3 satisfies CCA security and verifiability.*

Highly Efficient Heuristic Variant. Provided we assume the sigma protocol for language $\mathcal{L}_2$ is simulation extractable[1] with a straight-line (i.e. non-rewinding) extractor, we can omit proofs for languages $\mathcal{L}_2$ and $\mathcal{L}_3$. Note that simulation soundness of the sigma protocol can be proven, but extraction requires rewinding. Our heuristic has a flavor of "knowledge"-type assumptions which we believe is a reasonable compromise for a significant gain in efficiency. Similar assumptions about sigma protocols are quite common: For example, several works in threshold ECDSA signatures [43, 55] propose protocols requiring UC-secure (in particular straight-line simulation extractable) NIZK. However, favoring efficiency, their actual implementation uses non-UC-secure sigma protocols.

6 DISTRIBUTED RANDOMNESS GENERATION

We now show how a CCA timed commitment allows us to build an efficient distributed randomness generation protocol.

6.1 Definition

We consider a setting where there are n parties $(P_1, \ldots, P_n)$ want to jointly compute a random string. The definitions that we present, are tailored to our settings and allow us to model the following properties of interest:

- *Public-Coin Setup*: The protocol assumes a one-time (public-coin) setup that produces a short string (pp) that is made available to all participants.
- *Non-Interactive*: The protocol consists of a *single round* of interaction among users.
- *All-but-one Corruption*: The protocol is resilient against the corruption of all but one participants.

Syntactically, the protocol consists of the following interfaces. A setup algorithm RSetup is run at the beginning that outputs the public parameters pp to all the parties in the system. The parties locally run a randomness generation algorithm RGen that outputs a randomness commitment v, which is then published on a bulletin board or broadcast to all parties. Finally, each participant can locally run the randomness computation algorithm RComp to generate the final random value. The formal interfaces are given below.

Definition 6.1 (Distributed Randomness Generation). *A* distributed randomness generation *protocol* Π_{DRG} *consists of three PPT algorithms* (RSetup, RGen, RComp) *that are defined as follows.*

$pp \leftarrow \mathsf{RSetup}(1^\lambda)$*: the* setup *algorithm takes as input the security parameter* 1^λ *and outputs the public parameters* pp.

$v \leftarrow \mathsf{RGen}(pp)$*: the* randomness generation *algorithm takes as input the public parameters* pp*, and internally samples random coins to output a randomness commitment* v.

$r \leftarrow \mathsf{RComp}(pp, \{v_1, \ldots, v_n\})$*: the* distributed randomness computation *algorithm takes as input the public parameters* pp*, a set of values* $v_1, \ldots, v_n$ *and outputs a beacon value* r.

[1] A notion where the simulator is able to simulate proofs for an adversary and is also able to extract the witness from a proof output by the adversary [46].

$\underline{\mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}})}$: On input the security parameter 1^λ and the time parameter $1^{\mathbf{T}}$, do the following:

- Sample $pp \leftarrow \mathsf{PSetup}(1^\lambda, 1^{\mathbf{T}}, q)$ and parse $pp = (G, H, F, \tilde{q})$. Sample $crs' \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\tilde{\mathbb{G}}, \tilde{G}, \tilde{K}) \leftarrow \mathsf{GGen}(1^\lambda, \tilde{q})$.
- Set $crs = (crs', pp, \tilde{\mathbb{G}}, \tilde{G}, \tilde{K})$ and output crs.

$\underline{\mathsf{TCom}(crs, m)}$: On input a common reference string crs and a message m, do the following:

- Sample $r \leftarrow \mathbb{Z}_{\tilde{q}}$, and compute $Z \leftarrow \mathsf{PGen}(pp, m; r)$, where $Z = (Z_1, Z_2) = (G^r, \psi_q(H^r) \cdot F^m)$.
- Sample $k \leftarrow \mathbb{Z}_{\tilde{q}}$ and set $K := G^k$. Let $\alpha := \lfloor \log \tilde{q} \rfloor + 1$.
- For $i \in [\alpha]$, sample $(s_i, \tilde{s}_i) \leftarrow \mathbb{Z}_{\tilde{q}}$, and let r_i is the i-th bit of r. Compute

$$(c_{i,0}, c_{i,1}) = (G^{s_i}, K^{s_i} \cdot G^{r_i}) \text{ and } (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i} \cdot \tilde{G}^{r_i})$$

- Compute the NIZK proof π for the statement $stmt := (Z, G, K, \tilde{G}, \tilde{K}, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i\in[\alpha]})$ that certifies that

$$(G, K) \in \mathcal{L}_1 \text{ and } \left(G, K, \prod_{i=1}^{\alpha} c_{i,0}^{2^{i-1}}, \prod_{i=1}^{\alpha} c_{i,1}^{2^{i-1}} \cdot Z_1^{-1}\right) \in \mathcal{L}_2 \text{ and } (G, K, \tilde{G}, \tilde{K}, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i\in[\alpha]}) \in \mathcal{L}_3$$

- Set the commitment $c = (Z, K, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i\in[\alpha]})$ and return (c, π)

$\underline{\mathsf{TVfy}(crs, c, \pi)}$: On input a common reference string crs, a commitment $c := (Z, K, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i\in[\alpha]})$ (where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$), and a proof π, return 1 if and only if $\mathsf{Vfy}(crs, stmt, \pi) = 1$.

$\underline{\mathsf{TForceOp}(crs, c)}$: On input a common reference string crs and a commitment c, return $\mathsf{Solve}(pp, Z)$.

$\underline{\mathsf{TEval}(crs, f, (c_1, \ldots, c_n))}$: On input a common reference string crs, a function $f \in \mathcal{F}$, and a set of commitments $(c_1, \ldots, c_n)$, return $\mathsf{PEval}(pp, f, (Z_1, \ldots, Z_n))$.

Figure 3: Construction of CCA Timed Commitments

In terms of security, we want that the final random value generated by RComp is indistinguishable from a uniform random string. More precisely, we consider a time bound $\mathbf{T}$ on the duration of the randomness generation protocol and we consider an adversary whose parallel running time is bounded by $\mathbf{T}$. The adversary corrupts any proper subset of the parties involved and has access to an oracle that mimics the behaviour of honest parties: That is, the oracle runs the randomness generation algorithm RGen by internally sampling random coins and returning the output to the adversary. The adversary outputs the randomness commitment values of all the corrupt parties. A bit b is chosen randomly, and if $b = 0$, the value r_0 is returned to the adversary that is generated by running RComp on the adversary's randomness commitments and the responses of the oracle queries. If $b = 1$, then r_1 is sampled uniformly and returned to the adversary. The adversary outputs a bit b' as its guess. The distributed randomness generation protocol is said to be $\mathbf{T}$-Indistinguishable Randomness (IND-RAN) if the probability that $b = b'$ is negligibly close to $1/2$. The above intuition is captured in the formal definition below.

Definition 6.2 ($\mathbf{T}$-Indistinguishable Randomness). *A distributed randomness generation protocol* Π_{DRG} *satisfies* Indistinguishable randomness (IND-RAN) *with a gap* $\epsilon < 1$, *if there exists a negligible function* negl, *a polynomial* $\tilde{\mathbf{T}}$ *such that for all polynomials* $\mathbf{T} > \tilde{\mathbf{T}}$ *and for all* $\lambda, \mathbf{T} \in \mathbb{N}$, *and all PPT adversaries* $\mathcal{A}$ *with parallel running time bounded by* $\mathbf{T}^\epsilon$, *it holds that*

$$\Pr\left[\begin{array}{l|l} b = b' \wedge & pp \leftarrow \mathsf{RSetup}(1^\lambda) \\ V' \neq \emptyset & b \leftarrow \{0,1\} \\ & V \leftarrow \mathcal{A}^{\mathsf{RGen}(pp)}(pp) \\ & r_0 \leftarrow \mathsf{RComp}(pp, V \cup V') \\ & r_1 \leftarrow \{0,1\}^\lambda \\ & b' \leftarrow \mathcal{A}(pp, r_b) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda)$$

where V' *denotes the set of answers to queries to the* RGen *oracle.*

We discuss how IND-RAN notion captures standard properties for randomness generation considered in prior works.

Unpredictability. Prior works consider adversarial machines that cannot predict any non-trivial information about the final random value with a non-negligible probability. Our IND-RAN notion models the stronger notion of computational indistinguishability with respect to a uniformly sampled string. This trivially implies unpredictability against any $\mathbf{T}$-bounded (but possibly parallel) adversary. Note that the $\mathbf{T}$ condition is necessary, since after time $\mathbf{T}$ the randomness is revealed to all participants as the output of the protocol.

Bias-Resistance. A scheme for distributed randomness generation satisfying our IND-RAN notion also satisfies bias-resistance [68]: This is because if an adversary can bias the final outcome of the protocol even by a single bit, it can also distinguish it from a truly random string with the same probability.

All-but-one corruption. The winning condition in our IND-RAN notion requires the adversary to query the $\mathsf{RGen}(pp)$ at least once, and its output is included in the computation of the random string. This models the fact that at least one honest user must be in the system, while the rest can be corrupt.

6.2 Our Protocol

We present a distributed randomness generation protocol Π_{DRG} (Figure 4) where n parties $P_1, \ldots, P_n$ jointly compute a random integer in $\mathbb{Z}_q$, for some prime q. The protocol's maximum running time is $\mathbf{T}$ and uses our homomorphic CCA timed commitment (cf. Section 5).

$\underline{\mathsf{RSetup}(1^\lambda, 1^{\mathbf{T}})}$: The setup algorithm samples and returns $pp := crs \leftarrow \mathsf{TSetup}(1^\lambda, 1^{\mathbf{T}})$.

$\underline{\mathsf{RGen}(pp)}$: The randomness generation algorithm does the following:

- Sample $s \leftarrow \mathbb{Z}_q$.
- Generate a timed commitment $(c, \pi) \leftarrow \mathsf{TCom}(crs, s)$.
- Output $v := (c, \pi)$.

$\underline{\mathsf{RComp}(pp, \{v_1, \ldots, v_n\})}$: The distributed randomness computation algorithm does the following:

- For $i \in [n]$:
  - Parse each value $v_i := (c_i, \pi_i)$.
  - Check if $\mathsf{TVfy}(crs, c_i, \pi_i) \stackrel{?}{=} 1$, if not discard v_i.
- Compute $c^* \leftarrow \mathsf{TEval}(crs, +, c_1, \ldots, c_n)$, where + denotes addition over $\mathbb{Z}_q$.
- Force open $r^* \leftarrow \mathsf{TForceOp}(crs, c^*)$.
- Output r^*.

Figure 4: Distributed Randomness Generation protocol

The setup algorithm simply runs the setup of the timed commitment to generate the public parameters. Given that our timed commitment scheme (cf. Section 5) has a non-interactive transparent (public-coin) setup, our setup algorithm for Π_{DRG} inherits the same. The randomness generation algorithm is run individually by each party P_i and internally samples a random integer in $\mathbb{Z}_q$ and generates a timed commitment to such an integer, along with a proof π. Finally, the output of the protocol is computed by combining all timed commitments (such that the corresponding proof π verifies) and solving the resulting commitment via the force open algorithm.

Optimistic Efficiency. In the optimistic case where all the parties are honest, after everyone broadcasts their randomness commitment, the parties can simply broadcast the openings: The random integer s that they committed to in the timed commitments, and the random coins used in generating the timed commitment. Parties can verify the openings in a canonical way, by recomputing the timed commitments canonically and checking if this is what was received earlier. The parties can simply add the random integers from the openings in plain, to compute the final randomness. This way no party needs to run the force opening algorithm of the timed commitments, which is the computationally expensive step.

Public Verifiability. The protocol as described above requires all parties to run the force opening algorithm (which internally runs $\mathbf{T}$ sequential squarings) to output the final randomness. We can modify our protocol to add efficient *verifiability* for this computation if the party computing the squaring operation also produces a succinct and efficiently verifiable proof of correct computation. This is exactly the same class of functions that is computed in *verifiable delay functions (VDFs)* [15]. Wesolowski [74] gave a construction of VDF for sequential squaring in the class group setting, where the user performing the squaring operations can succinctly prove that the computation was performed correctly. Verifying this proof only takes logarithmic steps in the number of squarings performed.

Now, even in the pessimistic settings where some party does not broadcast their opening, only *a single* party has to run the (expensive) sequential computation and can convince all other participants of the correctness of the computation.

Scalability. The homomorphism of the timed commitments is exploited in the distributed randomness computation algorithm: Instead of force-opening each of the n timed commitments, the homomorphic evaluation results in *a single* timed commitment to force open. Therefore as n increases, the computational cost of computing the final randomness remains (approximately) the same. This evaluation operation is very efficient and adds only negligible overhead, as it only requires $2(n-1)$ group operations for evaluating over n commitments.

Analysis. Our security guarantee is stated formally in the following theorem. We defer the formal analysis to Appendix F.

Theorem 6.3. *Let* $(\mathsf{TSetup}, \mathsf{TCom}, \mathsf{TForceOp})$ *be a perfectly binding CCA timed commitment. Then the protocol* Π_{DRG} *has* indistinguishable randomness.

7 RELATED WORK

Timed Commitments and Time-Lock Puzzles. Time-lock puzzles can be constructed in RSA groups assuming the sequentiality of the squaring operation [66]. Boneh and Naor [17] built on the above construction to introduce the notion of timed commitments. The notion of homomorphic time-lock puzzles has been recently introduced in [57] where it was shown how to build linearly and multiplicatively homomorphic time-lock puzzles over RSA groups. In a later work, Brakerski et al. [19] showed how to construct fully homomorphic time-lock puzzles, additionally assuming multi-key fully homomorphic encryption. Due to the reliance on RSA groups, all of these schemes assume a trusted setup to sample the RSA modulus N.[2] The notion of non-malleable time-lock puzzles has been recently explored in a sequence of works [8, 45, 49]. This notion is intimately related to CCA timed commitments, although all of the scheme proposed do not support homomorphic operations, nor they have been implemented. Finally, we mention that time-lock puzzles can also be constructed from supersingular isogenies [25] and succinct randomized encodings [12], although the constructions are significantly less efficient than the RSA-based ones.

Randomness Generation. There has been a long line work studying distributed randomness generation starting from Blum [13] and Rabin [65] who introduced the notion. Threshold techniques like threshold publicly-verifiable secret sharing [27, 50, 69, 71] and threshold signature schemes [26, 47] have been used to generate

[2] It has been shown [52] that in many cases one can substitute RSA groups with RSA-UFO group (where the modulus is a large randomly chosen integer), which have the advantage of having a public-coin setup. It is plausible that a similar approach would give us homomorphic time-lock puzzles with public-coin setup, however the scheme would be completely impractical.

randomness in a distributed manner. These proposals require that at least a majority of the parties are honest. Note that this is inherent, since no protocol in the standard model (as opposed to approaches that make timing-like assumptions, such as ours) can yield secure coin tossing if all-but-one parties are corrupted [35].

Verifiable Random Functions (VRFs) have been used in Algorand [33] and Ouroboros Praos [39] to generate randomness as a byproduct of their consensus mechanism. However these protocols fail to achieve strong bias-resistance guarantees as the adversary could refuse to publish a block if the randomness outcome is undesirable to him [69]. Bonneau et al. [18] and Bentov et al. [10] show that we can extract almost uniformly distributed bits from a sequence of Bitcoin blocks, but the adversary can bias the result as shown in [62].

Verifiable Delay Functions (VDFs) that have been recently studied [15, 40, 63, 74] to build bias-resistant and unpredictable randomness generation as one of the main applications. Randao [5, 6] is one such proposal where users contribute randomness in plain and the aggregation of these values is hashed and used as a starting point to compute the VDF. Similar protocols where also considered that use VDFs [24, 53, 68]. However these proposals require a trusted setup (to generate the RSA modulus N) and even when all parties are honest (optimistic case), one party still has to compute the delay function. In practice, this means that a specialized hardware that performs squaring operations is running at all times.

UC secure time-lock puzzles [8] and non-malleable time-lock puzzles [45] were used to construct coin flipping protocol that satisfied optimistic efficiency, all-but-one corruption, and public verifiability. However their protocols do not scale well with the number of users as their time-lock puzzle constructions do not support homomorphic evaluation. Moreover, we are not aware of any implementation of their proposals.

8 EXPERIMENTAL EVALUATION

We implement our CCA Timed commitment construction, run experiments and report the evaluation results here.

Experimental Setup. We have implemented our CCA Timed Commitments using Sagemath with calls to PARI native C Library [61] for the operations in class groups. All benchmarks were done using a single thread on a standard laptop (Intel Core i5-6267U @ 2.90GHz). Our experiments have been run for security levels of $\lambda = 112$ and 128 bits. The fundamental discriminant Δ_K has therefore respective sizes of 1338 bits and 1827 bits following estimates from [11]. The prime q that defines the plaintext space $\mathbb{Z}_q$ is set to have 256 bits, which is a classical target in practice for randomness generation.

As the prime $\tilde{q}$ equals $B \cdot 2^\lambda$, it has size $\alpha = 786$ bits (resp. $\alpha = 1042$ bits with $\lambda = 128$). The Elgamal encryptions needed for the NIZK proof of well-formedness are done in a group $\tilde{\mathbb{G}}$ of prime order $\tilde{q}$ instantiated as a subgroup of $\mathbb{Z}_P^*$ with $P - 1 = \ell\tilde{q}$ for a ℓ such that the prime P is of size 2048 to have a 112 bit security against DL computations (resp. size 3072 for 128 bits security). We used SHA3-256 to implement the random oracles H_1 and H_2, and SHAKE256 for H_3.

Communication Size Costs. Our time-lock puzzle Z is composed of an element of $Cl(\Delta_K)$ and one of $Cl(\Delta_q)$. At the 112 bits (resp. 128 bits) security level, this is 3208 bits (resp. 4166 bits). Notice that using a recent compression technique of [42] this could be reduced by a factor 3/4. The sizes for our CCA timed-commitment can be found in Table 1: c contains a puzzle Z together with Elgamal encryptions of the bits of the randomness used to create Z both in the class group $\mathbb{G}$ and the prime order group $\tilde{\mathbb{G}}$ and π is the NIZK proof.

Table 1: Communication cost of CCA Timed Commitments. (H) denotes our heuristic variant.

λ (bits)	crs (kB)	c (kB)	π (kB)
112	1.24	667.88	1721.34
128	1.77	1276.94	3213
112 (H)	0.47	0.4	0.48
128 (H)	0.62	0.52	0.64

One can dramatically improve these sizes using a heuristic variant. In this variant we replace the complex NIZK proof by a sigma protocol that prove the well-formedness of the time-lock puzzle Z for which we assume straight-line extractability. More precisely, to prove that $(Z_1, Z_2) = (G^r, \psi_q(H^r) \cdot F^m)$ relatively to G and H for some r and m, we can straightforwardly adapt the proof from [29] that a CL ciphertext is well formed to our fast variant, which allows a direct extraction of the message m. This reduce the size of the crs as we do not need to define $\tilde{\mathbb{G}}$. Moreover, the size of c shrinks a lot as we do not need to re-encrypt bit by bit the randomness r. The proof π now only contains this NIZK of well-formedness of a ciphertext under the vast variant of CL.

Computation Time Costs. We set the parameter $\mathbf{T} = 2^{26}$ which corresponds to an time-lock opening time of roughly 45 minutes for $\lambda = 112$ (resp. one hour for $\lambda = 128$) in our local machine. This corresponds to the timing of TForceOp. The running time of TSetup is dominated by the generation of the puzzle: the group generators CGGen and GGen only take a couple of seconds. The running time of TCom and TVfy does not depend on $\mathbf{T}$ and it is dominated by the computation of a huge number of exponentiations. Note that contrary to the solving of the puzzle, these phases can be parallelized: the running times of TCom and TVfy can be reduced by a factor of N by working with N threads. The corresponding benchmarks can be found in Table 2. Despite the complexity of our proven NIZK proof of well-formedness, the timings remain practical, especially in a context where we can use parallelization for TCom and TVfy and applications with large time-lock opening time. We also report running time for the heuristic version (lines (H)) where we obtain a highly efficient protocol. We also report in Table 3 timings with $\mathbf{T} = 2^{19}$ which gives an opening time around 5 seconds which shows that TCom and TVfy are independent of $\mathbf{T}$.

Parallelisation. Our implementation uses only a single thread and is therefore non-optimized. We expect the performance to substantially improve with optimisations. In particular, independent exponentiations during the commitment generation can be performed using several threads in parallel. For instance, a workstation with 2 Dodeca Core processors, a 112 bits security commitment takes 5 sec using 48 threads. Furthermore, optimisations in the basic arithmetic

Table 2: Running time of CCA Timed Commitments on a single thread reported in seconds with $T = 2^{26}$. (H) denotes our heuristic variant.

λ (bits)	TSetup	TCom	TVfy	TForceOp
112	2617	244	194	2594
128	3691	600	468	3682
112 (H)	2596	0.194	0.116	2584
128 (H)	3641	0.341	0.203	3635

Table 3: Running time of CCA Timed Commitments on a single thread reported in seconds with $T = 2^{17}$ for our heuristic variant.

λ (bits)	TSetup	TCom	TVfy	TForceOp
112 (H)	7.99	0.2	0.120	5.272
128 (H)	13.474	0.337	0.205	7.259

in class groups is largely unexplored unlike in finite fields, leaving open a lot of improvements in this direction.

9 CONCLUSIONS

In this work we constructed a timed commitment scheme with a transparent setup, homomorphic evaluation properties, and satisfying CCA security. Along the way, we introduced new technical tools, such as a homomorphic time-lock puzzle scheme over class groups and a new simulation-extractable NIZK proof of well-formedness, that may be of independent interest. As an application, we proposed a new distributed randomness generation protocol that satisfies many desirable efficiency and security properties. As a next step, we plan to explore further applications of our randomness generation protocol and its integration in complex scenarios such as cryptocurrencies or blockchain consensus.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their comments in improving the work. This work was supported by the French ANR ALAMBIC project (ANR-16-CE39-0006). This work was also partially supported by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) under 442893093, and by the state of Bavaria at the Nuremberg Campus of Technology (NCT).

REFERENCES

[1] [n.d.]. https://academy.binance.com/en/articles/zk-snarks-and-zk-starks-explained.
[2] [n.d.]. https://tinyurl.com/6c2ydjx2.
[3] [n.d.]. The Chia Network Blockchain. https://www.chia.net/assets/ChiaGreenPaper.pdf.
[4] [n.d.]. Chia VDF Competition Guide. https://medium.com/@chia.net/chia-vdf-competition-guide-5382e1f4bd39.
[5] [n.d.]. Minimal VDF Randomness Beacon. https://ethresear.ch/t/minimal-vdf-randomness-beacon/3566.
[6] [n.d.]. Randao++. https://www.reddit.com/comments/4mdkku.
[7] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. 2011. How to Garble Arithmetic Circuits. In *52nd FOCS*, Rafail Ostrovsky (Ed.). IEEE Computer Society Press, Palm Springs, CA, USA, 120–129. https://doi.org/10.1109/FOCS.2011.40
[8] Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. 2020. TARDIS: A Foundation of Time-Lock Puzzles in UC. Cryptology ePrint Archive, Report 2020/537. https://eprint.iacr.org/2020/537.
[9] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2018. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046. https://eprint.iacr.org/2018/046.
[10] Iddo Bentov, Ariel Gabizon, and David Zuckerman. 2016. Bitcoin Beacon. arXiv:1605.04559 [cs.CR]
[11] Jean-François Biasse, Michael J. Jacobson, and Alan K. Silvester. 2010. Security Estimates for Quadratic Field Based Cryptosystems. In *ACISP 10 (LNCS, Vol. 6168)*, Ron Steinfeld and Philip Hawkes (Eds.). Springer, Heidelberg, Germany, Sydney, NSW, Australia, 233–247.
[12] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. 2016. Time-Lock Puzzles from Randomized Encodings. In *ITCS 2016*, Madhu Sudan (Ed.). ACM, Cambridge, MA, USA, 345–356. https://doi.org/10.1145/2840728.2840745
[13] Manuel Blum. 1982. Coin Flipping by Telephone. In *Proc. IEEE Spring COMPCOM*. 133–137.
[14] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In *20th ACM STOC*. ACM Press, Chicago, IL, USA, 103–112. https://doi.org/10.1145/62212.62222
[15] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable Delay Functions. In *CRYPTO 2018, Part I (LNCS, Vol. 10991)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 757–788. https://doi.org/10.1007/978-3-319-96884-1_25
[16] Dan Boneh, Benedikt Bünz, and Ben Fisch. 2019. Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains. In *CRYPTO 2019, Part I (LNCS, Vol. 11692)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 561–586. https://doi.org/10.1007/978-3-030-26948-7_20
[17] Dan Boneh and Moni Naor. 2000. Timed Commitments. In *CRYPTO 2000 (LNCS, Vol. 1880)*, Mihir Bellare (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 236–254. https://doi.org/10.1007/3-540-44598-6_15
[18] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. 2015. On Bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015. https://eprint.iacr.org/2015/1015.
[19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. 2019. Leveraging Linear Decryption: Rate-1 Fully-Homomorphic Encryption and Time-Lock Puzzles. In *TCC 2019, Part II (LNCS, Vol. 11892)*, Dennis Hofheinz and Alon Rosen (Eds.). Springer, Heidelberg, Germany, Nuremberg, Germany, 407–437. https://doi.org/10.1007/978-3-030-36033-7_16
[20] J. Buchmann and U. Vollmer. 2007. *Binary Quadratic Forms. An Algorithmic Approach.* Springer.
[21] Johannes Buchmann and Hugh C. Williams. 1988. A Key-Exchange System Based on Imaginary Quadratic Fields. *Journal of Cryptology* 1, 2 (June 1988), 107–118. https://doi.org/10.1007/BF02351719
[22] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. 2020. Transparent SNARKs from DARK Compilers. In *Advances in Cryptology – EUROCRYPT 2020*, Anne Canteaut and Yuval Ishai (Eds.). Springer International Publishing, Cham, 677–706.
[23] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. 2020. Transparent SNARKs from DARK Compilers. In *EUROCRYPT 2020, Part I (LNCS, Vol. 12105)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, Germany, Zagreb, Croatia, 677–706. https://doi.org/10.1007/978-3-030-45721-1_24
[24] Benedikt Bünz, Steven Goldfeder, and Joseph Bonneau. 2017. Proofs-of-delay and randomness beacons in Ethereum.
[25] Jeffrey Burdges and Luca De Feo. 2020. Delay Encryption. Cryptology ePrint Archive, Report 2020/638. https://eprint.iacr.org/2020/638.
[26] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2000. Random oracles in constantipole: practical asynchronous Byzantine agreement using cryptography (extended abstract). In *19th ACM PODC*, Gil Neiger (Ed.). ACM, Portland, OR, USA, 123–132. https://doi.org/10.1145/343477.343531
[27] Ignacio Cascudo and Bernardo David. 2017. SCRAPE: Scalable Randomness Attested by Public Entities. In *ACNS 17 (LNCS, Vol. 10355)*, Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi (Eds.). Springer, Heidelberg, Germany, Kanazawa, Japan, 537–556. https://doi.org/10.1007/978-3-319-61204-1_27
[28] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2019. Two-Party ECDSA from Hash Proof Systems and Efficient Instantiations. In *CRYPTO 2019, Part III (LNCS, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 191–221. https://doi.org/10.1007/978-3-030-26954-8_7
[29] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2020. Bandwidth-efficient threshold EC-DSA. Cryptology ePrint Archive, Report 2020/084. https://ia.cr/2020/084.
[30] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2020. Bandwidth-Efficient Threshold EC-DSA. In *PKC 2020, Part II (LNCS, Vol. 12111)*, Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas (Eds.). Springer, Heidelberg, Germany, Edinburgh, UK, 266–296. https://doi.org/10.1007/978-3-030-45388-6_10

[31] Guilhem Castagnos and Fabien Laguillaumie. 2015. Linearly Homomorphic Encryption from DDH. In *CT-RSA 2015 (LNCS, Vol. 9048)*, Kaisa Nyberg (Ed.). Springer, Heidelberg, Germany, San Francisco, CA, USA, 487–505. https://doi.org/10.1007/978-3-319-16715-2_26

[32] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. 2018. Practical Fully Secure Unrestricted Inner Product Functional Encryption Modulo p. In *ASIACRYPT 2018, Part II (LNCS, Vol. 11273)*, Thomas Peyrin and Steven Galbraith (Eds.). Springer, Heidelberg, Germany, Brisbane, Queensland, Australia, 733–764. https://doi.org/10.1007/978-3-030-03329-3_25

[33] Jing Chen and Silvio Micali. 2017. Algorand. arXiv:1607.01341 [cs.CR]

[34] Megan Chen, Ran Cohen, Jack Doerner, Yashvanth Kondi, Eysa Lee, Schuyler Rosefield, and Abhi Shelat. 2020. Multiparty Generation of an RSA Modulus. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing, Cham, 64–93.

[35] Richard Cleve. 1986. Limits on the Security of Coin Flips when Half the Processors Are Faulty (Extended Abstract). In *18th ACM STOC*. ACM Press, Berkeley, CA, USA, 364–369. https://doi.org/10.1145/12130.12168

[36] D. A. Cox. 1999. *Primes of the form $x^2 + ny^2$*. John Wiley & Sons.

[37] Ivan Damgård and Mats Jurik. 2001. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *PKC 2001 (LNCS, Vol. 1992)*, Kwangjo Kim (Ed.). Springer, Heidelberg, Germany, Cheju Island, South Korea, 119–136. https://doi.org/10.1007/3-540-44586-2_9

[38] Parthasarathi Das, Michael J. Jacobson Jr., and Renate Scheidler. 2019. Improved Efficiency of a Linearly Homomorphic Cryptosystem. In *Codes, Cryptology and Information Security*. Springer, To appear.

[39] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2018. Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain. In *EUROCRYPT 2018, Part II (LNCS, Vol. 10821)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer, Heidelberg, Germany, Tel Aviv, Israel, 66–98. https://doi.org/10.1007/978-3-319-78375-8_3

[40] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. 2019. Verifiable Delay Functions from Supersingular Isogenies and Pairings. In *ASIACRYPT 2019, Part I (LNCS, Vol. 11921)*, Steven D. Galbraith and Shiho Moriai (Eds.). Springer, Heidelberg, Germany, Kobe, Japan, 248–277. https://doi.org/10.1007/978-3-030-34578-5_10

[41] Cyprien Delpech de Saint Guilhem, Eleftheria Makri, Dragos Rotaru, and Titouan Tanguy. 2021. The return of Eratosthenes: Secure Generation of RSA Moduli using Distributed Sieving. Cryptology ePrint Archive, Report 2021/565. https://eprint.iacr.org/2021/565.

[42] Samuel Dobson, Steven D. Galbraith, and Benjamin Smith. 2020. Trustless Groups of Unknown Order with Hyperelliptic Curves. Cryptology ePrint Archive, Report 2020/196. https://eprint.iacr.org/2020/196.

[43] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. 2019. Threshold ECDSA from ECDSA Assumptions: The Multiparty Case. In *2019 IEEE Symposium on Security and Privacy (SP)*. 1051–1066. https://doi.org/10.1109/SP.2019.00024

[44] C. Dwork and M. Naor. 2000. Zaps and their applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*. 283–293. https://doi.org/10.1109/SFCS.2000.892117

[45] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. 2020. Non-Malleable Time-Lock Puzzles and Applications. Cryptology ePrint Archive, Report 2020/779. https://eprint.iacr.org/2020/779.

[46] Jens Groth. 2006. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT 2006 (LNCS, Vol. 4284)*, Xuejia Lai and Kefei Chen (Eds.). Springer, Heidelberg, Germany, Shanghai, China, 444–459. https://doi.org/10.1007/11935230_29

[47] Timo Hanke, Mahnush Movahedi, and Dominic Williams. 2018. DFINITY Technology Overview Series, Consensus System. arXiv:1805.04548 [cs.DC]

[48] Detlef Hühnlein, Michael J. Jacobson Jr., Sachar Paulus, and Tsuyoshi Takagi. 1998. A Cryptosystem Based on Non-maximal Imaginary Quadratic Orders with Fast Decryption. In *EUROCRYPT'98 (LNCS, Vol. 1403)*, Kaisa Nyberg (Ed.). Springer, Heidelberg, Germany, Espoo, Finland, 294–307. https://doi.org/10.1007/BFb0054134

[49] Jonathan Katz, Julian Loss, and Jiayu Xu. 2020. On the Security of Time-Lock Puzzles and Timed Commitments. In *TCC 2020, Part III (LNCS, Vol. 12552)*, Rafael Pass and Krzysztof Pietrzak (Eds.). Springer, Heidelberg, Germany, Durham, NC, USA, 390–413. https://doi.org/10.1007/978-3-030-64381-2_14

[50] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *CRYPTO 2017, Part I (LNCS, Vol. 10401)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 357–388. https://doi.org/10.1007/978-3-319-63688-7_12

[51] J.C Lagarias. 1980. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *Journal of Algorithms* 1, 2 (1980), 142–186. https://doi.org/10.1016/0196-6774(80)90021-8

[52] Russell W. F. Lai and Giulio Malavolta. 2019. Subvector Commitments with Application to Succinct Arguments. In *CRYPTO 2019, Part I (LNCS, Vol. 11692)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 530–560. https://doi.org/10.1007/978-3-030-26948-7_19

[53] Arjen K. Lenstra and Benjamin Wesolowski. 2015. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366. https://eprint.iacr.org/2015/366.

[54] Huijia Lin, Rafael Pass, and Pratik Soni. 2017. Two-Round and Non-Interactive Concurrent Non-Malleable Commitments from Time-Lock Puzzles. In *58th FOCS*, Chris Umans (Ed.). IEEE Computer Society Press, Berkeley, CA, USA, 576–587. https://doi.org/10.1109/FOCS.2017.59

[55] Yehuda Lindell and Ariel Nof. 2018. Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody. In *ACM CCS 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, Toronto, ON, Canada, 1837–1854. https://doi.org/10.1145/3243734.3243788

[56] Helger Lipmaa. 2012. Secure Accumulators from Euclidean Rings without Trusted Setup. In *ACNS 12 (LNCS, Vol. 7341)*, Feng Bao, Pierangela Samarati, and Jianying Zhou (Eds.). Springer, Heidelberg, Germany, Singapore, 224–240. https://doi.org/10.1007/978-3-642-31284-7_14

[57] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. 2019. Homomorphic Time-Lock Puzzles and Applications. In *CRYPTO 2019, Part I (LNCS, Vol. 11692)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 620–649. https://doi.org/10.1007/978-3-030-26948-7_22

[58] Kevin S. McCurley. 1989. Cryptographic key distribution and computation in class groups. In *Number Theory and Applications (Proc. NATO Advanced Study Inst. on Number Theory and Applications, Banff, 1988)*, Richard A. Molin (Ed.). Kluwer, Boston.

[59] Moni Naor and Moti Yung. 1990. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *22nd ACM STOC*. ACM Press, Baltimore, MD, USA, 427–437. https://doi.org/10.1145/100216.100273

[60] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT'99 (LNCS, Vol. 1592)*, Jacques Stern (Ed.). Springer, Heidelberg, Germany, Prague, Czech Republic, 223–238. https://doi.org/10.1007/3-540-48910-X_16

[61] PARI Group 2020. *PARI/GP version 2.11.4*. PARI Group, Univ. Bordeaux. available from http://pari.math.u-bordeaux.fr/.

[62] Cecile Pierrot and Benjamin Wesolowski. 2016. Malleability of the blockchain's entropy. Cryptology ePrint Archive, Report 2016/370. https://eprint.iacr.org/2016/370.

[63] Krzysztof Pietrzak. 2019. Simple Verifiable Delay Functions. In *ITCS 2019*, Avrim Blum (Ed.), Vol. 124. LIPIcs, San Diego, CA, USA, 60:1–60:15. https://doi.org/10.4230/LIPIcs.ITCS.2019.60

[64] David Pointcheval and Jacques Stern. 1996. Security Proofs for Signature Schemes. In *EUROCRYPT'96 (LNCS, Vol. 1070)*, Ueli M. Maurer (Ed.). Springer, Heidelberg, Germany, Saragossa, Spain, 387–398. https://doi.org/10.1007/3-540-68339-9_33

[65] Michael O. Rabin. 1983. Randomized Byzantine Generals. In *24th FOCS*. IEEE Computer Society Press, Tucson, Arizona, 403–409. https://doi.org/10.1109/SFCS.1983.48

[66] R. L. Rivest, A. Shamir, and D. A. Wagner. 1996. *Time-lock Puzzles and Timed-release Crypto*. Technical Report. Cambridge, MA, USA.

[67] Amit Sahai. 1999. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *40th FOCS*. IEEE Computer Society Press, New York, NY, USA, 543–553. https://doi.org/10.1109/SFFCS.1999.814628

[68] Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and E. Weippl. 2020. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. *IACR Cryptol. ePrint Arch.* 2020 (2020), 942.

[69] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl. 2020. HydRand: Efficient Continuous Distributed Randomness. In *2020 IEEE Symposium on Security and Privacy (SP)*. 73–89. https://doi.org/10.1109/SP40000.2020.00003

[70] Claus-Peter Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89 (LNCS, Vol. 435)*, Gilles Brassard (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 239–252. https://doi.org/10.1007/0-387-34805-0_22

[71] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris-Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. 2017. Scalable Bias-Resistant Distributed Randomness. In *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Jose, CA, USA, 444–460. https://doi.org/10.1109/SP.2017.45

[72] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. 2020. Verifiable Timed Signatures Made Practical. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1733–1750. https://doi.org/10.1145/3372297.3417263

[73] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. 2018. Doubly-Efficient zkSNARKs Without Trusted Setup. In *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 926–943. https://doi.org/10.1109/SP.2018.00060

[74] Benjamin Wesolowski. 2019. Efficient Verifiable Delay Functions. In *EUROCRYPT 2019, Part III (LNCS, Vol. 11478)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, Germany, Darmstadt, Germany, 379–407. https://doi.org/10.1007/978-3-030-17659-4_13

[75] Danyang Zhu, Yifeng Song, Jing Tian, Zhongfeng Wang, and Haobo Yu. 2020. An Efficient Accelerator of the Squaring for the Verifiable Delay Function Over a Class Group. In *2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 137–140. https://doi.org/10.1109/APCCAS50809.2020.9301680

A MORE PRELIMINARIES

Time-Lock Puzzles. We recall the definition of standard time-lock puzzles [12]. For conceptual simplicity we consider only schemes with binary solutions.

Definition A.1 (Time-Lock Puzzles). *A time-lock puzzle is a tuple of two algorithms* (PGen, Solve) *defined as follows.*

$Z \leftarrow \mathsf{PGen}(\mathbf{T}, s)$*: a probabilistic algorithm that takes as input a hardness-parameter* $\mathbf{T}$ *and a solution* $s \in \{0,1\}$*, and outputs a puzzle* Z*.*

$s \leftarrow \mathsf{Solve}(Z)$*: a deterministic algorithm that takes as input a puzzle* Z *and outputs a solution* s*.*

The correctness requirement is that for all $\lambda \in \mathbb{N}$, for all polynomials $\mathbf{T}$ in λ, and for all $s \in \{0,1\}$, it holds $s = \mathsf{Solve}(\mathsf{PGen}(\mathbf{T}, s))$. The security definition is described below.

Definition A.2 (Security). *A scheme* (PGen, Solve) *is secure with gap* $\varepsilon < 1$ *if there exists a polynomial* $\tilde{\mathbf{T}}(\cdot)$ *such that for all polynomials* $\mathbf{T}(\cdot) \geq \tilde{\mathbf{T}}(\cdot)$ *and every polynomial-size adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *of depth* $\leq \mathbf{T}^\epsilon(\lambda)$*, there exists a negligible function* negl*, such that for all* $\lambda \in \mathbb{N}$ *it holds that*

$$\Pr\left[b \leftarrow \mathcal{A}(Z) \middle|\ Z \leftarrow \mathsf{PGen}(\mathbf{T}(\lambda), b)\ \right] = \frac{1}{2} + \mathsf{negl}(\lambda).$$

Homomorphic Time-Lock Puzzles. We formally describe the notions we require from a homomorphic time-lock puzzle scheme.

Definition A.3 (Homomorphic Time-Lock Puzzles). *Let* $C = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ *be a class of circuits and let* S *be a finite domain. A homomorphic time-lock puzzle scheme* HTLP *with respect to* C *and with solution space* S *is tuple of four algorithms* (PSetup, PGen, Solve, PEval) *defined as follows.*

$pp \leftarrow \mathsf{PSetup}(1^\lambda, \mathbf{T})$*: a probabilistic algorithm that takes as input a security parameter* 1^λ *and a time hardness parameter* $\mathbf{T}$*, and outputs public parameters* pp*.*

$Z \leftarrow \mathsf{PGen}(pp, s)$*: a probabilistic algorithm that takes as input public parameters* pp*, and a solution* $s \in S$*, and outputs a puzzle* Z*.*

$s \leftarrow \mathsf{Solve}(pp, Z)$*: a deterministic algorithm that takes as input public parameters* pp *and a puzzle* Z *and outputs a solution* s*.*

$Z' \leftarrow \mathsf{PEval}(C, pp, Z_1, \ldots, Z_n)$*: a probabilistic algorithm that takes as input a circuit* $C \in C_\lambda$*, public parameters* pp *and a set of* n *puzzles* $(Z_1, \ldots, Z_n)$ *and outputs a puzzle* Z'*.*

Security requires that the solution of the puzzles is hidden for all adversaries that run in (parallel) time less than $\mathbf{T}$. We additionally require compactness that requires that the size of the homomorphically evaluated puzzles does not depend on the function that is evaluated.

Definition A.4 (Security of HTLP [57]). *An HTLP scheme consisting of* (PSetup, PGen, Solve, PEval)*, is secure with gap* $\varepsilon < 1$ *if there exists a polynomial* $\tilde{\mathbf{T}}(\cdot)$ *such that for all polynomials* $\mathbf{T}(\cdot) \geq \tilde{\mathbf{T}}(\cdot)$ *and every polynomial-size adversary* $\mathcal{A} = \{(\mathcal{A}_1, \mathcal{A}_2)_\lambda\}_{\lambda\in\mathbb{N}}$ *where the depth of* $\mathcal{A}_2$ *is bounded from above by* $\mathbf{T}^\epsilon(\lambda)$*, there exists a negligible function* negl*, such that for all* $\lambda \in \mathbb{N}$ *it holds that*

$$\Pr\left[b = b' \middle| \begin{array}{l} pp \leftarrow \mathsf{PSetup}(1^\lambda, \mathbf{T}(\lambda)) \\ (\tau, s_0, s_1) \leftarrow \mathcal{A}_1(1^\lambda, pp) \\ b \leftarrow\!\!\$\ \{0,1\} \\ Z^\star \leftarrow \mathsf{PGen}(pp, s_b) \\ b' \leftarrow \mathcal{A}_2(pp, Z^\star, \tau) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

and $(s_0, s_1) \in S^2$*.*

Definition A.5 (Compactness [57]). *Let* $C = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ *be a class of circuits (along with their respective representations). An HTLP scheme* (PSetup, PGen, Solve, PEval) *is compact (for the class* C*) if for all* $\lambda \in \mathbb{N}$*, all polynomials* $\mathbf{T}$ *in* λ*, all circuits* $C \in C_\lambda$ *and respective inputs* $(s_1, \ldots, s_n) \in S^n$*, all* pp *in the support of* $\mathsf{PSetup}(1^\lambda, \mathbf{T})$*, and all* Z_i *in the support of* $\mathsf{PGen}(pp, s_i)$*, the following two conditions are satisfied:*

- *There exists a fixed polynomial* $p(\cdot)$ *such that* $|Z| = p(\lambda, |C(s_1, \ldots, s_n)|)$*, where* $Z \leftarrow \mathsf{PEval}(C, pp, Z_1, \ldots, Z_n)$*.*
- *There exists a fixed polynomial* $\tilde{p}(\cdot)$ *such that the runtime of* $\mathsf{PEval}(C, pp, Z_1, \ldots, Z_n)$ *is bounded by* $\tilde{p}(\lambda, |C|)$*.*

Non-Interactive Zero-Knowledge Proofs. A NIZK proof [14] allows a prover to convince a verifier about the validity of a certain statement without revealing anything beyond that. We recall the syntax in the following.

Definition A.6 (NIZK). *Let* $\mathcal{L}$ *be an NP-language with relation* $\mathcal{R}$*. A NIZK system for* $\mathcal{R}$ *consists of the following efficient algorithms.*

$crs \leftarrow \mathsf{Setup}(1^\lambda)$*: On input the security parameter* 1^λ*, the setup algorithm returns a common reference string* crs*.*

$\pi \leftarrow \mathsf{Prv}(crs, stmt, wit)$*: On input the common reference string* crs*, a statement* $stmt$*, and a witness* wit*, the prover algorithm returns a proof* π*.*

$0/1 \leftarrow \mathsf{Vfy}(crs, stmt, \pi)$*: On input the common reference string* crs*, a statement* $stmt$*, and a proof* π*, the verifier algorithm returns a bit* $b \in \{0,1\}$*.*

Correctness requires that for all $\lambda \in \mathbb{N}$ and all pairs $(stmt, wit) \in \mathcal{R}$ it holds that

$$\Pr[\mathsf{Vfy}(crs, stmt, \mathsf{Prv}(crs, stmt, wit)) = 1] = 1$$

where $crs \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$.

We recall the definition of zero-knowledge in the following.

Definition A.7 (Zero-Knowledge). *A NIZK system for* $\mathcal{R}$ *is zero-knowledge if there exists a PPT algorithm* $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that for all pairs* $(stmt, wit) \in \mathcal{R}$ *and for all PPT distinguishers the following distributions are computationally indistinguishable*

$$\Big(crs \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{Prv}(crs, stmt, wit)\Big) \approx \Big(crs^*, \pi \leftarrow \mathsf{Sim}_1(crs, stmt, td)\Big)$$

where $(crs^*, td) \leftarrow \mathsf{Sim}_0(1^\lambda)$*.*

We require that the protocol satisfies the strong notion of simulation soundness [67].

Definition A.8 (Simulation Soundness). *A NIZK system for $\mathcal{R}$ is simulation-sound if there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$ and all PPT algorithms $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{Vfy}(crs, stmt, \pi) & (crs, td) \leftarrow \mathsf{Sim}_0(1^\lambda) \\ \wedge\ stmt \notin Q \ \wedge\ stmt \notin \mathcal{L} & (\pi, stmt) \leftarrow \mathcal{A}^{O(\cdot)}(crs) \end{array}\right] = \mathsf{negl}(\lambda)$$

where O takes as input a (possibly false) statement stmt and returns $\mathsf{Sim}_1(crs, stmt, td)$ *and we denote by Q the list of queries issued by $\mathcal{A}$.*

B ASSUMPTIONS

We give the formal definition of a DDH-hard prime order group.

Definition B.1 (DDH-hard prime order group). *We say that a group generation algorithm* $\mathsf{G\tilde{G}en}$ *is DDH hard if there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ the following holds:*

$$\Pr\left[b' = b \left| \begin{array}{c} (\tilde{\mathbb{G}}, \tilde{G}, \tilde{q}) \leftarrow \mathsf{G\tilde{G}en}(1^\lambda) \\ x, y, z \leftarrow \mathbb{Z}^*_{\tilde{q}} \\ \tilde{X}_0 := \tilde{X}_1 = \tilde{G}^x \\ \tilde{Y}_0 := \tilde{Y}_1 = \tilde{G}^y \\ \tilde{Z}_0 := \tilde{G}^{xy} \text{ and } \tilde{Z}_1 := \tilde{G}^z \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(\tilde{\mathbb{G}}, \tilde{q}, \tilde{G}, \tilde{X}_b, \tilde{Y}_b, \tilde{Z}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We can extend the above definition to the case of a class group where the order of the group is not known. In this case, the values x, y, z are sampled uniformly at random from a domain exponentially larger than the upper bound on the group order.

Definition B.2 (DDH-hard unknown order group). *We say that a group generation algorithm* CGGen *is DDH hard if there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$ all λ bit primes q, and all PPT adversaries $\mathcal{A}$ the following holds:*

$$\Pr\left[b' = b \left| \begin{array}{c} (\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}) \leftarrow \mathsf{CGGen}(1^\lambda, q) \\ x, y, z \leftarrow \mathbb{Z}^*_{\tilde{q}} \\ X_0 := X_1 = G^x \\ Y_0 := Y_1 = G^y \\ Z_0 := G^{xy} \text{ and } Z_1 := G^z \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}, X_b, Y_b, Z_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We also recall the subgroup membership assumption.

Definition B.3 (Hard Subgroup Membership ([32])). *We say that a group generation algorithm* CGGen *is HSM_{CL} hard if there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$, all λ bit primes q, and all PPT adversaries $\mathcal{A}$, the following holds:*

$$\Pr\left[b' = b \left| \begin{array}{c} (\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}) \leftarrow \mathsf{CGGen}(1^\lambda, q) \\ r_0 \leftarrow \mathbb{Z}_{q\tilde{q}} \text{ and } r_1 \leftarrow \mathbb{Z}_{\tilde{q}} \\ \delta_0 := \gamma^{r_0} \text{ and } \delta_1 := \gamma_q^{r_1} \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}, \delta_b, \mathsf{Solve}_{\mathsf{DL}}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

We recall the strong root assumption for class groups.

Definition B.4 (Strong root assumption [30]). *We say that the strong root assumption holds for the class group generation* CGGen *if there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ the following holds:*

$$\Pr\left[\begin{array}{c} G = U^\ell \\ \ell \neq 1, 2^k, \forall k \end{array} \middle| \begin{array}{c} (\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}) \leftarrow \mathsf{CGGen}(1^\lambda, q) \\ H \leftarrow \mathbb{G} \\ (U, \ell) \leftarrow \mathcal{A}(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}, H) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

We recall the 2^λ-low order assumption for class groups.

Definition B.5 (γ-Low order assumption [30]). *We say that the γ-low order assumption holds for the class group generation* CGGen *for a given γ if there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ the following holds:*

$$\Pr\left[\begin{array}{c} U^\ell = 1 \\ U \neq 1 \\ 1 < \ell < \gamma \end{array} \middle| \begin{array}{c} (\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}) \leftarrow \mathsf{CGGen}(1^\lambda, q) \\ (U, \ell) \leftarrow \mathcal{A}(\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

C CL FAST VARIANT

We describe here a slightly modifed version of the faster variant of CL encryption which is sketched in [31], and provide a clean security proof under the hard subroup membership assumption, HSM_{CL}, introduced in [32] (cf. Definition B.3).

The main difference with the scheme from [31] is the fact that instead of choosing $\varphi_q(\gamma)$ as a generator of $\mathbb{G}$ we choose G as the q-th power of this element (so $\mathrm{G} = \varphi_q(\gamma)^q$) in order for the IND-CPA proof to go through. Thanks to this slight modification, we are able to prove the security of this scheme under the HSM_{CL} assumption (see Theorem C.1), instead of the "non-standard" assumption stated in [31]. Remark that s is the (unknown) order of G and of $\gamma_q = \psi_q(G)$, since ψ_q is an injective homomorphism.

Theorem C.1. *Let* CGGen *be a HSM_{CL}-hard group generator, then the above cryptosystem is IND-CPA-secure.*

Proof. We describe a sequence of games whose transitions are then analysed to show that our fast variant of CL is secure under the HSM_{CL} assumption. Recall that $G = \varphi_q(\gamma)^q$ so that $G = \varphi_q(\psi_q(R))^q = R^{q^2}$ and q is prime to s and F is in the kernel of φ_q. The fact that the composition of φ_q and ψ_q is the exponentiation to the q is crucial in the proof.

$\underline{\mathsf{Hyb}_0}$: This is the original IND-CPA game.

$\underline{\mathsf{Hyb}_1}$: The public key is computed as follows: first sample $k' \leftarrow \mathbb{Z}_{\tilde{q}}$ and set $H = \varphi_q(\gamma)^{k'}$.

$\underline{\mathsf{Setup}(1^\lambda, q)}$:

- Let μ be the bit size of q. Pick p a $\eta(\lambda) - \mu$ bits prime such that $pq \equiv -1 \pmod 4$ and $(q/p) = -1$
- $\Delta_K := -pq$, $\Delta_q := q^2\Delta_K$
- Compute B an upper bound on the order of $Cl(\Delta_K)$ and set $\tilde{q} = 2^\lambda B$
- Generate a random square $R \in Cl(\Delta_K)$
- Compute $\gamma_q = \psi_q(R)$
- Set $F := (q^2, q)$ in $Cl(\Delta_q)$
- Set $\gamma := \gamma_q \cdot F$ and compute $G = \varphi_q(\gamma)^q$
- Set $\mathbb{G} = \langle G \rangle$
- Output $pp := (\mathbb{G}, G, F, \gamma, \gamma_q, \tilde{q})$

$\underline{\mathsf{KeyGen}(\mathbb{G}, G, F, \gamma, \gamma_q)}$:

- Sample $k \leftarrow \mathbb{Z}_{\tilde{q}}$
- Compute $H = G^k$
- Return $pk = H$ and $sk = k$

$\underline{\mathsf{Encrypt}(pk, m \in \mathbb{Z}_q)}$:

- Sample $r \leftarrow \mathbb{Z}_{\tilde{q}}$
- Compute $C_1 = G^r$ and $C_2 = \psi_q(H^r) \cdot F^m$
- Return $C = (C_1, C_2)$

$\underline{\mathsf{Decrypt}(sk, (C_1, C_2))}$:

- Return $\mathsf{Solve}_{\mathsf{DL}}(C_2 \cdot \psi_q(C_1^k)^{-1})$

Figure 5: Modified fast variant of CL

$\underline{\mathsf{Hyb}_2}$: Let $Z^\star = \gamma_q^{r'}$ with $r' \leftarrow \mathbb{Z}_{\tilde{q}}$. The challenge ciphertext is made up as follows: $C_1^\star = \varphi_q(Z^\star)$ and $C_2^\star = (Z^\star)^{k'} \cdot F^{m_b}$.

$\underline{\mathsf{Hyb}_3}$: We change the definition of $Z^\star$: $Z^\star = \gamma^{r'}$ with $r' \leftarrow \mathbb{Z}_{q\tilde{q}}$.

We now argue the indistinguishability of the hybrids.

$\underline{\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1}$: In Hyb_1, $H = \varphi_q(\gamma)^{k'} = \varphi_q(\gamma)^{kq} = G^k$ for some $k \in \mathbb{Z}_s$ since $\gcd(s, q) = 1$. Furthermore, k and k' follow the same distribution in $\mathbb{Z}_s$. Therefore, the public key has the right form. Note that the simulator does not know the "correct" secret key k.

$\underline{\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2}$:

In Hyb_2, $C_1^\star = \varphi_q(\gamma_q^{r'}) = \varphi_q(\gamma)^{r'} = \varphi_q(\gamma)^{qr} = G^r$ for a r that satisfies $r' = qr \mod s$. Again, it exists since $\gcd(q, s) = 1$. On the other hand, $C_2^\star = \gamma_q^{r'k'} \cdot F^{m_b}$.

But $\psi_q(H^r) = \psi_q(\varphi_q(\gamma)^{k'})^r = \psi_q(\varphi_q(\gamma))^{k'r}$.

Since $\psi_q(\varphi_q(a)) = a^q$ for all $a \in Cl(\Delta_K)$, this is equal to $\gamma^{k'rq} = \gamma_q^{k'rq} = \gamma_q^{k'r'}$. This means that $C^\star = (G^r, \psi_q(H^r) \cdot F^{m_b}$ is a genuine ciphertext of m_b for the public key H.

$\underline{\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3}$: The indistinguishability follows from reduction against the HSM_{CL} assumption.

We now prove that $C^\star$ perfectly hides b in Hyb_3. From the challenge ciphertext one gets $C_1^\star = \varphi_q(\gamma)^{r'}$ where $r' \leftarrow \mathbb{Z}_{q\tilde{q}}$. As $\varphi_q(\gamma)$ is of order s, from an information theoretical point of view, the only information known from the adversary on r' is modulo s. But r' is closed to uniform modulo qs, and $\gcd(q, s) = 1$, so r' modulo q is still uniformly distributed for the adversary.

Eventually, $C_2^\star = (\gamma^{r'})^{k'} \cdot F^{m_b} = (\gamma_q \cdot F)^{r'k'} \cdot F^{m_b} = \gamma_q^{r'k'} \cdot F^{m_b + r'k'}$. As $r'k'$ remains uniform modulo q it acts as a one-time pad on m_b (note that $k' \neq 0$ with overwhelming probability) , which means that the challenge ciphertext does not reveal any information on m_b.

□

D SECURITY ANALYSIS OF CCA TIMED COMMITMENT

In this section we present the formal proof for the security of our CCA Timed commitment construction Figure 3.

Proof of Theorem 5.6. The proof for CCA security proceeds by defining a series of hybrid distributions and then arguing about the indistinguishability of the neighbouring experiments.

$\underline{\mathsf{Hyb}_0}$: This is identical to the original CCA experiment, except that we fix the bit $b = 0$.

$\underline{\mathsf{Hyb}_1}$: In this hybrid we compute the NIZK proof for the challenge commitment using the simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1)$.

$\underline{\mathsf{Hyb}_2}$: In this hybrid we compute, for all $i \in [\alpha]$ (where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$), $(c_{i,0}, c_{i,1})$ and $(\tilde{c}_{i,0}, \tilde{c}_{i,1})$ as encryptions of 0 in the challenge commitment. I.e., we fix $r_i = 0$, regardless on the value of r.

$\underline{\mathsf{Hyb}_3}$: In this hybrid we sample $\tilde{K}$ as a $\tilde{G}^{\tilde{k}}$ for some uniformly at random integer $\tilde{k}$ from $\mathbb{Z}_{\tilde{q}}$.

$\underline{\mathsf{Hyb}_4}$: Here we change the way we simulate the oracle O. On input a valid commitment c, instead of using $\mathsf{Solve}(Z)$ to solve the puzzle, use $\tilde{k}$ to decrypt $(\tilde{c}_{i,0}, \tilde{c}_{i,1})$ to obtain r_i, for $i \in [\alpha]$ (where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$). Compute $r := \sum_{i=1}^{\alpha} 2^i \cdot r_i$. Now recover m from (Z_1, Z_2) by computing $\mathsf{Solve}_{\mathsf{DL}}(Z_2 \cdot \psi_q(H^r)^{-1})$.

$\underline{\mathsf{Hyb}_5}$: We switch Z in the challenge ciphertext from $\mathsf{PGen}(pp, m_0)$ to $\mathsf{PGen}(pp, m_1)$.

$\underline{\mathsf{Hyb}_6 \ldots \mathsf{Hyb}_9}$: We revert the changes made in hybrids $\mathsf{Hyb}_4 \ldots \mathsf{Hyb}_1$.

Observe that Hyb_9 is identical to the CCA experiment, except with the bit b fixed to $b = 1$. To conclude the proof, we now argue on the indistinguishability of the hybrid executions.

$\underline{\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1}$: This follows from the zero-knowledge property of the NIZK proofs.

$\underline{\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2}$: The indistinguishability follows from a standard hybrid argument (over each $i \in [2 \cdot \alpha]$, where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$) and a reduction against the DDH assumption (cf. Theorem B.1 and Theorem B.2).

$\underline{\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3}$: The two hybrids define two identical distributions, so the change here is only syntactical.

$\underline{\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4}$: The only difference between the two hybrids is in the simulation of the oracle O, therefore the two hybrids differ only in the case that the output of O differs on some input query of the adversary. Observe that this can happen only if the value r extracted in Hyb_4 is not the randomness used in generating (Z_1, Z_2), i.e., $G^r \neq Z_1$.

By the simulation-soundness of the NIZK, for $\alpha := \lfloor \log \tilde{q} \rfloor + 1$, we have that

$$\begin{aligned}
&\left(G, K, \prod_{i=1}^{\alpha} c_{i,0}^{2^i}, \prod_{i=1}^{\alpha} c_{i,1}^{2^i} \cdot Z_1^{-1}\right) \\
&= \left(G, K, \prod_{i=1}^{\alpha} G^{s_i \cdot 2^{-\rho_i} \cdot 2^i}, \prod_{i=1}^{\alpha} K^{s_i \cdot 2^{-\rho_i} \cdot 2^i} \cdot G^{r_i \cdot 2^i} \cdot Z_1^{-1}\right) \\
&= \left(G, K, G^{\sum_{i=1}^{\alpha} s_i 2^{-\rho_i} \cdot 2^i}, K^{\sum_{i=1}^{\alpha} s_i 2^{-\rho_i} \cdot 2^i} \cdot G^{\sum_{i=1}^{\alpha} r_i \cdot 2^i} \cdot Z_1^{-1}\right) \\
&= \left(G, K, G^{s'}, K^{s'} \cdot G^{\sum_{i=1}^{\alpha} r_i \cdot 2^i} \cdot Z_1^{-1}\right) \in \mathcal{L}_2
\end{aligned}$$

with $s' = \sum_{i=1}^{\alpha} s_i 2^{-\rho_i} \cdot 2^i$, which in particular means

$$G^{\sum_{i=1}^{\alpha} r_i \cdot 2^i} = Z_1$$

and therefore $(r_1, \ldots, r_\alpha)$ is the bit decomposition of the discrete logarithm of Z_1 in base G. Furthermore, we have that $(\tilde{c}_{i,0}, \tilde{c}_{i,1})$ encrypts the same bit as $(c_{i,0}, c_{i,1})$, for all $i \in [\alpha]$. It follows that decrypting $(c_{i,0}, c_{i,1})$ yields a valid bit decomposition of r, the discrete logarithm of Z_1 in base G, except with negligible probability.

$\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$: The indistinguishability follows from a reduction to the hiding property of the time-lock puzzle. The only non-trivial aspect of the reduction is the running time needed to answer the queries of the adversary to the oracle $\mathcal{O}$. Note however that the running time of the simulated oracle is independent of $\mathbf{T}$, so the running time of the reduction is only a polynomial (in λ) factor slower than that of the adversary.

Indistinguishability of the hybrids $\mathsf{Hyb}_5 \ldots \mathsf{Hyb}_9$ follows along the same lines. This concludes the proof for CCA security.

The proof for verifiability follows from the soundness of the NIZK proof system. Notice that the winning condition of the verifiability property requires $\mathsf{TVfy}(crs, c, \pi) = 1$ and $c \notin \mathsf{TCom}(crs, m)$. The latter condition means that the commitments is not well-formed according to TCom. Therefore, it must be the case that $\mathsf{Vfy}(crs, \mathsf{stmt}, \pi) = 1$ and one of the following holds:

$$(G, K) \notin \mathcal{L}_1$$

or

$$\left(G, K, \prod_{i=1}^{\alpha} c_{i,0}^{2^i}, \prod_{i=1}^{\alpha} c_{i,1}^{2^i} \cdot Z_1^{-1}\right) \notin \mathcal{L}_2$$

or

$$(G, K, \tilde{G}, \tilde{K}, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i \in [\alpha]}) \notin \mathcal{L}_3$$

where $\alpha := \lfloor \log \tilde{q} \rfloor + 1$. The above event immediately contradicts the soundness of at least one of the NIZK proof systems that we use for the languages. We can therefore conclude that the probability with which the above event occurs is at most negligible in the security parameter. This concludes the proof for verifiability. □

E EFFICIENT NIZK PROTOCOLS

Let $\alpha := \lfloor \log \tilde{q} \rfloor + 1$. We consider the statement

$$\mathsf{stmt} = (Z_1, Z_2, G, K, \tilde{G}, \tilde{K}, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i \in [\alpha]})$$

as defined in Section 5. For simplicity we split the statement that we want to prove in the following languages:

- Language $\mathcal{L}_1$ contains all statements $\mathsf{stmt}_1 := (G, K)$ such that K is generated by G, defined as
$$\mathcal{L}_1 := \left\{(G, K) \;\middle|\; \exists\, k, \rho \text{ s.t. } K = G^{k \cdot 2^{-\rho}}\right\}.$$
- Language $\mathcal{L}_2$ contains statements
$$\mathsf{stmt}_2 := (G, K, H_0, H_1)$$
defined as
$$\mathcal{L}_2 := \left\{(G, K, H_0, H_1) \;\middle|\; \exists\, s, \rho \text{ s.t. } H_0 = G^{s \cdot 2^{-\rho}} \text{ AND } H_1 = K^{s \cdot 2^{-\rho}}\right\},$$
where $H_0 := \prod_{i=1}^{\alpha} c_{i,0}^{2^{i-1}}$ and $H_1 := \prod_{i=1}^{\alpha} c_{i,1}^{2^{i-1}} \cdot Z_1^{-1}$.
- Language $\mathcal{L}_3$ contains statements
$$\mathsf{stmt}_3 := (G, K, \tilde{G}, \tilde{K}, \{c_{i,0}, c_{i,1}, \tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i \in [\alpha]}),$$
defined as
$$\mathcal{L}_3 := \left\{ \begin{array}{l|l} & \exists\, \{s_i, \tilde{s}_i, \rho_i\}_{i \in [\alpha]} \text{ s.t.} \\ & (c_{i,0}, c_{i,1}) = (G^{s_i \cdot 2^{-\rho_i}}, K^{s_i \cdot 2^{-\rho_i}}) \\ (G, K, \tilde{G}, \tilde{K}) & \quad \text{AND} \\ \{c_{i,0}, c_{i,1}\}_{i \in [\alpha]} & (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i}) \\ \{\tilde{c}_{i,0}, \tilde{c}_{i,1}\}_{i \in [\alpha]} & \quad \text{OR} \\ & (c_{i,0}, c_{i,1}) = (G^{s_i \cdot 2^{-\rho_i}}, K^{s_i \cdot 2^{-\rho_i}} \cdot G) \\ & \quad \text{AND} \\ & (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i} \cdot \tilde{G}) \end{array} \right\}$$

While we present individual protocols for each language, our system will prove the conjunction of such statements. This can be achieved by standard AND composition of sigma protocols. The protocols are presented in Figures 6 to 8 and they assume three hash functions $\mathsf{H}_1, \mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_{2^\lambda}$ and $\mathsf{H}_3 : \{0,1\}^* \to (\mathbb{Z}_{2^\lambda})^\alpha$ modelled as random oracles. These functions can be obtained by a single random oracle via standard domain separation techniques, but for simplicity we treat them as independent oracles. In all protocols, we assume that the prover checks that the elements of the statements belong to the correct groups as in standard discrete log based ZK proofs. For instance, for elements of the class groups, one has to check that there are squares, which can be done in polynomial time (cf. [51]). The setup algorithm solely consists of the sampling of the corresponding hash function, and it is therefore omitted. We recall the following standard lemma, proven e.g. in [7].

Lemma E.1. *Let $U_{[0,r]}$ be the uniform distribution on the interval $[0, r]$ and $\beta \in \mathbb{Z}$. Then the statistical distance between $U_{[0,r]}$ and $U_{[0,r]} + \beta$ is β/r.*

We now proceed with the analysis our protocols. We remark that many of these proofs are already well known in the literature (e.g. some proofs for the CL encryption scheme can be found in [28, 30]) and we present them here only for completeness.

Theorem E.2 (Zero-knowledge). *The protocol in Figure 6 satisfies statistical zero-knowledge in the random oracle model.*

Proof of Theorem E.2. The simulator on input (crs, stmt), picks $t'_K \leftarrow \mathbb{Z}_Q$ and $e' \leftarrow \mathbb{Z}_{2^\lambda}$. It then computes $K'_0 := G^{t'_K}/K^{e'}$ and sets the random oracle $\mathsf{H}_1(\mathsf{stmt}, K'_0) := e'$. It outputs $\pi := (K'_0, t'_K)$ as its proof. Notice that for a randomly sampled t in the honest proof the statistical distance between t'_K and $t + e \cdot x$ is $2^\lambda \cdot \tilde{q}/Q$ (following from Lemma E.1) which is negligible. Therefore the joint distribution of (K'_0, t'_K, e') computed by the simulator is statistically close to computing $(K_0, t + e \cdot x, e)$ honestly. □

$\mathsf{Prv}_{\mathcal{L}_1}(crs, stmt, wit)$: The prover routine does the following:

- Sample $t \leftarrow \mathbb{Z}_Q$, where $Q = \tilde{q} \cdot 2^{2\lambda}$, compute $K_0 := G^t$
- Compute $e \leftarrow \mathsf{H}_1(\mathsf{stmt}, K_0)$
- Compute $t_K := t + e \cdot x$
- Set the proof $\pi := (K_0, t_K, e)$

$\mathsf{Vfy}_{\mathcal{L}_1}(crs, stmt, \pi)$: The verifier routine does the following:

- Parse $\pi := (K_0, t_K, e)$
- Check if $e \stackrel{?}{=} \mathsf{H}_1(\mathsf{stmt}, K_0)$, if so continue, otherwise output 0
- Check if $G^{t_K} \stackrel{?}{=} K_0 \cdot K^e$. If successful, output 1, else output 0.

Figure 6: Prover and Verifier routine for NIZK proof for statements in language $\mathcal{L}_1$

Theorem E.3 (Simulation Soundness). *The protocol in Figure 6 satisfies simulation soundness provided the 2^λ-low order assumption and the strong root assumption holds in $\mathbb{G}$, in the random oracle model.*

Proof of Theorem E.3. In the following we assume without loss of generality that the reduction is given ahead of time the false statement stmt and the more general claim follows with a polynomial loss (by guessing the right query of the adversary to the random oracle). The proof consists of a reduction against the 2^λ-low order assumption and the strong root assumption. Consider a reduction $\mathcal{R}$ that on input $\mathbb{G}$, generates *crs* and gives it to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ may query statements stmt to the reduction and the reduction returns simulated proofs. The reduction sets and answers random oracle queries to H_1 via lazy sampling. At some point in the execution, the adversary makes a query of the form (stmt, K_0) to the random oracle H_1. The reduction forks the execution of the game by answering with two different integers $(e, e') \leftarrow \mathbb{Z}_Q$ such that $e' \neq e$. By the forking lemma [64], with inverse polynomial probability the adversary outputs two accepting proofs $\pi := (K_0, t_K, e)$ and $\pi' := (K_0, t'_K, e')$ on the statement stmt.

The reduction computes $(t_K - t'_K)$, $(e - e')$ and

$$\gamma := \gcd(t_K - t'_K, e - e').$$

We denote

$$\mu := G^{\frac{t_K - t'_K}{\gamma}} \cdot K^{-\frac{e-e'}{\gamma}}$$

which is either 1 or different from 1. In the case $\mu \neq 1$, we clearly have $\mu^\gamma = 1$. Given the maximum value of $(e - e')$ is at most 2^λ and γ divides $(e - e')$, the reduction outputs (μ, γ) as a solution to 2^λ-low order assumption.

Now suppose that $\mu = 1$. Let us denote $E := \frac{e-e'}{\gamma}$, so that $G^{\frac{t_K - t'_K}{\gamma}} = K^E$. We have two cases here,

(1) In the first case we suppose that $E = 2^\rho$ for some integer ρ. In this case we can compute $x := \frac{t_K - t'_K}{\gamma}$ such that $G^x = K^{2^\rho}$ or equivalently $G^{x \cdot 2^{-\rho}} = K$ as G, K are checked to be in the correct groups (in our applications with class groups, one checks that G and K are squares which means that they have odd orders). But since $\mathsf{stmt} \notin \mathcal{L}_1$, this case is not possible.
(2) In the second case. We have for some (α, β) that

$$\alpha(t_K - t'_K) + \beta(e - e') = \gamma$$

which can be efficiently computed by the extended Euclidean algorithm. Observe that

$$G^\gamma = G^{\alpha(t_K - t'_K) + \beta(e-e')}$$
$$G^\gamma = G^{\alpha(t_K - t'_K)} G^{\beta(e-e')}$$
$$G^\gamma = K^{\alpha(e-e')} G^{\beta(e-e')}$$
$$G^\gamma = (K^\alpha G^\beta)^{(e-e')}.$$

The reduction outputs $(K^\alpha G^\beta, E)$ as its solution to the strong root problem since E is not a power of 2 or a solution to the 2^λ-low order assumption as before. Thus we arrive at a contradiction, which proves the simulation soundness of the protocol. □

Theorem E.4 (Zero-knowledge). *The protocol in Figure 7 satisfies statistical zero-knowledge in the random oracle model.*

Proof of Theorem E.4. The simulator samples $t'_{G,K} \leftarrow \mathbb{Z}_Q$ and $e' \leftarrow \mathbb{Z}_{2^\lambda}$. It then computes $G'_0 := G^{t'_{G,K}}/(H_0)^{e'}$ and $K'_0 := K^{t'_{G,K}}/(H_1)^{e'}$ and sets the random oracle

$$\mathsf{H}_2(\mathsf{stmt}, G'_0, K'_0) := e'.$$

It outputs the proof $\pi := (G'_0, K'_0, t'_{G,K}, e')$. By Lemma E.1 the simulated proof is statistically close to the honest one. □

Theorem E.5 (Simulation Soundness). *The protocol in Figure 7 satisfies simulation soundness provided the 2^λ-low order assumption and the strong root assumption holds in $\mathbb{G}$, in the random oracle model.*

Proof of Theorem E.5. The proof follows along the lines of the argument for Theorem E.3 and it boils down to showing that it is possible to extract a solution to the 2^λ-low order assumption or the strong root problem given two accepting transcripts with the same first message $\pi := (G_0, K_0, t_{G,K}, e)$ and $\pi' := (G_0, K_0, t'_{G,K}, e')$. □

Theorem E.6 (Zero-knowledge). *The protocol in Figure 8 is zero-knowledge in the random oracle model.*

Proof of Theorem E.6. We describe the simulator for a single index $i \in [\alpha]$ and the algorithm can be extended to the more general case in a natural way. The simulator picks $d_{i,1}, d_{i,2} \leftarrow \mathbb{Z}_{2^\lambda}$, $r_{i,1}, r_{i,2} \leftarrow \mathbb{Z}_Q$ and $\tilde{r}_{i,1}, \tilde{r}_{i,2} \leftarrow \mathbb{Z}_{\tilde{q}}$. It then sets

$\underline{\mathsf{Prv}_{\mathcal{L}_2}}(crs, stmt, wit)$: The prover routine does the following:

- Sample $t \leftarrow \mathbb{Z}_Q$, where $Q = \tilde{q} \cdot 2^{2\lambda}$, compute $G_0 := G^t$ and $K_0 := K^t$
- Compute $e \leftarrow \mathsf{H}_2(\text{stmt}, G_0, K_0)$
- Compute $t_{G,K} := t + e \cdot s$
- Set the proof $\pi := (G_0, K_0, t_{G,K}, e)$

$\underline{\mathsf{Vfy}_{\mathcal{L}_2}}(crs, stmt, \pi)$: The verifier routine does the following:

- Parse $\pi := (G_0, K_0, t_{G,K}, e)$
- Check if $e \stackrel{?}{=} \mathsf{H}_2(\text{stmt}, G_0, K_0)$, if so continue, otherwise output 0
- Check if $G^{t_{G,K}} \stackrel{?}{=} G_0 \cdot H_0^e$ and $K^{t_{G,K}} \stackrel{?}{=} K_0 \cdot H_1^e$. If unsuccessful, output 0.

Figure 7: Prover and Verifier routine for NIZK proof for statements in language $\mathcal{L}_2$

- $e_i = (d_{i,1} + d_{i,2}) \mod 2^\lambda$
- $A_{i,1} := G^{r_{i,1}} \cdot (c_{i,0})^{d_{i,1}}$
- $B_{i,1} := K^{r_{i,1}} \cdot (c_{i,1})^{d_{i,1}}$
- $A_{i,2} := G^{r_{i,2}} \cdot (c_{i,0})^{d_{i,2}}$
- $B_{i,2} := K^{r_{i,2}} \cdot (c_{i,1}/G)^{d_{i,2}}$
- $\tilde{A}_{i,1} := \tilde{G}^{\tilde{r}_{i,1}} \cdot (\tilde{c}_{i,0})^{d_{i,1}}$
- $\tilde{B}_{i,1} := \tilde{K}^{\tilde{r}_{i,1}} \cdot (\tilde{c}_{i,1})^{d_{i,1}}$
- $\tilde{A}_{i,2} := \tilde{G}^{\tilde{r}_{i,2}} \cdot (\tilde{c}_{i,0})^{d_{i,2}}$
- $\tilde{B}_{i,2} := \tilde{K}^{\tilde{r}_{i,2}} \cdot (\tilde{c}_{i,1}/\tilde{G})^{d_{i,2}}$.

It sets the random oracle H_3 accordingly at the i-th point with e_i. The values $(A_{i,1}, B_{i,1}, A_{i,2}, B_{i,2}, \tilde{A}_{i,1}, \tilde{B}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,2})$ output by the simulator are statistically close to that in a honestly generated proof. This is because w_i if sampled uniformly from $\mathbb{Z}_Q$ is distributed statistically close to $r_{i,1} + s_i \cdot d_{i,1}$ (Lemma E.1). □

Theorem E.7 (Simulation Soundness). *The protocol in Figure 8 satisfies simulation soundness provided the 2^λ-low order assumption and the strong root assumption holds in $\mathbb{G}$, in the random oracle model.*

Proof of Theorem E.7. As before, we assume without loss of generality that the false statement (and the index i where the statement fails) is fixed ahead of time. This assumption can be lifted with a polynomial loss in the success probability of the reduction by guessing the correct query to the random oracle. In what follows, we assume that the false statement only pertains to the prime-order group $\tilde{\mathbb{G}}$. For the case of the class group elements, the argument is similar to the proof of Theorems E.3 and E.5. Our reduction $\mathcal{R}$ computes simulated proofs up until the point where the adversary made the random oracle query on the false statement. Here the reduction forks the execution and sets the i-th output of the random oracle to two different values ($e_i \neq e_i'$). By the forking Lemma [64], with inverse polynomial probability the adversary outputs two accepting proofs containing $\tilde{A}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,1}, \tilde{B}_{i,2}, d_{i,1}, d_{i,2}, \tilde{r}_{i,1}, \tilde{r}_{i,1}$ and $\tilde{A}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,1}, \tilde{B}_{i,2}, d'_{i,1}, d'_{i,2}, \tilde{r}'_{i,1}\ \tilde{r}'_{i,1}$ (among other elements from $\mathbb{G}$).

Since $e_i \neq e_i'$, it must be the case that either $d_{i,1} \neq d'_{i,1}$ or $d_{i,2} \neq d'_{i,2}$. If we have $d_{i,1} \neq d'_{i,1}$, then the reduction computes

$$\tilde{s}_i := \frac{\left(\tilde{r}_{i,1} - \tilde{r}'_{i,1}\right)}{\left(d'_{i,1} - d_{i,1}\right)}$$

from $\tilde{A}_{i,1}$, such that $\tilde{c}_{i,0} = \tilde{G}^{\tilde{s}_i}$. Similarly from $\tilde{B}_{i,1}$ we have $\tilde{c}_{i,1} = \tilde{K}^{\tilde{s}_i}$, therefore we have extracted the witness to the relation. If $d_{i,2} \neq d'_{i,2}$, then the reduction computes

$$\tilde{s}_i := \frac{\left(\tilde{r}_{i,2} - \tilde{r}'_{i,2}\right)}{\left(d'_{i,2} - d_{i,2}\right)}$$

from $\tilde{A}_{i,2}$, such that $\tilde{c}_{i,0} = \tilde{G}^{\tilde{s}_i}$. From $\tilde{b}_{i,2}$ we have that $K^{\tilde{s}_i} = (\tilde{c}_{i,1}/\tilde{G})$ and therefore $\tilde{s}_i$ is a valid witness for the relation. It is not possible for both $d_{i,1} \neq d'_{i,1}$ and $d_{i,2} \neq d'_{i,2}$, since this would mean we have $\tilde{c}_{i,1} = \tilde{K}^{\tilde{s}_i} = \tilde{K}^{\tilde{s}_i} \cdot G$. Therefore we are able to extract a valid witness in one of the branches, which contradicts the fact that the statement was false. □

F SECURITY ANALYSIS OF DISTRIBUTED RANDOMNESS GENERATION PROTOCOL

Proof of Theorem 6.3. We assume for simplicity that the adversary corrupts all but one parties and the honest party is P_1. We define the following series of hybrid distributions.

$\underline{\mathsf{Hyb}_0}$: Is identical to the IND-RAN experiment with the bit b fixed to $b = 0$, i.e. $\mathcal{A}$ is given the honestly computed r_0.

$\underline{\mathsf{Hyb}_1}$: This is identical to the previous hybrid except that now each of the commitments output by the adversary are individually force-opened using TForceOp. Let $(s_2, \ldots, s_n)$ be the resulting integers and let s_1 the integer sampled by the honest P_1 (in the call to the RGen protocol). Then the adversary is given

$$r_0 = \sum_{i=1}^{n} s_i.$$

$\underline{\mathsf{Hyb}_2}$: This is identical to the previous hybrid except that we compute

$$r_0 = \sum_{i=2}^{n} s_i + \tilde{s}$$

where $\tilde{s} \leftarrow \mathbb{Z}_q$ is sampled uniformly and independently from s_1.

$\underline{\mathsf{Hyb}_3}$: Here the adversary is given a uniformly sampled integer.

Note that the latter hybrid is identical to the experiment IND-RAN with the bit b fixed to $b = 1$. We now argue the indistinguishability of the hybrids.

$\mathsf{Prv}_{\mathcal{L}_3}(crs, stmt, wit)$: The prover routine does the following:

- For $i \in [\alpha]$, do the following:
  - If $r_i = 1$, i.e., $\Big((c_{i,0}, c_{i,1}) = (G^{s_i}, K^{s_i} \cdot G) \bigwedge (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i} \cdot \tilde{G})\Big)$, do the following:
    - Sample $w_i, r_{i,1} \leftarrow \mathbb{Z}_Q$, $\tilde{w}_i, \tilde{r}_{i,1} \leftarrow \mathbb{Z}_{\tilde{q}}$ and $d_{i,1} \leftarrow \mathbb{Z}_{2^\lambda}$
    - Set $A_{i,1} := G^{r_{i,1}} \cdot (c_{i,0})^{d_{i,1}}$, $B_{i,1} := K^{r_{i,1}} \cdot (c_{i,1})^{d_{i,1}}$
    - Set $\tilde{A}_{i,1} := \tilde{G}^{\tilde{r}_{i,1}} \cdot (\tilde{c}_{i,0})^{d_{i,1}}$, $\tilde{B}_{i,1} := \tilde{K}^{\tilde{r}_{i,1}} \cdot (\tilde{c}_{i,1})^{d_{i,1}}$
    - Set $A_{i,2} := G^{w_i}$, $B_{i,2} := K^{w_i}$
    - Set $\tilde{A}_{i,2} := \tilde{G}^{\tilde{w}_i}$, $\tilde{B}_{i,2} := \tilde{K}^{\tilde{w}_i}$
  - If $r_i = 0$, i.e., $\Big((c_{i,0}, c_{i,1}) = (G^{s_i}, K^{s_i}) \bigwedge (\tilde{c}_{i,0}, \tilde{c}_{i,1}) = (\tilde{G}^{\tilde{s}_i}, \tilde{K}^{\tilde{s}_i})\Big)$, do the following:
    - Sample $w_i, r_{i,2} \leftarrow \mathbb{Z}_Q$, $\tilde{w}_i, \tilde{r}_{i,2} \leftarrow \mathbb{Z}_{\tilde{q}}$ and $d_{i,2} \leftarrow \mathbb{Z}_{2^\lambda}$
    - Set $A_{i,1} := G^{w_i}$, $B_{i,1} := K^{w_i}$
    - Set $\tilde{A}_{i,1} := \tilde{G}^{\tilde{w}_i}$, $\tilde{B}_{i,1} := \tilde{K}^{\tilde{w}_i}$
    - Set $A_{i,2} := G^{r_{i,2}} \cdot (c_{i,0})^{d_{i,2}}$, $B_{i,2} := K^{r_{i,2}} \cdot (c_{i,1}/G)^{d_{i,2}}$
    - Set $\tilde{A}_{i,2} := \tilde{G}^{\tilde{r}_{i,2}} \cdot (\tilde{c}_{i,0})^{d_{i,2}}$, $\tilde{B}_{i,2} := \tilde{K}^{\tilde{r}_{i,2}} \cdot (\tilde{c}_{i,1}/\tilde{G})^{d_{i,2}}$
- Compute $(e_1, \ldots, e_{[\alpha]}) \leftarrow \mathsf{H}_3(\text{stmt}, \{A_{i,1}, B_{i,1}, A_{i,2}, B_{i,2}, \tilde{A}_{i,1}, \tilde{B}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,2}\}_{i\in[\alpha]})$
- For $i \in [\alpha]$, do the following:
  - If $r_i = 1$, do the following:
    - Compute $d_{i,2} := (e_i - d_{i,1}) \mod 2^\lambda$
    - Compute $r_{i,2} := w_i - s_i \cdot d_{i,2}$, $\tilde{r}_{i,2} := (\tilde{w}_i - \tilde{s}_i \cdot d_{i,2}) \mod \tilde{q}$
  - If $r_i = 0$, do the following:
    - Compute $d_{i,1} := (e_i - d_{i,2}) \mod 2^\lambda$
    - Compute $r_{i,1} := w_i - s_i \cdot d_{i,1}$, $\tilde{r}_{i,1} := (\tilde{w}_i - \tilde{s}_i \cdot d_{i,1}) \mod \tilde{q}$
- Output $\pi := \{e_i, A_{i,1}, B_{i,1}, A_{i,2}, B_{i,2}, \tilde{A}_{i,1}, \tilde{B}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,2}, d_{i,1}, d_{i,2}, r_{i,1}, r_{i,2}, \tilde{r}_{i,1}, \tilde{r}_{i,2}\}_{i\in[\alpha]}$

$\mathsf{Vfy}_{\mathcal{L}_3}(crs, stmt, \pi)$: The verifier routine does the following:

- Parse $\pi := \{e_i, A_{i,1}, B_{i,1}, A_{i,2}, B_{i,2}, \tilde{A}_{i,1}, \tilde{B}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,2}, d_{i,1}, d_{i,2}, r_{i,1}, r_{i,2}, \tilde{r}_{i,1}, \tilde{r}_{i,2}\}_{i\in[\alpha]}$
- Check if $(e_1, \ldots, e_\alpha) \stackrel{?}{=} \mathsf{H}_3(\text{stmt}, \{A_{i,1}, B_{i,1}, A_{i,2}, B_{i,2}, \tilde{A}_{i,1}, \tilde{B}_{i,1}, \tilde{A}_{i,2}, \tilde{B}_{i,2}\}_{i\in[\alpha]})$
- For $i \in [\alpha]$, check if all the following hold, and output 0 otherwise:
  - $e_i \stackrel{?}{=} (d_{i,1} + d_{i,2}) \mod 2^\lambda$
  - $A_{i,1} \stackrel{?}{=} G^{r_{i,1}} \cdot (c_{i,0})^{d_{i,1}}$
  - $B_{i,1} \stackrel{?}{=} K^{r_{i,1}} \cdot (c_{i,1})^{d_{i,1}}$
  - $A_{i,2} \stackrel{?}{=} G^{r_{i,2}} \cdot (c_{i,0})^{d_{i,2}}$
  - $B_{i,2} \stackrel{?}{=} K^{r_{i,2}} \cdot (c_{i,1} \cdot G^{-1})^{d_{i,2}}$
  - $\tilde{A}_{i,1} \stackrel{?}{=} \tilde{G}^{\tilde{r}_{i,1}} \cdot (\tilde{c}_{i,0})^{d_{i,1}}$
  - $\tilde{B}_{i,1} \stackrel{?}{=} \tilde{K}^{\tilde{r}_{i,1}} \cdot (\tilde{c}_{i,1})^{d_{i,1}}$
  - $\tilde{A}_{i,2} \stackrel{?}{=} \tilde{G}^{\tilde{r}_{i,2}} \cdot (\tilde{c}_{i,0})^{d_{i,2}}$
  - $\tilde{B}_{i,2} \stackrel{?}{=} \tilde{K}^{\tilde{r}_{i,2}} \cdot (\tilde{c}_{i,1} \cdot \tilde{G}^{-1})^{d_{i,2}}$
- If all the above conditions hold, output 1, else output 0.

Figure 8: Prover and Verifier routine for NIZK proof for statements in language $\mathcal{L}_3$

$\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$: By the perfect correctness of the commitment scheme, the hybrids only differ in the case where one of the commitments output by the adversary is not well-formed. However, such a commitment is always rejected unless the adversary computes a proof π for a false statement, which contradicts the verifiability of the CCA timed commitment scheme.

$\mathsf{Hyb}_1 \approx_{\mathbf{T}^\epsilon} \mathsf{Hyb}_2$: We show this indinstinguishability via a reduction to the CCA security of the timed commitment scheme. Let $\mathcal{A}$ be a PPT adversary with depth less than $\mathbf{T}^\epsilon$ (for some $\epsilon < 1$) that distinguishes between the two hybrids. The reduction $\mathcal{R}$ against the CCA security of the timed commitment proceeds as follows. The reduction obtains *crs* of the timed commitment scheme. When the adversary queries the RGen oracle, the reduction locally samples $(s_0, s_1) \leftarrow \mathbb{Z}_q$ and sends (s_0, s_1) to its challenger. It receives (c, π) from its challenger and sends (c, π) to the adversary as reply to the oracle query. The adversary outputs $V := \{(c_2, \pi_2), \ldots, (c_n, \pi_n)\}$. The reduction forwards each of these to its own oracle O. If the oracle responds with $\perp$ for any of the pairs (c_i, π_i), the reduction sets $m_i = 0$. Otherwise, the reduction receives m_i as a response and

defines a set $\{m_2, \ldots, m_n\}$. The reduction sets

$$r = \sum_{i=2}^{n} m_i + s_0$$

and returns r to the adversary. The adversary responds with a bit b' and the reduction outputs b' as its own answer to the challenger. This concludes the description of $\mathcal{R}$.

Notice that the reduction is efficient and it running time is only a polynomial (in λ) factor slower than $\mathcal{A}$. The reduction violates the CCA security of the timed commitments with the same probability as $\mathcal{A}$ violates the IND-RAN security. To see this, observe that we have two cases where (c, π) embeds s_0 or s_1. If (c, π) indeed embeds s_0, then r is distributed as in hybrid Hyb_1, otherwise r is distributed uniformly, as in Hyb_2. This is a contradiction to the CCA security of the timed commitments.

$\underline{\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3}$: Since $\tilde{s}$ is uniformly chosen and $\mathbb{Z}_q$ defines a field, the two hybrid distributions are identical. □