Simple, Fast, Efficient, and Tightly-Secure Non-Malleable Non-Interactive Timed Commitments

Peter Chvojka[1] and Tibor Jager[2]

[1] IMDEA Software Institute, chvojka.p@gmail.com
[2] University of Wuppertal, jager@uni-wuppertal.de

Abstract. Timed commitment schemes, introduced by Boneh and Naor (CRYPTO 2000), can be used to achieve fairness in secure computation protocols in a simple and elegant way. The only known non-malleable construction in the standard model is due to Katz, Loss, and Xu (TCC 2020). This construction requires general-purpose zero knowledge proofs with specific properties, and it suffers from an inefficient commitment protocol, which requires the committing party to solve a computationally expensive puzzle.

We propose new constructions of non-malleable non-interactive timed commitments, which combine (an extension of) the Naor-Yung paradigm used to construct IND-CCA secure encryption with a non-interactive ZK proofs for a simple algebraic language. This yields much simpler and more efficient non-malleable timed commitments in the standard model.

Furthermore, our constructions also compare favourably to known constructions of timed commitments in the random oracle model, as they achieve several further interesting properties that make the schemes very practical. This includes the possibility of using a homomorphism for the forced opening of multiple commitments in the sense of Malavolta and Thyagarajan (CRYPTO 2019), and they are the first constructions to achieve *public verifiability*, which seems particularly useful to apply the homomorphism in practical applications.

1 Introduction

Timed commitments make it possible to commit to a message with respect to some time parameter $T \in \mathbb{N}$, such that (1) the commitment is *binding* for the committing party, (2) it is *hiding* the committed message for T units of time (*e.g.*, seconds, minutes, days), but (3) it can also forcibly be opened after time T in case the committing party refuses to open the commitment or becomes unavailable. This idea goes back to a seminal work by Rivest, Shamir, and Wagner [RSW96] introducing the strongly related notion of *time-lock puzzles*, and Boneh and Naor [BN00] extended this idea to *timed commitments*, which have the additional feature that an opening to the commitment can be efficiently verified (and thus the commitment can be opened efficiently).

Achieving fairness via timed commitments. One prime application of timed commitmens is to achieve *fairness* in secure two- or multi-party protocols. For instance, consider a simple sealed-bid auction protocol with n bidders $B_1, \ldots, B_n$, where every bidder B_i commits to its bid x_i and publishes the commitment $c_i = \mathsf{Com}(x_i, r_i)$ using randomness r_i. When all bidders have published their commitments, everyone reveals their bid x_i along with r_i, such that everyone can publicly verify that the claimed bid x_i is indeed consistent with the initial commitment c_i. The bidder with the maximal bid wins the auction. For this to be most practical, we want commitments to be *non-interactive*.

Now suppose that after the first $(n-1)$ bidders $B_1, \ldots, B_{n-1}$ have opened their commitments (x_i, r_i), the last bidder B_n claims that it has "lost" its randomness r_{i^*}, *e.g.*, by accidentally deleting it. However, B_n also argues strongly and quite plausibly that it has made the highest bid x_{i^*}. This is a difficult situation to resolve in practice:

- B_n *might indeed be honest.* In this case, it would be fair to accept its highest bid x_{i^*}. One could argue that it is B_n's own fault and thus it should not win the auction, but at the same time a seller might strongly argue to accept the bid, as it is interested in maximising the price, and if B_n's claim is indeed true, then discarding the real highest bit could be considerd unfair by the seller.
- *However,* B_n *might also be cheating.* Maybe it didn't commit to the highest bid, and now B_n tries to "win" the auction in an unfair way.

Timed commitments can resolve this situation very elegantly and without the need to resort to a third party that might collude with bidders, and thus needs to be trusted, or which might not even be available in certain settings, *e.g.*, in fully decentralized protocols, such as blockchain-based applications. In a timed commitment scheme, the parties create their commitments $c_i = \mathsf{Com}(x_i, r_i, T)$ with respect to a suitable time parameter T for the given application. In case one party is not able to or refuses to open its commitment, the other parties can force the commitment open in time T and thus resolve a potential dispute.

1.1 Requirements on Practical Timed Commitments

Several challenges arise when constructing timed commitments that can be used in practical applications.

Consistency of standard and forced opening. A first challenge to resolve when constructing a timed commitment scheme is to guarantee that the availability of an alternative way to open a commitment, by using the forced decommitment procedure, does not break the *binding* property. Standard and forced opening must be guaranteed to reveal the same message. Otherwise, a malicious party could create a commitment where standard and forced openings yields different values. Then it could decide in the opening phase whether it provide the "real" opening, or whether it refuses to open, such that the other parties will perform the forced opening.

Non-interactivity. Having non-interactive commitments is generally desirable to obtain protocols that do not require all parties to be online at the same time. Furthermore, certain applications inherently require the commitment scheme to be non-interactive. This includes, for example, protocols where the commitments are published in a public ledger (*e.g.*, a decentralized blockchain). Several examples of such applications are described in [MT19]. Non-interactivity also avoids concurrent executions of the commitment protocol, which simplifies the security model significantly.

Non-malleability. Non-malleability of a commitment guarantees that no party can turn a given commitment c that decommits to some value x into another commitment c' which decommits to a different value x', such that x and x' are related in some meaningful way. For instance, in the above example of an auction, a malicious party B_n could first wait for all other parties to publish their commitments. Then it would select the commitment c_i which most likely contains the highest bid x_i, and exploit the malleability of to create a new commitment c_n, which is derived from c_i and opens to $x_i + 1$. Hence, B_n would be able win the auction with a bid that is only slightly larger than the 2^{nd} highest bit, which does not meet the intuitive security expectations on a secure auctioning protocol.

In order to achieve non-malleability for timed commitments, a recent line of works has explored the idea of *non-malleable time-locked commitments* and *puzzles* [KLX20,TCLM21,EFKP20,BDD+21]. Existing constructions of timed commitments are either malleable, rely on the random oracle model, have highly non-tight security proof, which constructs a reduction that solves multiple instances of a puzzle, or require the sender of the commitment to invest as much effort to commit to a value as for the receiver to forcibly open the commitment. The only known standard model construction by Katz *et al.* [KLX20] relies on non-interactive zero-knowledge proofs (NIZKs) for *general* NP relations with very specific properties.

Force opening many commitments at once via homomorphism. Yet another interesting property that can make timed commitments more practical is a possibility to aggregate multiple commitments into a *single* one, such that it is sufficient to force open only this commitment. The idea of homomorphic time-lock *puzzles* was introduced by Malavolta and Thyagarajan [MT19] and later adopted to the setting of non-interactive timed commitments in [TCLM21].

A homomorphic timed commitment scheme allows to efficiently evaluate a circuit C over a set of commitments $c_1, \ldots, c_n$, where c_i is a commitment to some value x_i for all i, to obtain a commitment c to $C(x_1, \ldots, x_n)$. If there are multiple parties $B_{i_1}, \ldots, B_{i_z}$ that refuse to open their commitments and it is not necessary to recover the full committed messages $x_{i_1}, \ldots, x_{i_z}$, but recovering $C(x_{i_1}, \ldots, x_{i_z})$ is sufficient, then one can use the homomorphism to compute a signle commitment c that needs to be opened. Malavolta and Thyagarajan [MT19] describe several interesting applications, includings e-voting and sealed-bid auctions over blockchains, multi-party coin flipping, and multi-party contract signing.

Public verifiability of commitments. Another property is *public verifiability* of a timed commitment, which requires that one can efficiently check whether a commitment is well-formed, such that a forced decommitment will yield a correct result.

Without public verifiability, timed commitments might not provide practical solutions for certain applications. For instance, a malicious party could output a malformed commitment that cannot be opened in time T, such that a protocol would fail again in case the malicious party refuses to open the commitment. This could pose a problem in time-sensitive applications, in particular if a large time parameter T is used, and also give rise do Denial-of-Service attacks. Note that public verifiability is particularly relevant for homomorphic commitments. When many commitments are aggregated into a single one, then it is essentiall that all these commitments are well-formed, as otherwise the forced opening may fail. Public verifiability allows to efficiently decide which subset of commitments is well-formed, and thus to include only these in the homomorphic aggregate.

Note that the requirement of public verifiability rules out several natural ways to achieve non-malleability, such as the Fujisaki-Okamoto transform [FO99,FO13] used by Ephraim *et al.* [EFKP20]. It seems that even in the random oracle model ZK proofs are required.

Public verifiability of forced opening. In scenarios when the forced opening is executed by untrusted party, it is desirable to be able efficiently check that forced opening has been executed properly without redoing an expensive sequential computation. This particularly useful when the forced opening computation is outsourced to untrusted server. This property was first suggested for time-lock puzzles by [EFKP20].

1.2 Our Contributions

We provide a simpler and more efficient approach to construct practical non-malleable timed commitments. We give the first constructions that simultaneously achieve non-interactivity, non-malleability, linear (*i.e.*, additive) or multiplicative homomorphism, public verifiability of commitments and public verifiability of forced opening. Moreover, all our reductions avoid the need to answer decommitment queries using the slow forced decommitment algorithm, which yields much tighter security. Instead of relying on expensive ZK proofs for general NP languages as prior work, we show how to use Fiat-Shamir [FS87] NIZKs derived from Sigma protocols for simple algebraic languages. Our constructions can be instantiated in the standard model by leveraging techniques from Libert *et al.* [LNPY21] and more efficiently in the random oracle model.

In more detail, we make the following contributions.

1. We begin by extending the formal definitions of prior work to cover *public verifiability* of forced opening in the setting of non-malleable non-interactive timed commitments.

2. We then give four constructions of non-interactive non-malleable timed commitments. All our constructions rely on a variation of the double encryption paradigm by Naor and Yung [NY90], which was also used by Katz *et al.* [KLX20] and Thyagarajan *et al.* [TCLM21].
   However, in contrast to [KLX20], we do not start from a timed public key encryption scheme, but build our timed commitment from scratch. This enables us avoid two out of the three NIZK proofs in their construction, and lets us replace the third by a proof for a variation of the DDH relation over groups of unknown order. We are able to instantiate the given NIZK both in the standard model and in the random oracle model [BR93]. Like the construction from [KLX20] we support public verifiability of commitments. Another important advantage of our constructions over that of Katz *et al.* [KLX20] is that it allows for fast commitment, whereas [KLX20] requires to execute an expensive sequential computation in order to commit to a message. Additionally, we achieve public verifiability of forced opening and homomorphic properties.

In comparison, the non-interactive construction of David *et al.* [BDD+21] is in the programmable random oracle model, while ours can also be instantiated in the standard model. David *et al.* achieve fast commitments, however the construction does not provide public verifiability of commitments, public verifiability of forced opening nor homomorphic properties. The work of Ephraim et al. [EFKP20] does support fast commitments and public verifiability of forced opening, but is also in the (auxiliary non-programmable) random oracle model and does not support public verifiability of commitments and homomorphic properties. Thyagarajan *et al.* [TCLM21] construct the first CCA-secure non-interactive timed commitment with transparent setup, meaning that randomness used in the setup can be made public. Their construction relies on class groups and CCA security is achieved using the Naor-Young paradigm. Additionally, the construction is linearly homomorphic. This is very similar to our work. The main disadvantage of this approach compared to our constructions is that the size of the resulting commitments is linear in security parameter and the security proof is extremely non-tight, since it relies on slow forced decommitment in several steps of the security proof. Moreover, it supports a significantly smaller message space and the construction is in the random oracle model. We provide a summary of the properties of our constructions in comparison to previous works in Table 1. Additionally, in Table 2 we provide a comparison of an instantiation of our construction of linearly homomorphic NITCs in the ROM with an instantiation of the construction from [TCLM21] which is also linearly homomorphic and in ROM. We compare the size of crs, commitments Com, proofs π_{Com}, and messages for security level $\lambda = 128$ bits and taking into account a security loss in the security proofs. Since in the majority of game hops of the security proof of [TCLM21] decommitment queries are answered using forced decommitment, the corresponding security loss is $Q \cdot T$ where Q is the number of decommitment queries and T is the time parameter of NITC. As an example, we assume that $Q = T = 2^{32}$, which results in the security loss of 2^{64}. Therefore to achieve 128

bits of security, one has to instantiate assumptions in [TCLM21] for security parameter $\lambda = 192$ bits.[3] According to [BJS10] the fundamental discriminant Δ_K for this security parameter has size of 3598 bits, and similarly to [TCLM21] we define the message space $\mathbb{Z}_q$ for q which is 256 bits. Hence, Δ_q has size of $3588 + 2 \times 256 = 4110$ bits, size of $\tilde{q}$ is $\alpha = 3598/2 + 192 = 1991$ bits, and $\mathbb{Z}_p^*$ is instantiated for a prime p of size 3072 bits. To instantiate our construction it is sufficient to use recommended modulus size of 3072 bits, since our security proof is tight. We remark, that our constructions provide significantly smaller commitments and proofs and larger message space even if we don't take the security loss into account.

Construction	**Hom.**	**Std.**	**Setup**	**Com?**	**FDec?**	$\lvert\mathsf{Com}\rvert$	$\lvert\pi_{\mathsf{Com}}\rvert$	t_{Com}	**Tight**
[EFKP20]	—	✗	—	✗	✓	$O(1)$	—	$O(\log T)$	✓
[KLX20]	—	✓	priv.	✓	✗	$O(1)$	$O(1)$	$O(T)$	✓
[TCLM21]	linear	✗	pub.	✓	✗	$O(\lambda)$	$O(\lambda)$	$O(1)$	✗
Section 3.3	linear	✓	priv.	✓	✓	$O(1)$	$O(\log \lambda)$	$O(1)$	✓
Section 3.4	mult.	✓	priv.	✓	✓	$O(1)$	$O(\log \lambda)$	$O(1)$	✓
Section 4.3	linear	✗	priv.	✓	✓	$O(1)$	$O(1)$	$O(1)$	✓
Section 4.4	mult.	✗	priv.	✓	✓	$O(1)$	$O(1)$	$O(1)$	✓

Table 1. Comparison of our constructions with related work. Column **Hom.** indicates whether the construction provides a linear/multiplicative homomorphism, **Std.** whether the construction has a standard-model proof, **Com?** whether it is publicly verifiable that commitments are well-formed, **FDec?** efficient public verifiability of forced decommitments, $\lvert\mathsf{Com}\rvert$ is the size of commitments, $\lvert\pi_{\mathsf{Com}}\rvert$ the size of proofs, t_{Com} the running time of the commitment algorithm, and **Tight** whether the proof avoids running the forced decommitment algorithm to respond to CCA queries.

Construction	$\lvert\mathsf{crs}\rvert$ (kB)	$\lvert\mathsf{Com}\rvert$ (kB)	$\lvert\pi_{\mathsf{Com}}\rvert$ (kB)	$\lvert m\rvert$ (bits)
[TCLM21]	2.32	3321.41	8846.96	256
Section 4.3	1.92	1.54	1.55	3072

Table 2. Comparison of our construction with [TCLM21] for security level $\lambda = 128$ bits and taking into account the security loss for $Q = T = 2^{32}$.

1.3 Technical Overview

The *binding* property of our commitment scheme will be relatively easy to argue, therefore let us focus on the *hiding* property and non-malleability. Like in [KLX20], we prove this by considering an IND-CCA security experiment,

[3] The choice of Q and T such that $QT = 2^{64}$ is convenient because it yields $\lambda = 192$ and [BJS10] provides concrete parameters for this security parameter.

where the adversary has access to a forced decommitment oracle. Even though the forced decommitment can be performed in polynomial time, this polynomial may be very large, if the time parameter T is large. Since the experiment needs to be perform a forced decommitment for *every* decommitment query of the adversary, this would incur a very significant overhead and a highly lossy reduction. Hence, following Katz *et al.* [KLX20], we aim to build commitment schemes where a reduction can perform a fast decommitment.

Recall that a classical approach to achieve IND-CCA security is to apply the Naor-Yung paradigm [NY90]. A natural approach to construct non-malleable timed commitments is therefore to apply this paradigm as follows. A commitment $c = (c_1, c_2, \pi)$ to a message m consists of a time-lock puzzle c_1 opening to m, a public key encryption of m, and a simulation-sound zero knowledge proof π that both contain the same message m, everything with respect to public parameters contained in a public common reference string. This scheme may potentially achieve all desired properties:

- Consistency of regular and forced opening can be achieved by using a suitable time-lock puzzle and public-key encryption scheme.
- The commitment is non-interactive.
- IND-CCA security follows from the standard Naor-Yung argument.
- The time-lock puzzle in the above construction can be instantiated based on repeated squaring [RSW96], possibly using the variant of [MT19] that combines repeated squaring with Paillier encryption [Pai99] to achieve a linear homomorphism.
- Public verifiability can be achieved by using a suitable proof system for π.

Furthermore, in the IND-CCA security proof, we can perform fast opening by decrypting c_2 with the secret key of the public key encryption scheme, which is indistinguishable from a forced opening using c_1 by the soundness of the proof. However, it turns out that concretely instantiating this scheme in a way that yields a practical construction is non-trivial and requires a very careful combination of different techniques.

Triple Naor-Yung. First of all, note that repeated squaring modulo a composite number $N = PQ$, where P and Q are different primes, is currently the only available choice to achieve a practical time-lock puzzle, hence we are bound to using this puzzle to instantiate c_1. Conveniently, this puzzle allows for a linear (*i.e.*, additive) homomorphism by following [MT19]. Then, in order to be able to instanatiate π efficiently, it would be convenient to use a standard Sigma protocol, which can then be made non-interactive via the Fiat-Shamir transform [FS87] in the random oracle model, or by leveraging techniques from Libert *et al.* [LNPY21] in the standard model. Since practically efficient Sigma protocols are only known for algebraic languages, such as that defined by the DDH relation, for example, we have to choose c_2 in a way which is "algebraically compatible" with c_1 and the available proofs π. If we instantiate c_1 with the homomorphic TLP from [MT19], then a natural candidate would be to instantiate c_2 also with Paillier encryption. Here we face the first technical difficulty:

– Efficient proof systems for π are only available, if both c_1 and c_2 use the same modulus N. Hence, we have to instantiate both with the same modulus.
– When arguing that c_1 hides the committed message m in the Naor-Yung argument of the security proof, we will have to replace c_1 with a random puzzle, using the *strong sequential squaring* (SSS) assumption. At the same time, we have to be able to respond to decommitment queries using the decryption key of c_2. But this decryption key is the factorization P, Q of the common modulus N, and we cannot reduce to the hardness of SSS while knowing the factorization of N.

The first candidate approach to overcome this difficulty is to replace the Paillier encryption used in c_2 with an encryption scheme that does not require knowledge of the factorization of N, such as the "Paillier ElGamal" scheme from [MT19], which is defined over the subgroup $\mathbb{J}_N$ of elements of $\mathbb{Z}_N$ having Jacobi symbol 1, and which uses a discrete logarithm to decrypt but still requires the factorization of N to be hidden in order to be secure.

However, now we run into another difficulty. In the Naor-Yung argument, we will also have to replace c_2 with an encryption of a random message, in order to argue that our commitment scheme is hiding. In this part of the proof, we cannot know the secret key of c_2, that is, neither the aforementioned discrete logarithm, nor the factorization of N. However, we also cannot use c_1 to respond to decommitment queries, because then we would have to solve the time-lock puzzle, which cannot be done fast without knowledge of the factorization of N.

We resolve this problem by using "triple Naor-Yung". In our linearly homomorphic constructions, a commitment to m will have the form (c_1, c_2, c_3, π), where c_1 and c_2 are Paillier-ElGamal encryptions of m, and c_3 is the Paillier-style time-lock puzzle based on repeated squaring from [MT19]. All are with respect to the same modulus N, and thus allow for an efficient Sigma-protocol-based proof π that c_1, c_2, and c_3 all contain the same message. In the Naor-Yung-style IND-CCA security proof, we will first replace c_3 with a random puzzle, while using the discrete logarithm of the public key that corresponds to c_1 to perform fast decommitments. When we then replace c_2 with an encryption of a random message, we use the discrete logarithm of the public key that corresponds to c_1 to answer decommitment queries. Finally, we switch to using the discrete logarithm of the secret key corresponding to c_2 for decommitment queries, and replace c_1 with an encryption of a random message. Hence, throughout the argument we never require the factorization of N for fast decommitments.

Standard Naor-Yung works for multiplicative homomorphism. Next, we observe that the standard (*i.e.*, "two-ciphertext") Naor-Yung approach works, if a *multiplicative* homomorphism (or no homomorphism at all) is required. Concretely, a commitment will have the form (c_1, c_2, π), where c_1 is an ElGamal encryption and c_2 uses the "sequential-squaring-with-ElGamal-encryption" idea of [MT19]. By replacing the underlying group to the subgroup $\mathbb{QR}_N$ of quadratic residues modulo N, we can rely on the DDH assumption in $\mathbb{QR}_N$ and thus do not require the factorization of N to be hidden when replacing the ElGamal encryption c_1

with an encryption of a random message. While the construction idea and high-level arguments are very similar, the underlying groups and detailed arguments are somewhat different, and thus we have to give a separate proof.

On separate proofs in the standard model and the ROM The constructions sketched above can be instantiated relatively efficiently in the standard model, using the one-shot Fiat-Shamir arguments in the standard model by Libert *et al.* [LNPY21]. However, these proofs repeat the underlying Sigma protocol a logarithmic number of times, and thus it would be interesting to also consider constructions in the random oracle model. Since the syntactical definitions and properties of proof systems in the random oracle model are slightly different from that in [LNPY21], we give separate proofs for both random oracle constructions as well.

Shared randomness To obtain commitments of smaller size we additionally apply the shared randomness technique from [BMV16], where instead of producing two or three independent encryptions of the same message, we reuse the same randomness for encryption. This allows to save one group element in case of the standard Naor-Yung constructions and two group elements in the case of triple Naor-Yung.

1.4 Further related work

Time-lock puzzles based on randomized encodings were introduced in [BDGM19], but all known constructions of timed commitments rely on the repeated squaring puzzles of [RSW96]. Timed commitments are also related to time-lock encryption scheme [LJKW18] and time-released encryption [CJSS21], albeit with different properties. The construction in [LJKW18] is based on an external "computational reference clock" (instantiated with a public block chain), whose output can be used to decrypt, such that decrypting parties do not have to perform expensive computations by solving a puzzle. The constructions of Chvojka *et al.* [CJSS21] are based on repeated squaring, however, the main difference is that the time needed for decryption starts to run from the point when *setup* is executed and not from the point when ciphertext is created.

2 Preliminaries

We denote our security parameter by λ. For $n \in \mathbb{N}$ we write 1^n to denote the n-bit string of all ones. For any element x in a set X, we use $x \xleftarrow{\$} X$ to indicate that we choose x uniformly at random from X. For simplicity we model all algorithms as Turing machines, however, all adversaries are modeled as non-uniform polynomial-size circuits to simplify concrete time bounds in the security definitions of non-interactive timed commitments and the strong sequential squaring assumption. All algorithms are randomized, unless explicitly defined as deterministic. For any PPT algorithm A, we define $x \leftarrow A(1^\lambda, a_1, \ldots, a_n)$ as the

execution of A with inputs security parameter λ, $a_1, \ldots, a_n$ and fresh randomness and then assigning the output to x. We write $[n]$ to denote the set of integers $\{1, \ldots, n\}$ and $\lfloor x \rfloor$ to denote the greatest integer that is less than or equal to x.

Non-interactive timed commitments. The following definition of a non-interactive timed commitment scheme is from [KLX20].

Definition 1. *A non-interactive timed commitments scheme* $\mathtt{NITC}$ *with message space* $\mathcal{M}$ *is a tuple of algorithms* $\mathtt{NITC} = (\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *with the following syntax.*

- $\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T)$ *is a probabilistic algorithm that takes as input the security parameter* 1^λ *and a hardness parameter* T *and outputs a common reference string* crs *and a secret key.*
- $(c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}}) \leftarrow \mathsf{Com}(\mathsf{crs}, m)$ *is a probabilistic algorithm that takes as input a common reference string* crs *and a message* m *and outputs a commitment* c *and proofs* $\pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}}$.
- $0/1 \leftarrow \mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}})$ *is a deterministic algorithm that takes as input a common reference string* crs*, a commitment* c *and proof* π_{Com} *and outputs* 0 *(reject) or* 1 *(accept).*
- $0/1 \leftarrow \mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}})$ *is a deterministic algorithm that takes as input a common reference string* crs*, a commitment* c*, a message* m *and proof* π_{Dec} *and outputs* 0 *(reject) or* 1 *(accept).*
- $m \leftarrow \mathsf{FDec}(\mathsf{crs}, c, \pi_{\mathsf{Com}})$ *is a deterministic forced decommit algorithm that takes as input a common reference string* crs *and a ciphertext* c *and outputs* $m \in \mathcal{M} \cup \{\perp\}$ *in time at most* $T \cdot \mathsf{poly}(\lambda)$.

We say $\mathtt{NITC}$ *is correct if for all* $\lambda, T \in \mathbb{N}$ *and all* $m \in \mathcal{M}$ *holds:*

$$\Pr\left[\begin{array}{r} \mathsf{FDec}(\mathsf{crs}, c) = m \\ \wedge\ \mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) = 1 : \\ \wedge\ \mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}}) = 1 \end{array} \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T) \\ (c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}}) \leftarrow \mathsf{Com}(\mathsf{crs}, m) \end{array}\right] = 1.$$

The following definition is based on [KLX20], however, adjusted to computational model considered by Bitansky *et al.* [BGJ+16].

Definition 2. *A non-interactive timed commitment scheme* $\mathtt{NITC}$ *is* IND-CCA secure *with gap* $0 < \epsilon < 1$ *if there exists a polynomial* $\tilde{T}(\cdot)$ *such that for all polynomials* $T(\cdot) \geq \tilde{T}(\cdot)$ *and every non-uniform polynomial-size adversary* $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$*, where the depth of* $\mathcal{A}_{2,\lambda}$ *is at most* $T^\epsilon(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *it holds*

$$\mathbf{Adv}^{\mathtt{NITC}}_{\mathcal{A}} = \left| \Pr\left[b = b' \ : \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T(\lambda)) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot,\cdot)}_{1,\lambda}(\mathsf{crs}) \\ b \xleftarrow{\$} \{0,1\} \\ (c^*, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}}) \leftarrow \mathsf{Com}(\mathsf{crs}, m_b) \\ b' \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_{2,\lambda}(c^*, \pi^*_{\mathsf{Com}}, \mathsf{st}) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where $|m_0| = |m_1|$ *and the oracle* $\mathsf{DEC}(c, \pi_{\mathsf{Com}})$ *returns the result of* $\mathsf{FDec}(\mathsf{crs}, c)$ *if* $\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) = 1$*, otherwise it returns* $\perp$*, with the restriction that* $\mathcal{A}_{2,\lambda}$ *is not allowed to query the oracle* $\mathsf{DEC}(\cdot,\cdot)$ *for a decommitment of the challenge commitment* $(c^*, \pi^*_{\mathsf{Com}})$*.*

As already observed in [KLX20], a challenge for a security proof of a concrete timed commitment construction is that the reduction must be able to answer decommitment queries to $\mathsf{DEC}(\cdot,\cdot)$ in time which is independent of T, as otherwise one is not able to obtain a sound proof when reducing to a time-sensitive assumption, such as the strong sequential squaring assumption. This in particular means that decommitment queries in the security proof can not be simply answered by executing the forced decommitment algorithm FDec, as its runtime depends on T, but there must exist another way.

Remark 1. We note that our definition of the decommitment oracle DEC slightly differs from the original definition in [KLX20], since we require that the oracle at first checks if the commitment is well formed and only then returns the result of FDec. All our constructions can achieve also the original definition, to this end we would simply include the proof π that the commitment is well-formed in the commitment and then directly perform the check if a commitment is well formed in algorithm FDec. However, in that case π_{Com} would be empty and the whole idea of the separation of a commitment from a proof of well-formedness would be meaningless. [4]

Definition 3. *We define the* $\mathsf{BND\text{-}CCA}_{\mathcal{A}}(\lambda)$ *experiment as follows:*

1. $\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T(\lambda))$*;*
2. $(m, c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}}, m', \pi'_{\mathsf{Dec}}) \leftarrow \mathcal{A}_\lambda^{\mathsf{DEC}(\cdot,\cdot)}(\mathsf{crs})$*, where the oracle* $\mathsf{DEC}(c, \pi_{\mathsf{Com}})$ *returns* $\mathsf{FDec}(\mathsf{crs}, c)$ *if* $\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) = 1$*, otherwise it returns* $\perp$*;*
3. *Output 1 iff* $\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) = 1$ *and either:*
   - $m \neq m' \ \wedge \ \mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}}) = \mathsf{DecVrfy}(\mathsf{crs}, c, m', \pi'_{\mathsf{Dec}}) = 1$*;*
   - $\mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}}) = 1 \ \wedge \ \mathsf{FDec}(\mathsf{crs}, c) \neq m$*.*

A non-interactive timed commitment scheme $\mathtt{NITC}$ *is* BND-CCA secure *if for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there is a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{NITC}}_{\mathcal{A}} = \Pr\left[\mathsf{BND\text{-}CCA}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Next we define a new property of NITCs, which allows for efficient verification that a forced decommitment was executed correctly, without the need to execute expensive sequential computation. This property was first suggested for time-lock puzzles by [EFKP20] and denoted as public verifiability.

Definition 4. *A non-interactive timed commitments scheme* $\mathtt{NITC}$ *is* publicly verifiable *if* FDec *additionally outputs a proof* π_{FDec} *and has an additional algorithm* $\mathsf{FDecVrfy}$ *with the following syntax:*

[4] Note that [KLX20] FDec also implicitly checks well-formedness, as it runs a decryption algorithm, which verifies the NIZK proof.

– $0/1 \leftarrow \mathsf{FDecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{FDec}})$ *is a deterministic algorithm that takes as input a common reference string* crs*, a commitment* c*, a message* m*, and a proof* π_{FDec} *and outputs 0 (reject) or 1 (accept) in time* $\mathsf{poly}(\log T, \lambda)$.

Moreover, a publicly verifiable NITC must have the following properties:

– Completeness *for all* $\lambda, T \in \mathbb{N}$ *and all* $m \in \mathcal{M}$ *holds:*

$$\Pr\left[\mathsf{FDecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{FDec}}) = 1 : \begin{array}{r}\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T) \\ (c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}}) \leftarrow \mathsf{Com}(\mathsf{crs}, m) \\ (m, \pi_{\mathsf{FDec}}) \leftarrow \mathsf{FDec}(\mathsf{crs}, c)\end{array}\right] = 1.$$

– Soundness *for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there is a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\Pr\left[\begin{array}{r}\mathsf{FDecVrfy}(\mathsf{crs}, c, m', \pi'_{\mathsf{FDec}}) = 1 \qquad \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T) \\ \wedge\ \mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) = 1 : (c, \pi_{\mathsf{Com}}, m', \pi'_{\mathsf{FDec}}) \leftarrow \mathcal{A}_\lambda(\mathsf{crs}) \\ \wedge\ m \neq m' \qquad (m, \pi_{\mathsf{FDec}}) \leftarrow \mathsf{FDec}(\mathsf{crs}, c)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

The following definition is inspired by the definition of homomorphic time-lock puzzles of Malavolta *et al.* [MT19].

Definition 5. *A non-interactive timed commitments scheme* $\mathtt{NITC}$ *is homomorphic with respect to a class of circuits* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$*, if there is an additional algorithm* Eval *with the following syntax:*

– $c \leftarrow \mathsf{Eval}(\mathsf{crs}, C, c_1, \ldots, c_n)$ *is a probabilistic algorithm that takes as input a common reference string* crs*, a circuit* $C \in \mathcal{C}_\lambda$*, and set of* n *commitments* $(c_1, \ldots, c_n)$*. It outputs a commitment* c.

Additionally, a homomorphic NITC fulfils the following properties:
Correctness: *for all* $\lambda, T \in \mathbb{N}$*,* $C \in \mathcal{C}_\lambda$*,* $(m_1, \ldots, m_n) \in \mathcal{M}^n$*, all* crs *in the support of* $\mathsf{PGen}(1^\lambda, T)$*, all* c_i *in the support of* $\mathsf{Com}(\mathsf{crs}, m_i)$ *we have:*

1. *There exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{FDec}(\mathsf{crs}, \mathsf{Eval}(\mathsf{crs}, C, c_1, \ldots, c_n)) \neq C(m_1, \ldots, m_n)\right] \leq \mathsf{negl}(\lambda).$$

2. *The exists a fixed polynomial* poly *such that the runtime of* $\mathsf{FDec}(\mathsf{crs}, c)$ *is bounded by* $\mathsf{poly}(\lambda, T)$*, where* $c \leftarrow \mathsf{Eval}(\mathsf{crs}, C, c_1, \ldots, c_n)$.

Compactness: *for all* $\lambda, T \in \mathbb{N}$*,* $C \in \mathcal{C}_\lambda$*,* $(m_1, \ldots, m_n) \in \mathcal{M}^n$*, all* crs *in the support of* $\mathsf{PGen}(1^\lambda, T)$*, all* c_i *in the support of* $\mathsf{Com}(\mathsf{crs}, m_i)$*, the following two conditions are satisfied:*

1. *The exists a fixed polynomial* $\hat{\mathsf{poly}}$ *such that* $|c| = \hat{\mathsf{poly}}(\lambda, |C(m_1, \ldots, m_n)|)$*, where* $c \leftarrow \mathsf{Eval}(\mathsf{crs}, C, c_1, \ldots, c_n)$.
2. *The exists a fixed polynomial* $\tilde{\mathsf{poly}}$ *such that the runtime of* $\mathsf{Eval}(\mathsf{crs}, C, c_1, \ldots, c_n)$ *is bounded by* $\tilde{\mathsf{poly}}(\lambda, |C|)$.

Complexity assumptions. We base our constructions on the strong sequential squaring assumption. Let p, q be safe primes (i.e., such that $p = 2p'+1, q = 2q'+1$ for primes p', q'). We denote by $\varphi(\cdot)$ Euler's totient function, by $\mathbb{Z}_N^*$ the group $\{x \in \mathbb{Z}_N : \gcd(N, x) = 1\}$ and by $\mathbb{J}_N$ the cyclic subgroup of elements of $\mathbb{Z}_N^*$ with Jacobi symbol 1 which has order $|\mathbb{J}_N| = \frac{\varphi(N)}{2} = \frac{(p-1)(q-1)}{2}$. By $\mathbb{QR}_N$ we denote the cyclic group of quadratic residues modulo N which has order $|\mathbb{QR}_N| = \frac{\varphi(N)}{4} = \frac{(p-1)(q-1)}{4}$. To efficiently sample a random generator g from $\mathbb{J}_N$, it is sufficient to be able sample random element from $\mathbb{J}_N \setminus \mathbb{QR}_N$, since with all but negligible probability a random element of $\mathbb{J}_N \setminus \mathbb{QR}_N$ is a generator. Moreover, when the factors p, q are known, then it easy to check if the given element is a generator of $\mathbb{J}_N$ by testing possible orders.

To sample a random element of $\mathbb{J}_N \setminus \mathbb{QR}_N$, we can sample $r \xleftarrow{\$} \mathbb{Z}_N^*$ and let $g := -r^2 \bmod N$. Now notice that $r^2 \bmod N$ is a random element in the group of the quadratic residues and $-1 \bmod N \in \mathbb{J}_N \setminus \mathbb{QR}_N$. To see this, notice that for any safe prime p it holds that $p = 3 \bmod 4$. By Euler's criterion we have $\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} \bmod p$ for odd primes p and every x which is coprime to p. Therefore $\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = -1$, meaning that $-1 \bmod N \in \mathbb{J}_N \setminus \mathbb{QR}_N$. By multiplying a fixed element of $\mathbb{J}_N \setminus \mathbb{QR}_N$ with a random element of $\mathbb{QR}_N$ we obtain a random element of $\mathbb{J}_N \setminus \mathbb{QR}_N$.

As mentioned above, to sample a random element from $\mathbb{QR}_N$, we can sample $r \xleftarrow{\$} \mathbb{Z}_N^*$ and let $g := r^2 \bmod N$. Again g is a generator of $\mathbb{QR}_N$ with all but negligible probability. When the factors p, q are known, then it easy to check if the given element is a generator of $\mathbb{QR}_N$ by checking if $g^{p'} \neq 1 \bmod N \wedge g^{q'} \neq 1 \bmod N$. Therefore we are able to efficiently sample a random generator of $\mathbb{QR}_N$.

Since our constructions relies on the strong sequential squaring assumption either in the group $\mathbb{J}_N$ [MT19] or in the group $\mathbb{QR}_N$ [KLX20] for brevity we state the strong sequential squaring assumption in the group $\mathbb{G}$, where $\mathbb{G}$ is one of the mentioned groups. Let GenMod be a probabilistic polynomial-time algorithm which on input 1^λ outputs two λ-bit safe primes p and q, modulus $N = pq$ and a random generator g of the group $\mathbb{G}$.

$\mathsf{ExpSSS}^b_{\mathcal{A}}(\lambda)$:

$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(N, T(\lambda), g)$
$x \xleftarrow{\$} \mathbb{G}$
if $b = 0 : y := x^{2^{T(\lambda)}} \bmod N$
if $b = 1 : y \xleftarrow{\$} \mathbb{G}$
return $b' \leftarrow \mathcal{A}_{2,\lambda}(x, y, \mathsf{st})$

Fig. 1. Security experiment for the strong sequential squaring assumption.

Definition 6 (Strong Sequential Squaring Assumption (SSS)). *Consider the security experiment* $\mathsf{ExpSSS}^b_{\mathcal{A}}(\lambda)$ *in Figure 1. The* strong sequential squaring assumption *with gap* $0 < \epsilon < 1$ *holds relative to* GenMod *if there exists a polynomial* $\tilde{T}(\cdot)$ *such that for all polynomials* $T(\cdot) \geq \tilde{T}(\cdot)$ *and for every non-uniform polynomial-size adversary* $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$*, where the depth of* $\mathcal{A}_{2,\lambda}$ *is at most* $T^{\epsilon}(\lambda)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpSSS}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpSSS}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Next we define the DDH experiment in the group $\mathbb{J}_N$, as originally stated by Castagnos *et al.* [CPP16]. Castagnos *et al.* have shown that this problem is hard assuming that DDH is hard in the subgroups of $\mathbb{Z}^*_N$ of order p' and q' and that the quadratic residuosity problem is hard in $\mathbb{Z}^*_N$. We also define DDH experiment in the group of quadratic residues modulo N where the factors of N are given to an adversary. Castagnos *et al.* [CPP16] have shown that DDH problem is hard in $\mathbb{QR}_N$ assuming that DDH is hard in the large prime-order subgroups of $\mathbb{Z}^*_N$. This is shown as part of the proof of their Theorem 9. We remark that even though in the mentioned proof the prime factors p, q are not given to DDH adversary in the group $\mathbb{QR}_N$, but the proof relies on the fact that the constructed reduction knows factors p, q. Therefore the proof is valid even if p, q are given to DDH adversary in $\mathbb{QR}_N$ as input.

$\mathsf{ExpJ_N DDH}^b_{\mathcal{A}}(\lambda)$:	$\mathsf{ExpQR_N DDH}^b_{\mathcal{A}}(\lambda)$:
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$	$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\alpha, \beta \xleftarrow{\$} [\varphi(N)/2]$	$\alpha, \beta \xleftarrow{\$} [\varphi(N)/4]$
if $b = 0 : \gamma = a \cdot b \bmod \varphi(N)/2$	if $b = 0 : \gamma = a \cdot b \bmod \varphi(N)$
if $b = 1 : \gamma \xleftarrow{\$} [\varphi(N)/2]$	if $b = 1 : \gamma \xleftarrow{\$} [\varphi(N)/4]$
return $b' \leftarrow \mathcal{A}_\lambda(N, g, g^\alpha, g^\beta, g^\gamma)$	return $b' \leftarrow \mathcal{A}_\lambda(N, p, q, g, g^\alpha, g^\beta, g^\gamma)$

Fig. 2. Security experiments for DDH in $\mathbb{J}_N$ and $\mathbb{QR}_N$.

Definition 7 (Decisional Diffie-Hellman in $\mathbb{J}_N$). *Consider the security experiment* $\mathsf{ExpJ_N DDH}^b_{\mathcal{A}}(\lambda)$ *in Figure 2. The decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{J}_N$ *if for every non-uniform polynomial-size adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpJ_N DDH}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpJ_N DDH}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Definition 8 (Decisional Diffie-Hellman in $\mathbb{QR}_N$). *Consider the security experiment* $\mathsf{ExpQR_N DDH}^b_{\mathcal{A}}(\lambda)$ *in Figure 2. The decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{QR}_N$ *if for every non-uniform polynomial-size adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpQR_N DDH}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpQR_N DDH}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

$\mathsf{ExpDCR}^b_{\mathcal{A}}(\lambda)$:
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$y_1 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$
$y_0 = y_1^N \bmod N^2$
return $b' \leftarrow \mathcal{A}_\lambda(N, y_b)$

Fig. 3. Security experiment for DCR.

Definition 9 (Decisional Composite Residuosity Assumption). *Consider the security experiment* $\mathsf{ExpDCR}^b_{\mathcal{A}}(\lambda)$ *in Figure 3. The decisional composite residuosity assumption holds relative to* GenMod *if for every non-uniform polynomial-size adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{DCR}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpDCR}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpDCR}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

When designing an efficient simulation sound NIZK for our scheme, we rely on factoring assumption.

Definition 10 (Factoring Assumption). *The* factoring assumption *holds relative to* GenMod *if for every non-uniform polynomial-size adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{Factor}}_{\mathcal{A}} = \Pr\left[N = p'q' \ : \begin{array}{r}(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)\\ p', q' \leftarrow \mathcal{A}_\lambda(N),\\ \textit{such that } p', q' \in \mathbb{N}; p', q' > 1\end{array}\right] \leq \mathsf{negl}(\lambda).$$

To argue that our proof system fulfils required properties, we make of use the following lemma, which states that it is possible factorize N if a positive multiple of $\varphi(N)$ is known. The proof of this lemma is part of an analysis of [KL14, Theorem 8.50].

Lemma 1. *Let* $(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$ *and let* $M = \alpha\varphi(N)$ *for some positive integer* $\alpha \in \mathbb{Z}^+$*. There exists a PPT algorithm* $\mathsf{Factor}(N, M)$ *which, on input* (N, M)*, outputs* $p', q' \in \mathbb{N}$*,* $p', q' > 1$ *such that* $N = p'q'$ *with probability at least* $1 - 2^{-\lambda}$*.*

On sampling random exponents for $\mathbb{J}_N$ and $\mathbb{QR}_N$. Since in our construction the order $\varphi(N)/2$ of the group $\mathbb{J}_N$ and the order $\varphi(N)/4$ of $\mathbb{QR}_N$ are unknown, we use the set $[\lfloor N/2 \rfloor]$, respectively $[\lfloor N/4 \rfloor]$, whenever we should sample from the sets $[\varphi(N)/2]$, respectively $[\varphi(N)/4]$ without knowing the factorization of N. Sampling from $[\lfloor N/2 \rfloor]$ is statistically indistinguishable from sampling from $[\varphi(N)/2]$ and similarly sampling from $[\lfloor N/4 \rfloor]$ is statistically indistinguishable from sampling from $[\varphi(N)/4]$.

Definition 11 (Statistical Distance). *Let X and Y be two random variables over a finite set S. The statistical distance between X and Y is defined as*

$$\mathbb{SD}(X, Y) = \frac{1}{2} \sum_{s \in S} \left| \Pr[X = s] - \Pr[Y = s] \right| .$$

Lemma 2. *Let p, q be primes, $N = pq$, $\ell \in \mathbb{N}$ such that $\gcd(\ell, \varphi(N)) = \ell$ and X and Y be random variables defined on domain $[\lfloor N/\ell \rfloor]$ as follows:*

$$\Pr[X = r] = 1/\lfloor N/\ell \rfloor \ \ \forall r \in [\lfloor N/\ell \rfloor] \text{ and } \Pr[Y = r] = \begin{cases} \ell/\varphi(N) & \forall r \in [\varphi(N)/\ell] \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\mathbb{SD}(X, Y) \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

We defer the proof of this lemma to Supplementary Material A.

3 Standard Model Constructions

In this section we construct two non-malleable non-interactive timed commitment schemes whose security can be proven in standard model and which are either linearly (i.e., additively) or multiplicatively homomorphic. he constructions rely on non-interactive zero-knowledge proofs in the common reference string model.

3.1 Non-Interactive Zero-Knowledge Proofs

We recall the definition of a simulation-sound non-interactive proof system (SS-NIZK) that we take from Libert *et al.* [LNPY21].

Definition 12. *A* non-interactive zero-knowledge proof system *Π for an NP language L associated with a relation $\mathcal{R}$ is a tuple of four PPT algorithms $(\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_L, \mathsf{Prove}, \mathsf{Vrfy})$, which work as follows:*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, L)$ *takes a security parameter 1^λ and the description of a language L. It outputs a a common reference string* crs.

– $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, s, w)$ *is a PPT algorithm which takes as input the common reference string* crs*, a statement* s*, and a witness* w *such that* $(s, w) \in \mathcal{R}$ *and outputs a proof* π*.*
– $0/1 \leftarrow \mathsf{Vrfy}(\mathsf{crs}, s, \pi)$ *is a deterministic algorithm which takes as input the common reference string* crs*, a statement* s *and a proof* π *and outputs either 1 or 0, where 1 means that the proof is "accepted" and 0 means it is "rejected".*

Moreover, Π *should satisfy the following properties.*

– Completeness: *for all* $(s, w) \in \mathcal{R}$ *holds:*

$$\Pr[\mathsf{Vrfy}(\mathsf{crs}, s, \pi) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, L), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, s, w)] = 1.$$

– Soundness: *for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in N$

$$\mathbf{Snd}_{\mathcal{A}}^{\mathtt{NIZK}} = \Pr\left[\begin{matrix} s \notin L \wedge \\ \mathsf{Vrfy}(\mathsf{crs}, s, \pi) = 1 \end{matrix} : \begin{matrix} (\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, L) \\ (\pi, s) \leftarrow \mathcal{A}_\lambda(\mathsf{crs}, \tau_L) \end{matrix}\right] \leq \mathsf{negl}(\lambda),$$

where τ_L *is membership testing trapdoor.*
– Zero-Knowledge: *there is a PPT simulator* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$*, such that for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*:*

$$\begin{aligned}\mathbf{ZK}_{\mathcal{A}}^{\mathtt{NIZK}} = \Big|&\Pr\left[\mathcal{A}_\lambda^{\mathsf{Prove}(\mathsf{crs},\cdot,\cdot),}(\mathsf{crs}, \tau_L) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, L)\right] \\ &- \Pr\left[\mathcal{A}_\lambda^{\mathcal{O}(\mathsf{crs},\tau,\cdot,\cdot),}(\mathsf{crs}, \tau_L) = 1 : (\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda, L)\right]\Big| \leq \mathsf{negl}(\lambda).\end{aligned}$$

Here τ_L *is a membership testing trapdoor for language* L*;* $\mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$ *is an oracle that outputs* $\perp$ *on input* $(s, w) \notin \mathcal{R}$ *and outputs a valid proof* $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, s, w)$ *otherwise;* $\mathcal{O}(\mathsf{crs}, \tau, \cdot, \cdot)$ *is an oracle that outputs* $\perp$ *on input* $(s, w) \notin \mathcal{R}$ *and outputs a simulated proof* $\pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, \tau, s)$ *on input* $(s, w) \in \mathcal{R}$*. Note that the simulated proof is generated independently of the witness* w*.*

Remark 2. We have slightly modified the soundness and zero-knowledge definitions compared to [LNPY21]. Our soundness definition is adaptive and an adversary is given as input also a membership testing trapdoor τ_L. This notion is implied by the simulation-soundness as defined in Definition 13. Our zero-knowledge definition provides a membership testing trapdoor τ_L as an input for an adversary, whereas the definition of [LNPY21] lets an adversary generate the language L itself. The definition of [LNPY21] works in our constructions too, but we prefer to base our constructions on a slightly weaker definition.

Definition 13 (One-Time Simulation Soundness). *A NIZK for an NP language* L *with zero-knowledge simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *is* one-time simulation sound*, if for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$

there exists a negligible function $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{SimSnd}_{\mathcal{A}}^{\mathtt{NIZK}} = \Pr\left[\begin{matrix} s \notin L \wedge \\ (s,\pi) \neq (s',\pi') \wedge : \\ \mathsf{Vrfy}(\mathsf{crs}, s, \pi) = 1 \end{matrix} \begin{matrix} (\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda, L) \\ (s,\pi) \leftarrow \mathcal{A}_\lambda^{\mathsf{Sim}_2(\mathsf{crs},\tau,\cdot)}(\mathsf{crs}, \tau_L) \end{matrix}\right] \leq \mathsf{negl}(\lambda),$$

where τ_L *is a membership testing trapdoor for language* L *and* $\mathsf{Sim}_2(\mathsf{crs}, \tau, \cdot)$ *is a single query oracle which on input* s' *returns* $\pi' \leftarrow \mathsf{Sim}(\mathsf{crs}, \tau, s')$.

Libert *et al.* [LNPY21] show that given an additively homomorphic encryption scheme, one can build a trapdoor Sigma protocol for the language defined below. Moreover, any trapdoor Sigma protocol can be turned into an unbounded simulation sound NIZK which directly implies existence of a one-time simulation sound NIZK. Since we use the term *trapdoor Sigma protocol* only as intermediate notion and never instantiate it, we do not state formal definition and only reference it for brevity. For more details about trapdoor Sigma protocols see e.g. [LNPY21].

Lemma 3 (Lemma D.1 [LNPY21]). *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an additively homomorphic encryption scheme where the message space* M*, randomness space* R *and the ciphertext space* C *form groups* $(M,+),(R,+)$ *and* $(C,\cdot)$*. Let the encryption scheme be such that for any public key* pk *generated using* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, any messages* $m_1, m_2 \in M$ *and randomness* $r_1, r_2 \in R$ *holds*

$$\mathsf{Enc}(\mathsf{pk}, m_1; r_1) \cdot \mathsf{Enc}(\mathsf{pk}, m_2; r_2) = \mathsf{Enc}(\mathsf{pk}, m_1 + m_2; r_1 + r_2).$$

Let S *be a finite set of public cardinality such that uniform sampling from* S *is computationally indistinguishable from uniform sampling from* R*. Then there is an trapdoor Sigma protocol for the language* $L := \{c \in C | \exists r \in R : c = \mathsf{Enc}(\mathsf{pk}, 0; r)\}$ *of encryptions of zero, where* pk *is fixed by the language.*

Remark 3. We note that Libert *et al.* required that the order of the group $(R,+)$ is public and that this group is efficiently samplable, which is used in their proof of the zero-knowledge property. This is however, not necessary, since it is sufficient to be able to sample from a distribution which is computationally indistinguishable from the uniform distribution. This results in computational indistinguishability of real and simulated transcripts. In case of our constructions, we will sample randomness from a distribution which is statistically close and hence indistinguishable from the uniform distribution over R, which yields that the real and the simulated transcripts are statistically indistinguishable.

Additionally, Libert *et al.* construct a simulation sound non-interactive argument system from any trapdoor Sigma protocol relying on a strongly unforgeable one-time signature, a lossy public-key encryption scheme, an admissible hash function and a correlation intractable hash function.

Theorem 1 (Thm B.1, Thm. B.2 [LNPY21]). *Let* $(\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_L, \mathsf{Prove}, \mathsf{Vrfy})$ *be a trapdoor Sigma protocol for an NP language* L*. Then given a strongly unforgeable one-time signature scheme,* $\mathcal{R}$*-lossy public-key encryption scheme, a*

correlation intractable hash function and an admissible hash function, there is an unbounded simulation sound non-interactive zero-knowledge proof system for the language L.

We note that in order to achieve negligible soundness error, it is needed to run the underlying trapdoor Sigma protocol $\mathcal{O}(\log \lambda)$ times in parallel. One run of the trapdoor Sigma protocol of Libert *et al.* for L, as defined above, corresponds to sending one ciphertext of the homomorphic encryption scheme and one random element $r \in R$.

3.2 Standard-Model Instantiation of SS-NIZKs

In this section we provide simulation sound NIZK proof systems for languages L_1 and L_2 that are used in our constructions. The languages are defined in the following way:

$$L_1 = \left\{ (c_0, c_1, c_2, c_3) | \exists (m, r) : \begin{array}{r} (\wedge_{i=1}^{3} c_i = h_i^{rN}(1+N)^m \bmod N^2) \wedge \\ c_0 = g^r \bmod N \end{array} \right\} \text{ and}$$

$$L_2 = \left\{ (c_0, c_1, c_2) | \exists (m, r) : (\wedge_{i=1}^{2} c_i = h_i^r m \bmod N) \wedge c_0 = g^r \bmod N \right\},$$

where g, h_1, h_2, h_3, N are parameters defining the languages.

Note that L_1 consists of all ciphertexts $(c_0, c_1 \cdot (c_2)^{-1}, c_3 \cdot (c_2)^{-1})$ that are encryptions of zero, where the corresponding public key is defined as $\mathsf{pk} := (g, (h_1 \cdot (h_2)^{-1}), (h_3 \cdot (h_2)^{-1}), N)$ and encryption is defined as $\mathsf{Enc}(\mathsf{pk} := (g, h, h'), m) : c := g^r \bmod N, c' := h^{rN}(1+N)^m \bmod N^2, c' := h'^{rN}(1+N)^m \bmod N^2$. L_2 consists of all ciphertexts $(c_0, c_1 \cdot (c_2)^{-1})$ that are encryptions of zero, where the corresponding public key is defined as $\mathsf{pk} := (g, (h_1 \cdot (h_2)^{-1})), N)$ and encryption is defined as $\mathsf{Enc}(\mathsf{pk} := (g, h), m) : c := g^r, c' := hg^m \bmod N$. Hence, both encryption schemes are additively homomorphic and by Lemma 3 we obtain a trapdoor Sigma protocol for the languages L_1, L_2. By Theorem 1 this yields unbounded simulation-sound NIZKs for these languages.

3.3 Construction of Linearly Homomorphic Non-Malleable NITC

We start with a construction of linearly homomorphic non-malleable NITC. In our construction depicted in Figure 4 we rely on a one-time simulation-sound NIZK for the following language:

$$L = \left\{ (c_0, c_1, c_2, c_3) | \exists (m, r) : \begin{array}{r} (\wedge_{i=1}^{3} c_i = h_i^{rN}(1+N)^m \bmod N^2) \wedge \\ c_0 = g^r \bmod N \end{array} \right\},$$

where g, h_1, h_2, h_3, N are parameters specifying the language.

$\underline{\mathsf{PGen}(1^\lambda, T)}$
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\varphi(N) := (p-1)(q-1)$
$k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$
$t := 2^T \bmod \varphi(N)/2$
For $i \in [2] : h_i := g^{k_i} \bmod N$
$h_3 := g^t \bmod N$
$\mathsf{crs}_{\mathtt{NIZK}} \leftarrow \mathtt{NIZK.Setup}(1^\lambda, L)$
return $\mathsf{crs} := (N, T, g, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}})$

$\underline{\mathsf{Com}(\mathsf{crs}, m)}$
$r \xleftarrow{\$} [\lfloor N/2 \rfloor]$
$c_0 := g^r \bmod N$
For $i \in [3] : c_i := h_i^{rN}(1+N)^m \bmod N^2$
$c := (c_0, c_1, c_2, c_3), w := (m, r)$
$\pi_{\mathsf{Com}} \leftarrow \mathtt{NIZK.Prove}(\mathsf{crs}_{\mathtt{NIZK}}, c, w)$
$\pi_{\mathsf{Dec}} := r$
return $(c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}})$

$\underline{\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}})}$
return $\mathtt{NIZK.Vrfy}(\mathsf{crs}_{\mathtt{NIZK}}, c, \pi_{\mathsf{Com}})$

$\underline{\mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}})}$
Parse c as (c_0, c_1, c_2, c_3)
if $\wedge_{i=1}^3 c_i = h_i^{\pi_{\mathsf{Dec}} N}(1+N)^m \bmod N^2$
$\wedge c_0 = g^{\pi_{\mathsf{Dec}}} \bmod N$
    return 1
return 0

$\underline{\mathsf{FDec}(\mathsf{crs}, c)}$
Parse c as (c_0, c_1, c_2, c_3)
Compute $\pi_{\mathsf{FDec}} := c_0^{2^T} \bmod N$
$m := \frac{c_3 \cdot \pi_{\mathsf{FDec}}^{-N} (\bmod N^2) - 1}{N}$
return (m, π_{FDec})

$\underline{\mathsf{FDecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{FDec}})}$
Parse c as $(c_0, c_1, c_2, c_3$
if $c_3 = \pi_{\mathsf{FDec}}^N (1+N)^m \bmod N^2$
    return 1
return 0

$\underline{\mathsf{Eval}(\mathsf{crs}, \oplus_N, c_1, \ldots, c_n)}$
Parse c_i as $(c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3})$
Compute $c_0 := \prod_{i=1}^n c_{i,0} \bmod N, c_1 := \perp, c_2 := \perp, c_3 := \prod_{i=1}^n c_{i,3} \bmod N^2$
return $c := (c_0, c_1, c_2, c_3)$

Fig. 4. Construction of Linearly Homomorphic NITC in Standard Model.
$\oplus_N$ refers to addition mod$N$

Theorem 2. *If* $(\mathtt{NIZK.Setup}, \mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ *is a one-time simulation-sound non-interactive zero-knowledge proof system for* L*, the strong sequential squaring assumption with gap* ϵ *holds relative to* GenMod *in* $\mathbb{J}_N$*, the Decisional Composite Residuosity assumption holds relative to* GenMod*, and the Decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{J}_N$*, then* $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *defined in Figure 4 is an IND-CCA-secure non-interactive timed commitment scheme with gap* $\underline{\epsilon}$*, for any* $\underline{\epsilon} < \epsilon$*.*

Proof. Completeness is implied by the completeness of the NIZK and can be verified by straightforward inspection.

To prove security we define a sequence of games $\mathsf{G}_0 - \mathsf{G}_{12}$. For $i \in \{0, 1, \ldots, 12\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$ outputs b' in the game G_i such that $b = b'$.

$\underline{\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, i)}$
Parse c as (c_0, c_1, c_2, c_3)
if $\mathtt{NIZK}.\mathsf{Vrfy}(\mathsf{crs}_{\mathtt{NIZK}}, (c_0, c_1, c_2, c_3), \pi_{\mathsf{Com}}) = 1$
    Compute $y := c_0^{k_i} \bmod N$
    return $\frac{c_i \cdot y^{-N} (\bmod N^2) - 1}{N}$
return $\perp$

Fig. 5. Decommitment oracle

Game 0. Game G_0 corresponds to the original security experiment where decommitment queries are answered using FDec.

Game 1. In game G_1 decommitment queries are answered using the algorithm Dec defined in Figure 5 with $i := 1$, meaning that secret key k_1 and ciphertext c_1 are used, to answer decommitment queries efficiently.

Lemma 4.

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

Notice that if c_1 and c_3 contain the same message, both oracles answer decommitment queries consistently. Let E denote the event that the adversary $\mathcal{A}$ asks a decommitment query (c, π_{Com}) such that its decommitment using the key k_1 is different from its decommitment using FDec. Since G_0 and G_1 are identical until E happens, we bound the probability of E. Concretely, we have

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ that breaks soundness of the NIZK. It is given as input $\mathsf{crs}_{\mathtt{NIZK}}$ together with a membership testing trapdoor $\tau_L := (k_1, k_2, t)$ where $t := 2^T \bmod \varphi(N)/2$. The adversary $\mathcal{B}_\lambda(\mathsf{crs}_{\mathtt{NIZK}}, \tau_L)$ proceeds as follows:

1. It computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N, h_3 := g^t \bmod N$ using the membership testing trapdoor $\tau_L := (k_1, k_2, t)$ and sets $\mathsf{crs} := (N, T, g, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}})$.
2. Then it runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. It samples $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/2 \rfloor]$ and computes $c_0^* := g^r, c_1^* := h_1^{rN}(1+N)^{m_b}, c_2^* := h_2^{rN}(1+N)^{m_b}, c_3^* := h_3^{rN}(1+N)^{m_b}$. It sets $(s := (c_0^*, c_1^*, c_2^*, c_3^*), w := (m, r))$ and runs $\pi^* \leftarrow \mathtt{NIZK}.\mathsf{Prove}(s, w)$.
4. It runs $b' \leftarrow \mathcal{A}_{2,\lambda}(s, \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
5. Finally, it checks whether there exists a decommitment query (c, π_{Com}) such that $\mathsf{DEC}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 2)$. If E occurs, then this is the case, and it returns (c, π_{Com}). Notice that this check can be done efficiently with the knowledge of t, since instead of running FDec, $\mathcal{B}$ can verify the proof and compute $c_3 c_0^{-t} \bmod N$ which produces the same output as FDec.

$\mathcal{B}$ simulates G_1 perfectly and if the event E happens, then it outputs a valid proof for a statement which is not in the specified language L. Therefore we get

$$\Pr[\mathsf{E}] \leq \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

Game 2. Game G_2 proceeds exactly as the previous game but we run the zero-knowledge simulator $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda, L)$ in PGen and produce a simulated proof for the challenge commitment as $\pi^* \leftarrow \mathsf{Sim}_2(\mathsf{crs}, \tau, (c_0^*, c_1^*, c_2^*, c_3^*))$. By zero-knowledge security of underlying NIZK we directly obtain

Lemma 5.

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \mathbf{ZK}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against the zero-knowledge security of NIZK as follows: $\mathcal{B}_\lambda(\mathsf{crs}_{\mathtt{NIZK}}, \tau_L)$:

1. It sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}})$ and runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1, which is included in $\tau_L = (k_1, k_2, t)$.
2. It samples $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/2 \rfloor]$ and computes $c_0^* := g^r, c_1^* := h_1^{rN}(1+N)^{m_b}, c_2^* := h_2^{rN}(1+N)^{m_b}, c_3^* := h_3^{rN}(1+N)^{m_b}$. It submits $(s := (c_0^*, c_1^*, c_2^*, c_3^*), w := (m, r))$ to its oracle and obtains proof π^* as answer.
3. Then it runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
4. Finally, it returns the truth value of $b = b'$.

If the proof π^* is generated using $\mathtt{NIZK}.\mathsf{Prove}$, then $\mathcal{B}$ simulates G_1 perfectly. Otherwise π^* is generated using Sim_1 and $\mathcal{B}$ simulates G_2 perfectly. This proves the lemma.

Game 3. In G_3 we sample r uniformly at random from $[\varphi(N)/2]$.

Lemma 6.

$$|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Since the only difference between the two games is in the set from which we sample r, to upper bound the advantage of adversary we can use Lemma 2 with $\ell := 2$, which directly yields the required bound.

Game 4. In G_4 we sample $y_3 \xleftarrow{\$} \mathbb{J}_N$ and compute c_3^* as $y_3^N(1+N)^{m_b}$.

Let $\tilde{T}_{\mathsf{SSS}}(\lambda)$ be the polynomial whose existence is guaranteed by the SSS assumption. Let $\mathsf{poly}_{\mathcal{B}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Steps 1–2 and answer decommitment queries in Step 3 of the adversary $\mathcal{B}_{2,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathcal{B}}(\lambda))^{1/\epsilon}$. Set $\tilde{T}_{\mathtt{NITC}} := \max(\tilde{T}_{\mathsf{SSS}}, \underline{T})$.

Lemma 7. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ for some $T(\cdot) \geq \underline{T}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}}.$$

The adversary $\mathcal{B}_{1,\lambda}(N, T(\lambda), g)$ proceeds as follows:

1. It samples $k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N, h_3 := g^{2^{T(\lambda)}} \bmod N$, runs $(\mathsf{crs}_{\mathtt{NIZK}}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}})$. Notice that value h_3 is computed by repeated squaring.
2. It runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Finally, it outputs $(N, g, k_1, k_2, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}}, \tau, m_0, m_1, \mathsf{st})$

The adversary $\mathcal{B}_{2,\lambda}(x, y, (N, g, k_1, k_2, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}}, \tau, m_0, m_1, \mathsf{st}))$:

1. Samples $b \xleftarrow{\$} \{0, 1\}$, computes $c_0^* := x, c_1^* := x^{k_1 N}(1+N)^{m_b}, c_2^* := x^{k_2 N}(1+N)^{m_b}, c_3^* := y^N(1+N)^{m_b}$.
2. Runs $\pi^* \leftarrow \mathsf{Sim}_2(\mathsf{crs}_{\mathtt{NIZK}}, \tau, (c_0^*, c_1^*, c_2^*, c_3^*))$.
3. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*), \mathsf{st})$ and answers decommitment queries using k_1.
4. Returns the truth value of $b = b'$.

Since g is a generator of $\mathbb{J}_N$ and x is sampled uniformly at random from $\mathbb{J}_N$ there exists some $r \in [\varphi(N)/2]$ such that $x = g^r$. Therefore when $y = x^{2^T} = (g^{2^T})^r \bmod N$, then $\mathcal{B}$ simulates G_3 perfectly. Otherwise y is random value and $\mathcal{B}$ simulates G_4 perfectly.

Now we analyse the running time of the constructed adversary. Adversary $\mathcal{B}_1$ computes h_3 by $T(\lambda)$ consecutive squarings and because $T(\lambda)$ is polynomial in λ, $\mathcal{B}_1$ is efficient. Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathcal{B}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}(\lambda) \leq 2T^{\epsilon}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{NITC}}(\cdot) \geq \tilde{T}_{\mathsf{SSS}}(\cdot)$ as required.

Game 5. In G_5 we sample $y_3 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and compute c_3^* as $y_3(1+N)^{m_b}$.

Lemma 8.

$$|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq \mathbf{Adv}^{\mathsf{DCR}}_{\mathcal{B}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ against DCR. $\mathcal{B}_\lambda(N, y)$ works as follows:

1. It samples $g, y_3, x \xleftarrow{\$} \mathbb{J}_N, k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N, h_3 := g^{2^T} \bmod N$, runs $(\mathsf{crs}_{\mathtt{NIZK}}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}})$. Notice that value h_3 is computed by repeated squaring.

2. It runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Then it samples $b \xleftarrow{\$} \{0,1\}, w \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that $\left(\frac{y}{N}\right) = \left(\frac{w}{N}\right)$. We remark that computing Jacobi symbol can be done efficiently without knowing factorization of N.
4. It computes $c_0^* := x, c_1^* := x^{k_1 N}(1+N)^{m_b}, c_2^* := x^{k_2 N}(1+N)^{m_b}, c_3^* := yw^N(1+N)^{m_b}$. Runs $\pi^* \leftarrow \mathsf{Sim}_2(\mathsf{crs}_{\mathtt{NIZK}}, \tau, (c_0^*, c_1^*, c_2^*, c_3^*))$.
5. It runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
6. Then it returns the truth value of $b = b'$.

If $y = v^N \bmod N^2$ then $yw^N = v^N w^N = (vw)^N$ and hence yw^N is N-th residue. Moreover, the Jacobi symbol of yw is 1, since the Jacobi symbol is multiplicatively homomorphic. Therefore $\mathcal{B}$ simulates G_4 perfectly.

Otherwise, if y is uniform random element in $\mathbb{Z}^*_{N^2}$, then yw^N is also uniform among all elements of $\mathbb{Z}^*_{N^2}$ that have Jacobi symbol 1 and $\mathcal{B}$ simulates G_5 perfectly. This proves the lemma.

We remark that at this point c_3^* does not reveal any information about b. Here we use that if $x = y \bmod N$ then $\left(\frac{x}{N}\right) = \left(\frac{y}{N}\right)$ and that there is an isomorphism $f : \mathbb{Z}^*_N \times \mathbb{Z}_N \to \mathbb{Z}^*_{N^2}$ given by $f(u,v) = u^N(1+N)^v = u^N(1+vN) \bmod N^2$ (see e.g. [KL14, Proposition 13.6]). Since $f(u,v) \bmod N = u^N + u^N vN \bmod N = u^N \bmod N$, that means that Jacobi symbol $\left(\frac{f(u,v)}{N}\right)$ depends only on u. Hence if $\left(\frac{f(u,v)}{N}\right) = 1$, then it must hold that $\left(\frac{f(u,r)}{N}\right) = 1$ for all $r \in \mathbb{Z}_N$. This implies that a random element $f(u,v)$ in $\mathbb{Z}^*_{N^2}$ with $\left(\frac{f(u,v)}{N}\right) = 1$ has a uniformly random distribution of v in $\mathbb{Z}_N$. Therefore if $yw^N = u^N(1+N)^v \bmod N^2$ then $yw^N(1+N)^{m_b} = u^N(1+N)^{m_b+v} \bmod N^2$. Since v is uniform in $\mathbb{Z}_N$, $(m_b + v)$ is also uniform in $\mathbb{Z}_N$, which means that ciphertext c_3^* does not reveal any information about b.

Game 6. In G_6 we sample k_2 uniformly at random from $[\varphi(N)/2]$.

Lemma 9.

$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Again using a statistical argument this lemma directly follows from Lemma 2 with $\ell := 2$.

Game 7. In G_7 we sample $y_2 \xleftarrow{\$} \mathbb{J}_N$ and compute c_2^* as $y_2^N(1+N)^{m_b}$.

Lemma 10.

$$|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_7 = 1]| \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against DDH in the group $\mathbb{J}_N$.

$\mathcal{B}_\lambda(N, g, g^\alpha, g^\beta, g^\gamma)$ proceeds as follows:

1. It samples $k_1 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_3 := g^{2^T} \bmod N$, runs $(\mathsf{crs}_{\mathtt{NIZK}}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{crs} := (N, T, g, h_1, h_2 := g^\alpha, h_3, \mathsf{crs}_{\mathtt{NIZK}})$.
2. It runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. It samples $b \xleftarrow{\$} \{0,1\}, y_3 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and computes $(c_0^*, c_1^*, c_2^*, c_3^*) := (g^\beta, (g^\beta)^{k_1 N}(1+N)^{m_b}, (g^\gamma)^N (1+N)^{m_b}, y_3(1+N)^{m_b})$. Runs $\pi^* \leftarrow \mathsf{Sim}_2(\mathsf{crs}_{\mathtt{NIZK}}, \tau, (c_0^*, c_1^*, c_2^*, c_3^*))$.
4. It runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
5. It returns the truth value of $b = b'$.

If $\gamma = \alpha\beta$, then $\mathcal{B}$ simulates G_6 perfectly. Otherwise g^γ is uniform random element in $\mathbb{J}_N$ and $\mathcal{B}$ simulates G_7 perfectly. This proves the lemma.

Game 8. In G_8 we sample k_2 uniformly at random from $[\lfloor N/2 \rfloor]$. Again by Lemma 2 with $\ell := 2$ we get

Lemma 11.
$$|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_8 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Game 9. In G_9 we sample $y_2 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and compute c_2^* as $y_2(1+N)^{m_b}$.

Lemma 12.
$$|\Pr[\mathsf{G}_8 = 1] - \Pr[\mathsf{G}_9 = 1]| \leq \mathbf{Adv}^{\mathsf{DCR}}_{\mathcal{B}}.$$

This can be proven in similar way as Lemma 8. We remark that at this point c_2^* does not reveal any information about b, with the same argument as in the transition from G_4 to G_5.

Game 10. In G_{10} we answer decommitment queries using Dec (Figure 5) with $i := 2$ which means that secret key k_2 and ciphertext c_2 are used.

Lemma 13.
$$|\Pr[\mathsf{G}_9 = 1] - \Pr[\mathsf{G}_{10} = 1]| \leq \mathbf{SimSnd}^{\mathtt{NIZK}}_{\mathcal{B}}.$$

Let E denote the event that adversary $\mathcal{A}$ asks a decommitment query (c, π_{Com}) such that its decommitment using the key k_1 is different from its decommitment using the key k_2. Since G_9 and G_{10} are identical until E happens, it is sufficient to bound the probability of E. Concretely,

$$|\Pr[\mathsf{G}_9 = 1] - \Pr[\mathsf{G}_{10} = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ that breaks one-time simulation soundness of the NIZK. It is given as input $\mathsf{crs}_{\mathtt{NIZK}}$ together with a membership testing trapdoor $\tau_L := (k_1, k_2, t)$, where $t := 2^T \bmod \varphi(N)/2$. The adversary $\mathcal{B}^{\mathsf{Sim}_2}_\lambda(\mathsf{crs}_{\mathtt{NIZK}}, \tau_L)$ proceeds as follows:

1. It computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N, h_3 := g^t \bmod N$ using the membership testing trapdoor τ_L and sets $\mathsf{crs} := (N, T, g, h_1, h_2, h_3, \mathsf{crs}_{\mathtt{NIZK}})$.
2. It runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_2.
3. It samples $b \xleftarrow{\$} \{0,1\}, x \xleftarrow{\$} \mathbb{J}_N, y_2, y_3 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ and computes $(c_0^*, c_1^*, c_2^*, c_3^*) := (x, x^{k_1 N}(1+N)^{m_b}, y_2(1+N)^{m_b}, y_3(1+N)^{m_b})$. Forwards $(c_0^*, c_1^*, c_2^*, c_3^*)$ to simulation oracle Sim_2 and obtains a proof π^*.
4. It runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_2.
5. If there exists a decommitment query (c, π_{Com}) such that $\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 1) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 2)$, then it returns (c, π_{Com}). Note that such a query exists iff E happens.

$\mathcal{B}$ simulates G_{10} perfectly and if the event E happens, it outputs a valid proof for a statement which is not in the specified language L. Therefore we get

$$\Pr[\mathsf{E}] \leq \mathbf{SimSnd}^{\mathtt{NIZK}}_{\mathcal{B}}.$$

Game 11. In G_{11} we sample k_1 uniformly at random from $[\varphi(N)/2]$. The following again follows directly from Lemma 2 with $\ell := 2$.

Lemma 14.

$$|\Pr[\mathsf{G}_{10} = 1] - \Pr[\mathsf{G}_{11} = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Game 12. In G_{12} we sample $y_1 \xleftarrow{\$} \mathbb{J}_N$ and compute c_1^* as $y_1^N(1+N)^{m_b}$.

Lemma 15.

$$|\Pr[\mathsf{G}_{11} = 1] - \Pr[\mathsf{G}_{12} = 1]| \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}}.$$

This can be proven in similar way as Lemma 10.

Game 13. In G_{13} we sample $y_1 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and compute c_1^* as $y_1(1+N)^{m_b}$.

Lemma 16.

$$|\Pr[\mathsf{G}_{12} = 1] - \Pr[\mathsf{G}_{13} = 1]| \leq \mathbf{Adv}^{\mathsf{DCR}}_{\mathcal{B}}.$$

This can be proven in similar way as Lemma 8. We remark that at this point c_1^* does not reveal any information about b, with the same arguments as above.

Lemma 17.

$$\Pr[\mathsf{G}_{13} = 1] = \frac{1}{2}.$$

Clearly, c_0^* is uniform random element in $\mathbb{J}_N$ and hence it does not contain any information about the challenge message. Since y_1, y_2, y_3 are sampled uniformly at random from $\mathbb{Z}_{N^2}^*$ the ciphertexts c_1^*, c_2^*, c_3^* are also uniform random elements in $\mathbb{Z}_{N^2}^*$ and hence do not contain any information about the challenge message m_b. Therefore, an adversary can not do better than guessing.

By combining Lemmas 4 - 17 we obtain the following:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathtt{NITC}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \leq \sum_{i=0}^{12} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]| + \left|\Pr[\mathsf{G}_{13} - \frac{1}{2}\right| \\ &\leq \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}} + \mathbf{ZK}_{\mathcal{B}}^{\mathtt{NIZK}} + \mathbf{Adv}_{\mathcal{B}}^{\mathsf{SSS}} + \mathbf{SimSnd}_{\mathcal{B}}^{\mathtt{NIZK}} + 2\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}} + 3\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DCR}} \\ &\quad + 4\left(\frac{1}{p} + \frac{1}{q} - \frac{1}{N}\right).\end{aligned}$$

Theorem 3. ($\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}$) *defined in Figure 4 is a BND-CCA-secure non-interactive timed commitment scheme.*

Proof. We show that the construction is actually perfectly binding. This is straightforward to show since Paillier encryption is perfectly binding. Therefore there is exactly one message/randomness pair (m, r) which can pass the check in $\mathsf{DecVrfy}$. Therefore the first winning condition of the BND-CCA experiment happens with probability 0. Moreover, since PGen is executed by the challenger, the value h_3 is computed correctly and therefore FDec reconstructs always the correct message m. Therefore the second winning condition of BND-CCA experiment happens with probability 0 as well.

Theorem 4. *If* $\mathtt{NIZK} = (\mathtt{NIZK.Setup}, \mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ *is a non-interactive zero-knowledge proof system for* L*, then* ($\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}, \mathsf{FDecVrfy}$) *defined in Figure 4 is a publicly verifiable non-interactive timed commitment scheme.*

Proof. Completeness is straightforward to verify. To prove soundness, notice that if the commitment verifies, then we know that $c_0 = g^r$ and $c_3 = h_3^r(1+N)^m$ for honestly generated g and h_3 and some r and m. Otherwise, an adversary would be able to break soundness of the proof system. Since there is an isomorphism $f : \mathbb{Z}_N^* \times \mathbb{Z}_N \to \mathbb{Z}_{N^2}^*$ given by $f(a, b) = a^N(1+N)^b \bmod N^2$ (see e.g. [KL14, Proposition 13.6]) there exist unique values π_{FDec} and m such that $c_3 = \pi_{\mathsf{FDec}}^N(1+N)^m \bmod N^2$. Therefore adversary is not able to provide a different message m' fulfilling the required equation.

Finally, note that $\mathsf{FDecVrfy}$ is efficient, with a running time which is independent of T.

It is straightforward to verify that considering the Eval algorithm, our construction yields a l*inearly homomorphic* NITC, which follows from the linear homorphism of Paillier, as also used in [MT19].

Theorem 5. *The NITC* ($\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}, \mathsf{FDecVrfy}, \mathsf{Eval}$) *defined in Figure 4 is a linearly homomorphic non-interactive timed commitment scheme.*

3.4 Construction of Multiplicatively Homomorphic Non-Malleable NITC

The construction described in this section is similar to that from Section 3.3, except that we replace Paillier encryption with ElGamal to obtain a multiplicative homomorphism and the construction is based on standard Naor-Yung paradigm. Our construction is given in Figure 6 and we rely on a one-time simulation sound NIZK for the following language:

$$L = \left\{(c_0, c_1, c_2) | \exists (m, r) : (\wedge_{i=1}^{2} c_i = h_i^r m \bmod N) \wedge c_0 = g^r \bmod N\right\},$$

where g, h_1, h_2, N are parameters specifying the language.

$\underline{\mathsf{PGen}(1^\lambda, T)}$	$\underline{\mathsf{Com}(\mathsf{crs}, m)}$
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$	$r \xleftarrow{\$} [\lfloor N/4 \rfloor]$
$\varphi(N) := (p-1)(q-1)$	$c_0 := g^r \bmod N$
$k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$	For $i \in [2] : c_i := h_i^r m \bmod N$
$t := 2^T \bmod \varphi(N)/4$	$c := (c_0, c_1, c_2), w := (m, r)$
$h_1 := g^{k_1} \bmod N$	$\pi_{\mathsf{Com}} \leftarrow \mathtt{NIZK}.\mathsf{Prove}(\mathsf{crs}_{\mathtt{NIZK}}, c, w)$
$h_2 := g^t \bmod N$	$\pi_{\mathsf{Dec}} := r$
$\mathsf{crs}_{\mathtt{NIZK}} \leftarrow \mathtt{NIZK}.\mathsf{Setup}(1^\lambda, L)$	return $(c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}})$
return $\mathsf{crs} := (N, T, g, h_1, h_2, \mathsf{crs}_{\mathtt{NIZK}})$	
$\underline{\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}})}$	$\underline{\mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}})}$
return $\mathtt{NIZK}.\mathsf{Vrfy}(\mathsf{crs}_{\mathtt{NIZK}}, c, \pi)$	Parse c as (c_0, c_1, c_2)
	if $\wedge_{i=1}^{2} c_i = h_i^{\pi_{\mathsf{Dec}}} m \bmod N \wedge c_0 = g^{\pi_{\mathsf{Dec}}} \bmod N$
	return 1
	return 0
$\underline{\mathsf{FDec}(\mathsf{crs}, c)}$	$\underline{\mathsf{Eval}(\mathsf{crs}, \otimes_N, c_1, \ldots, c_n)}$
Parse c as (c_0, c_1, c_2)	Parse c_i as $(c_{i,0}, c_{i,1}, c_{i,2})$
Compute $y := c_0^{2^T} \bmod N$	Compute $c_0 := \prod_{i=1}^{n} c_{i,0} \bmod N, c_1 := \perp$
$m := c_2 \cdot y^{-1} \bmod N$	Compute $c_2 := \prod_{i=1}^{n} c_{i,2} \bmod N$
return m	return $c := (c_0, c_1, c_2)$

Fig. 6. Construction of Multiplicatively Homomorphic NITC in Standard Model. $\otimes_N$ refers to multiplication mod$N$

Theorem 6. *If* $(\mathtt{NIZK}.\mathsf{Setup}, \mathtt{NIZK}.\mathsf{Prove}, \mathtt{NIZK}.\mathsf{Vrfy})$ *is a one-time simulation-sound non-interactive zero-knowledge proof system for* L*, the strong sequential squaring assumption with gap* ϵ *holds relative to* GenMod *in* $\mathbb{QR}_N$*, and the Decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{QR}_N$*, then* $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *defined in Figure 6 is an IND-CCA-secure non-interactive timed commitment scheme with* $\underline{\epsilon}$*, for any* $\underline{\epsilon} < \epsilon$*.*

The proof can be found in Supplementary Material B.

Theorem 7. ($\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}$) *defined in Figure 6 is a BND-CCA-secure non-interactive timed commitment scheme.*

Proof. We show that the construction is perfectly binding. This is straightforward to show since ElGamal encryption is perfectly binding. Therefore there is exactly one message/randomness pair (m, r) which can pass the check in $\mathsf{DecVrfy}$. Therefore the first winning condition of BND-CCA experiment happens with probability 0. Moreover, since PGen is executed by the challenger, the value h_3 is computed correctly and therefore FDec reconstructs always the correct message m. Therefore the second winning condition of BND-CCA experiment happens with probability 0 as well.

It is straightforward to verify that considering Eval algorithm, our construction yields multiplicatively homomorphic NITC.

Theorem 8. *The NITC* ($\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}, \mathsf{FDecVrfy}, \mathsf{Eval}$) *defined in Figure 6 is a multiplicatively homomorphic non-interactive timed commitment scheme.*

Remark 4 (Public Verifiability). It is natural to ask if it is possible to make the construction in Figure 6 publicly verifiable. Since the output m of FDec is perfectly determined by value $y := c_0^{2^T} \bmod N$, it is possible to achieve public verifiability if one can efficiently check that indeed y equals to $c_0^{2^T} \bmod N$ without executing T squarings. However, this is exactly what proofs of exponentiation of Pietrzak [Pie19] and Wesolowski [Wes19] do. Concretely, [Pie19,Wes19] propose efficient proofs systems for the language $L' := \{(G, a, b, T) | a, b \in G \wedge b = a^{2^T}\}$ where G is some group where low order assumption [Pie19] or adaptive root assumption [Wes19] hold. We remark, that for both suggested proof systems G can be instantiated for example as $\mathbb{Z}_N^*/\{-1, 1\}$ [Wes19,BBF18] or as it is proposed by Pietrzak G can be instantiated as a group of signed quadratic residues $\mathbb{QR}_N^+ := \{|x| : x \in \mathbb{QR}_N\}$. One can argue that the strong sequential squaring assumption holds in $\mathbb{QR}_N^+$ (see e.g. [Pie19,EFKP20]). Therefore by adjusting the construction in Figure 6 to work in the group $\mathbb{QR}_N^+$, one can obtain publicly verifiable NITC by outputing in FDec the value y together with a proof of exponentiation that $y = c_0^{2^T} \bmod N$ and $\mathsf{FDecVrfy}$ just checks that the proof of exponentiation is valid and at the same time $c_2 = y \cdot m \bmod N$. For completeness we provide a description of these algorithms in Figure 7, where we use ($\mathsf{PoE.Prover}, \mathsf{PoE.Vrfy}$) to denote a proof system for language L'. Both Pietrzak's and Wesolowski's proof system are interactive protocols which might be made non-interactive using Fiat-Shamir transformation. Thus we obtain a publicly verifiable NITC in ROM.

$\underline{\mathsf{FDec}(\mathsf{crs}, c)}$	$\underline{\mathsf{FDecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{FDec}})}$
Parse c as (c_0, c_1, c_2)	Parse c as (c_0, c_1, c_2)
$y := c_0^{2^T} \bmod N$, $\pi_{\mathtt{PoE}} = \mathtt{PoE}.\mathsf{Prove}(c_0, y)$	if $c_2 = m \cdot y \bmod N \wedge \mathtt{PoE}.\mathsf{Vrfy}((c_0, y), \pi_{\mathtt{PoE}})$
$\pi_{\mathsf{FDec}} := (y, \pi_{\mathtt{PoE}}), m := c_2 \cdot y{-}1 \bmod N$	return 1
return (m, π_{FDec})	return 0

Fig. 7. FDec and FDecVrfy of Publicly Verifiable NITC

4 Random Oracle Model Constructions

We are able to obtain more efficient construction of non-malleable non-interactive timed commitments when we instantiate the non-interactive zero-knowledge proof systems in the random oracle model [BR93]. Since the underlying languages and proofs differ in some subtle but crucial ways from the standard model constructions, we provide the constructions together with proofs in full detail.

4.1 Non-Interactive Zero-Knowledge Proofs in the Random Oracle Model

Definition 14. *A* non-interactive proof system *for an NP language L with relation $\mathcal{R}$ is a pair of algorithms* $(\mathsf{Prove}, \mathsf{Vrfy})$*, which work as follows:*

- $\pi \leftarrow \mathsf{Prove}(s, w)$ *is a PPT algorithm which takes as input a statement s and a witness w such that $(s, w) \in \mathcal{R}$ and outputs a proof π.*
- $\mathsf{Vrfy}(s, \pi) \in \{0, 1\}$ *is a deterministic algorithm which takes as input a statement s and a proof π and outputs either 1 or 0, where 1 means that the proof is "accepted" and 0 means it is "rejected".*

We say that a non-interactive proof system is complete*, if for all $(s, w) \in \mathcal{R}$ holds:*

$$\Pr[\mathsf{Vrfy}(s, \pi) = 1 : \pi \leftarrow \mathsf{Prove}(s, w)] = 1.$$

We say that a non-interactive proof system is sound *if for all non-uniform polynomial-size adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in N$*

$$\mathbf{Snd}_{\mathcal{A}}^{\mathtt{NIZK}} = \Pr\left[s \notin L \wedge \mathsf{Vrfy}(s, \pi) = 1 : (\pi, s) \leftarrow \mathcal{A}_\lambda\right] \leq \mathsf{negl}(\lambda).$$

Next we define the *zero-knowledge* property for non-interactive proof system in the random oracle model. The simulator Sim of a non-interactive zero-knowledge proof system is modelled as a stateful algorithm which provides two modes, namely $(\pi, \mathsf{st}) \leftarrow \mathsf{Sim}(1, \mathsf{st}, s)$ for answering proof queries and $(v, \mathsf{st}) \leftarrow \mathsf{Sim}(2, \mathsf{st}, u)$ for answering random oracle queries. The common state st is updated after each operation.

Definition 15 (Zero-Knowledge in the ROM). *Let* $(\mathsf{Prove}, \mathsf{Vrfy})$ *be a non-interactive proof system for a relation* $\mathcal{R}$ *which may make use of a hash function* $H : \mathcal{U} \to \mathcal{V}$. *Let* $\mathsf{Funs}[\mathcal{U}, \mathcal{V}]$ *be the set of all functions from the set* $\mathcal{U}$ *to the set* $\mathcal{V}$. *We say that* $(\mathsf{Prove}, \mathsf{Vrfy})$ *is* non-interactive zero-knowledge proof in the random oracle model (NIZK)*, if there exists an efficient simulator* Sim *such that for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{ZK}_{\mathcal{A}}^{\mathtt{NIZK}} = \left| \Pr\left[\mathcal{A}_\lambda^{\mathsf{Prove}^H(\cdot,\cdot),H(\cdot)} = 1\right] - \Pr\left[\mathcal{A}_\lambda^{\mathsf{Sim}_1(\cdot,\cdot),\mathsf{Sim}_2(\cdot)}\right] = 1 \right| \leq \mathsf{negl}(\lambda),$$

where

- H *is a function sampled uniformly at random from* $\mathsf{Funs}[\mathcal{U}, \mathcal{V}]$,
- Prove^H *corresponds to the* Prove *algorithm, having oracle access to* H,
- $\pi \leftarrow \mathsf{Sim}_1(s, w)$ *takes as input* $(s, w) \in \mathcal{R}$, *and outputs the first output of* $(\pi, \mathsf{st}) \leftarrow \mathsf{Sim}(1, \mathsf{st}, s)$,
- $v \leftarrow \mathsf{Sim}_2(u)$ *takes as input* $u \in \mathcal{U}$ *and outputs the first output of* $(v, \mathsf{st}) \leftarrow \mathsf{Sim}(2, \mathsf{st}, u)$.

Definition 16 (One-Time Simulation Soundness). *Let* $(\mathsf{Prove}, \mathsf{Vrfy})$ *be a non-interactive proof system for an NP language* L *with zero-knowledge simulator* Sim. *We say that* $(\mathsf{Prove}, \mathsf{Vrfy})$ *is* one-time simulation sound *in the random oracle model, if for all non-uniform polynomial-size adversaries* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{SimSnd}_{\mathcal{A}}^{\mathtt{NIZK}} = \Pr\left[\begin{matrix} s \notin L \wedge (s, \pi) \neq (s', \pi') \\ \wedge \mathsf{Vrfy}^{\mathsf{Sim}_2(\cdot)}(s, \pi) = 1 \end{matrix} : (s, \pi) \leftarrow \mathcal{A}_\lambda^{\mathsf{Sim}_1(\cdot),\mathsf{Sim}_2(\cdot)}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Sim}_1(\cdot)$ *is a single query oracle which on input* s' *returns the first output of* $(\pi', \mathsf{st}) \leftarrow \mathsf{Sim}(1, \mathsf{st}, s')$ *and* $\mathsf{Sim}_2(u)$ *returns the first output of* $(v, \mathsf{st}) \leftarrow \mathsf{Sim}(2, \mathsf{st}, u)$.

4.2 Efficient Instantiation of SS-NIZK in the ROM

In this section we provide efficient simulation sound NIZK proof systems in the ROM for languages L_3 and L_4 that are used in our constructions. The languages are defined in the following way:

$$L_3 = \left\{ (h_1, h_2, c_0, c_1, c_2, c_3) | \exists (m, r) : \begin{matrix} (\wedge_{i=1}^3 c_i = h_i^{rN}(1+N)^m \bmod N^2) \wedge \\ c_0 = g^r \bmod N \end{matrix} \right\} \text{ and}$$

$$L_4 = \left\{ (h_1, h_2, c_0, c_1, c_2) | \exists (m, r) : (\wedge_{i=1}^2 c_i = h_i^r m \bmod N) \wedge c_0 = g^r \bmod N \right\},$$

where g, h_3, N are parameters defining the language. For the language L_3 this is equivalent to proving that $(c_0^N, (h_1 \cdot (h_2)^{-1})^N, (c_1 \cdot (c_2)^{-1}))$ and $(c_0^N, (h_3 \cdot (h_2)^{-1})^N, (c_3 \cdot (c_2)^{-1}))$ are two DDH tuples where all computations are done mod N^2. Similarly, for the language L_4 this is equivalent to proving that $(c_0, (h_1 \cdot$

$(h_2)^{-1}), (c_1 \cdot (c_2)^{-1}))$ is a DDH tuple where all computations are done mod N. Therefore, we can instantiate the required NIZKs using Sigma protocols that prove that given tuples are DDH tuples. These Sigma protocols can be turned into efficient simulation-sound NIZKs in the ROM using the Fiat-Shamir transformation [FS87]. We remark that we have to design a Sigma protocols in a hidden order group setting, since the orders of the groups $\mathbb{J}_N$ and $\mathbb{QR}_N$ are known neither by the prover, nor by the verifier. Current constructions of Sigma protocols in hidden order groups are far less efficient than standard Sigma protocols. However, we observe that to obtain simulation-sound NIZKs, it is sufficient to design a Sigma protocol which has a negligible soundness error and we do not have to care about special soundness. Therefore we are able to avoid the strong RSA assumption which is often needed in Sigma protocols in hidden order groups to prove special soundness. As a result we are able to avoid both using an unnecessary large modulus N and a large number of sequential repetitions as discussed in [BKS+09]. This results in particularly short proofs.

We recall at first definition of a Sigma protocol.

Definition 17. *A Sigma protocol for an NP language L and a corresponding binary relation $\mathcal{R}$ is an interactive 3-move protocol Σ between two interactive algorithms which we will call* Prover *and* Verifier*.* Prover *is given input a statement and a witness (s, w) and the* Verifier *is given as input a statement s. The protocol works as follows:*

1. Prover *starts the protocol by computing a message a, called the* commitment*, and sends a to* Verifier*.* commitment*, and a state* st*. The message a is sent to the* Verifier*.*
2. Verifier *receives a and samples a* challenge *c uniformly from a finite challenge space $\mathcal{C}$ and sends it to the* Prover*.*
3. Prover *receives the challenge c end computes a* response *$z \in \mathcal{Z}$. It sends the response z to* Verifier*.*
4. Verifier *runs a deterministic function $\mathsf{V}(s, a, c, z)$ which outputs 0 or 1 meaning reject or accept, respectively.*

A triple (a, c, z) is called a transcript *of the Sigma protocol. A transcript (a, c, z) is called an* accepting *transcript for s if it cause $\mathsf{V}(s, a, c, z) = 1$. We say that a Sigma protocol is* complete *if for all $(s, w) \in \mathcal{R}$ it holds that whenever a $\mathsf{Prover}(s, w)$ and a $\mathsf{Verifier}(s)$ interact, then the* Verifier *always accepts.*

Definition 18 (Honest Verifier Zero-Knowledge). *We say that a Sigma protocol for an NP language L is* honest verifier zero-knowledge (HVZK)*, if there exists a PPT simulator* Sim *which takes as input $s \in L$ and outputs transcript (a, c, z) that that has the same distribution as honest transcript resulting from interactions between* Prover *and* Verifier *on common input s.*

Definition 19 (Soundness). *We say that a Sigma protocol for an NP language L is* sound*, if a proof for a statement $x \notin L$ output by any (even unbounded) adversary is accepted only with negligible probability.*

We define also slightly stronger soundness definition where adversary is given an auxiliary input.

Definition 20 (Soundness with respect to auxiliary input). *We say that a Sigma protocol for an NP language L is* sound with respect to auxiliary input aux*, if a proof for a statement $x \notin L$ output by any (even unbounded) adversary which is given* aux *as input is accepted only with negligible probability. We use* $\mathbf{Snd}^{\sigma}_{\mathcal{A}}$ *to denote the advantage of* $\mathcal{A}$*.*

Definition 21 (Quasi Unique Responses). *Let Σ be a Sigma protocol. We say that Σ has* quasi unique responses *if for every non-uniform polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$*

$$\mathbf{OUR}^{\Sigma}_{\mathcal{B}} = \Pr\left[\begin{array}{r}\mathsf{V}(s,a,c,z) = \mathsf{V}(s,a,c,z') = 1\\ \wedge z \neq z' : (s,a,c,z,z') \leftarrow \mathcal{A}_\lambda\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 22 (Unpredictable Commitments). *Let Σ be a Sigma protocol for a relation $\mathcal{R}$ with transcripts $(a,c,z) \in A \times \mathcal{C} \times \mathcal{Z}$. We say that Σ has δ-*unpredictable commitments *if for all $(s,w) \in \mathcal{R}$ and for all $a^* \in A$ holds that with probability at most δ, an interaction between $\mathsf{Prover}(s,w)$ and $\mathsf{Verifier}(s)$ produce a transcript (a,c,z) with $fm^* = a$.*

Faust *et al.* [FKMV12] have shown that Sigma protocols fulfilling the above definitions can be turned into simulation-sound NIZK via the Fiat-Shamir transform.

Theorem 9 ([FKMV12]). *Consider a non-trivial three-round public-coin honest verifier zero-knowledge interactive proof system* $(\mathsf{Prover},\mathsf{Verifier})$ *for an NP language L, with quasi unique responses. In the random oracle model, the proof system* $(\mathsf{Prove},\mathsf{Vrfy})$ *derived from* $(\mathsf{Prover},\mathsf{Verifier})$ *via the Fiat-Shamir transform is a simulation-sound NIZK with respect to its canonical simulator* Sim*.*

Now we are ready to describe our Sigma protocol $\Sigma = (\mathsf{Prover},\mathsf{Verifier})$ for language L_3. At first, we recall that $f : \mathbb{Z}^*_N \times \mathbb{Z}_N \to \mathbb{Z}^*_{N^2}$ defined as $f(x,y) = x^N(1+N)^y$ is an isomorphism. Since g is generator of $\mathbb{J}_N$ and has order $\varphi(N)/2$, also $g^N \bmod N^2$ has order $\varphi(N)/2$.

1. Prover samples $\alpha \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $a_0 := g^{N\alpha}, a_1 := (h_1 \cdot h_2^{-1})^{N\alpha}, a_2 := (h_3 \cdot h_2^{-1})^{N\alpha} \bmod N^2$ and sends (a_0,a_1,a_2) to $\mathsf{Verifier}$.
2. $\mathsf{Verifier}$ samples $v \xleftarrow{\$} [2^d]$ and sends v to Prover.
3. Prover computes the response $z := \alpha + v \cdot r$ and sends it to $\mathsf{Verifier}$.
4. $\mathsf{Verifier}$ accepts if and only if $g^{Nz} = a_0 \cdot c_0^{Nv} \wedge (h_1 \cdot h_2^{-1})^{Nz} = a_1 \cdot (c_1 \cdot c_2^{-1})^v \wedge (h_3 \cdot h_2^{-1})^{Nz} = a_2 \cdot (c_3 \cdot c_2^{-1})^v$ where all computation are done $\bmod N^2$ and $z \in [\lfloor N/2 \rfloor + v\,\lfloor N/2 \rfloor]$.

Theorem 10. *If the factoring assumption holds relative* GenMod*, then the above defined protocol* $\Sigma = (\mathsf{Prover},\mathsf{Verifier})$ *is a Sigma protocol for language L_3 that is perfectly complete, honest verifier zero-knowledge, sound with respect to auxiliary input* $\mathsf{aux} = (p,q)$ *and has quasi-unique responses.*

Proof. Completeness is straightforward to verify:

1. $g^{Nz} = g^{N(\alpha+v\cdot r)} = g^{N\alpha} + (g^r)^{Nv} = a_0 \cdot c_0^{Nv} \bmod N^2$;
2. $(h_1 \cdot h_2^{-1})^{Nz} = (h_1 \cdot h_2^{-1})^{N(\alpha+v\cdot r)} = (h_1 \cdot h_2^{-1})^{N\alpha} \cdot (h_1^{Nr} \cdot (h_2^{Nr})^{-1})^v = a_1 \cdot (h_1^{Nr} \cdot m \cdot (h_2^{Nr} \cdot m)^{-1})^v = a_1 \cdot (c_1 \cdot c_2^{-1})^v \bmod N^2$;
3. $(h_3 \cdot h_2^{-1})^{Nz} = (h_3 \cdot h_2^{-1})^{N(\alpha+v\cdot r)} = (h_3 \cdot h_2^{-1})^{N\alpha} \cdot (h_3^{Nr} \cdot (h_2^{Nr})^{-1})^v = a_1 \cdot (h_3^{Nr} \cdot m \cdot (h_2^{Nr} \cdot m)^{-1})^v = a_2 \cdot (c_3 \cdot c_2^{-1})^v \bmod N^2$;
4. Since $r, \alpha \in [\lfloor N/2 \rfloor]$ and $v \in [2^d]$, therefore $z := \alpha + v \cdot r \in [\lfloor N/2 \rfloor + v \lfloor N/2 \rfloor]$.

HVZK. The simulator $\mathsf{Sim}((h_1, h_2, c_0, c_1, c_2, c_3))$ works as follows:

1. Samples $v \xleftarrow{\$} [2^d], z \xleftarrow{\$} [\lfloor N/2 \rfloor + v \lfloor N/2 \rfloor]$.
2. Computes $a_0 := g^{Nz} \cdot c_0^{-v}, a_1 := (h_1 \cdot h_2^{-1})^{Nz} \cdot (c_1 \cdot c_2^{-1})^{-v}, a_2 := (h_3 \cdot h_2^{-1})^{Nz} \cdot (c_3 \cdot c_2^{-1})^{-v} \bmod N^2$ and returns $((a_0, a_1, a_2), v, z)$.

It is easy to verify that the produced transcript is valid and has the same distribution as an honest transcript.

Soundness. Let $((a_0, a_1, a_2), v, z)$ is accepting transcript for $(h_1, h_2, c_0, c_1, c_2, c_3) \notin L_3$. That means $c_0 = g^{Nr_0}, (c_1 \cdot (c_2)^{-1}) = (h_1 \cdot (h_2)^{-1})^{Nr_1} \bmod N, (c_3 \cdot (c_2)^{-1}) = (h_3 \cdot (h_2)^{-1})^{Nr_2} \bmod N$ and $r_0 \neq r_1 \mod \varphi(N)/2$ or $r_0 \neq r_2 \mod \varphi(N)/2$. Let $a_0 = g^{N\alpha_0} \bmod N^2, a_1 = (h_1 \cdot (h_2)^{-1})^{N\alpha_1} \bmod N^2, a_2 = (h_3 \cdot (h_2)^{-1})^{N\alpha_2} \bmod N^2$. Considering the first verification equation, taking discrete logarithms to base g^N, the second verification equation with discrete logarithms to base $(h_1 \cdot h_2^{-1})^N$, and the third verification equation with discrete logarithms to base $(h_3 \cdot h_2^{-1})^N$, we obtain following equations:

$$z = \alpha_0 + r_0 v \bmod \varphi(N)/2 \tag{1}$$

$$z = \alpha_1 + r_1 v \bmod \varphi(N)/2 \tag{2}$$

$$z = \alpha_2 + r_2 v \bmod \varphi(N)/2 \tag{3}$$

Computing (1)-(2) we obtain

$$0 = (\alpha_0 - \alpha_1) + (r_0 - r_1)v \bmod \varphi(N)/2$$

$$v = \frac{\alpha_1 - \alpha_0}{r_0 - r_1} \bmod \varphi(N)/2$$

Computing (1)-(3) we obtain

$$0 = (\alpha_0 - \alpha_2) + (r_0 - r_2)v \bmod \varphi(N)/2$$

$$v = \frac{\alpha_2 - \alpha_0}{r_0 - r_2} \bmod \varphi(N)/2$$

Now notice that at least one of $(r_0 - r_1)$ and $(r_0 - r_2)$ is not zero and values $\alpha_0, \alpha_1, \alpha_2, r_0, r_1, r_2$ are fixed before challenge v is provided by $\mathsf{Verifier}$. Therefore, in order to provide an accepting proof, the adversary has to predict value v. If we choose $2^d < \varphi(N)/2$, then v is unique and therefore the probability that the adversary produces an accepting transcript for a statement that is not in L_3 is at most 2^{-d}, which can be set to be negligible in λ. This holds even for unbounded adversaries.

Soundness with auxiliary input $\mathsf{aux} = (p, q)$ Notice that proof of soundness holds even if an adversary is given as input factorization of N.

Quasi Unique Responses. We show that our Sigma protocol has quasi unique responses, otherwise we are able to factorize N. Assume that an adversary can output a statement $((h_1, h_2, c_0, c_1, c_2, c_3), (a_0, a_1, a_2), v, z, z')$ such that $((a_0, a_1, a_2), v, z)$ and $((a_0, a_1, a_2), v, z')$ are accepting transcripts for $(h_1, h_2, c_0, c_1, c_2, c_3)$ and $z \neq z'$. Therefore using the first verification equation it holds $g^{Nz} = a_0 \cdot c_0^v = g^{Nz'} \bmod N \implies z = z' \bmod \varphi(N)/2$. Hence, $z - z' = \alpha \cdot \varphi(N)/2 \implies 2(z - z') = \alpha \cdot \varphi(N)$ for some $\alpha \neq 0$. Applying Lemma 1 for $M = 2(z - z')$ we are able to factorize N.

This concludes the proof.

Next we describe our Sigma protocol $\Sigma = (\mathsf{Prover}, \mathsf{Verifier})$ for language L_4:

1. Prover samples $\alpha \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $a_0 := g^\alpha \bmod N, a_1 := (h_1 \cdot h_2^{-1})^\alpha \bmod N$ and sends (a_0, a_1) to $\mathsf{Verifier}$.
2. $\mathsf{Verifier}$ samples $v \xleftarrow{\$} [2^d]$ and sends v to Prover.
3. Prover computes the response $z := \alpha + v \cdot r$ and sends it to $\mathsf{Verifier}$.
4. $\mathsf{Verifier}$ accepts if and only if $g^z = a_0 \cdot c_0^v \bmod N \wedge (h_1 \cdot h_2^{-1})^z = a_1 \cdot (c_1 \cdot c_2^{-1})^v \bmod N \wedge z \in [\lfloor N/4 \rfloor + v \lfloor N/4 \rfloor]$.

Theorem 11. *If the factoring assumption holds relative* GenMod*, then the above defined protocol* $\Sigma = (\mathsf{Prover}, \mathsf{Verifier})$ *is a Sigma protocol for language* L_4 *that is perfectly complete, honest verifier zero-knowledge, sound with auxiliary input* $aux := (p, q)$*, has quasi-unique responses and* $\frac{2}{p'q'}$*-unpredictable commitments.*

Proof. We prove required properties.

Completeness. We verify the given requirements:

1. $g^z = g^{\alpha + v \cdot r} = g^\alpha + (g^r)^v = a_0 \cdot c_0^v \bmod N$;
2. $(h_1 \cdot h_2^{-1})^z = (h_1 \cdot h_2^{-1})^{\alpha + v \cdot r} = (h_1 \cdot h_2^{-1})^\alpha \cdot (h_1^r \cdot (h_2^r)^{-1})^v = a_1 \cdot (h_1^r \cdot m \cdot (h_2^r \cdot m)^{-1})^v = a_1 \cdot (c_1 \cdot c_2^{-1})^v$;
3. since $r, \alpha \in [\lfloor N/4 \rfloor]$ and $v \in [2^d]$, therefore $z := \alpha + v \cdot r \in [\lfloor N/4 \rfloor + v \lfloor N/4 \rfloor]$.

HVZK. The simulator $\mathsf{Sim}((h_1, h_2, c_0, c_1, c_2))$ works as follows:

1. Samples $v \xleftarrow{\$} [2^d], z \xleftarrow{\$} [\lfloor N/4 \rfloor + v \lfloor N/4 \rfloor]$.
2. Computes $a_0 := g^z \cdot c_0^{-v}, a_1 := (h_1 \cdot h_2^{-1})^z \cdot (c_1 \cdot c_2^{-1})^{-v}$ and returns $((a_0, a_1), v, z)$.

It is easy to verify that produced transcript is accepting and moreover it has the same distribution as a honest transcript.

Soundness. Let $((a_0, a_1), v, z)$ be an accepting transcript for $(h_1, h_2, c_0, c_1, c_2) \notin L_4$. That means $c_0 = g^{r_0}, (c_1 \cdot (c_2)^{-1}) = (h_1 \cdot (h_2)^{-1})^{r_1} \bmod N$, and $r_0 \neq r_1 \bmod \varphi(N)/4$. Let $a_0 = g^{\alpha_0} \bmod N, a_1 = (h_1 \cdot (h_2)^{-1})^{\alpha_1} \bmod N$. Considering the first verification equation with discrete logarithms to base g and the second verification equation with discrete logarithm to base $(h_1 \cdot h_2^{-1})$ we obtain following equations:

$$z = \alpha_0 + r_0 v \bmod \varphi(N)/4 \tag{4}$$
$$z = \alpha_1 + r_1 v \bmod \varphi(N)/4 \tag{5}$$
$$\tag{6}$$

Computing (1)-(2) we obtain

$$0 = (\alpha_0 - \alpha_1) + (r_0 - r_1)v \bmod \varphi(N)/4$$
$$v = \frac{\alpha_1 - \alpha_0}{r_0 - r_1} \bmod \varphi(N)/4$$

Now notice that $(r_0 - r_1)$ is not zero and values $\alpha_0, \alpha_1, r_0, r_1$, are fixed before challenge v is provided by $\mathsf{Verifier}$. Therefore in order to provide accepting proof, the adversary has to correctly guess value v. If we choose $2^d < \varphi(N)/4$ value v is unique and therefore probability the adversary produce accepting transcript for a statement that is not in L_4 is at most 2^{-d} which can be set to be negligible in λ. This holds even for unbounded adversaries.

Soundness with auxiliary input $\mathsf{aux} = (p, q)$ Notice that proof of soundness holds even if an adversary is given as input factorization of N.

Quasi Unique Responses. We show that our Sigma protocol has quasi unique responses, otherwise we are able to factorize N. Assume that an adversary can output a statement $((h_1, h_2, c_0, c_1, c_2), (a_0, a_1), v, z, z')$ such that $((a_0, a_1), v, z)$ and $((a_0, a_1), v, z')$ are accepting transcripts for $(h_1, h_2, c_0, c_1, c_2)$ and $z \neq z'$. Therefore using the first verification equation it holds $g^z = a_0 \cdot c_0^v = g^{z'} \bmod N \implies z = z' \bmod \varphi(N)/4$. Hence, $z - z' = \alpha \cdot \varphi(N)/4 \implies 4(z - z') = \alpha \cdot \varphi(N)$ for some $\alpha \neq 0$. Applying Lemma 1 for $M = 4(z - z')$ we are able to factorize N.

Unpredictable Commitments . Elements a_0, a_1 are essentially random elements in $\mathbb{QR}_N$. But since we sample exponent α from $[\lfloor N/4 \rfloor]$ which is slightly bigger than $\varphi(N)/4$ some elements are twice so likely to be sampled. Therefore $\delta = 2\frac{4}{\varphi(N)/4} = \frac{2}{p'q'}$.

By Theorem 9 our Sigma protocols can be turned into simulation-sound NIZKs with alogrithms $(\mathsf{Prove}, \mathsf{Vrfy})$ in the random oracle model via the Fiat-Shamir transform. Moreover since the underlying sigma protocols are sound with respect to auxiliary input $\mathsf{aux} := (p, q)$ the NIZKs obtained via Fiat-Shamir transformation are also sound with respect to auxiliary input in the sense of Definition 20.

4.3 Construction of Linearly Homomorphic Non-Malleable NITC

We define language for our construction of a linearly homomorphic NITC depicted in Figure 8 which relies on a one-time simulation sound NIZKs in the ROM in the following way:

$$L = \left\{ (h_1, h_2, c_0, c_1, c_2, c_3) | \exists (m, r) : \begin{array}{r} (\wedge_{i=1}^{3} c_i = h_i^{rN}(1+N)^m \bmod N^2) \wedge \\ c_0 = g^r \bmod N \end{array} \right\},$$

where g, h_3, N are parameters specifying the language.

$\underline{\mathsf{PGen}(1^\lambda, T)}$
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\varphi(N) := (p-1)(q-1)$
$k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$
$t := 2^T \bmod \varphi(N)/2$
For $i \in [2] : h_i := g^{k_i} \bmod N$
$h_3 := g^t \bmod N$
return $\mathsf{crs} := (N, T, g, h_1, h_2, h_3)$

$\underline{\mathsf{Com}(\mathsf{crs}, m)}$
$r \xleftarrow{\$} [\lfloor N/2 \rfloor]$
$c_0 := g^r \bmod N$
For $i \in [3] : c_i := h_i^{rN}(1+N)^m \bmod N^2$
$\Phi := (h_1, h_2, c_0, c_1, c_2, c_3), w := (m, r)$
$\pi_{\mathsf{Com}} \leftarrow \mathtt{NIZK}.\mathsf{Prove}(\Phi, w)$
$c := (c_0, c_1, c_2, c_3)$
$\pi_{\mathsf{Dec}} := r$
return $(c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}})$

$\underline{\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}})}$
Parse c as (c_0, c_1, c_2, c_3)
return $\mathtt{NIZK}.\mathsf{Vrfy}((h_1, h_2, c_0, c_1, c_2, c_3), \pi)$

$\underline{\mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}})}$
Parse c as (c_0, c_1, c_2, c_3)
if $\wedge_{i=1}^{3} c_i = h_i^{\pi_{\mathsf{Dec}} N}(1+N)^m \bmod N^2$
$\wedge c_0 = g^{\pi_{\mathsf{Dec}}} \bmod N$
    return 1
return 0

$\underline{\mathsf{FDec}(\mathsf{crs}, c)}$
Parse c as (c_0, c_1, c_2, c_3)
Compute $\pi_{\mathsf{FDec}} := c_0^{2^T} \bmod N$
$m := \frac{c_3 \cdot \pi_{\mathsf{FDec}}^{-N} (\bmod N^2) - 1}{N}$
return (m, π_{FDec})

$\underline{\mathsf{FDecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{FDec}})}$
Parse c as (c_0, c_1, c_2, c_3)
if $c_3 = \pi_{\mathsf{FDec}}^{N}(1+N)^m \bmod N^2$
    return 1
return 0

$\underline{\mathsf{Eval}(\mathsf{crs}, \oplus_N, c_1, \ldots, c_n)}$
Parse c_i as $(c_{i,0}, c_{i,1}, c_{i,2}, c_{i,3})$
Compute $c_0 := \prod_{i=1}^{n} c_{i,0} \bmod N, c_1 := \perp, c_2 := \perp, c_3 := \prod_{i=1}^{n} c_{i,3} \bmod N^2$
return $c := (c_0, c_1, c_2, c_3, \pi)$

Fig. 8. Construction of Linearly Homomorphic NITC in ROM.
$\oplus_N$ refers to addition mod$N$

Theorem 12. *If* $\mathtt{NIZK} = (\mathtt{NIZK}.\mathsf{Prove}, \mathtt{NIZK}.\mathsf{Vrfy})$ *is a one-time simulation-sound non-interactive zero-knowledge proof system for* L *which is sound with respect*

to auxiliary input $\mathsf{aux} := (p, q)$*, the strong sequential squaring assumption with gap* ϵ *holds relative to* GenMod *in* $\mathbb{J}_N$*, the Decisional Composite Residuosity assumption holds relative to* GenMod*, and the Decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{J}_N$*, then* (PGen, Com, ComVrfy, DecVrfy, FDec) *defined in Figure 8 is an IND-CCA-secure non-interactive timed commitment scheme with* $\underline{\epsilon}$*, for any* $\underline{\epsilon} < \epsilon$*.*

The proof can be found in Supplementary Material C.

Theorem 13. (PGen, Com, ComVrfy, DecVrfy, FDec) *defined in Figure 8 is a BND-CCA-secure non-interactive timed commitment scheme.*

Proof. This can be proven in the same way as Theorem 3.

Theorem 14. *If* $\mathtt{NIZK} = (\mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ *is a non-interactive zero-knowledge proof system for* L*, then* (PGen, Com, ComVrfy, DecVrfy, FDec, FDecVrfy) *defined in Figure 8 is a publicly verifiable non-interactive timed commitment scheme.*

Proof. This can be proven in the same way as Theorem 4.

It is straightforward to verify that considering Eval algorithm, our construction yields linearly homomorphic NITC.

Theorem 15. *The NITC* (PGen, Com, ComVrfy, DecVrfy, FDec, FDecVrfy, Eval) *defined in Figure 8 is a linearly homomorphic non-interactive timed commitment scheme.*

4.4 Construction of Multiplicatively Homomorphic Non-Malleable NITC

We define language for our construction of a multiplicatively homomorphic NITC which relies on a Sigma protocol in the following way:

$$L = \left\{(h_1, h_2, c_0, c_1, c_2) | \exists (m, r) : (\wedge_{i=1}^{3} c_i = h_i^r m \bmod N) \wedge c_0 = g^r \bmod N\right\},$$

where g, N are parameters specifying the language.

Our construction is given in Figure 9 where $(\mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ is the Fiat-Shamir transform of a Sigma protocol for language L. Since it is not straightforward to provide a security proof directly with respect to standard definition of one-time simulation sound NIZK, we provide the security proof in the random oracle model relying on the properties of the underlying Sigma protocol.

Theorem 16. *If* $\Sigma =$ (Prover, Verifier) *is a Sigma protocol for* L *with quasi unique responses and* δ*-unpredictable commitments which is sound with respect to auxiliary input* $\mathsf{aux} = (p, q)$ *and is honest verifier zero-knowledge,* $H : \mathbb{QR}_N^7 \to [2^d]$ *is a hash function modelled as a random oracle, the strong sequential squaring assumption with gap* ϵ *holds relative to* GenMod *in* $\mathbb{QR}_N$*, and the Decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{QR}_N$*, then* (PGen, Com, ComVrfy, DecVrfy, FDec) *defined in Figure 9 is an IND-CCA-secure non-interactive timed commitment scheme with* $\underline{\epsilon}$*, for any* $\underline{\epsilon} < \epsilon$*.*

$\underline{\mathsf{PGen}(1^\lambda, T)}$
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\varphi(N) := (p-1)(q-1)$
$k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$
$t := 2^T \bmod \varphi(N)/4$
$h_1 := g^{k_1} \bmod N$
$h_2 := g^t \bmod N$
return $\mathsf{crs} := (N, T, g, h_1, h_2)$

$\underline{\mathsf{Com}(\mathsf{crs}, m)}$
$r \xleftarrow{\$} [\lfloor N/2 \rfloor]$
$c_0 := g^r \bmod N$
For $i \in [2] : c_i := h_i^r m \bmod N$
$\Phi := (h_1, h_2, c_0, c_1, c_2), w := (m, r)$
$\pi_{\mathsf{Com}} \leftarrow \mathtt{NIZK}.\mathsf{Prove}(\Phi, w)$
$c := (c_0, c_1, c_2)$
$\pi_{\mathsf{Dec}} := r$
return $(c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}})$

$\underline{\mathsf{ComVrfy}(\mathsf{crs}, c, \pi_{\mathsf{Com}})}$
Parse c as (c_0, c_1, c_2)
return $\mathtt{NIZK}.\mathsf{Vrfy}((h_1, h_2, c_0, c_1, c_2), \pi_{\mathsf{Com}})$

$\underline{\mathsf{DecVrfy}(\mathsf{crs}, c, m, \pi_{\mathsf{Dec}})}$
Parse c as (c_0, c_1, c_2)
if $\wedge_{i=1}^2 c_i = h_i^{\pi_{\mathsf{Dec}}} m \bmod N \wedge c_0 = g^{\pi_{\mathsf{Dec}}} \bmod N$
return 1
return 0

$\underline{\mathsf{FDec}(\mathsf{crs}, c)}$
Parse c as (c_0, c_1, c_2)
Compute $y := c_0^{2^T} \bmod N$
$m := c_2 \cdot y^{-1} \bmod N$
return m

$\underline{\mathsf{Eval}(\mathsf{crs}, \otimes_N, c_1, \ldots, c_n)}$
Parse c_i as $(c_{i,0}, c_{i,1}, c_{i,2})$
Compute $c_0 := \prod_{i=1}^n c_{i,0} \bmod N, c_1 := \perp$
Compute $c_2 := \prod_{i=1}^n c_{i,2} \bmod N$
return $c := (c_0, c_1, c_2)$

Fig. 9. Construction of Multiplicatively Homomorphic NITC in ROM.
$\otimes_N$ refers to multiplication mod$N$

The proof can be found in Supplementary Material D.

Theorem 17. $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *defined in Figure 9 is a BND-CCA-secure non-interactive timed commitment scheme.*

Proof. This can be proven in the same way as Theorem 7.

It is straightforward to verify that considering Eval algorithm, our construction yields multiplicatively homomorphic NITC.

Theorem 18. *The NITC* $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}, \mathsf{FDecVrfy}, \mathsf{Eval})$ *defined in Figure 9 is a multiplicatively homomorphic non-interactive timed commitment scheme.*

Remark 5 (Public Verifiability). The construction can be made publicly verifiable in the same way as suggested in Remark 4.

Acknowledgements

We would like to thank Julian Loss for helpful discussions in an early stage of this work.

References

BBF18. Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. https://eprint.iacr.org/2018/712.

BDD+21. Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. TARDIS: A foundation of time-lock puzzles in UC. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 429–459. Springer, Heidelberg, October 2021.

BDGM19. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 407–437. Springer, Heidelberg, December 2019.

BGJ+16. Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *ITCS 2016*, pages 345–356. ACM, January 2016.

BJS10. Jean-François Biasse, Michael J. Jacobson, and Alan K. Silvester. Security estimates for quadratic field based cryptosystems. In Ron Steinfeld and Philip Hawkes, editors, *ACISP 10*, volume 6168 of *LNCS*, pages 233–247. Springer, Heidelberg, July 2010.

BKS+09. Endre Bangerter, Stephan Krenn, Ahmad-Reza Sadeghi, Thomas Schneider, and Joe-Kai Tsay. On the design and implementation of efficient zero-knowledge proofs of knowledge. In *SPEED-CC 2009*, 2009.

BMV16. Silvio Biagioni, Daniel Masny, and Daniele Venturi. Naor-yung paradigm with shared randomness and applications. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 62–80. Springer, Heidelberg, August / September 2016.

BN00. Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, Heidelberg, August 2000.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CJSS21. Peter Chvojka, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Versatile and sustainable timed-release encryption and sequential time-lock puzzles. ESORICS 2021, 2021. https://eprint.iacr.org/2020/739.

CPP16. Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 308–338. Springer, Heidelberg, August 2016.

EFKP20. Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Non-malleable time-lock puzzles and applications. Cryptology ePrint Archive, Report 2020/779, 2020. https://eprint.iacr.org/2020/779.

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

KL14. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman and Hall/CRC Press, 2014.

KLX20. Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 390–413. Springer, Heidelberg, November 2020.

LJKW18. Jia Liu, Tibor Jager, Saqib A Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Designs, Codes and Cryptography*, 86(11):2549–2586, 2018.

LNPY21. Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. One-shot Fiat-Shamir-based NIZK arguments of composite residuosity in the standard model. 2021.

MT19. Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 620–649. Springer, Heidelberg, August 2019.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

Pie19. Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, January 2019.

RSW96. Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, 1996.

TCLM21. Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. Efficient CCA timed commitments in class groups. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2663–2684. ACM Press, November 2021.

Wes19. Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

A Proof of Lemma 2

Lemma 2. *Let p, q be primes, $N = pq$, $\ell \in \mathbb{N}$ such that $\gcd(\ell, \varphi(N)) = \ell$ and X and Y be random variables defined on domain $[\lfloor N/\ell \rfloor]$ as follows:*

$$\Pr[X = r] = 1/\lfloor N/\ell \rfloor \ \forall r \in [\lfloor N/\ell \rfloor] \text{ and } \Pr[Y = r] = \begin{cases} \ell/\varphi(N) & \forall r \in [\varphi(N)/\ell] \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\mathbb{SD}(X, Y) \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Proof. The following computation proves the lemma:

$$\begin{aligned}
\mathbb{SD}(X,Y) &= \frac{1}{2} \sum_{r \in [\lfloor N/\ell \rfloor]} |\Pr[X = r] - \Pr[Y = r]| = \\
\frac{1}{2}\left(\sum_{r=1}^{\varphi(N)/\ell} |\Pr[X = r] - \Pr[Y = r]| + \sum_{r=\varphi(N)/\ell+1}^{\lfloor N/\ell \rfloor} |\Pr[X = r] - \Pr[Y = r]| \right) &= \\
\frac{1}{2}\left(\sum_{r=1}^{\varphi(N)/\ell} \left| \frac{1}{\lfloor N/\ell \rfloor} - \frac{\ell}{\varphi(N)} \right| + \sum_{r=\varphi(N)/\ell+1}^{\lfloor N/\ell \rfloor} \left| \frac{1}{\lfloor N/\ell \rfloor} - 0 \right| \right) &\leq \\
\frac{1}{2}\left(\sum_{r=1}^{\varphi(N)/\ell} \left| \frac{\ell}{N} - \frac{\ell}{\varphi(N)} \right| + \sum_{r=\varphi(N)/\ell+1}^{\lfloor N/\ell \rfloor} \left| \frac{1}{\lfloor N/\ell \rfloor} - 0 \right| \right) &= \\
\frac{1}{2}\left(\varphi(N)/\ell \left| \frac{\ell(\varphi(N) - N)}{\varphi(N)N} \right| + (\lfloor N/\ell \rfloor - \varphi(N)/\ell) \frac{1}{\lfloor N/\ell \rfloor} \right) &= \\
\frac{1}{2}\left(\frac{(N - \varphi(N))}{N} + 1 - \frac{\varphi(N)/\ell}{\lfloor N/\ell \rfloor} \right) \leq \frac{1}{2}\left(\frac{(N - \varphi(N))}{N} + 1 - \frac{\varphi(N)/\ell}{N/\ell} \right) &= \\
\frac{1}{2}\frac{2(N - \varphi(N))}{N} = \frac{(N - (N - p - q + 1))}{N} = \frac{p + q - 1}{N} = \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.
\end{aligned}$$

B Proof of Theorem 6

Completeness is implied by the completeness of the NIZK and can be verified by inspection.

To prove security we define a sequence of games $\mathsf{G}_0 - \mathsf{G}_8$. For $i \in \{0, 1, \ldots, 8\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$ outputs b' in the game G_i such that $b = b'$.

Game 0. Game G_0 corresponds to the original security experiment where decommitment queries are answered using FDec.

Game 1. In game G_1 decommitment queries are answered using the algorithm Dec defined in Figure 10 with $i := 2, \mathsf{sk} := t$ which means that secret key t and ciphertext c_2 are used, to answer decommitment queries efficiently.

$\underline{\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, i, \mathsf{sk})}$
Parse c as (c_0, c_1, c_2)
if $\mathtt{NIZK}.\mathsf{Vrfy}(\mathsf{crs}_{\mathtt{NIZK}}, (c_0, c_1, c_2), \pi_{\mathsf{Com}}) = 1$
    Compute $y := c_0^{\mathsf{sk}} \bmod N$
    return $c_i \cdot y^{-1} \bmod N$
return $\perp$

Fig. 10. Decommitment oracle

Lemma 18.

$$\Pr[\mathsf{G}_0 = 1] = \Pr[\mathsf{G}_1 = 1].$$

Notice that both DEC and Dec answer decommitment queries in the exactly same way, hence the change is only syntactical.

Game 2. Game G_2 proceeds exactly as the previous game but we run the zero-knowledge simulator $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda, L)$ in PGen and produce a simulated proof for the challenge commitment as $\pi^* \leftarrow \mathsf{Sim}_2(\mathsf{crs}, \tau, (c_0^*, c_1^*, c_2^*))$. By the zero-knowledge security of the NIZK we directly obtain

Lemma 19.

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \mathbf{ZK}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against the zero-knowledge security of NIZK as follows: $\mathcal{B}_\lambda(\mathsf{crs}_{\mathtt{NIZK}}, \tau_L)$:

1. Set $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, \mathsf{crs}_{\mathtt{NIZK}})$, run $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$, and answer decommitment queries using t which is included in $\tau_L = (k_1, t)$.
2. Sample $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/4 \rfloor]$ and compute $c_0^* := g^r, c_1^* := h_1^r m_b, c_2^* := h_2^r m_b \bmod N$. Then submit $(s := (h_1, h_2, c_0^*, c_1^*, c_2^*), w := (m, r))$ to the oracle to obtain proof π^*.
3. Run $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$, answering decommitment queries using t.
4. Return the truth value of $b = b'$.

If the proof π^* is generated using $\mathtt{NIZK}.\mathsf{Prove}$, then $\mathcal{B}$ simulates G_1 perfectly. Otherwise, π^* is generated using Sim_1 and $\mathcal{B}$ simulates G_2 perfectly. This proves the lemma.

Game 3. In G_3 we sample k_1 uniformly at random from $[\varphi(N)/4]$.

Lemma 20.

$$|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

This lemma directly follows from Lemma 2 with $\ell := 4$.

Game 4. In G_4 we sample $y_1 \xleftarrow{\$} \mathbb{QR}_N$ and compute c_1^* as $y_1 m_b$.

Lemma 21.

$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against DDH in the group $\mathbb{QR}_N$.

$\mathcal{B}_\lambda(N, p, q, g, g^\alpha, g^\beta, g^\gamma)$:

1. Computes $\varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/4, h_2 := g^t \bmod N$ and sets $\mathsf{crs} := (N, T, g, h_1 := g^\alpha, h_2)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using t.
3. Samples $b \xleftarrow{\$} \{0,1\}$ and computes $(c_0^*, c_1^*, c_2^*) := (g^\beta, g^\gamma \cdot m_b, (g^\beta)^t \cdot m_b)$. Runs $\pi^* \leftarrow \mathsf{Sim}(1, \mathsf{st}', (c_0^*, c_1^*, c_2^*))$.
4. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using t.
5. Returns the truth value of $b = b'$. We remark that at this point c_1^* does not reveal any information about m_b.

If $\gamma = \alpha\beta$ then $\mathcal{B}$ simulates G_3 perfectly. Otherwise g^γ is uniform random element in $\mathbb{QR}_N$ and $\mathcal{B}$ simulates G_4 perfectly. This proofs the lemma. We remark that at this point c_1^* does not reveal any information about m_b.

Game 5. In G_5 we sample k_1 uniformly at random from $[\lfloor N/4 \rfloor]$.

Lemma 22.

$$|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

This lemma directly follows from Lemma 2 with $\ell := 4$.

Game 6. In G_6 we answer decommitment queries using Dec (Figure 10) with $i := 1, \mathsf{sk} := k_1$ which means that secret key k_1 and ciphertext c_1 are used.

Lemma 23.

$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq \mathbf{SimSnd}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

Let E denote the event that adversary $\mathcal{A}$ asks a decommitment query (c, π_{Com}) such that its decommitment using the key k_1 is different from its decommitment using the key t. Since G_5 and G_6 are identical until E does not happen, by the standard argument it is sufficient to upper bound the probability of happening E. Concretely,

$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ that breaks one-time simulation soundness of the NIZK and it is given as input $\mathsf{crs}_{\mathtt{NIZK}}$ together with a membership testing trapdoor $\tau_L := (k_1, t)$ where $t := 2^T \bmod \varphi(N)/4$.

The adversary $\mathcal{B}_\lambda^{\mathsf{Sim}_2}(\mathsf{crs}_{\mathtt{NIZK}}, \tau_L)$:

1. Computes $h_1 := g^{k_1} \bmod N, h_2 := g^t \bmod N$ using the membership testing trapdoor τ_L and sets $\mathsf{crs} := (N, T, g, h_1, h_2, \mathsf{crs}_{\mathtt{NIZK}})$.

2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Samples $b \xleftarrow{\$} \{0,1\}, x, y_1 \xleftarrow{\$} \mathbb{QR}_N$ and computes $(c_0^*, c_1^*, c_2^*) := (x, y_1 m_b, x^t m_b)$. Forwards (c_0^*, c_1^*, c_2^*) to simulation oracle Sim_2 and obtains a proof π^*.
4. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
5. Find a decommitment query (c, π_{Com}) such that $\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 1, k_1) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 2, t)$ and returns (c, π_{Com}).

$\mathcal{B}$ simulates G_6 perfectly and if the event E happens, it outputs a valid proof for a statement which is not in the specified language L. Therefore

$$\Pr[\mathsf{E}] \leq \mathbf{SimSnd}_{\mathcal{B}}^{\mathtt{NIZK}},$$

which concludes the proof of the lemma.

Game 7. In G_7 we sample r uniformly at random from $[\varphi(N)/4]$.

Lemma 24.

$$|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_7 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Since the only difference between the two games is in the set from which we sample r, to upper bound the advantage of adversary we can use Lemma 2 with $\ell := 4$, which directly yields required upper bound.

Game 8. In G_8 we sample $y_2 \xleftarrow{\$} \mathbb{QR}_N$ and compute c_2^* as $y_2 m_b$.

Let $\tilde{T}_{\mathsf{SSS}}(\lambda)$ be the polynomial whose existence is guaranteed by the SSS assumption. Let $\mathsf{poly}_{\mathcal{B}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Steps 1–2 and answer decommitment queries in Step 3 of the adversary $\mathcal{B}_{2,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathcal{B}}(\lambda))^{1/\epsilon}$. Set $\tilde{T}_{\mathtt{NITC}} := \max(\tilde{T}_{\mathsf{SSS}}, \underline{T})$.

Lemma 25. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$, where depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ for some $T(\cdot) \geq \underline{T}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_8 = 1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{SSS}}.$$

The adversary $\mathcal{B}_{1,\lambda}(N, T(\lambda), g)$:

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{2^{T(\lambda)}} \bmod N$, runs $(\mathsf{crs}_{\mathtt{NIZK}}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, \mathsf{crs}_{\mathtt{NIZK}})$. Notice that value h_2 is computed by repeated squaring.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Outputs $(N, g, k_1, h_1, h_2, \mathsf{crs}_{\mathtt{NIZK}}, \tau, m_0, m_1, \mathsf{st})$

The adversary $\mathcal{B}_{2,\lambda}(x, y, (N, g, k_1, h_1, h_2, \mathsf{crs}_{\mathtt{NIZK}}, \tau, m_0, m_1, \mathsf{st}))$:

1. Samples $b \xleftarrow{\$} \{0,1\}, y_1 \xleftarrow{\$} \mathbb{QR}_N$, computes $c_0^* := x, c_1^* := y_1 m_b, c_2^* := y m_b$.

2. Runs $\pi^* \leftarrow \mathsf{Sim}(\mathsf{crs}_{\mathtt{NIZK}}, \tau, (c_0^*, c_1^*, c_2^*))$.
3. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*), \mathsf{st})$ and answers decommitment queries using k_1.
4. Returns the truth value of $b = b'$.

Since g is a generator of $\mathbb{QR}_N$ and x is sampled uniformly at random from $\mathbb{QR}_N$ there exists some $r \in [\varphi(N)/4]$ such that $x = g^r$. Therefore when $y = x^{2^T} = (g^{2^T})^r \bmod N$, then $\mathcal{B}$ simulates G_7 perfectly. Otherwise y is random value and $\mathcal{B}$ simulates G_8 perfectly. We remark that at this point c_2^* does not reveal any information about m_b.

Now we analyse the running time of the constructed adversary. Adversary $\mathcal{B}_1$ computes h_2 by $T(\lambda)$ consecutive squarings and because $T(\lambda)$ is polynomial in λ, $\mathcal{B}_1$ is efficient. Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathcal{B}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}(\lambda) \leq 2T^{\epsilon}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{NITC}}(\cdot) \geq \tilde{T}_{\mathsf{SSS}}(\cdot)$ as required.

Lemma 26.

$$\Pr[\mathsf{G}_8 = 1] = \frac{1}{2}.$$

Clearly, c_0^* is uniform random element in $\mathbb{QR}_N$ and hence it does not contain any information about the challenge message. Since y_1, y_2 are sampled uniformly at random from $\mathbb{QR}_N$ the ciphertexts c_1^*, c_2^* are also uniform random elements in $\mathbb{QR}_N$ and hence do not contain any information about the challenge message m_b. Therefore, an adversary can not do better than guessing.

By combining Lemmas 18 - 26 we obtain the following:

$$\begin{aligned}\mathbf{Adv}^{\mathtt{NITC}}_{\mathcal{A}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \leq \sum_{i=0}^{7} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]| + \left|\Pr[\mathsf{G}_8 - \frac{1}{2}\right| \\ &\leq \mathbf{ZK}^{\mathtt{NIZK}}_{\mathcal{B}} + \mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}} + \mathbf{SimSnd}^{\mathtt{NIZK}}_{\mathcal{B}} + \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}} + 3\left(\frac{1}{p} + \frac{1}{q} - \frac{1}{N}\right),\end{aligned}$$

which concludes the proof.

C Proof of Theorem 12

Completeness is implied by the completeness of the NIZK and can be verified by inspection.

To prove security we define a sequence of games G_0–G_{13}. For $i \in \{0, 1, \ldots, 13\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$ outputs b' in the game G_i such that $b = b'$.

Game 0. Game G_0 corresponds to the original security experiment where decommitment queries are answered using FDec.

$\underline{\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, i)}$
Parse c as (c_0, c_1, c_2, c_3)
if $\mathtt{NIZK}.\mathsf{Vrfy}((h_1, h_2, c_0, c_1, c_2, c_3), \pi_{\mathsf{Com}}) = 1$
    Compute $y := c_0^{k_i} \bmod N$
    return $\frac{c_i \cdot y^{-N} (\bmod N^2) - 1}{N}$
return $\perp$

Fig. 11. Decommitment oracle

Game 1. In game G_1 decommitment queries are answered using the algorithm Dec defined in Figure 11 with $i := 1$, meaning that secret key k_1 and ciphertext c_1 are used, to answer decommitment queries efficiently.

Lemma 27.

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

Notice that if c_1 and c_3 contain the same message, both oracles answer decommitment queries consistently. Let E denote the event that the adversary $\mathcal{A}$ asks a decommitment query (c, π_{Com}) such that its decommitment using the key k_1 is different from its decommitment using FDec. Since G_0 and G_1 are identical until E happens, we bound the probability of E. Concretely, we have

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ that breaks soundness with respect to auxiliary input $\mathsf{aux} := (p, q)$ of the NIZK. The adversary $\mathcal{B}_\lambda(p, q)$ proceeds as follows:

1. Samples $k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N$, $\varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/2$ and sets $\mathsf{crs} := (N, T, g, h_1, h_2, h_3)$ where h_3 is given by L.
2. Then it runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. It samples $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/2 \rfloor]$ and computes $c_0^* := g^r, c_1^* := h_1^{rN}(1+N)^{m_b}, c_2^* := h_2^{rN}(1+N)^{m_b}, c_3^* := h_3^{rN}(1+N)^{m_b}$. It sets $(s := (h_1, h_2, c_0^*, c_1^*, c_2^*, c_3^*), w := (m, r))$ and runs $\pi^* \leftarrow \mathtt{NIZK}.\mathsf{Prove}(s, w)$.
4. Next, it runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
5. Finally, it checks whether there exists a decommitment query (c, π_{Com}) such that $\mathsf{DEC}(c, \pi_{\mathsf{Com}}, 1) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 1)$. If E occurs, then this is the case, and it returns $((h_1, h_2, c_0, c_1, c_2, c_3), \pi_{\mathsf{Com}})$. Notice that finding such a query can be done efficiently with the knowledge of t since instead of running FDec it is possible to verify the proof and simply compute $\frac{c_3 \cdot (c_0^t)^{-N} (\bmod N^2) - 1}{N}$ which produce the same result as FDec.

$\mathcal{B}$ simulates G_1 perfectly and if the event E happens, then it outputs a valid proof for a statement which is not in the specified language L. Therefore we get

$$\Pr[\mathsf{E}] \leq \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

Game 2. Game G_2 proceeds exactly as the previous game but we use the zero-knowledge simulator $(\pi^*, \mathsf{st}) \leftarrow \mathsf{Sim}(1, \mathsf{st}, (h_1, h_2, c_0^*, c_1^*, c_2^*, c_3^*))$ to produce a simulated proof for the challenge commitment and $\mathsf{Sim}(2, \mathsf{st}, \cdot)$ to answer random oracle queries. By zero-knowledge security of underlying NIZK we directly obtain

Lemma 28.

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \mathbf{ZK}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against zero-knowledge security of NIZK as follows:

1. Samples $k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, h_3)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Samples $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/2 \rfloor]$ and computes $c_0^* := g^r, c_1^* := h_1^{rN}(1+N)^{m_b}, c_2^* := h_2^{rN}(1+N)^{m_b}, c_3^* := h_3^{rN}(1+N)^{m_b}$. It submits $(s := (h_1, h_2, c_0^*, c_1^*, c_2^*, c_3^*), w := (m, r))$ to its oracle and obtains proof π^* as answer.
4. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
5. Returns the truth value of $b = b'$.

If the proof π^* is generated using $\mathtt{NIZK.Prove}$, then $\mathcal{B}$ simulates G_1 perfectly. Otherwise π^* is generated using Sim_1 and $\mathcal{B}$ simulates G_2 perfectly. This proofs the lemma.

Game 3. In G_3 we sample r uniformly at random from $[\varphi(N)/2]$.

Lemma 29.

$$|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Since the only difference between the two games is in the set from which we sample r, to upper bound the advantage of adversary we can use Lemma 2 with $\ell := 2$, which directly yields required upper bound.

Game 4. In G_4 we sample $y_3 \xleftarrow{\$} \mathbb{J}_N$ and compute c_3^* as $y_3^N(1+N)^{m_b}$.

Let $\tilde{T}_{\mathsf{SSS}}(\lambda)$ be the polynomial whose existence is guaranteed by the SSS assumption. Let $\mathsf{poly}_{\mathcal{B}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Steps 1–2 and answer decommitment queries in Step 3 of the adversary $\mathcal{B}_{2,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathcal{B}}(\lambda))^{1/\epsilon}$. Set $\tilde{T}_{\mathtt{NITC}} := \max(\tilde{T}_{\mathsf{SSS}}, \underline{T})$.

Lemma 30. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$, where depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ for some $T(\cdot) \geq \underline{T}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{SSS}}.$$

The adversary $\mathcal{B}_{1,\lambda}(N, T(\lambda), g)$:

1. Samples $k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N$, $h_3 := g^{2^{T(\lambda)}} \bmod N$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2, h_3)$. Notice that value h_3 is computed by repeated squaring.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Outputs $(N, g, k_1, k_2, h_1, h_2, h_3, m_0, m_1, \mathsf{st})$

The adversary $\mathcal{B}_{2,\lambda}(x, y, (N, g, k_1, k_2, h_1, h_2, h_3, m_0, m_1, \mathsf{st}))$:

1. Samples $b \xleftarrow{\$} \{0,1\}$, computes $c_0^* := x, c_1^* := x^{k_1 N}(1+N)^{m_b}, c_2^* := x^{k_2 N}(1+N)^{m_b}, c_3^* := y^N(1+N)^{m_b}$.
2. Runs $\pi^* \leftarrow \mathsf{Sim}(1, \mathsf{st}', (h_1, h_2, c_0^*, c_1^*, c_2^*, c_3^*))$.
3. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
4. Returns the truth value of $b = b'$.

Since g is a generator of $\mathbb{J}_N$ and x is sampled uniformly at random from $\mathbb{J}_N$ there exists some $r \in [\varphi(N)/2]$ such that $x = g^r$. Therefore when $y = x^{2^T} = (g^{2^T})^r \bmod N$, then $\mathcal{B}$ simulates G_3 perfectly. Otherwise y is random value and $\mathcal{B}$ simulates G_4 perfectly.

Now we analyse the running time of the constructed adversary. Adversary $\mathcal{B}_1$ computes h_3 by $T(\lambda)$ consecutive squarings and because $T(\lambda)$ is polynomial in λ, $\mathcal{B}_1$ is efficient. Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathcal{B}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}(\lambda) \leq 2T^{\epsilon}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{NITC}}(\cdot) \geq \tilde{T}_{\mathsf{SSS}}(\cdot)$ as required.

Game 5. In G_5 we sample $y_3 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and compute c_3^* as $y_3(1+N)^{m_b}$.

Lemma 31.

$$|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq \mathbf{Adv}^{\mathsf{DCR}}_{\mathcal{B}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against DCR.

$\mathcal{B}_\lambda(N, y)$:

1. Samples $g, y_3, x \xleftarrow{\$} \mathbb{J}_N, k_1, k_2 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{k_2} \bmod N, h_3 := g^{2^T} \bmod N$ and sets $\mathsf{crs} := (N, T, g, h_1, h_2, h_3)$. Notice that value h_3 is computed by repeated squaring.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Samples $b \xleftarrow{\$} \{0,1\}, w \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that $\left(\frac{y}{N}\right) = \left(\frac{w}{N}\right)$. We remark that computing Jacobi symbol can be done efficiently without knowing factorization of N.
4. Computes $c_0^* := x, , c_1^* := x^{k_1 N}(1+N)^{m_b}, c_2^* := x^{k_2 N}(1+N)^{m_b}, c_3^* := yw^N(1+N)^{m_b}$. Runs $\pi^* \leftarrow \mathsf{Sim}(1, \mathsf{st}', (h_1, h_2, c_0^*, c_1^*, c_2^*, c_3^*))$.
5. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
6. Returns the truth value of $b = b'$.

If $y = v^N \bmod N^2$ then $yw^N = v^N w^N = (vw)^N$ and hence yw^N is N-th residue. Moreover, the Jacobi symbol of yw is 1, since the Jacobi symbol is multiplicatively homomorphic. Therefore $\mathcal{B}$ simulates G_4 perfectly.

Otherwise, if y is uniform random element in $\mathbb{Z}^*_{N^2}$, then yw^N is also uniform among all elements of $\mathbb{Z}^*_{N^2}$ that have Jacobi symbol 1 and $\mathcal{B}$ simulates G_5 perfectly. This proves the lemma.

We remark that at this point c^*_3 does not reveal any information about b. Here we use that if $x = y \bmod N$ then $\left(\frac{x}{N}\right) = \left(\frac{y}{N}\right)$ and that there is an isomorphism $f : \mathbb{Z}^*_N \times \mathbb{Z}_N \to \mathbb{Z}^*_{N^2}$ given by $f(u, v) = u^N(1+N)^v = u^N(1+vN) \bmod N^2$ (see e.g. [KL14, Proposition 13.6]). Since $f(u, v) \bmod N = u^N + u^N vN \bmod N = u^N \bmod N$, that means that Jacobi symbol $\left(\frac{f(u,v)}{N}\right)$ depends only on u. Hence if $\left(\frac{f(u,v)}{N}\right) = 1$ then it must hold that $\left(\frac{f(u,r)}{N}\right) = 1$ for all $r \in \mathbb{Z}_N$. This implies that a random element $f(u, v)$ in $\mathbb{Z}^*_{N^2}$ with $\left(\frac{f(u,v)}{N}\right) = 1$ has a uniformly random distribution of v in $\mathbb{Z}_N$. Therefore if $yw^N = u^N(1+N)^v \bmod N^2$ then $yw^N(1+N)^{m_b} = u^N(1+N)^{m_b+v} \bmod N^2$. Since v is uniform in $\mathbb{Z}_N$, $(m_b + v)$ is also uniform in $\mathbb{Z}_N$, which means that ciphertext c^*_3 does not reveal any information about b.

Game 6. In G_6 we sample k_2 uniformly at random from $[\varphi(N)/2]$.

Lemma 32.

$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Again using a statistical argument this lemma directly follows from Lemma 2 with $\ell := 2$.

Game 7. In G_7 we sample $y_2 \xleftarrow{\$} \mathbb{J}_N$ and compute c^*_2 as $y_2^N(1+N)^{m_b}$.

Lemma 33.

$$|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_7 = 1]| \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ against DDH in the group $\mathbb{J}_N$.

$\mathcal{B}_\lambda(N, g, g^\alpha, g^\beta, g^\gamma)$:

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/2 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_3 := g^{2^T} \bmod N$ and sets $\mathsf{crs} := (N, T, g, h_1, h_2 := g^\alpha, h_3)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Samples $b \xleftarrow{\$} \{0, 1\}, y_3 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and computes $(c^*_0, c^*_1, c^*_2, c^*_3) := (g^\beta, (g^\beta)^{k_1 N}(1+N)^{m_b}, (g^\gamma)^N(1+N)^{m_b}, y_3(1+N)^{m_b})$. Runs $\pi^* \leftarrow \mathsf{Sim}(1, \mathsf{st}', (h_1, h_2, c^*_0, c^*_1, c^*_2, c^*_3))$.
4. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c^*_0, c^*_1, c^*_2, c^*_3), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
5. Returns the truth value of $b = b'$.

If $\gamma = \alpha\beta$, then $\mathcal{B}$ simulates G_6 perfectly. Otherwise g^γ is uniform random element in $\mathbb{J}_N$ and $\mathcal{B}$ simulates G_7 perfectly. This proofs the lemma.

Game 8. In G_8 we sample k_2 uniformly at random from $[\lfloor N/2 \rfloor]$.

Lemma 34.

$$|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_8 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

This lemma directly follows from Lemma 2.

Game 9. In G_9 we sample $y_2 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ such that it has Jacobi symbol 1 and compute c_2^* as $y_2(1+N)^{m_b}$.

Lemma 35.

$$|\Pr[\mathsf{G}_8 = 1] - \Pr[\mathsf{G}_9 = 1]| \leq \mathbf{Adv}^{\mathsf{DCR}}_{\mathcal{B}}.$$

This can be proven in similar way as Lemma 31. We remark that at this point c_2^* does not reveal any information about m_b.

Game 10. In G_{10} we answer decommitment queries using Dec (Figure 11) with $i := 2$ which means that secret key k_2 and ciphertext c_2 are used.

Lemma 36.

$$|\Pr[\mathsf{G}_9 = 1] - \Pr[\mathsf{G}_{10} = 1]| \leq \mathbf{SimSnd}^{\mathtt{NIZK}}_{\mathcal{B}}.$$

Let E denote the event that adversary $\mathcal{A}$ asks a decommitment query (c, π_{Com}) such that its decommitment using the key k_1 is different from its decommitment using the key k_2. Since G_9 and G_{10} are identical until E does not happen, by the standard argument it is sufficient to upper bound the probability of happening E. Concretely,

$$|\Pr[\mathsf{G}_9 = 1] - \Pr[\mathsf{G}_{10} = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ which breaks one-time simulation soundness of the NIZK.

The adversary $\mathcal{B}^{\mathsf{Sim}_1, \mathsf{Sim}_2}_{\lambda}$:

1. Computes $\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, T)$ as defined in the construction where the value h_3 is computed using repeated squaring instead.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_2.
3. Samples $b \xleftarrow{\$} \{0,1\}, x \xleftarrow{\$} \mathbb{J}_N, y_2, y_3 \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ and computes $(c_0^*, c_1^*, c_2^*, c_3^*) := (x, x^{k_1 N}(1+N)^{m_b}, y_2(1+N)^{m_b}, y_3(1+N)^{m_b})$. Forwards $(h_1, h_2, c_0^*, c_1^*, c_2^*, c_3^*)$ to simulation oracle Sim_1 and obtains a proof π^*.
4. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_2.
5. Find a decommitment query (c, π_{Com}) such that $\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 1) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 2)$ and returns $((h_1, h_2, c_0, c_1, c_2, c_3), \pi_{\mathsf{Com}})$.

$\mathcal{B}$ simulates G_{10} perfectly and if the event E happens, it outputs a valid proof for a statement which is not in the specified language L. Therefore

$$\Pr[\mathsf{E}] \leq \mathbf{SimSnd}^{\mathtt{NIZK}}_{\mathcal{B}},$$

which concludes the proof of the lemma.

Game 11. In G_{11} we sample k_1 uniformly at random from $[\varphi(N)/2]$.

Lemma 37.

$$|\Pr[\mathsf{G}_{10}=1]-\Pr[\mathsf{G}_{11}=1]| \leq \frac{1}{p}+\frac{1}{q}-\frac{1}{N}.$$

This lemma directly follows from Lemma 2 with $\ell := 2$.

Game 12. In G_{12} we sample $y_1 \xleftarrow{\$} \mathbb{J}_N$ and compute c_1^* as $y_1^N(1+N)^{m_b}$.

Lemma 38.

$$|\Pr[\mathsf{G}_{11}=1]-\Pr[\mathsf{G}_{12}=1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}.$$

This can be proven in similar way as Lemma 33.

Game 13. In G_{13} we sample $y_1 \xleftarrow{\$} \mathbb{Z}_{N^2}^*$ such that it has Jacobi symbol 1 and compute c_1^* as $y_1(1+N)^{m_b}$.

Lemma 39.

$$|\Pr[\mathsf{G}_{12}=1]-\Pr[\mathsf{G}_{13}=1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DCR}}.$$

This can be proven in similar way as Lemma 31. We remark that at this point c_1^* does not reveal any information about m_b.

Lemma 40.

$$\Pr[\mathsf{G}_{13}=1] = \frac{1}{2}.$$

Clearly, c_0^* is uniform random element in $\mathbb{J}_N$ and hence it does not contain any information about the challenge message. Since y_1, y_2, y_3 are sampled uniformly at random from $\mathbb{Z}_{N^2}^*$ the ciphertexts c_1^*, c_2^*, c_3^* are also uniform random elements in $\mathbb{Z}_{N^2}^*$ and hence do not contain any information about the challenge message m_b. Therefore, an adversary can not do better than guessing.

By combining Lemmas 27 - 40 we obtain the following:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathtt{NITC}} &= \left|\Pr[\mathsf{G}_0=1]-\frac{1}{2}\right| \leq \sum_{i=0}^{12}|\Pr[\mathsf{G}_i=1]-\Pr[\mathsf{G}_{i+1}=1]| + \left|\Pr[\mathsf{G}_{13}-\frac{1}{2}\right| \\ &\leq \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}} + \mathbf{ZK}_{\mathcal{B}}^{\mathtt{NIZK}} + \mathbf{Adv}_{\mathcal{B}}^{\mathsf{SSS}} + \mathbf{SimSnd}_{\mathcal{B}}^{\mathtt{NIZK}} + 2\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}} + 3\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DCR}} \\ &\quad + 4\left(\frac{1}{p}+\frac{1}{q}-\frac{1}{N}\right).\end{aligned}$$

which concludes the proof.

D Proof of Theorem 16

Completeness is implied by the completeness of the NIZK and can be verified by inspection.

To prove security we define a sequence of games $\mathsf{G}_0 - \mathsf{G}_8$. For $i \in \{0,1,\ldots,8\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ outputs b' in the game G_i such that $b = b'$.

In the following we assume that the underlying sigma protocol produces transcripts of the form (a, v, z), where any of these values can be vectors.

$\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, i, \mathsf{sk})$
Parse c as (c_0, c_1, c_2)
if $\mathtt{NIZK}.\mathsf{Vrfy}((h_1, h_2, c_0, c_1, c_2), \pi_{\mathsf{Com}}) = 1$
    Compute $y := c_0^{\mathsf{sk}} \bmod N$
    return $c_i \cdot y^{-1} \bmod N$
return $\perp$

Fig. 12. Decommitment oracle

Game 0. Game G_0 corresponds to the original security experiment where decommitment queries are answered using FDec.

Game 1. In game G_1 decommitment queries are answered using the algorithm Dec defined in Figure 12 with $i := 1, \mathsf{sk} := k_1$ which means that secret key k_1 and ciphertext c_1 are used, to answer decommitment queries efficiently.

Lemma 41.

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq Q \cdot \mathbf{Snd}_{\mathcal{B}}^{\Sigma}.$$

Notice that if c_1 and c_2 contain the same message, both oracles answer decommitment queries consistently. Let E denote the event that the adversary $\mathcal{A}$ asks a decommitment query $(c := (c_0, c_1, c_2), \pi_{\mathsf{Com}} := (a, z))$ such that its decommitment using the key k_1 is different from its decommitment using FDec. Since G_0 and G_1 are identical until E happens, we bound the probability of E. Concretely, we have

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ that breaks soundness with respect to auxiliary input $\mathsf{aux} := (p, q)$ of the Sigma protocol. W.l.o.g. we assume that whenever event E happens, $\mathcal{A}$ previously asked the random oracle H on input $(h_1, h2, c_0, c_1, c_2, a)$. The argument for this is that it is straightforward to transform any adversary that violates this condition into an adversary that makes one additional query to H and wins with the same probability. Let $\mathcal{A}$ makes at most Q random oracle queries. The adversary $\mathcal{B}_\lambda(p, q)$ proceeds as follows:

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, \varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/4, h_2 := g^t \bmod N$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2)$.
2. Then it runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Samples $i \xleftarrow{\$} [Q]$ and answers random oracle queries $(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j)$ in the following way:
   - If $i = j$ it runs the protocol with the honest verifier $\mathsf{Verifier}$ for statement $(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j)$ using as a commitment value a^j. It obtains challenge v from $\mathsf{Verifier}$ and it programs the oracle $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j) := v$ if . Answer the query with v.

– Otherwise, it returns $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j)$ if it is set. If this is not the case samples v_j, sets $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j) := v_j$ and returns v_j.

4. It samples $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/4 \rfloor]$ and computes $c_0^* := g^r, c_1^* := h_1^r m_b, c_2^* := h_2^r m_b$. It sets $(s := (h_1, h_2, c_0^*, c_1^*, c_2^*), w := (m, r))$ and produces honest proof π^*.
5. Next, it runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1.
6. Finally, it checks whether there exists a decommitment query $(c := (c_0, c_1, c_2), \pi_{\mathsf{Com}} = (a, z))$ such that $\mathsf{DEC}(\mathsf{crs}, c, \pi_{\mathsf{Com}}) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 1, k_1)$. This check can be done efficiently with the knowledge of t. If E occurs and $\mathcal{B}_\lambda$ has guessed index i correctly, then value z allows $\mathcal{B}_\lambda$ succeed in the attack game.

Suppose that the query $(h_1, h_2, c_0, c_1, c_2, a)$ has been asked to the random oracle as i^*-th query and that $i = i^*$. Then it holds

$$\mathbf{Snd}_{\mathcal{B}}^{\Sigma} = \Pr[\mathsf{E} \wedge i = i^*] = \Pr[\mathsf{E}] \Pr[i = i^*] = \frac{1}{Q} \Pr[\mathsf{E}],$$

where the first equality holds since the events are independent.

Game 2. Game G_2 proceeds exactly as the previous game but we use the HVZK simulator Sim to produce a simulated proof for the challenge commitment $(h_1, h_2, c_0^*, c_1^*, c_2^*)$ and upon receiving simulated transcript (a^*, v^*, z^*) from the simulator we try to program the random oracle in the following way: if $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = \perp$ then $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) := v^*$ and setting $\pi^* = (a^*, z^*)$. Now notice that, since the transcripts have the exactly same distributions, the games proceeds exactly the same until our programming of random oracle is successful. Let denote by E the event that we are not able to set correct value for H. E happens only in the case that adversary already asked a random oracle query at the point $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*)$. Let Q is the number of random oracle queries asked by $\mathcal{A}$. Since the Sigma protocol has δ-unpredictable commitments the probability of this event is by union bound is less than $Q\delta$. Hence we obtain

Lemma 42.

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq Q\delta.$$

Game 3. In G_3 we sample r uniformly at random from $[\varphi(N)/4]$.

Lemma 43.

$$|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

Since the only difference between the two games is in the set from which we sample r, to upper bound the advantage of adversary we can use Lemma 2 with $\ell := 4$, which directly yields required upper bound.

Game 4. In G_4 we sample $y_2 \xleftarrow{\$} \mathbb{QR}_N$ and compute c_2^* as $y_2 m_b$.

Let $\tilde{T}_{\mathsf{SSS}}(\lambda)$ be the polynomial whose existence is guaranteed by the SSS assumption. Let $\mathsf{poly}_{\mathcal{B}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Steps 1–2 and answer decommitment queries in Step 3 of the adversary $\mathcal{B}_{2,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathcal{B}}(\lambda))^{1/\underline{\epsilon}}$. Set $\tilde{T}_{\mathtt{NITC}} := \max(\tilde{T}_{\mathsf{SSS}}, \underline{T})$.

Lemma 44. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\underline{\epsilon}}(\lambda)$ for some $T(\cdot) \geq \underline{T}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}}.$$

The adversary $\mathcal{B}_{1,\lambda}(N, T(\lambda), g)$:

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{2^{T(\lambda)}} \bmod N$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2)$. Notice that value h_2 is computed by repeated squaring.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1. Random oracle queries are answered as before.
3. Outputs $(N, g, k_1, h_1, h_2, m_0, m_1, \mathsf{st})$

The adversary $\mathcal{B}_{2,\lambda}(x, y, (N, g, k_1, h_1, h_2, m_0, m_1, \mathsf{st}))$:

1. Samples $b \xleftarrow{\$} \{0, 1\}$, computes $c_0^* := x, c_1^* := x^{k_1} m_b, c_2^* := y m_b$.
2. Runs $(a^*, v^*, z^*) \leftarrow \mathsf{Sim}(h_1, h_2, c_0^*, c_1^*, c_2^*)$. If $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = \perp$ sets $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = v^*$. Sets $\pi^* = (a^*, z^*)$.
3. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1. Random oracle queries are answered as before.
4. Returns the truth value of $b = b'$.

Since g is a generator of $\mathbb{QR}_N$ and x is sampled uniformly at random from $\mathbb{QR}_N$ there exists some $r \in [\varphi(N)/2]$ such that $x = g^r$. Therefore when $y = x^{2^T} = (g^{2^T})^r \bmod N$, then $\mathcal{B}$ simulates G_3 perfectly. Otherwise y is random value and $\mathcal{B}$ simulates G_4 perfectly. We remark that at this point c_2^* does not reveal any information about m_b.

Now we analyse the running time of the constructed adversary. Adversary $\mathcal{B}_1$ computes h_3 by $T(\lambda)$ consecutive squarings and because $T(\lambda)$ is polynomial in λ, $\mathcal{B}_1$ is efficient. Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathcal{B}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T}^{\underline{\epsilon}}(\lambda) + T^{\underline{\epsilon}}(\lambda) \leq 2T^{\underline{\epsilon}}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{NITC}}(\cdot) \geq \tilde{T}_{\mathsf{SSS}}(\cdot)$ as required.

Game 5. Let $(c^*, \pi_c^* om = (a^*, z^*))$ is the challenge commitment. In G_5 we abort experiment if adversary asks any decommitment query $(c, \pi_{\mathsf{Com}} = (a, z))$ such that $c = c^*, a = a^*$ and $z \neq z^*$. We denote this event by E.

Lemma 45.

$$|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq \mathbf{OUR}^{\sigma}_{\mathcal{B}}.$$

Since G_4 and G_5 are identical until E does not happen, by the standard argument it is sufficient to upper bound the probability of happening E. Concretely,

$$|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ which breaks quasi unique responses property of the Sigma protocol. The adversary $\mathcal{B}_\lambda$ proceeds as follows:

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{2^T} \bmod N$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2)$. Notice that h_2 is computed by repeated squaring.
2. Then it runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1. Random oracle queries are answered as before.
3. Samples $b \xleftarrow{\$} \{0,1\}$, computes $c_0^* := x, c_1^* := x^{k_1} m_b, c_2^* := y m_b$.
4. Runs $(a^*, v^*, z^*) \leftarrow \mathsf{Sim}(h_1, h_2, c_0^*, c_1^*, c_2^*)$. If $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = \perp$ sets $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = v^*$. Sets $\pi^* = (a^*, z^*)$.
5. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1. Random oracle queries are answered as before.
6. Returns the truth value of $b = b'$.
7. Finally, it checks whether there is a decommitment query $(c, \pi = (a, z))$ such that $c = c^*, a = a^*$ and $z \neq z^*$. If E occurs, then this is the case, and it returns $((h_1, h_2, c_0^*, c_1^*, c_2^*), a^*, v^*, z^*, z)$ where $v^* = H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*)$.

Notice, that if E happens that $\mathcal{B}_\lambda$ indeed provides two transcripts which has different responses which yields

$$\Pr[\mathsf{E}] \leq \mathbf{OUR}_{\mathcal{B}}^{\Sigma}.$$

Game 6. In game G_6 decommitment queries are answered using the algorithm Dec defined in Figure 12 with $i := 2, \mathsf{sk} := t$ which means that secret key t and ciphertext c_2 are used, to answer decommitment queries efficiently.

Lemma 46.

$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq Q \cdot \mathbf{Snd}_{\mathcal{B}}^{\Sigma}.$$

Let E denote the event that adversary $\mathcal{A}$ asks a decommitment query $(c, \pi_{\mathsf{Com}} = (a_0, a_1, z))$ such that its decommitment using the key k_1 is different from its decommitment using the key t. Since G_5 and G_6 are identical until E does not happen, by the standard argument it is sufficient to upper bound the probability of happening E. Concretely,

$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq \Pr[\mathsf{E}].$$

We construct an adversary $\mathcal{B}$ that breaks soundness with respect to auxiliary input $\mathsf{aux} := (p, q)$ of the Sigma protocol. W.l.o.g. we assume that whenever event E happens, $\mathcal{A}$ previously asked the random oracle H on input $(h_1, h_2, c_0, c_1, c_2, a)$. The argument for this is that it is straightforward to transform any adversary that violates this condition into an adversary that makes one additional query to H and wins with the same probability. Let $\mathcal{A}$ makes at most Q random oracle queries. The adversary $\mathcal{B}_\lambda(p, q)$ proceeds as follows:

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, \varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/4, h_2 := g^t \bmod N$ and sets $\mathsf{crs} := (N, T(\lambda), g, h_1, h_2)$.
2. Then it runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using k_1.
3. Samples $i \xleftarrow{\$} [Q]$ and answers random oracle queries $(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j,)$ in the following way:
   - If $i = j$ it runs the protocol with the honest verifier $\mathsf{Verifier}$ for statement $(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j)$ using as a commitment value a^j. It obtains challenge v from $\mathsf{Verifier}$ and it programs the oracle $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j) := v$. Answer the query with v.
   - Otherwise, if $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j) \neq \perp$ returns $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j)$. Else it samples v_j, sets $H(h_1^j, h_2^j, c_0^j, c_1^j, c_2^j, a^j) := v_j$ and returns v_j.
4. Samples $b \xleftarrow{\$} \{0,1\}, x, y_2 \xleftarrow{\$} \mathbb{QR}_N$ and computes $(c_0^*, c_1^*, c_2^*) := (x, x^{k_1} m_b, y_2 m_b)$.
5. Runs $(a^*, v^*, z^*) \leftarrow \mathsf{Sim}(h_1, h_2, c_0^*, c_1^*, c_2^*)$. If $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = \perp$ sets $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = v^*$. Sets $\pi^* = (a^*, z^*)$.
6. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using k_1 and random oracle queries in the same way as before.
7. Finally, it checks whether there exists a decommitment query $(c, \pi_{\mathsf{Com}} = (a, z))$ such that $\mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 1, k_1) \neq \mathsf{Dec}(\mathsf{crs}, c, \pi_{\mathsf{Com}}, 2, t)$. If E occurs, then this is the case and it returns z as a response to the challenge v.

Suppose that the query $(h_1, h_2, c_0, c_1, c_2, a)$ has been asked to the random oracle as i^*-th query and that $i = i^*$. Then $\mathcal{B}_\lambda$ guesses a correct index for which $\mathcal{A}$ asks decommitment query for a statement which is not in the specified language L. Therefore we get

$$\mathbf{Snd}_{\mathcal{B}}^{\Sigma} = \Pr[\mathsf{E} \wedge i = i^*] = \Pr[\mathsf{E}] \Pr[i = i^*] = \frac{1}{Q} \Pr[\mathsf{E}],$$

where the first equality holds since the events are independent.

Game 7. In G_7 we sample k_1 uniformly at random from $[\varphi(N)/4]$.

Lemma 47.

$$|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_7 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

This lemma directly follows from Lemma 2 with $\ell := 4$.

Game 8. In G_8 we sample $y_1 \xleftarrow{\$} \mathbb{QR}_N$ and compute c_1^* as $y_1 m_b$.

Lemma 48.

$$|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_8 = 1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against DDH in the group $\mathbb{QR}_N$.

$\mathcal{B}_\lambda(N, p, q, g, g^\alpha, g^\beta, g^\gamma)$:

1. Computes $\varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/4, h_2 := g^t \bmod N$ and sets $\mathsf{crs} := (N, T, g, h_1 := g^\alpha, h_2)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{crs})$ and answers decommitment queries using t. Random oracle queries are answered as before.
3. Samples $b \xleftarrow{\$} \{0,1\}$ and computes $(c_0^*, c_1^*, c_2^*) := (g^\beta, g^\gamma \cdot m_b, (g^\beta)^t \cdot m_b)$.
4. Runs $(a^*, v^*, z^*) \leftarrow \mathsf{Sim}(h_1, h_2, c_0^*, c_1^*, c_2^*)$. If $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = \perp$ sets $H(h_1, h_2, c_0^*, c_1^*, c_2^*, a^*) = v^*$. Sets $\pi^* = (a^*, z^*)$.
5. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*), \pi^*, \mathsf{st})$ and answers decommitment queries using t. Random oracle queries are answered as before.
6. Returns the truth value of $b = b'$. We remark that at this point c_1^* does not reveal any information about m_b.

If $\gamma = \alpha\beta$ then $\mathcal{B}$ simulates G_7 perfectly. Otherwise g^γ is uniform random element in $\mathbb{QR}_N$ and $\mathcal{B}$ simulates G_8 perfectly. This proofs the lemma. We remark that at this point c_1^* does not reveal any information about m_b.

Lemma 49.

$$\Pr[\mathsf{G}_8 = 1] = \frac{1}{2}.$$

Clearly, c_0^* is uniform random element in $\mathbb{QR}_N$ and hence it does not contain any information about the challenge message. Since y_1, y_2 are sampled uniformly at random from $\mathbb{QR}_N$ the ciphertexts c_1^*, c_2^* are also uniform random elements in $\mathbb{QR}_N$ and hence do not contain any information about the challenge message m_b. Therefore, an adversary can not do better than guessing.

By combining Lemmas 41 - 49 we obtain the following:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{\mathtt{NITC}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \leq \sum_{i=0}^{7} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]| + \left|\Pr[\mathsf{G}_8 - \frac{1}{2}\right| \\ &\leq 2Q \cdot \mathbf{Snd}_{\mathcal{B}}^{\Sigma} + Q\delta + \mathbf{Adv}_{\mathcal{B}}^{\mathsf{SSS}} + \mathbf{OUR}_{\mathcal{B}}^{\Sigma} + \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}} + 2\left(\frac{1}{p} + \frac{1}{q} - \frac{1}{N}\right),\end{aligned}$$

which concludes the proof.