Private Coin Verifiable Delay Function

Peter Chvojka

chvojka.p@gmail.com

Abstract. We construct the first tight verifiable delay function (VDF) where the evaluation algorithm only evaluates sequentially the function and hence outputs and empty proof, verification is independent of time parameter T and setup has constant size parameters. Our VDF is based on repeated squaring in hidden order groups, but it requires that coins used to sample a random instance must be kept secret in order to guarantee sequentiality. We denote such a VDF as a *private coin* verifiable delay function and show that it can be used to obtain multiplicatively homomorphic non-interactive timed commitment with efficient publicly verifiable force decommitment algorithm.

1 Introduction

A verifiable delay function (VDF) is a function which takes some specified time T to evaluate on a random input x and moreover the evaluation algorithm produces together with an output y a proof π which allows to efficiently check that (x, y) is valid input/output pair of the VDF. This primitive has been introduced by Boneh *et al.* in [BBBF18] and has found several interesting applications. Soon after the introduction of this primitive, Pietrzak [Pie19] and Wesolowski [Wes19] constructs VDFs based on repeated squaring in hidden order groups. Their constructions differs in the size of generated proofs, in verification time, in time needed to compute a proof and in assumptions on which their security is based. Concretely, Pietrzak's construction is based on the low order assumption, the verification time is $2\log_2 T$ small exponentiations, the time needed to compute a proof is $2\sqrt{T}$ and the size of the proof is $log_2 T$ group elements of the underlying group. Wesolowski's construction is based on the adaptive root assumption, the verification time is $\log_2 T$ multiplications and two small exponentiations, the time needed to compute a proof is $2T$ group operations (this can be reduce to T group operations using a windowing method) and the size of the proof is single element of the underlying group. Döttling *et al.* [DGMV20] study a *tight* verifiable delay functions where the prover's computational complexity is not much more than the sequentiality bound T. Moreover, they show that VDF can not be constructed in black-box way from random oracles. De Feo *et al.* [DMPS19] construct the first tight verifiable delay function where the evaluation algorithm does not need to compute any proof and hence the verification time is independent of T. The construction is based on supersingular isogenies, however, its public parameters grow linearly with the time parameter T. The notion of *continuous* verifiable delay function (cVDF) was introduced by Ephraim *et al.*

in [EFKP20] and it has an additional property that the intermediate steps of the computation are publicly and continuously verifiable. Construction of cVDF of Ephraim *et al.* is based on repeated squaring assumption and Fiat-Shamir heuristic.

Till now there remained open questions if it is possible to build a tight VDF based on the squaring assumption in hidden order groups whose verification does not require any proof and if it is possible to build such a tight VDF whose public parameters are constant size. We answer both of this questions positively, however, under the condition that the coins which are used to sample random instance have to remain secret in order to preserve sequentiality property. We denote such a VDF as a *private coin* verifiable delay function. This requirement unfortunately makes applicability of our VDF in typical scenarios significantly harder and it would require a multi-party computation protocol for instance sampling. However, we show that there might be applications outside of typical scenarios.

Concretely, we show that our VDF can be used in building non-interactive timed commitment (NITC) [KLX20] which achieves efficient public verifiability of force decommitment. A non-interactive timed commitment is a commitment which can be additionally opened by anyone after some deadline T has passed. This is usually achieved by executing some sequential computation which takes time roughly T. In a public verifiable NITC this computation can be outsourced to an untrusted server that can prove that the computation was performed correctly. Chvojka *et al.* [CJ23] propose a construction of multiplicatively homomorphic public verifiable NITC in which the party which executes forced decommitment has to compute a proof of exponentiation in order to achieve public verifiability. As we show, this can be avoided by a small modification of the construction. The main disadvantage of our approach is that one has to rely on generic purpose NIZKs.

We remark that our constructions fulfil the standard VDF definition, since these definitions are not explicit about the way how random instances are sampled. We believe that these VDFs are not only interesting from the theoretical point of view, but the ideas used in their constructions might find useful applications in different scenarios.

2 Technical Overview

We construct two verifiable dealy functions using the same ideas. Our first VDF achieves only weaker soundness of Pietrzak [Pie19], which assumes that an instance is honestly generated. Then we show, how this construction can be modified to achieve stronger soundness as defined by Boneh *et al.* [BBBF18]. Our constructions are based on the repeated squaring in hidden order groups. Similarly to [Pie19,Wes19], the evaluation algorithm of VDF is computing value $x^{2^T} \bmod N$ on input x. However, to obtain VDF with an empty proof, we rely on rerandomization technique used in building time-lock puzzles [MT19] and non-interactive timed commitments [CJ23] based on the strong sequential squaring

assumption (SSS). The basic idea is to sample a generator of the group $\mathbb{G}$ in which the SSS assumption holds and then compute $h := g^{2^T} \bmod N$. Then to commit to a message m, these elements are both rerandomized by computing $c_0 = g^r \bmod N$, $y := h^r \bmod N$, and y is used to hide a message m by computing $c_1 := ym \bmod N$. The commitment is pair (c_0, c_1). To recover message, it is sufficient to compute $y := c_0^{2^T} \bmod N$ by repeated squaring and computing $m := c_1 y^{-1} \bmod N$.

The crucial observation is, that value y which is at committing time computed efficiently just by rerandomizing h using randomness r, can be used to verify that repeated squaring of c_0 was executed correctly. For now, assume that there is a designated entity which checks the validity of the output of VDF and this entity is responsible for sampling the instance and at the same time can keep some secret value. This is in some sense similar to designated verifier NIZKs. Then if the instance is computed as $x := g^r \bmod N$ and the verifier stores value r, then the correctness of VDF ouput y can be easily check by $y = h^r \bmod N$. Notice that the verification time is independent of time parameter T. Now the question is, how to make from this VDF with designated verifier a standard VDF, or in other words how to make the verification step public. Since value y can be computed at the time of sampling instance x, one can provide some information about y as part of the instance in a way which does not hurt sequentiality. This can be done by revealing the value $x_1 := \mathsf{F}(y)$, where F is the one-way function. Observe that this does not harm sequentiality, since it is infeasible to compute value y from the evaluation of one-way function and at the same time it allows for efficient verification of the validity of VDF ouput y by evaluating the expression $x_1 = \mathsf{F}(y)$. In case that F is collision resistant function and we are guaranteed that instance x_0, x_1 is correctly generated, it is impossible to find y' such that $y' \neq y$ and $x_1 = \mathsf{F}(y')$. Hence, the soundness property of VDF is attained. However, if an instance can be maliciously generated, then the soundness does not need to hold anymore. To avoid this issue, we supplement our instance with a NIZK proof which guarantees that instance is generated correctly. Function F can be instantiated in several ways as we discuss in Section 6 and hence we obtains VDFs based on different assumptions.

Next we show, how the ideas used in building our VDFs can be applied to transform a multiplicatively homomorphic publicly verifiable non-interactive timed commitment (MHPVNITC) of Chvojka and Jager [CJ23] in such a way that force decommitment algorithm does not need to compute any additional proof in order to guarantee public verifiability property of forced decommitment. A non-interactive timed commitment [KLX20] allows to commit a message in such a way that commitment can be opened in two ways: 1) using an opening and a message, 2) using forced decommitment which can be executed by anyone and takes time roughly T to perform without knowledge of opening value. MHPVNITC of Chvojka *et al.* is based on Naor-Yung paradigm [NY90] with shared randomness [BMV16] and works as follows. The setup algorithm produces elements $g, h_1 := g^k \bmod N, h_2 := g^{2^T} \bmod N$, where g is a generator of group of quadratic residues $\mathbb{QR}_N$ and k is sampled uniformly at

random from $[|\mathbb{QR}_N|]$. A commitment to a message m is generated as $c_0 := g^r \bmod N, c_1 := h_1^r m \bmod N, c_2 := h_2^r m \bmod N$. To force decommit the message one can compute value $y := c_0^{2^T} \bmod N$ by repeated squaring and then output $m := c_3 y^{-1} \bmod N$. However, if we are interested in public verifiability of force decommitmnet, then a proof of exponentiation must be provided in addition. In order to avoid this additional computation we commit to a message m as follows. We use concrete instantiation of function F from our VDFs using the RSA trapdoor function with exponent e. In that case the instance of VDF is $(x_0 := g^r \bmod N, x_1 := (h^r)^e \bmod N, \pi_{\texttt{NIZK}})$, where $\pi_{\texttt{NIZK}}$ guarantees that instance is correctly generated. In a timed commitment we commit to a message as $c_0 := g^r \bmod N, c_1 := h_1^r m \bmod N, c_2 := (h_2^r)^e m \bmod N$ (notice that $c_0 := x_0, c_2 := x_1 m$). But since the value x_1 in commitment is blinded with value m we have to provide an additional value which ensures public verifiability of forced decommitment and at the same time preserves non-malleability. This can be achieved by adding value $c_3 := H(h_2^r)$ where H is a hash function modelled as a random oracle. Since the construction relies on Naor-Yung, we have to use one-time simulation sound NIZK proving that (c_0, c_1, c_2, c_3) is well formed commitment. We remark, that now the underlying language for our NIZK is not anymore algebraic as it was in cite and therefore we make of use a generic purpose NIZKs. Another disadvantage is that we require proving something about a hash function which is modelled as a random oracle and hence this can be considered only heuristically secure.

3 Preliminaries

We use λ to denote the security parameter. For $\lambda \in \mathbb{N}$ we write 1^λ to denote the λ-bit string of all ones. To indicate that we choose x uniformly at random from X for any element x in a set X, we use $x \xleftarrow{\$} X$. We write $[n]$ to denote the set of integers $\{1, \ldots, n\}$ and $\lfloor n \rfloor$ to denote the greatest integer that is less than or equal to n. All algorithms are randomized, unless explicitly defined as deterministic. For simplicity we model all algorithms as Turing machines, however, all adversaries are modelled 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. 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 $\lambda, a_1, \ldots, a_n$ and fresh randomness and then assigning the output to x.

3.1 Verifiable Delay Function

We define a verifiable delay function in similar fashion as Boneh *et al.* [BBBF18], however, we additinaly consider Gen algorith that generates instances from the domain of the function as in [Pie19].

Definition 1. *A verifiable delay function* $\texttt{VDF}$ *is a tuple of four algorithms* $\texttt{VDF} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Eval}, \mathsf{Vrfy})$ *with the following syntax.*

– $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T)$ *is a probabilistic algorithm that takes as input the security parameter* 1^λ *and a hardness parameter* T *and outputs public parameters.*
– $x \leftarrow \mathsf{Gen}(\mathsf{pp})$ *is a probabilistic algorithm that takes as input public parameters* pp *and samples a value* x*.*
– $(y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x)$ *takes as input public parameters* pp *and value* x *and outputs value* y *and (possibly empty) proof* π*.*
– $0/1 \leftarrow \mathsf{Vrfy}(\mathsf{pp}, x, y, \pi)$ *is a deterministic algorithm that takes as input public parameters pp, values* x, y*, and proof* π *and outputs* 0 *(reject) or* 1 *(accept).*

We say VDF *is correct if for all* $\lambda, T \in \mathbb{N}$ *holds:*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{pp}, x, y, \pi) = 1 : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ x \leftarrow \mathsf{Gen}(\mathsf{pp}) \\ (y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x) \end{array}\right] = 1.$$

Since the soundness definitions of Boneh *et al.* and Pietrzak differs, we consider both of them, however soundness definition of Pietrzak is adjusted to our setting. The main difference between them is that in [Pie19] the challenge instance x is generated in the experiment and in [BBBF18] an adversary outputs an instance x.

Definition 2 (Soundness [Pie19]). *A* VDF *is sound if for all* $T \in \mathbb{N}$ *and all PPT algorithms* $\mathcal{A}$ *that runs in time* $\mathsf{poly}(T, \lambda)$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{VDF-SND}}_{\mathcal{A}} = \Pr\left[\begin{array}{r} \mathsf{Vrfy}(\mathsf{pp}, x, y', \pi') = 1 \\ \wedge y \neq y' \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ x \leftarrow \mathsf{Gen}(\mathsf{pp}) \\ (y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x) \\ (y', \pi') \leftarrow \mathcal{A}(\mathsf{pp}, T, x) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 3 (Soundness [BBBF18]). *A* VDF *is sound if for all* $T \in \mathbb{N}$ *and all PPT algorithms* $\mathcal{A}$ *that runs in time* $\mathsf{poly}(T, \lambda)$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{VDF-SND}}_{\mathcal{A}} = \Pr\left[\begin{array}{r} \mathsf{Vrfy}(\mathsf{pp}, x, y', \pi') = 1 \\ \wedge y \neq y' \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ (x, y', \pi') \leftarrow \mathcal{A}(\mathsf{pp}, T, x) \\ (y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 4 (Sequentiality). *A* VDF *is 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 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* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{VDF-SEQ}}_{\mathcal{A}} = \Pr\left[\mathsf{Vrfy}(\mathsf{pp}, x, y, \pi) = 1 : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ \mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp}) \\ x \leftarrow \mathsf{Gen}(\mathsf{pp}) \\ (y, \pi) \leftarrow \mathcal{A}_{2,\lambda}(x, \mathsf{st}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 5 (Private Coin VDF). *A* $\texttt{VDF}$ *is private coin if* Gen *algorithm additionally outputs a trapdoor value* τ *and there is an additional algorithm* $\mathsf{TrapEval}$ *with the following syntax.*

- $y \leftarrow \mathsf{TrapEval}(\mathsf{pp}, x, \tau)$ *is a deterministic algorithm that takes as input public parameters* pp*, a value* $x \in \mathcal{X}$ *and a trapdoor* τ *and outputs a value* $y \in \mathcal{Y}$ *in time* $\mathsf{poly}(\lambda)$ *that is independent of* T*.*

A private coin VDF is correct, if it fulfils correctness definition of VDF and additionally it holds

$$\Pr\left[y = y' : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T) \\ (x, \tau) \leftarrow \mathsf{Gen}(\mathsf{pp}) \\ (y, \pi) \leftarrow \mathsf{Eval}(\mathsf{pp}, x) \\ y' \leftarrow \mathsf{TrapEval}(\mathsf{pp}, x, \tau) \end{array}\right] = 1.$$

3.2 Non-Malleable Publicly Verifiable Non-Interactive Timed Commitment

Non-Malleable Non-Interactive Timed Commitment (NITC) has been introduced by Boneh *et al.* [KLX20] and later homomorphic NITC was firstly considered in [TCLM21] and public verifiable NITC by Chvojka and Jager [CJ23]. We define security definitions of NITC in the same way as in [CJ23].

Definition 6. *A non-interactive timed commitments scheme* $\texttt{NITC}$ *with message space* $\mathcal{M}$ *is a tuple of algorithms* $\texttt{NITC} = (\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *with the following syntax.*

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

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

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

Definition 7. *A non-interactive timed commitment scheme* $\texttt{NITC}$ *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*

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

where $|m_0| = |m_1|$ *and the oracle* $\mathsf{DEC}(c, \pi_{\mathsf{Com}})$ *returns the result of* $\mathsf{FDec}(\mathsf{pp}, c)$ *if* $\mathsf{ComVrfy}(\mathsf{pp}, c, \pi_{\mathsf{Com}}) = 1$*, 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* $(c^*, \pi^*_{\mathsf{Com}})$*.*

Definition 8. *We define the* $\mathsf{BND\text{-}CCA}_{\mathcal{A}}(\lambda)$ *experiment as follows:*

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

A non-interactive timed commitment scheme $\texttt{NITC}$ *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* $\lambda \in \mathbb{N}$

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

Definition 9. *A non-interactive timed commitments scheme* $\texttt{NITC}$ *is* publicly verifiable *if* FDec *additionally outputs a proof* π_{FDec} *and has an additional algorithm* $\mathsf{FDecVrfy}$ *with the following syntax:*

- $0/1 \leftarrow \mathsf{FDecVrfy}(\mathsf{pp}, c, m, \pi_{\mathsf{FDec}})$ *is a deterministic algorithm that takes as input public parameters* pp*, 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:*

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

– 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* $\lambda \in \mathbb{N}$

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

Definition 10. *A non-interactive timed commitments scheme* 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:*

– $c \leftarrow \mathsf{Eval}(\mathsf{pp}, C, c_1, \ldots, c_n)$ *is a probabilistic algorithm that takes as input public parameters* pp*, 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* pp *in the support of* $\mathsf{PGen}(1^\lambda, T)$*, all* c_i *in the support of* $\mathsf{Com}(\mathsf{pp}, m_i)$ *we have:*

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

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

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

Compactness: *for all* $\lambda, T \in \mathbb{N}$*,* $C \in \mathcal{C}_\lambda$*,* $(m_1, \ldots, m_n) \in \mathcal{M}^n$*, all* pp *in the support of* $\mathsf{PGen}(1^\lambda, T)$*, all* c_i *in the support of* $\mathsf{Com}(\mathsf{pp}, m_i)$*, 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* $c \leftarrow \mathsf{Eval}(\mathsf{pp}, C, c_1, \ldots, c_n)$*.*
2. *The exists a fixed polynomial* $\tilde{\mathsf{poly}}$ *such that the runtime of* $\mathsf{Eval}(\mathsf{pp}, C, c_1, \ldots, c_n)$ *is bounded by* $\tilde{\mathsf{poly}}(\lambda, |C|)$*.*

3.3 Non-Interactive Zero-Knowledge Proof

In this work we consider Non-Interactive Zero-Knowledge proof systems (NIZKs) both in standard model and in the random oracle model (ROM). Reason for it is that for an language L in our construction of verifiable delay function with efficient membership test, one can propose an efficient Sigma protocol which can be transformed into NIZK in the ROM and also in the standard model. Moreover, in our PVNITC construction we rely on NIZK in standard model. We begin with definition of standard model NIZKs that we take from Libert *et al.* [LNPY22] that are adjusted in the same way as in [CJ23]. In comparison to another NIZK definitions in standard model, in definitions of Libert *et al.* and Chvojka *et al.* an adversary gets some auxiliary information about a language called a membership testing trapdoor τ_L. Later we propose an instantiation of NIZK for our application which fulfils these definitions with respect to some specific language trapdoor.

Definition 11. *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*.*

Definition 12 (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* $\pi' \leftarrow \mathsf{Sim}(\mathsf{crs}, \tau, s')$*.*

Non-Interactive Zero-Knowledge Proofs in the Random Oracle Model. Next we recall definition of a NIZK in the ROM.

Definition 13. *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 14 (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 $\lambda \in \mathbb{N}$*

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

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 $(v, \mathsf{st}) \leftarrow \mathsf{Sim}(2, \mathsf{st}, u)$.*

3.4 Complexity assumptions

Our constructions of VDF are based on the strong sequential squaring assumption in group $\mathbb{G}$ which can be instantiated either by the group of quadratic residues or by the group elements with Jacobi symbol 1. Let p, q be safe primes (i.e., such that $p = 2p'+1, q = 2q'+1$ for primes p', q')and $\varphi(\cdot)$ be Euler's totient function. We denote by $\mathbb{QR}_N$ 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 element x from $\mathbb{QR}_N$, one can sample $r \xleftarrow{\$} \mathbb{Z}_N^*$ and let $x := r^2 \bmod N$. 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 $x^{p'} \neq 1 \bmod N \wedge x^{q'} \neq 1 \bmod N$. Therefore we are able to efficiently sample a random generator of $\mathbb{QR}_N$. We denote by $\mathbb{J}_N$ the cyclic subgroup of $\mathbb{Z}_N^*$ of elements with Jacobi symbol 1 which has order $|\mathbb{J}_N| = \frac{\varphi(N)}{2} = \frac{(p-1)(q-1)}{2}$. To efficiently sample a random generator g from $\mathbb{J}_N$, one can sample $r \xleftarrow{\$} \mathbb{Z}_N^*$ and let $g := -r^2 \bmod N$. 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 $\mathbb{G}$. Now we state the strong sequential squaring assumption in $\mathbb{G}$.

$\mathsf{ExpSSS}^b_{\mathcal{A}}(\lambda)$:
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(N, T(\lambda), g)$
$x \xleftarrow{\$} \mathbb{G}$
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 15 (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* $\lambda \in \mathbb{N}$

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

The construction of multiplicatively homomorphic PVNITC of Chvojka and Jager [CJ23] relies on the Decisional Diffie-Hellman assumption in $\mathbb{QR}_N$ as stated below.

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

Fig. 2. Security experiment for DDH

Definition 16 (Decisional Diffie-Hellman in $\mathbb{QR}_N$). *Consider the security experiment* $\mathsf{ExpDDH}^b_{\mathcal{A}}(\lambda)$ *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* $\lambda \in \mathbb{N}$

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

$\underline{\mathsf{ExpRSA}_{\mathcal{A}}(\lambda)\text{:}}$
$(p, q, N, g, e) \leftarrow \mathsf{GenMod}(1^\lambda)$
$y \xleftarrow{\$} \mathbb{G}$
$x \leftarrow \mathcal{A}_\lambda(N, e, y, g)$
return the truth value of $x^e = y \bmod N$

Fig. 3. Security experiment for RSA

In our VDF constructions we make use of collision resistant one way function which can be instantiated based for example on the RSA assumption or the square-root assumption. We recall both of them below. In the RSA experiment definition we use $\mathbb{G}$ to denote either $\mathbb{J}_N$ or $\mathbb{QR}_N$. The original definition of the RSA assumption is in the group of $\mathbb{Z}^*_N$ which implies that it holds in both $\mathbb{J}_N$ and $\mathbb{QR}_N$. Moreover, we adjust the assumption slightly by requiring that GenMod outputs a generator of $\mathbb{G}$.

Definition 17 (RSA assumption). *Consider the security experiment* $\mathsf{ExpRSA}_{\mathcal{A}}(\lambda)$ *in Figure 3. The RSA assumption holds relative to* GenMod *in* $\mathbb{G}$ *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* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{RSA}}_{\mathcal{A}} = \Pr[\mathsf{ExpRSA}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Definition 18 (Square-root assumption). *Consider the security experiment* $\mathsf{ExpSR}_{\mathcal{A}}(\lambda)$ *in Figure 4. The square-root assumption holds relative to* GenMod

$\underline{\mathsf{ExpSR}_{\mathcal{A}}(\lambda):}$
$(p, q, N) \leftarrow \mathsf{GenMod}(1^{\lambda})$
$y \xleftarrow{\$} \mathbb{QR}_N$
$x \leftarrow \mathcal{A}_{\lambda}(N, y)$
return the truth value of $x^2 = y \bmod N$

Fig. 4. Security experiment for square-root assumption

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 $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{SR}}_{\mathcal{A}} = \Pr[\mathsf{ExpSR}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Sampling random exponents for $\mathbb{QR}_N$ and $\mathbb{J}_N$. As shown in [CJ23] we can use the set $[\lfloor N/4 \rfloor]$ whenever we should sample from the set $[\varphi(N)/4]$ without knowing the factorization of N. Similarly, we can use the set $[\lfloor N/2 \rfloor]$ whenever we should sample from the set $[\varphi(N)/2]$. Sampling from $[\lfloor N/4 \rfloor]$ and $[\lfloor N/2 \rfloor]$ is statistically indistinguishable from sampling from $[\varphi(N)/4]$ and $\varphi(N)/2$ respectively as stated in Lemma 1 below which is proven in [CJ23].

Definition 19 (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 1. *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

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

Definition 20 (One-way function). *A function $\mathsf{F} : \mathcal{X} \to \mathcal{Y}$ is one-way function if the following holds:*

- *There exist a polynomial time algorithm that computes F.*
- *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 $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{\mathsf{OWF}}_{\mathcal{A}} = \Pr\left[\mathsf{F}(x') = y : \begin{array}{r} x \xleftarrow{\$} \mathcal{X} \\ x' \leftarrow \mathcal{A}_{\lambda}(\mathsf{F}(x)) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 21 (Collision resistant function). *A function $\mathsf{F} : \mathcal{X} \to \mathcal{Y}$ is a collision resistant function, 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 $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{A}} = \Pr[\mathsf{F}(x) = \mathsf{F}(x') : (x, x') \leftarrow \mathcal{A}_{\lambda}] \leq \mathsf{negl}.$$

4 VDF Construction

Let N be a product of two large primes. We construct a VDF whose domain is determined by elements $g, h \in \mathbb{G}^2$ and a collision resistant one-way function F produced in Setup and consists of pairs (x_0, x_1) such that $x_0 = g^r \bmod N$ and $x_1 = \mathsf{F}(h^r \bmod N)$ for some $r \in R$. We require that the domain of F is group $\mathbb{G}$. Random sampling from the domain can be done efficiently by sampling at first $r \xleftarrow{\$} R$, where sampling from the set R is statistically indistinguishable from sampling from $[|\mathbb{G}|]$ and then by computing x_0 and x_1 as described above. The construction is given in Figure 5.

$\underline{\mathsf{Setup}(1^\lambda, T)}$
$(p, q, N, \mathbb{G}, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$t := 2^T \bmod |\mathbb{G}|$
$h := g^t \bmod N$
return $(\mathsf{pp} := (N, \mathsf{F}, g, h))$

$\underline{\mathsf{Gen}(\mathsf{pp})}$
$r \xleftarrow{\$} R$
$x_0 := g^r \bmod N, x_1 := \mathsf{F}(h^r \bmod N)$
return $x := (x_0, x_1), \tau := r$

$\underline{\mathsf{Eval}(\mathsf{pp}, x = (x_0, x_1))}$
return $y := x_0^{2^T} \bmod N$

$\underline{\mathsf{Vrfy}(\mathsf{pp}, x = (x_0, x_1), y)}$
if $x_1 = \mathsf{F}(y)$
    return 1
return 0

$\underline{\mathsf{TrapEval}(\mathsf{pp}, x = (x_0, x_1), \tau)}$
return $(y, \pi) := (h^\tau \bmod N, \perp)$

Fig. 5. VDF

Theorem 1. *If the strong sequential squaring assumption with gap ϵ holds relative to* GenMod*, and* F *is a collision resistant one-way function, then* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Eval}, \mathsf{Vrfy})$ *defined in Figure 5 is a secure private coin verifiable delay function that is sound in the sense of Definition 2.*

Proof. We show that VDF in Figure 5 is complete, sound in the sense of Definition 2 and fulfils sequentiality definition.

Completeness. It is straightforward to verify.

Soundness. We show that if there is an adversary $\mathcal{A}$ which breaks soundness then we can break collision resistance of F. This is straightforward, since in Definition 2 the instance $x = (x_0, x_1)$ is honestly generated and hence $\mathsf{F}(y) = x_1 = \mathsf{F}(y')$ for $y \neq y'$. Therefore y, y' is required collision. Hence, we conclude that

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{VDF-SND}} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{CR}}.$$

Sequentiality. We consider a sequence of games $\mathsf{G}_0 - \mathsf{G}_3$.

Game 0. Game G_0 corresponds to original security experiment.

Game 1. In G_1 we sample exponent r for instance x, uniformly at random from $[|\mathbb{G}|]$.

Lemma 2.

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

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 1, which directly yields the required bound.

Game 2. In G_2 we sample $y \xleftarrow{\$} \mathbb{G}$ and compute the challenge as $x := (g^r \bmod N, \mathsf{F}(y))$.

Lemma 3.

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

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

1. Computes $h := g^{2^{T(\lambda)}} \bmod N$ and sets $\mathsf{pp} := (N, \mathsf{F}, g, h)$.
2. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$.
3. Outputs $(N, \mathsf{F}, g, h, \mathsf{st})$

The adversary $\mathcal{B}_{2,\lambda}(x, y, (N, \mathsf{F}, g, h, \mathsf{st}))$:

1. Sets $x_0^* := x$ and computes $x_1^* := \mathsf{F}(y) \bmod N$.
2. Runs $y^* \leftarrow \mathcal{A}_{2,\lambda}((x_0^*, x_1^*), \mathsf{st})$.
3. Returns the truth value of $y = y^*$.

Since g is a generator of $\mathbb{G}$ and x is sampled uniformly at random from $\mathbb{G}$ there exists some $r \in [|\mathbb{G}|]$ such that $x = g^r \bmod N$. Therefore when $y = x^{2^T} = (g^{2^T})^r \bmod N$, then $\mathcal{B}$ simulates G_1 perfectly. Otherwise y is random value and $\mathcal{B}$ simulates G_2 perfectly. This proofs the lemma.

Game 3. In G_3 we sample exponent r for x_0^*, uniformly at random from $[R]$.

Lemma 4.

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

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 1, which directly yields the required bound.

Lemma 5.

$$\Pr[\mathsf{G}_3 = 1] = \mathbf{Adv}^{\mathsf{OWF}}_{\mathcal{B}'}.$$

The adversary $\mathcal{B}'(y)$:

1. Samples $g \xleftarrow{\$} \mathbb{G}$ (with high probability g is generator).
2. Computes $h := g^{2^T} \bmod N$ by repeated squaring.
3. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(N, \mathsf{F}, g, h)$.
4. Computes $x_0^* := g^r \bmod N$ where $r \xleftarrow{\$} [R]$.
5. Returns $y^* \leftarrow \mathcal{A}_{2,\lambda}((x_0^*, y), \mathsf{st})$.

$\mathcal{B}'$ simulates G_3 perfectly and whenever $\mathcal{A}$ outputs correct value for our VDF then it necessary holds that $\mathsf{F}(y^*) = y \bmod N$ which proofs the lemma.

By combining Lemmas 2 - 5 we obtain the following:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{VDF-SEQ}} = \Pr[\mathsf{G}_0 = 1] \leq \sum_{i=0}^{3} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]| + \Pr[\mathsf{G}_2 = 1]$$
$$\leq 2(\frac{1}{p} + \frac{1}{q} - \frac{1}{N}) + \mathbf{Adv}_{\mathcal{B}}^{\mathsf{SSS}} + \mathbf{Adv}_{\mathcal{B}'}^{\mathsf{OWF}},$$

which concludes the proof.

Remark 1. The construction in Figure 5 does not achieve soundness as stated in Definition 3. The problem is, that it is not possible to verify that instance x has the correct form. An adversary can provide $x = (g^r \bmod N, \mathsf{F}(y))$ for some random value y and since F is a one-way function we are not able to notice that this instance is maliciously constructed. An adversary can then output value y as output of VDF and this value verifies. Output of Eval algorithm is however $(g^r)^{2^T} \bmod N$ which is with all but negligible probability different from y and hence we are able to break soundness. Another problem when we want to construct a VDF in group $\mathbb{QR}_N$ is that an adversary can output an element in $\mathbb{J}_N \setminus \mathbb{QR}_N$ and we are not able to notice this fact, since there is no efficient membership test for elements in $\mathbb{QR}_N$.

To fix this issue, we change the domain of the VDF by adding the proof π that certifies that given instance is well formed and hence allows for efficient membership testing.

5 VDF with an Efficient Membership Test

Even though the previous construction fulfils the soundness definition of Definition 2 of a trapdoor VDF, it is difficult to efficiently check if $(x_0, x_1) = (g^r \bmod N, \mathsf{F}(h^r \bmod N))$, since F is a one-way function. Depending on an application, this might be problematic and therefore we provide also VDF which avoids this issue by simply certifying the fact that (x_0, x_1) is correctly generated using a NIZK proof. As we show, in this way we obtain VDF which achieves soundness as specified in Definition 3. As before, let N be a product of two primes. The domain is determined by elements $g, h \in \mathbb{G}$ produced in Setup and by a collision resistant one-way function F, and consists of triples (x_0, x_1, π) such

that the tuple $x_0 = g^r \bmod N \wedge x_1 = \mathsf{F}(h^r \bmod N)$ for some r and π is a proof certifying this fact. We require that the domain of F is $\mathbb{G}$. Random sampling from the domain can be done efficiently by sampling at first $r \xleftarrow{\$} R$, where sampling from the set R is statistically indistinguishable from sampling from $[|\mathbb{G}|]$ and then by computing x_0, x_1, π as described above. Let $\mathtt{NIZK} = (\mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ is a non-interactive zero-knowledge proof system for the language

$$L = \{(x_0, x_1) | \exists r : x_0 = g^r \bmod N \wedge x_1 = \mathsf{F}(h^r \bmod N)\},$$

where g, h, N, F are parameters specifying the language. The construction is given in Figure 6.

$\underline{\mathsf{Setup}(1^\lambda, T)}$	$\underline{\mathsf{Gen}(\mathsf{pp})}$
$(p, q, N, \mathbb{G}, g) \leftarrow \mathsf{GenMod}(1^\lambda)$	$r \xleftarrow{\$} R$
$t := 2^T \bmod \lvert\mathbb{G}\rvert$	$x_0 := g^r \bmod N, x_1 := \mathsf{F}(h^r \bmod N)$
$h := g^t \bmod N$	$\pi \leftarrow \mathtt{NIZK.Prove}((x_0, x_1), r)$
return $(\mathsf{pp} := (N, \mathsf{F}, g, h))$	return $x := (x_0, x_1, \pi), \tau := r$
$\underline{\mathsf{Eval}(\mathsf{pp}, x = (x_0, x_1, \pi_{\mathtt{NIZK}}))}$	$\underline{\mathsf{Vrfy}(\mathsf{pp}, x = (x_0, x_1, \pi_{\mathtt{NIZK}}), y)}$
return $y := (g^r)^{2^T} \bmod N$	if $\mathtt{NIZK.Vrfy}((x_0, x_1), \pi_{\mathtt{NIZK}}) = 1 \wedge x_1 = \mathsf{F}(y)$
	return 1
	return 0
$\underline{\mathsf{TrapEval}(\mathsf{pp}, x = (x_0, x_1, \pi), \tau)}$	
return $(y, \pi) := (h^\tau \bmod N, \perp)$	

Fig. 6. VDF with an Efficient Membership Test

Theorem 2. *If* $\mathtt{NIZK} = (\mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ *is a non-interactive zero-knowledge proof system for language* L*, the strong sequential squaring assumption with gap* ϵ *holds relative to* GenMod*, and* F *is a collision resistant one-way function, then* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Eval}, \mathsf{Vrfy})$ *defined in Figure 6 is a secure private coin verifiable delay function which satisfies soundness in the sense of Definition 3.*

Proof. We show that VDF in Figure 6 is complete, sound in the sense of Definition 3 and fulfils sequentiality definition.

Completeness. It follows from the completeness of the corresponding proof system.

Soundness. Let denote by SND the event that $\mathcal{A}$ breaks the soundness in Definition 3. Let E denote an event that $\mathcal{A}$ outputs an instance $x = (x_0, x_1, \pi)$ that is not a valid tuple.

If E does not happen, then if $\mathcal{A}$ outputs y' which verifies and is different from y this means $\mathsf{F}(y) = x_1 = \mathsf{F}(y')$. Hence, y, y' is valid collision and we can conclude $\Pr[\mathsf{SND} \wedge \overline{\mathsf{E}}] = \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{B}}$.

In case that E happens, then (x_0, x_1) is not well formed instance and we can break the soundness of the NIZK. Hence $\Pr[\mathsf{E}] \leq \mathbf{Snd}^{\mathtt{NIZK}}_{\mathcal{B}'}$. We can conclude

$$\begin{aligned}\mathbf{Adv}^{\mathsf{VDF-SND}}_{\mathcal{A}} &= \Pr[\mathsf{SND}] = \Pr[\mathsf{SND} \wedge \mathsf{E}] + \Pr[\mathsf{SND} \wedge \overline{\mathsf{E}}] \\ &\leq \Pr[\mathsf{E}] + \Pr[\mathsf{SND} \wedge \overline{\mathsf{E}}] = \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{B}} + \mathbf{Snd}^{\mathtt{NIZK}}_{\mathcal{B}'}.\end{aligned}$$

Sequentiality. We consider a sequence of games $\mathsf{G}_0 - \mathsf{G}_4$.

Game 0. Game G_0 corresponds to original security experiment.

Game 1. Game G_1 proceeds exactly as the previous game but we use zero-knowledge simulator to produce a simulated proof for the challenge instance.

Lemma 6.

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

The adversary $\mathcal{B}$:

1. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(N, \mathsf{F}, g, h)$.
2. Samples $r \xleftarrow{\$} R$ and computes $x_0^* := g^r \bmod N, x_1^* := \mathsf{F}(h^r \bmod N)$.
3. It submits $(s := (x_0^*, x_1^*), w := r)$ to its oracle and obtains proof π^* as answer.
4. Runs $y^* \leftarrow \mathcal{A}_{2,\lambda}(\mathsf{st}, (x_0^*, x_1^*, \pi^*))$
5. Outputs $y^* = h^r \mod N$.

If the proof π^* is generated using $\mathtt{NIZK}.\mathsf{Prove}$, then $\mathcal{B}$ simulates G_0 perfectly. Otherwise π^* is generated using Sim_1 and $\mathcal{B}$ simulates G_1 perfectly. This proves the lemma.

Game 2. In G_2 we sample exponent r for instance x, uniformly at random from $[|\mathbb{G}|]$.

Lemma 7.

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

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 1, which directly yields the required bound.

Game 3. In G_3 we sample $y \xleftarrow{\$} \mathbb{G}$ and compute the challenge as $x^* := (g^r \bmod N, \mathsf{F}(y), \pi)$.

Lemma 8.

$$|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}'}.$$

The adversary $\mathcal{B}'_{1,\lambda}(N, T(\lambda), g)$:

1. Computes $h := g^{2^{T(\lambda)}} \bmod N$ and sets $\mathsf{pp} := (N, \mathsf{F}, g, h)$.
2. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$.
3. Outputs $(N, \mathsf{F}, g, h, \mathsf{st})$

The adversary $\mathcal{B}'_{2,\lambda}(x, y, (N, \mathsf{F}, g, h, \mathsf{st}))$:

1. Sets $x_0^* := x$ and computes $x_1^* := \mathsf{F}(y)$.
2. Runs $\pi^* \leftarrow \mathtt{NIZK.Sim}(1, \mathsf{st}', (x_0^*, x_1^*))$.
3. Runs $y^* \leftarrow \mathcal{A}_{2,\lambda}((x_0^*, x_1^*, \pi^*), \mathsf{st})$.
4. Returns the truth value of $y = y^*$.

Since g is a generator of $\mathbb{G}$ and x is sampled uniformly at random from $\mathbb{G}$ there exists some $r \in [|G|]$ such that $x = g^r \bmod N$. Therefore when $y = x^{2^T} = (g^{2^T})^r \bmod N$, then $\mathcal{B}'$ simulates G_2 perfectly. Otherwise y is random value and $\mathcal{B}'$ simulates G_3 perfectly. This proofs the lemma.

Game 4. In G_4 we sample exponent r for x_0^*, uniformly at random from $[R]$.

Lemma 9.

$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

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 1, which directly yields the required bound.

Lemma 10.

$$\Pr[\mathsf{G}_4 = 1] = \mathbf{Adv}^{\mathsf{OWF}}_{\tilde{\mathcal{B}}}.$$

The adversary $\tilde{\mathcal{B}}(y)$:

1. Samples $g \xleftarrow{\$} \mathbb{G}$ (with high probability g is generator).
2. Computes $h := g^{2^T} \bmod N$ by repeated squaring.
3. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(N, \mathsf{F}, g, h)$.
4. Computes $x_0^* := g^r \bmod N$ where $r \xleftarrow{\$} R$.
5. Runs $\pi^* \leftarrow \mathtt{NIZK.Sim}(1, \mathsf{st}', (x_0^*, y))$.
6. Returns $y^* \leftarrow \mathcal{A}_{2,\lambda}((x_0^*, y, \pi^*), \mathsf{st})$.

$\tilde{\mathcal{B}}$ simulates G_4 perfectly and whenever $\mathcal{A}$ outputs correct value for our VDF then it necessary holds that $\mathsf{F}(y^*) = y \bmod N$ which proofs the lemma.

By combining Lemmas 6 - 10 we obtain the following:

$$\mathbf{Adv}^{\mathsf{VDF-SEQ}}_{\mathcal{A}} = \Pr[\mathsf{G}_0 = 1] \leq \sum_{i=0}^{4} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]| + \Pr[\mathsf{G}_4 = 1]$$

$$\leq 2(\frac{1}{p} + \frac{1}{q} - \frac{1}{N}) + \mathbf{Adv}^{\mathbf{ZK}}_{\mathcal{B}} + \mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}'} + \mathbf{Adv}^{\mathsf{OWF}}_{\tilde{\mathcal{B}}},$$

which concludes the proof.

Remark 2. Proposed VDFs require the trusted setup, since the factorization of N must be kept secret. One can achieve also the transparent setup if the VDFs are instantiated over the class group of an imaginary quadratic number field. In that case the value $h := g^{2^T}$ computed in the setup must be computed by repeated squaring and hence the runtime of the setup is $O(T)$.

6 Instantiation of function F

In this section we discuss several ways how to instantiate a collision resistant one-way function F.

Using RSA trapdoor one-way function. If we choose e, such that $\gcd(e, \varphi(N)) = 1$ (for example e = 3), then function $x^e \bmod N$ is a well-known one-way function based on RSA assumption (Definition 17) in the group $\mathbb{G}$. Moreover, this function is permutation on $\mathbb{Z}_N^*$ therefore there are no collisions and can be instantiated over both $\mathbb{QR}_N$ and $\mathbb{J}_N$. When F is instantiated in this way, the language for a NIZK proof system is equivalent to showing that (g, h^e, x_0, x_1) is a DDH tuple and an efficient Sigma protocol for this language was given for example in [CJ23, Theorem 11]. The proposed Sigma protocol can be transformed into NIZK in ROM using Fiat-Shamir transformation [FS87] or into NIZK in the standard model using technique of Libert *et al.*. [LNPY22].

Using Rabin's one-way function . Another candidate for F is Rabin's one-way function which is essentialy computing $x^2 \bmod N$. This function is one-way function since computing square roots modulo N is considered to be a hard problem (Definition 18) and actually it is equivalent to factoring. Moreover, if the domain of this function is $\mathbb{QR}_N$, then the function is a permutation and there are no collisions. This is because assuming that p, q are safe primes, then N is a Blume integer. VDF obtained in this way over $\mathbb{QR}_N$ which is given in Figure 5 is sound in the sense of Definition 2. However, if we instantiate construction in Figure 5 over $\mathbb{J}_N$, then we don't achieve soundness as defined in Definition 2. Moreover, we are not able to achieve soundness in the sense of Definition 3 also for VDF constructions given in Figure 6 over both $\mathbb{QR}_N$ and $\mathbb{J}_N$. The problem is that Rabin's function is not collision resistant over $\mathbb{J}_N$ and since we do not know how to efficiently check a membership in $\mathbb{QR}_N$, even the use of Rabin's function over $\mathbb{QR}_N$ is problematic as we explain now. An attacker can construct a collision simply by taking $x \in \mathbb{QR}_N$ and computing $y := x^2 \bmod N$. Then $(-x)^2 = y \bmod N$ and $-x \in \mathbb{J}_N \setminus \mathbb{QR}_N$. Since we do not know how to distinguish elements of $\mathbb{QR}_N$ and $\mathbb{J}_N \setminus \mathbb{QR}_N$ the pair $(x, -x)$ is a collision for y.

One could avoid this issue by defining the output of F either as a set $(x, -x)$ and adjusting the output of Eval algorithm of VDF to be also set. Another option would be to accept as a valid output for example $\max(x, -x)$ or $\min(x, -x)$. In this way if Vrfy of VDF checks that $x \in \mathbb{J}_N$ then this is permutation. Moreover, even if we would omit this check in VDF and an attacker would output another square root of y, then we would be able to break the factoring assumption. Hence, a VDF constructed in this way which omits the membership test in $\mathbb{J}_N$ is collision resistant under the factoring assumption. Another option is to work with the group of signed quadratic residues $\mathbb{QR}_N^+$ as done in [Pie19], that is isomorphic to $\mathbb{QR}_N$ and moreover one can efficiently test the membership in this group.

Similarly as before, the language for a NIZK proof system is equivalent to showing that (g, h^2, x_0, x_1) is a DDH tuple and an efficient Sigma protocol for

this language was given for example in [CJ23, Theorem 11]. The proposed Sigma protocol can be transformed into NIZK in ROM using Fiat-Shamir transformation [FS87] or into NIZK in the standard model using technique of Libert *et al.*. [LNPY22]. Now notice that the strong sequential squaring assumption implies the factoring assumption which is equivalent to the square-root assumption. Therefore we can say that a VDF obtained in this way which is given in Figure 5 is secure based on the strong sequential squaring assumption. And VDF obtained in this way which is given in Figure 6 (with adjustions defined above) is secure based on the strong sequential squaring assumption in the random oracle model if the underlying NIZK is obtained via Fiat-Shamir transformation.

Using a collision resistant hash function . The straightforward candidate for F is also a collision-resistant hash function which is one-way. Disadvantage of this approach is that when constructing a VDF with efficient instance membership test we need NIZK which enables to prove statements containing a hash functions. These NIZKs are usually less efficient than NIZKs obtained from Sigma protocolas via Fiat-Shamir transformation.

7 Construction of Multiplicatively Homomorphic Non-Malleable NITC

In this section we show how to apply ideas from our VDF's constructions to obtain a multiplicatively homomorphic publicly verifiable NITC which has efficient FDec and $\mathsf{FDecVrfy}$ algorithms. To adjust the construction of Chvojka and Jager [CJ23] we could use any function F whose domain and range are $\mathbb{QR}_N$. From the discussed options in Section 6 the simplest option is to use the RSA trapdoor function. In Figure 7 is our construction of publicly verifiable NITC and we rely on a one-time simulation sound NIZK for the following language:

$$L = \left\{ (c_0, c_1, c_2, c_3) | \exists (m, r) : \begin{array}{c} (c_0 = g^r \bmod N \wedge c_1 = h_1^r m \bmod N \\ \wedge c_2 = h_2^{re} m \bmod N \wedge c_3 = H(h_2^r \bmod N) \end{array} \right\},$$

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

We highlight the changes which are added to the original construction of [CJ23] using blue color.

The most significant changes compared to original proposal of Chvojka *et al.* are in FDec and $\mathsf{FDecVrfy}$ algorithms and therefore Figure 8 provides their original content as stated in [CJ23].

Theorem 3. *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* $\mathbb{QR}_N$*, the Decisional Diffie-Hellman assumption holds relative to* GenMod *in* $\mathbb{QR}_N$*, and* H *is modelled as a random oracle, then* $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *defined in Figure 7 is an IND-CCA-secure non-interactive timed commitment scheme with* $\underline{\epsilon}$*, for any* $\underline{\epsilon} < \epsilon$*.*

$\underline{\mathsf{PGen}(1^\lambda, T)}$
$(p, q, N, g) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\varphi(N) := (p-1)(q-1)$
$k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$
$t := 2^T \bmod \varphi(N)/4$
$h_1 := g^{k_1} \bmod N$
$h_2 := g^t \bmod N$
Pick e s.t. $\gcd(e, \varphi(N)) = 1$, e.g. $e := 3$
$\mathsf{crs} \leftarrow \mathtt{NIZK}.\mathsf{Setup}(1^\lambda, L)$
return $\mathsf{pp} := (N, T, g, h_1, h_2, \mathsf{crs}, e, H)$

$\underline{\mathsf{Com}(\mathsf{pp}, m)}$
$r \xleftarrow{\$} [\lfloor N/4 \rfloor]$
$c_0 := g^r \bmod N$
For $i \in [2] : y_i := h_i^r \bmod N$
$c_1 := y_1 \cdot m \bmod N, c_2 := y_2^e \cdot m \bmod N$
$c_3 := H(y_2)$
$c := (c_0, c_1, c_2, c_3), w := (m, r)$
$\pi_{\mathsf{Com}} \leftarrow \mathtt{NIZK}.\mathsf{Prove}(\mathsf{crs}, c, w)$
$\pi_{\mathsf{Dec}} := r$
return $(c, \pi_{\mathsf{Com}}, \pi_{\mathsf{Dec}})$

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

$\underline{\mathsf{DecVrfy}(\mathsf{pp}, c, m, \pi_{\mathsf{Dec}})}$
Parse c as (c_0, c_1, c_2, c_3)
if $c_0 = g^{\pi_{\mathsf{Dec}}} \bmod N \wedge c_1 = h_1^{\pi_{\mathsf{Dec}}} m \bmod N$
$c_2 = h_2^{e\pi_{\mathsf{Dec}}} m \bmod N \wedge c_3 = H(h_2^{\pi_{\mathsf{Dec}}})$
return 1
return 0

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

$\underline{\mathsf{FDecVrfy}(\mathsf{pp}, c, m, \pi_{\mathsf{FDec}})}$
Parse c as (c_0, c_1, c_2, c_3)
if $c_2 = \pi_{\mathsf{FDec}}^e \cdot m \bmod N \wedge c_3 = H(\pi_{\mathsf{FDec}})$
return 1
return 0

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

Fig. 7. Construction of Multiplicatively Homomorphic NITC
$\otimes_N$ refers to multiplication mod$N$

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

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

Fig. 8. FDec and FDecVrfy of Chvojka *et al.* [CJ23]

The IND-CCA security can be proved in similar fashion as in [CJ23]. 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 $b = b'$.

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

Fig. 9. 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 9 with $i := 2, \mathsf{sk} := t$ which means that secret key t and ciphertext c_2 are used, to answer decommitment queries efficiently.

Lemma 11.

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

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 $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda)$ 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 the zero-knowledge security of the NIZK we directly obtain

Lemma 12.

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

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against the zero-knowledge security of NIZK. $\mathcal{B}_\lambda$ is given as input crs together with membership testing trapdoor with respect to auxiliary input $\tau_L := (k_1, t)$ where $t := 2^T \bmod \varphi(N)/4$ and it works as follows: $\mathcal{B}_\lambda(\mathsf{crs}, \tau_L)$:

1. Sets $\mathsf{pp} := (N, T(\lambda), g, h_1, h_2, \mathsf{crs}, e, H)$, runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$, and answers decommitment queries using t which is included in τ_L.
2. Samples $b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} [\lfloor N/4 \rfloor]$ and compute $c_0^* := g^r, c_1^* := h_1^r m_b, c_2^* := h_2^{re} m_b \bmod N, c_3^* := H(h_2^r \bmod N)$. Then submits $(s := (c_0^*, c_1^*, c_2^*, c_3^*), w := (m, r))$ to the oracle to obtain proof π^*.
3. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$, answering decommitment queries using t.
4. 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 k_1 uniformly at random from $[\varphi(N)/4]$.

Lemma 13.
$$|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

This lemma directly follows from Lemma 1 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 14.
$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}}.$$

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. Chooses e s.t. $\gcd(e, \varphi(N)) = 1$, e.g. $e := 3$.
2. Computes $\varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/4, h_2 := g^t \bmod N$, runs $(\mathsf{crs}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{pp} := (N, T, g, h_1 := g^\alpha, h_2, \mathsf{crs}, e, H)$.
3. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$ and answers decommitment queries using t.
4. Samples $b \xleftarrow{\$} \{0,1\}$ and computes $(c_0^*, c_1^*, c_2^*, c_3^*) := (g^\beta, g^\gamma \cdot m_b, (g^\beta)^{te} \cdot m_b, H((g^\beta)^e))$. Runs $\pi^* \leftarrow \mathtt{NIZK}.\mathsf{Sim}_2(\mathsf{crs}, (c_0^*, c_1^*, c_2^*, c_3^*), \tau)$.
5. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((c_0^*, c_1^*, c_2^*, c_3^*), \pi^*, \mathsf{st})$ and answers decommitment queries using t.
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_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 15.
$$|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq \frac{1}{p} + \frac{1}{q} - \frac{1}{N}.$$

This lemma directly follows from Lemma 1 with $\ell := 4$.

Game 6. In G_6 we answer decommitment queries using Dec (Figure 9) with $i := 1, \mathsf{sk} := k_1$ which means that secret key k_1 and ciphertext c_1 are used.

Lemma 16.
$$|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq \mathbf{SimSnd}_{\mathcal{B}}^{\mathtt{NIZK}}.$$

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 crs together with a membership testing trapdoor $\tau_L := (k_1, t)$ where $t := 2^T \bmod \varphi(N)/4$.

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

1. Sets $\mathsf{pp} := (N, T, g, h_1, h_2, \mathsf{crs}, e, H)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$ and answers decommitment queries using k_1.
3. Samples $b \xleftarrow{\$} \{0,1\}, x, y_1 \xleftarrow{\$} \mathbb{QR}_N$ and computes $(c_0^*, c_1^*, c_2^*, c_3^*) := (x, y_1 m_b, x^{te} m_b, H(x^t))$. Forwards $(c_0^*, c_1^*, c_2^*, c_3^*)$ to simulation oracle Sim_2 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_1.
5. Find a decommitment query (c, π_{Com}) such that $\mathsf{Dec}(\mathsf{pp}, c, \pi_{\mathsf{Com}}, 1, k_1) \neq \mathsf{Dec}(\mathsf{pp}, c, \pi_{\mathsf{Com}}, 2, t)$ 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 17.

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

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 1 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 $y_2^e m_b$ and c_3^* as $H(y_2)$.

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 18. *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. Chooses e s.t. $\gcd(e, \varphi(N)) = 1$, e.g. $e := 3$.
2. 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 $(\mathsf{crs}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{pp} := (N, T(\lambda), g, h_1, h_2, \mathsf{crs}, e, H)$. Notice that value h_2 is computed by repeated squaring.
3. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$ and answers decommitment queries using k_1.
4. Outputs $(N, g, k_1 h_1, h_2, \mathsf{crs}, e, H, \tau, m_0, m_1, \mathsf{st})$

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

1. Samples $b \xleftarrow{\$} \{0, 1\}, y_1 \xleftarrow{\$} \mathbb{QR}_N$, computes $c_0^* := x, c_1^* := y_1 m_b, c_2^* := y^e m_b$.
2. Runs $\pi^* \leftarrow \mathtt{NIZK}.\mathsf{Sim}_2(\mathsf{crs}, (c_0^*, c_1^*, c_2^*, c_3^*), \tau)$.
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 $\mathbb{QR}_N$ 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.

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 19.

$$\Pr[\mathsf{G}_8 = 1] \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{RSA}} + \frac{1}{2}.$$

Let E be an event that adversary $\mathcal{A}$ asks a random oracle query for value y. We show that if E happens, then we can build $\mathcal{B}$ against RSA assumption. On the other hand, if E does not happen, then $\mathcal{A}$ does not have any information about committed message.

At first assume that E does not happen. Since $H(y_2)$ is a uniform random element over the range of the hash function, it does not reveal any information about y_2 and does not contain any information about m_b. 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 $\mathbb{QR}_N$ and $H(y_2)$ does not reveal any information about y_2, 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 with probability $1/2$.

If E happens, then we build $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ against RSA.

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

1. Samples $k_1 \xleftarrow{\$} [\lfloor N/4 \rfloor]$, computes $h_1 := g^{k_1} \bmod N, h_2 := g^{2^T} \bmod N$, runs $(\mathsf{crs}, \tau) \leftarrow \mathtt{NIZK}.\mathsf{Sim}_1(1^\lambda, L)$ and sets $\mathsf{pp} := (N, g, h_1, h_2, \mathsf{crs}, e, H)$. Notice that value h_2 is computed by repeated squaring.

2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pp})$ and answers decommitment queries using k_1. It answers random oracle queries x by lazy sampling. If for any x holds $x^e = y \bmod N$, then it returns x.
3. Samples $b \xleftarrow{\$} \{0,1\}, c_0^*, y_1 \xleftarrow{\$} \mathbb{QR}_N$ and computes $c_1^* := y_1 m_b, c_2^* := y m_b, c_3^* \xleftarrow{\$} \mathcal{V}$, where $\mathcal{V}$ is the image of the RO. Runs to simulation oracle $\mathtt{NIZK}.\mathsf{Sim}_2(\mathsf{crs}, (c_0^*, c_1^*, c_2^*, c_3^*), \tau)$ 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_1. RO queries are answered by lazy sampling and consistently with previous RO queries. If for any query x holds $x^e = y \bmod N$, then it returns x.

Now notice that if E happens, then $\mathcal{B}$ indeed outputs x such that $x^e = y \bmod N$. Therefore $\Pr[\mathsf{E}] = \mathbf{Adv}_{\mathcal{B}}^{\mathsf{RSA}}$. Hence,

$$\begin{aligned}\Pr[\mathsf{G}_8 = 1] &= \Pr[\mathsf{G}_8 = 1 \wedge \mathsf{E}] + \Pr[\mathsf{G}_8 = 1 \wedge \overline{\mathsf{E}}] \\ &\leq \Pr[\mathsf{E}] + \Pr[\mathsf{G}_8 = 1 \wedge \overline{\mathsf{E}}] = \mathbf{Adv}_{\mathcal{B}}^{\mathsf{RSA}} + \frac{1}{2}.\end{aligned}$$

Notice that at the same time $\Pr[\mathsf{G}_8 = 1] \geq \frac{1}{2}$, because $\mathcal{A}$ can always do random guessing independent of happening E. Therefore $\left|\Pr[\mathsf{G}_8 = 1] - \frac{1}{2}\right| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{RSA}}$. By combining Lemmas 11 - 19 we obtain the following:

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

which concludes the proof.

Theorem 4. $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec})$ *defined in Figure 7 is a BND-CCA-secure non-interactive timed commitment scheme.*

Proof. Our adjusted construction remains perfectly binding and this can be argued in the same way as it was argued in [CJ23]. Since ElGamal encryption is perfectly binding, 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 hence 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 5. *The NITC* $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}, \mathsf{FDecVrfy}, \mathsf{Eval})$ *defined in Figure 7 is a multiplicatively homomorphic non-interactive timed commitment scheme.*

Theorem 6. *If* $\mathtt{NIZK} = (\mathtt{NIZK.Prove}, \mathtt{NIZK.Vrfy})$ *is a non-interactive zero-knowledge proof system for* L *and* H *is a collision resistant hash function, then* $(\mathsf{PGen}, \mathsf{Com}, \mathsf{ComVrfy}, \mathsf{DecVrfy}, \mathsf{FDec}, \mathsf{FDecVrfy})$ *defined in Figure 7 is a publicly verifiable non-interactive timed commitment scheme.*

Proof. Completeness is straightforward to verify.

To prove the soundness lets assume that $\mathcal{A}$ provides m' such that $m \neq m'$. Since $\mathsf{FDecVrfy}$ outputs 1 and therefore there are two different y, y' such that $c_2 = y^e m \bmod N \wedge c_3 = (y')^e m \bmod N$. There are two cases to consider:

1. $H(y) = H(y')$. We denote this event by COLL. Notice if COLL happens then we can break a collision resistance of the hash function. Adversary provides y' as the proof π_{FDec} and y can be computed by repeated squaring.
2. COLL does not happen. This however means that $c_3 = H(m')$ and $c_2 = h^{re} m \bmod N$ which breaks the soundness of the NIZK.

Therefore we can conclude that both cases are negligible. Let PVSND denotes the event that $\mathcal{A}$ break soundness of the construction. Then

$$\begin{aligned}\Pr[\mathsf{PVSND}] &= \Pr[\mathsf{PVSND} \wedge \mathsf{COLL}] + \Pr[\mathsf{PVSND} \wedge \overline{\mathsf{COLL}}] \\ &\leq \Pr[\mathsf{COLL}] + \Pr[\mathsf{PVSND} \wedge \overline{\mathsf{COLL}}] \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{CRHF}} + \mathbf{Snd}_{\mathcal{B}}^{\mathtt{NIZK}}.\end{aligned}$$

Instantiating NIZK . Since the language L for our constructions is not purely algebraic we have to rely on NIZK for general relations and one such a option is simulation-extractable SNARK of Groth and Maller [GM17]. Its security is based on the assumptions in bilinear groups and its independent of the knowledge of discrete logarithms of elements h_1, h_2 that specify the language L. Therefore zero-knowledge and simulation-extractability, and hence also simulation-soundness and soundness, holds even if we provide this information as an auxiliary input to an adversary.

Remark 3. We note that since language NIZK proof system has to prove the claim which includes a hash function modelled as a random oracle, the whole construction is only heuristically secure.

References

BBBF18. Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

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: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 62–80, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.

CJ23. Peter Chvojka and Tibor Jager. Simple, fast, efficient, and tightly-secure non-malleable non-interactive timed commitments. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 500–529, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.

DGMV20. Nico Döttling, Sanjam Garg, Giulio Malavolta, and Prashant Nalini Vasudevan. Tight verifiable delay functions. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 65–84, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany.

DMPS19. Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 248–277, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

EFKP20. Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 581–612, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

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: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 390–413, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

LNPY22. Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. One-shot fiat-shamir-based NIZK arguments of composite residuosity and logarithmic-size ring signatures in the standard model. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 488–519, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

MT19. Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 620–649, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

Pie19. Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019: 10th Innovations in Theoretical Computer Science Conference*, volume 124, pages 60:1–60:15, San Diego, CA, USA, January 10–12, 2019. LIPIcs.

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: 28th Conference on Computer and Communications Security*, pages 2663–2684, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

Wes19. Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 379–407, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.