Towards Practical Homomorphic Time-Lock Puzzles: Applicability and Verifiability

Yi Liu[1,3], Qi Wang[1,2], and Siu-Ming Yiu[3]

[1] Research Institute of Trustworthy Autonomous Systems & Guangdong Provincial
Key Laboratory of Brain-inspired Intelligent Computation,
Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
liuy7@mail.sustech.edu.cn
wangqi@sustech.edu.cn
[2] National Center for Applied Mathematics (Shenzhen),
Southern University of Science and Technology, Shenzhen 518055, China
[3] Department of Computer Science,
The University of Hong Kong, Pokfulam, Hong Kong SAR, China
smyiu@cs.hku.hk

Abstract. Time-lock puzzle schemes allow one to encrypt messages for the future. More concretely, one can efficiently generate a time-lock puzzle for a secret/solution s, such that s remains hidden until a specified time T has elapsed, even for any parallel adversaries. However, since computation on secrets within multiple puzzles can be performed only when *all* of these puzzles are solved, the usage of classical time-lock puzzles is greatly limited. Homomorphic time-lock puzzle (HTLP) schemes were thus proposed to allow evaluating functions over puzzles directly without solving them.

However, although efficient HTLP schemes exist, more improvements are still needed for practicability. In this paper, we improve HTLP schemes to broaden their application scenarios from the aspects of *applicability* and *verifiability*. In terms of applicability, we design the *first* multiplicatively HTLP scheme with the solution space over $\mathbb{Z}_n^*$, which is more expressible than the original one, *e.g.*, representing integers. Then, to fit HTLP into scenarios requiring verifiability that is missing in existing schemes, we propose three *simple* and *fast* protocols for both the additively HTLP scheme and our multiplicatively HTLP scheme, respectively. The first two protocols allow a puzzle solver to convince others of the correctness of the solution or the invalidity of the puzzle so that others do not need to solve the puzzle themselves. The third protocol allows a puzzle generator to prove the validity of his puzzles. It is shown that a puzzle in our scheme is only 1.25KB, and one multiplication on puzzles takes simply 0.01ms. Meanwhile, the overhead of each protocol is less than 0.6KB in communication and 40ms in computation. Hence, HTLP still demonstrates excellent efficiency in both communication and computation with these versatile properties.

Keywords: Public-key cryptography · (Homomorphic) time-lock puzzles · Repeated modular squaring · Zero-knowledge.

1 Introduction

The notion of time-lock puzzle [18] was proposed in 1996 and has become a widely used cryptographic primitive. Time-lock puzzle schemes allow one to *efficiently* encrypt a secret/solution[4] s for the future, such that s remains hidden until a specified time T has elapsed, even for parallel adversaries. In other words, an efficiently generated time-lock puzzle can force a solver to complete a computational task that costs no less than time T to recover the solution. With this property, time-lock puzzles lead to tremendous applications, such as sealed-bid auction [18], fair contract signing [3], zero-knowledge argument [7], non-malleable commitment [13], publicly verifiable covert security [8,19].

However, the performance of classical time-lock puzzle schemes within large-scale protocols is far from satisfactory. For example, we may need to perform computation on secrets within many puzzles. This computation can only be done after all puzzles are solved, leading to colossal computation overhead.

In 2019, Malavolta and Thyagarajan [15] introduced a new notion called homomorphic time-lock puzzles (HTLP). Using HTLP schemes, we can perform computation on hidden secrets within puzzles and derive the computation result by solving only the resulting puzzle. They proposed two partially HTLP schemes: an additively HTLP scheme with the solution space $\mathbb{Z}_n$ and a multiplicatively HTLP scheme with the solution space $\mathbb{J}_n$. Here n is a strong RSA modulus, and $\mathbb{J}_n = \{x \in \mathbb{Z}_n^* \mid J_n(x) = 1\}$, where $J_n(x)$ denotes the Jacobi symbol of x. Then they showed that with the help of the homomorphic property, these two efficient schemes could be used in several scenarios, such as e-voting, multi-party coin flipping, and fair contract signing. Moreover, HTLP, as a key component, has recently been involved in several cryptographic protocols, such as verifiable timed signatures [22], fair and sound secret sharing [12].

Nevertheless, for the two partially HTLP schemes, limitations in terms of practicability still exist.

- The solution space of the multiplicatively HTLP scheme in [15] is the multiplicative group $\mathbb{J}_n$. However, the solution space $\mathbb{Z}_n^*$ is preferred than $\mathbb{J}_n$, since $\mathbb{Z}_n^*$ is more expressible in representation, such as for integers. In addition, we note that some RSA-related schemes, *e.g.*, the threshold RSA signature scheme [21], are over $\mathbb{Z}_n^*$, and they require a HTLP scheme with solution space $\mathbb{Z}_n^*$ for the multi-party contract signing paradigm introduced in [15].
- It is common for HTLP-enabled applications (*e.g.*, [15,22]) that all parties perform the same computation on puzzles and derive an identical solution. However, even if the identical solution has been obtained by one party, there is a lack of methods for others to quickly verify the correctness of this solution and thus avoid solving puzzles.
- For the additively HTLP schemes in [15], malicious parties may generate invalid puzzles, which also invalidate the resulting puzzle, while solvers can know this fact only after time T. Hence, to reach a consensus that a puzzle is invalid, all parties have to carry out the time-T puzzle-solving process.

[4] In this paper, “secret” and “solution” are the same concept and used interchangeably.

1.1 Our Results

In this work, we provide the following practical solutions to address the aforementioned limitations of the HTLP schemes [15].

Applicability We propose the *first* multiplicatively HTLP scheme with the more expressible solution space $\mathbb{Z}_n^*$ for a strong RSA modulus n.

Verifiability To avoid the redundant cost of the puzzle-solving process, we provide three *simple* and *fast* protocols for both the additively HTLP scheme with the solution space $\mathbb{Z}_n$ in [15] and our multiplicatively HTLP scheme, respectively, to verify the following three properties.

(1) *Correctness.* A puzzle solver is able to convince other parties of the *correctness of the solution* that he solves from a puzzle.

(2) *Invalidity.* Upon finding that a puzzle is invalid, one can convince other parties of the *invalidity of the puzzle.*

(3) *Validity.* A puzzle generator can convince other parties of the *validity of the puzzle* he generated.

Note that all of these protocols can be compiled by Fiat-Shamir heuristic [9] to be non-interactive. The first two protocols enable parties to outsource the puzzle-solving process to one party. Moreover, the second one can be combined with smart contracts to penalize malicious puzzle generators via such proofs of invalid puzzles. Furthermore, puzzle generators can use the third protocol to prove their innocence at the beginning of the puzzle generation.

According to our implementations and analysis in Section 5, a puzzle of the new scheme is only 1.25KB, and one multiplication on puzzles costs only 0.01ms. Meanwhile, each of the three properties has less than 0.6KB communication overhead and less than 40ms computation overhead. Therefore, equipped with applicability and verifiability, the application scenarios of HTLP can be greatly broadened, and HTLP schemes are thereby more practical.

1.2 Technical Overview

The original multiplicatively HTLP scheme with the solution space $\mathbb{J}_n$ can be regarded as a combination of the ElGamal encryption scheme [10] over $\mathbb{J}_n$ and the RSW time-lock puzzle scheme [18]. Roughly speaking, the puzzle for a solution $s \in \mathbb{J}_n$ is of the form $Z = (u = g^r, v = h^r s) \in \mathbb{J}_n^2$ for a random r and public parameters $(g, h = g^{2^T}) \in \mathbb{J}_n^2$. Then s is recovered via $s \leftarrow vu^{-2^T}$ by repeated squaring of u for T times. Intuitively, the reason why the secret $s \in \mathbb{J}_n$ can be hidden is similar to the semantic security of the ElGamal encryption scheme based on the decisional Diffie-Hellman (DDH) assumption over $\mathbb{J}_n$. However, extending the solution space to $\mathbb{Z}_n^*$ is highly nontrivial, since the DDH assumption over $\mathbb{Z}_n^*$ does not hold. More concretely, given $v = h^r s$ for $s \in \mathbb{Z}_n^*$, we can easily learn the Jacobi symbol of s from $J_n(v)$. We overcome this barrier by encoding $s \in \mathbb{Z}_n^*$ using elements in $\mathbb{J}_n$. Given an element $\chi \in \mathbb{Z}_n^* \backslash \mathbb{J}_n$, $s \in \mathbb{Z}_n^*$ can be encoded by $(m = s\chi^\sigma, \sigma) \in \mathbb{J}_n \times \{0, 1\}$, where $\sigma = (-J_n(s) + 1)/2$, *i.e.*, $\sigma = 0$ if $J_n(s) = 1$ and $\sigma = 1$ if $J_n(s) = -1$. Since $m \in \mathbb{J}_n$, we can use the original multiplicatively

HTLP scheme for $\mathbb{J}_n$ to encrypt m, such that the multiplicatively homomorphic property for m is preserved. Similarly, σ can be carried by an additively HTLP, such that addition of σ is supported. Now a puzzle in this new scheme is a tuple of two puzzles carrying m and σ, respectively. The multiplication of secrets within puzzles of this new scheme is performed entry-wise for multiplication and addition for the two puzzles inside, respectively. To solve a puzzle in the new scheme, we solve the two inside puzzles to obtain $m = \prod_i m_i = \prod_i s_i \chi^{\sum_i \sigma_i}$ and $\sigma = \sum_i \sigma_i$, and the resulting secret $s = \prod_i s$ can be easily derived.

The first two properties (solution correctness and puzzle invalidity) for verifiability share a similar idea to verifiable delay function (VDF) [1]. VDF is a function that needs a prescribed time to compute, even for parallel computers, while anyone can quickly verify the output. For the VDF scheme in [23], given a tuple (g, h, T), a VDF evaluator can generate a proof for the statement $h = g^{2^T}$. This pattern is also adopted in time-lock puzzle schemes over RSA groups. Hence, a puzzle solver can utilize this technique to help verifiers speed up the puzzle-solving process to verify these two properties.

The third property (puzzle validity) requires zero knowledge to preserve secrets while proving puzzle validity. At first glance, it seems that this is similar to classical zero-knowledge protocols for the knowledge of discrete logarithms. However, since puzzles of the two HTLP schemes are over $\mathbb{Z}_n^*$ and $\mathbb{Z}_{n^2}^*$, where n is an RSA modulus, the order of these two groups are hidden from parties, and thus protocols and security proofs are *much more involved.* We solve this problem by carefully choosing the value interval of related parameters and proving the zero-knowledge property in a statistical sense. Then to extract the witness in the security proof, we perform division of exponents directly over integers rather than modulo the order and then show that it is divisible unless the prover breaks the strong RSA assumption (see the details in Section 4.2).

1.3 Related Work

Besides the two partially HTLP schemes, a fully HTLP scheme based on indistinguishability obfuscation was proposed in [15], and another one based on fully homomorphic encryption was given in [4]. They are both based on costly primitives and are mostly of theoretical interest at present.

Very recently, a generic construction of HTLP schemes was proposed in [5]. This construction uses existing classical time-lock puzzle schemes and homomorphic encryption schemes in a black-box manner. Its setup algorithm generates a key pair of the homomorphic encryption scheme, together with a time-lock puzzle for the random coins used in the key generation, and outputs the public key and puzzle as public parameters. Then homomorphic puzzles of this construction are ciphertexts encrypting secrets via the public key. Parties can solve the puzzle for random coins, derive the private key from random coins, and then decrypt puzzles (ciphertexts) using the private key. We remark that the setup is for one-time use, and all secrets are revealed after time T from the setup. Hence, it can only be applied to scenarios where all puzzles are generated simultaneously, and public parameters should be periodically re-initialized. Moreover, we often require a multi-party protocol to perform the setup, which is costly for one-time use and complicated to prevent malicious parties from obtaining public parameters in advance to gain advantages. Alternatively, HLTP schemes in [15] only need *one* setup of public parameters, and a secret within a puzzle is hidden for time T, starting from the generation of that puzzle.

2 Preliminaries

We write $x \leftarrow_{\$} S$ for uniformly sampling an element x from the set S and use $[a, b]$ for integers a and b to denote the set $\{a, a+1, \ldots, b-1\}$. The variable κ represents the computational security parameter. A non-negative function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is called *negligible* if $\mathsf{negl}(\kappa) = o(\kappa^{-c})$ for every constant $c > 0$. We say that $1 - f$ is *overwhelming* if the function f is negligible. We call an integer $n = pq$ a strong RSA modulus, where p and q are distinct safe primes having equal length. Here primes p and q are called safe if $p = 2p' + 1$ and $q = 2q' + 1$ for primes p' and q', respectively. Let $\mathbb{J}_n = \{x \in \mathbb{Z}_n^* \mid J_n(x) = 1\}$, where $J_n(x)$ is the Jacobi symbol of x. If n is a strong RSA modulus, then $\mathbb{J}_n$ is a cyclic multiplicative group with order $\phi(n)/2 = 2p'q'$. Let $g_0 \leftarrow_{\$} \mathbb{Z}_n^*$ and $g \leftarrow -g_0^2$, then g is a generator of $\mathbb{J}_n$ with overwhelming probability. For the RSA modulus n, $\lceil n/2 \rceil$ is statistically close to $2p'q'$. The security of protocols in this paper is proved under standard security definitions (see [14] for more information). Computational assumptions involved in this paper are summarized in Appendix A. We state Lemma 1 as follows, and its proof is in Appendix C.1. This lemma is extensively used in the proof of Theorem 4 (for the property "puzzle validity") to prove the security of the protocol under the strong RSA assumption via reduction.

Lemma 1. *For an RSA instance (n, e, y), finding $x \in \mathbb{Z}_n^*$ and e' coprime to e, such that $x^e \equiv y^{e'}$ is equivalent to finding an e-th root of y modulus n.*

3 Homomorphic Time-Lock Puzzles

We use the definition of HTLP scheme in [15] (see also Appendix B) for our scheme. Briefly, an HLTP scheme achieves the homomorphic property for a class of functions/circuits and should be *compact*, *i.e.*, puzzles have the same length, and homomorphic operations do not depend on the time parameter T while the time of puzzle-solving only depends on T. Of course, the scheme should be *correct*, *i.e.*, solving a puzzle generated by the puzzle generation algorithm with the input of a secret s should obtain s.

3.1 Additively HTLP Scheme with Solution Space $\mathbb{Z}_n$

First, we briefly restate the additively HTLP scheme with the solution space $\mathbb{Z}_n$ in [15]. Its protocols with respect to verifiability are given in Section 4.2. We denote the circuit class of the additively HTLP scheme by $+_n$, which indicates the circuit for addition in $\mathbb{Z}_n$. For example, let $f : \mathbb{Z}_n^N \to \mathbb{Z}_n$ be $f(s_1, \ldots, s_N) =$

$s_1 + \cdots + s_N \bmod n$, where variables $s_i \in \mathbb{Z}_n$ for $1 \leq i \leq N$, and N can be arbitrary polynomials in κ. Then $f \in +_n$.

- $\mathsf{Setup}(1^\kappa, T)$: On input 1^κ and a time parameter T, sample a strong RSA modulus $n = pq$, where $p = 2p' + 1$ and $q = 2q' + 1$. Then pick a random generator g of $\mathbb{J}_n$ and compute $h \leftarrow g^{2^T} \bmod n$, where $2^T \bmod 2p'q'$ can be computed first to speed up the computation. Finally, output the public parameter $pp = (T, n, g, h)$.
- $\mathsf{Gen}(pp, s)$: On input pp and a secret $s \in \mathbb{Z}_n$, pick $r \leftarrow_{\$} [0, \lceil n/2 \rceil]$, compute $u \leftarrow g^r \bmod n$ and $v \leftarrow h^{r \cdot n} \cdot (1+n)^s \bmod n^2$, and output a puzzle $Z = (u, v)$.
- $\mathsf{Solve}(pp, Z)$: On input pp and a puzzle $Z = (u, v)$, compute $w \leftarrow u^{2^T} \bmod n$ by repeated squaring. Then compute $s \leftarrow \frac{[v/w^n \bmod n^2]-1}{n}$. If $s \in \mathbb{Z}_n$, output s. Otherwise, output $\perp$ to indicate that Z is invalid.
- $\mathsf{Eval}(pp, f, Z_1, \ldots, Z_N)$: On input pp, function $f \in +_n$, and a set of puzzle $(Z_i)_{i=1,\ldots,N}$, do the following two steps.
  1. Parse $pp = (T, n, g, h)$, $Z_i = (u_i, v_i)$ for $i \in 1, \ldots, N$.
  2. Output the resulting puzzle $Z = (u, v)$, where $u = \prod_{i=1}^N u_i \bmod n$ and $v = \prod_{i=1}^N v_i \bmod n^2$.

It is easy to verify the additively homomorphic property. Given two puzzles $Z_1 = (u_1 = g^{r_1}, v_1 = h^{r_1 \cdot n}(1+n)^{s_1})$ and $Z_2 = (u_2 = g^{r_2}, v_2 = h^{r_2 \cdot n}(1+n)^{s_2})$ for secrets s_1 and s_2, respectively, the puzzle $Z_3 = (u_1 u_2, v_1 v_2) = (g^{r_1+r_2}, h^{(r_1+r_2)n}(1 + n)^{s_1+s_2}) \in \mathbb{J}_n \times \mathbb{Z}_{n^2}^*$ is the puzzle for the secret $s_3 = s_1 + s_2 \bmod n$ and $\mathsf{Solve}(pp, Z_3) = s_3$.

3.2 Our Multiplicatively HTLP Scheme with Solution Space $\mathbb{Z}_n^*$

We now present our multiplicatively HTLP scheme MHTLP with the solution space $\mathbb{Z}_n^*$, following the idea in Section 1.2. We denote the circuit class of our multiplicatively HTLP scheme by $\times_n$, which indicate the circuit for multiplication of (less than n) elements in $\mathbb{Z}_n$. For example, let $f : {\mathbb{Z}_n^*}^N \rightarrow \mathbb{Z}_n^*$ be $f(s_1, \ldots, s_N) = \prod_i^N s_i \bmod n$, where variables $s_i \in \mathbb{Z}_n$ for $1 \leq i \leq N$, and we need to ensure that $N < n$. Then $f \in \times_n$.

- $\mathsf{Setup}(1^\kappa, T)$: On input 1^κ and a time parameter T, sample a strong RSA modulus $n = pq$, pick a random generator g of $\mathbb{J}_n$ and $\chi \leftarrow_{\$} \mathbb{Z}_n^*$ for $J_n(\chi) = -1$, compute $h \leftarrow g^{2^T} \bmod n$, and output $pp = (T, n, g, h, \chi)$.
- $\mathsf{Gen}(pp, s)$, On input pp and a secret $s \in \mathbb{Z}_n^*$, sample $r \leftarrow_{\$} [0, \lceil n/2 \rceil]$ and $r' \leftarrow_{\$} [0, \lceil n/2 \rceil]$, and then compute $u \leftarrow g^r \bmod n$, $u' \leftarrow g^{r'} \bmod n$, and $\sigma = (-J_n(s)+1)/2$. Given σ, compute $v \leftarrow h^r \cdot \chi^\sigma s \bmod n$ and $\theta \leftarrow h^{r' \cdot n}(1+n)^\sigma \bmod n^2$. Finally, output a puzzle $Z = (u, u', v, \theta)$.
- $\mathsf{Solve}(pp, Z)$: On input pp and a puzzle $Z = (u, u', v, \theta)$, compute $w \leftarrow u^{2^T} \bmod n$ and $w' \leftarrow {u'}^{2^T} \bmod n$ by repeated squaring. Then compute $d \leftarrow \frac{[\theta/{w'}^n \bmod n^2]-1}{n}$. If $d \notin \mathbb{Z}_n$, output $\perp$ to indicate that Z is invalid. Otherwise, output the solution $s \leftarrow v w^{-1} \chi^{-d} \bmod n$.

- $\mathsf{Eval}(pp, f, Z_1, \ldots, Z_N)$: On input pp, a function $f \in \times_n$, and a set of puzzle $(Z_i)_{i=1,\ldots,N}$, do the following two steps.
  1. Parse $pp = (T, n, g, h, \chi)$, $Z_i = (u_i, u'_i, v_i, \theta_i)$ for $i \in 1, \ldots, N$.
  2. Output $Z = (u, u', v, \theta)$, where $u = \prod_{i=1}^{N} u_i \bmod n$, $u' = \prod_{i=1}^{N} u'_i \bmod n$, $v = \prod_{i=1}^{N} v_i \bmod n$, and $\theta = \prod_{i=1}^{N} \theta_i \bmod n^2$.

It is straightforward to verify the multiplicatively homomorphic property. For instance, given puzzles $Z_i = (u_i, u'_i, v_i, \theta_i) = (g^{r_i}, g^{r'_i}, h^{r_i} \cdot \chi^{\sigma_i} s_i, h^{r'_i \cdot n}(1+n)^{\sigma_i}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^*$ for the secret s_i for $i = 1, \ldots, N$, the puzzle Z defined as

$$
\begin{aligned}
Z &= (\prod_{i=1}^{N} u_i, \prod_{i=1}^{N} u'_i, \prod_{i=1}^{N} v_i, \prod_{i=1}^{N} \theta_i) = (\prod_{i=1}^{N} g^{r_i}, \prod_{i=1}^{N} g^{r'_i}, \prod_{i=1}^{N} h^{r_i} \chi^{\sigma_i} s_i, \prod_{i=1}^{N} h^{r'_i n}(1+n)^{\sigma_i}) \\
&= (g^{\sum_{i=1}^{N} r_i}, g^{\sum_{i=1}^{N} r'_i}, h^{\sum_{i=1}^{N} r_i} \chi^{\sum_{i=1}^{N} \sigma_i} \prod_{i=1}^{N} s_i, h^{n \cdot \sum_{i=1}^{N} r'_i}(1+n)^{\sum_{i=1}^{N} \sigma_i}) \\
&= (g^r, g^{r'}, h^r \chi^\sigma \prod_{i=1}^{N} s_i, h^{n \cdot r'}(1+n)^\sigma) = (u, u', v, \theta) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^* \,, \qquad (1)
\end{aligned}
$$

where $r = \sum_{i=1}^{N} r_i$, $r' = \sum_{i=1}^{N} r'_i$, and $\sigma = \sum_{i=1}^{N} \sigma_i$, is the puzzle for the secret $s = \prod_{i=1}^{N} s_i \bmod n$, since $\mathsf{Solve}(pp, Z) = s$. Note that this scheme supports homomorphic multiplication for $N < n$. Since n is exponentially large in κ, assuming $N < n$ does not limit practical usage. Here we recall the definition of reusable security for HTLP schemes and present Theorem 1 for our scheme.

Definition 1 ([15]). *An HTLP scheme* (Setup, Gen, Solve) *with a solution space $\mathbb{S}$ is reusable-secure with gap $\varepsilon < 1$ if there exists a polynomial $\tilde{T}(\cdot)$, such that for all polynomial $T(\cdot) \geq \tilde{T}(\cdot)$ and all polynomial-size adversaries $(\mathcal{A}_1, \mathcal{A}_2) = \{(\mathcal{A}_1, \mathcal{A}_2)_\kappa\}_{\kappa \in \mathbb{N}}$, where the depth of $\mathcal{A}_2$ is bounded from above by $T^\varepsilon(\kappa)$, we have*

$$
\Pr\left[b \leftarrow \mathcal{A}_2(Z, \tau) : \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\kappa, T(\kappa)); \\ (\tau, s_0, s_1) \leftarrow \mathcal{A}_1(pp); \\ b \leftarrow_{\$} \{0,1\}; Z \leftarrow \mathsf{Gen}(pp, s_b); \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\kappa) \,,
$$

where $s_0, s_1 \in \mathbb{S}$.

Theorem 1. MHTLP *is a reusable-secure HTLP scheme with the solution space $\mathbb{Z}_n^*$ supporting homomorphic multiplication of $N(< n)$ elements in n.*

Proof. The correctness is straightforward. For the puzzle $Z = (u, u', v, \theta)$ as in (1) above, the Solve algorithm computes $w \equiv u^{2^T} \equiv g^{2^T r} \equiv h^r \pmod{n}$ and $w' \equiv u'^{2^T} \equiv g^{2^T r'} \equiv h^{r'} \pmod{n}$. Then the algorithm computes $\theta / w'^n \bmod n^2 = (1+n)^\sigma \bmod n^2 = 1 + \sigma n$, where $\sigma = \sum_{i=1}^{N} \sigma_i < n$. Hence, we can derive $d = \frac{[\theta / w'^n \bmod n^2] - 1}{n} = \sigma$ and $s \equiv \sum_{i=1}^{N} s_i \equiv v w^{-1} \chi^{-d} \pmod{n}$.

The compactness is evident. It is easy to verify that the length of puzzles derived from homomorphic operations is the same as that of a puzzle output by

Gen and does not depend on the number of multipliers N. The running time of the Solve algorithm does not depend on N, either. Moreover, the running time of homomorphic multiplications does not depend on the time parameter T.

Finally, we prove the reusable security by the following sequence of hybrids.

Hybrid$_0$ This is the original experiment in Definition 1.

Hybrid$_1$ The element v now is generated via $v \leftarrow_\$ \mathbb{J}_n$ instead of computing $v \leftarrow h^r \cdot \chi^\sigma s \bmod n$.

We claim that **Hybrid**$_1$ is indistinguishable from **Hybrid**$_0$ for adversaries. Otherwise, given a distinguisher $(\mathcal{A}_1, \mathcal{A}_2)$ for **Hybrid**$_0$ and **Hybrid**$_1$, where the depth of $\mathcal{A}_2$ is less than T, we can construct the following polynomial-size adversary $(\mathcal{D}_1, \mathcal{D}_2)$ to break the strong sequential squaring assumption.

Upon receiving (n, g, T), $\mathcal{D}_1$ computes $h \leftarrow g^{2^T} \bmod n$, picks $\chi \leftarrow_\$ \mathbb{Z}_n^*$ for $J_n(\chi) = -1$, and sets $pp \leftarrow (T, n, g, h, \chi)$. Then $\mathcal{D}_1$ runs $\mathcal{A}_1(pp)$, obtains (τ, s_0, s_1) from $\mathcal{A}_1$, and outputs $\tau' = (h, \tau, s_0, s_1)$. Then $\mathcal{D}_2$ will be invoked with input (x, y, τ'). $\mathcal{D}_2$ sets $u \leftarrow x$, $v \leftarrow y \cdot \chi^{\sigma_b} s_b$, $u' \leftarrow g^{r'} \bmod n$, and $\theta \leftarrow h^{r' \cdot n}(1+n)^{\sigma_b} \bmod n^2$ for $r' \leftarrow_\$ [0, \lceil n/2 \rceil]$ and $b \leftarrow_\$ \{0,1\}$. Here $\sigma_b = (-J_n(s_b)+1)/2$. Then $\mathcal{D}_2$ invokes $\mathcal{A}_2$ with input $Z = (u, u', v, \theta)$ and τ, and outputs whatever $\mathcal{A}_2$ outputs. Note that $\mathcal{D}_1$ is efficient since T is a polynomial, and the depth of $\mathcal{D}_2$ is identical (up to a constant factor) to that of $\mathcal{A}_2$. We have the following two cases for this reduction.

- The case $y = x^{2^T} \bmod n$: Since g is a generator of $\mathbb{J}_n$ and $x \in \mathbb{J}_n$, there exists r, such that $x = g^r \bmod n$. Then the puzzle is of the form:

$$\begin{aligned} Z = (u, u', v, \theta) &= (x, g^{r'}, x^{2^T} \cdot \chi^{\sigma_b} s_b, h^{r' \cdot n}(1+n)^{\sigma_b}) \\ &= (g^r, g^{r'}, h^r \cdot \chi^{\sigma_b} s_b, h^{r' \cdot n}(1+n)^{\sigma_b}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^*\,, \end{aligned}$$

  which is identically distributed as that in **Hybrid**$_0$.
- The case $y \leftarrow_\$ \mathbb{J}_n$: The puzzle in this case is of the form:

$$\begin{aligned} Z = (u, u', v, \theta) &= (x, g^{r'}, y \cdot \chi^{\sigma_b} s_b, h^{r' \cdot n}(1+n)^{\sigma_b}) \\ &= (g^r, g^{r'}, z, h^{r' \cdot n}(1+n)^{\sigma_b}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^*\,, \end{aligned}$$

  where $z = y \cdot \chi^{\sigma_b} s_b \bmod n$ can be regarded as a random element in $\mathbb{J}_n$ since y is independently random. Hence, Z is identically distributed as that in **Hybrid**$_1$.

Therefore, $(\mathcal{D}_1, \mathcal{D}_2)$ wins the strong sequential squaring experiment with probability significantly higher than $\frac{1}{2}$, contradicting to the assumption.

Hybrid$_2$ The element θ is generated via $\theta \leftarrow \rho^n(1+n)^\sigma \bmod n^2$ for $\rho \leftarrow_\$ \mathbb{J}_n$ instead of computing $\theta \leftarrow h^{r'}(1+n)^\sigma$.

We claim that **Hybrid**$_2$ is indistinguishable from **Hybrid**$_1$ for adversaries. Otherwise, given a distinguisher $(\mathcal{A}_1, \mathcal{A}_2)$ where the depth of $\mathcal{A}_2$ is less than T, we can construct a polynomial-size adversary $(\mathcal{D}_1, \mathcal{D}_2)$ to break the strong sequential squaring assumption as follows. $\mathcal{D}_1$ receives (n, g, T), generates the public parameter pp, and derives $\tau' = (h, \tau, s_0, s_1)$ as that for the reduction in $\textbf{Hybrid}_1$. Then $\mathcal{D}_2$ is invoked with (x, y, τ'). $\mathcal{D}_2$ sets u and v as the experiment of $\textbf{Hybrid}_1$. Let $u' \leftarrow x$ and $\theta \leftarrow y^n(1+n)^{\sigma_b} \bmod n$ for $b \leftarrow_{\$} \{0, 1\}$. Here $\sigma_b = (-J_n(s_b)+1)/2$. Then $\mathcal{D}_2$ invokes $\mathcal{A}_2$ with input $Z = (u, u', v, \theta)$ and τ, and outputs whatever $\mathcal{A}_2$ outputs. Note that $\mathcal{D}_1$ is efficient since T is a polynomial, and the depth of $\mathcal{D}_2$ is identical (up to a constant factor) to that of $\mathcal{A}_2$. We have the following two cases for this reduction.

- The case $y = x^{2^T} \bmod n$: Since g is a generator of $\mathbb{J}_n$ and $x \in \mathbb{J}_n$, there exists r', such that $x = g^{r'} \bmod n$. Then the puzzle is of the form:

$$\begin{aligned} Z = (u, u', v, \theta) &= (u, x, v, y^n(1+n)^{\sigma_b}) = (u, x, v, x^{2^T n}(1+n)^{\sigma_b}) \\ &= (u, g^{r'}, v, h^{r' \cdot n}(1+n)^{\sigma_b}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^* \,. \end{aligned}$$

  Hence, Z is identically distributed as that in $\textbf{Hybrid}_1$.
- The case $y \leftarrow_{\$} \mathbb{J}_n$: The puzzle in this case is of the form:

$$Z = (u, u', v, \theta) = (u, g^{r'}, v, z^n(1+n)^{\sigma_b}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^* \,,$$

  where $z \leftarrow_{\$} \mathbb{J}_n$. Hence, Z is identically distributed as that in $\textbf{Hybrid}_2$.

Therefore, $(\mathcal{D}_1, \mathcal{D}_2)$ wins the strong sequential squaring experiment with probability significantly higher than $\frac{1}{2}$, which contradicts to the assumption.

$\textbf{Hybrid}_3$ The element θ now is generated via $\theta \leftarrow_{\$} \mathbb{Z}_{n^2}^*$ instead of computing $\theta \leftarrow \rho^n(1+n)^{\sigma} \bmod n^2$ for $\rho \leftarrow_{\$} \mathbb{J}_n$.

We claim that $\textbf{Hybrid}_3$ is indistinguishable from $\textbf{Hybrid}_2$ for adversaries. Otherwise, given a distinguisher $(\mathcal{A}_1, \mathcal{A}_2)$, we can construct a probabilistic polynomial-time adversary $\mathcal{D}$ to break the decisional composite residuosity assumption. Upon receiving the pair (n, y), $\mathcal{D}$ picks $g \leftarrow_{\$} \mathbb{J}_n$ and $\chi \leftarrow_{\$} \mathbb{Z}_n^*$ for $J_n(\chi) = -1$, computes $h \leftarrow g^{2^T} \bmod n$, and sets $pp \leftarrow (T, n, g, h, \chi)$. $\mathcal{D}$ also chooses $\hat{y} \leftarrow_{\$} \mathbb{Z}_n^*$, such that $J_n(\hat{y}) = J_n(y)$. Then $\mathcal{D}$ invokes $\mathcal{A}_1(pp)$ and receives (τ, s_0, s_1). Let u, u', and v be generated as the reduction in $\textbf{Hybrid}_2$ and $\theta \leftarrow y\hat{y}^n(1+n)^{\sigma_b} \bmod n^2$ for $b \leftarrow_{\$} \{0, 1\}$. Here $\sigma_b = (-J_n(s_b)+1)/2$. Let $Z = (u, u', v, \theta)$. $\mathcal{D}$ invokes $\mathcal{A}_2$ with input Z and τ, and outputs whatever $\mathcal{A}_2$ outputs. The two cases for this reduction are as follows.

- The case $y = x^n \bmod n^2$ for $x \leftarrow_{\$} \mathbb{Z}_n^*$: The puzzle is of the form:

$$Z = (u, u', v, y\hat{y}^n(1+n)^{\sigma_b}) = (u, u', v, (x\hat{y})^n(1+n)^{\sigma_b}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^* \,.$$

  Here $x\hat{y} \in \mathbb{J}_n$ is a random element. Hence, Z is identically distributed as that in $\textbf{Hybrid}_2$.
- The case $y \leftarrow_{\$} \mathbb{Z}_{n^2}^*$: The puzzle is of the form:

$$Z = (u, u', v, y\hat{y}^n(1+n)^{\sigma_b}) = (u, u', v, \tilde{y}) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^* \,,$$

  where $\tilde{y} = y\hat{y}^n(1+n)^{\sigma_b} \bmod n^2$. Since $y \leftarrow_{\$} \mathbb{Z}_{n^2}^*$, $\tilde{y}$ is a random element in $\mathbb{Z}_{n^2}^*$. Hence, Z is identically distributed as that in $\textbf{Hybrid}_3$.

Therefore, $\mathcal{D}$ wins the DCR experiment with probability significantly higher than $\frac{1}{2}$, which contradicts to the assumption.

Since the puzzle in **Hybrid**$_3$ information-theoretically hides the solution s_b, the proofs is then completed. □

Remark 1. Both the additively HTLP scheme [15] and our MHTLP scheme need a one-time trusted setup. Their Setup algorithms can be executed by a trusted party or a group of parties using actively secure multi-party computation protocols.

4 Protocols for Verifiability

In this section, we introduce three protocols for verifiability for the additively HTLP scheme (Section 3.1) and our MHTLP scheme (Section 3.2), respectively. Note that for the RSA modulus $n = (2p'+1)(2q'+1)$, we let p', q' be larger than 2^κ. Note that all protocols are public-coin, which means that all challenges from the verifier are chosen uniformly random. Thus, they can be compiled by Fiat-Shamir heuristic [9] to be non-interactive and secure against malicious verifiers.

4.1 Building Block

We recall the protocol for VDF proposed by Wesolowski [23] as a building block. This protocol is a succinct public-coin interactive argument for the language

$$\mathcal{L}_{\mathsf{EXP}} = \left\{(T, n, u, w) : w = u^{2^T} \bmod n\right\} ,$$

where $u, w \in \mathbb{Z}_n$ and n is a strong RSA modulus. We denote this protocol by Π_{EXP}. Let $\mathsf{Prime}(2\kappa)$ be the set of the first $2^{2\kappa}$ primes. The description of the protocol between a prover P and a verifier V is as follows.

1. V randomly picks a prime ℓ from $\mathsf{Prime}(2\kappa)$ and sends ℓ to P.
2. Let $2^T = q\ell + r$, where $q, r \in \mathbb{Z}$ and $0 \leq r < \ell$. P computes $\pi \leftarrow u^q \bmod n$ and sends it to V.
3. V computes $r \leftarrow 2^T \bmod \ell$. Then V outputs accept if $\pi \in \mathbb{Z}_n$ and $w = \pi^\ell u^r \bmod n$, and reject otherwise.

We note that $\pi \leftarrow u^q \bmod n$ can be efficiently computed by $T \cdot 3/\log(T)$ group operations that are allowed to be parallelized (see [23] for more information).

4.2 Protocols for Additively HTLP Scheme

Correctness This protocol allows a puzzle solver to prove to others that the solution $s \in \mathbb{Z}_n$ he derived from solving a puzzle $Z = (u, v) \in \mathbb{J}_n \times \mathbb{Z}^*_{n^2}$ is correct. We denote this protocol by Π_{ACorSol} and its corresponding language is

$$\mathcal{L}_{\mathsf{ACorSol}} = \left\{(T, n, u, v, s) : v = u^{2^T \cdot n}(1+n)^s \bmod n^2\right\} ,$$

for an RSA modulus n. The idea is that the puzzle solver can reveal $w = u^{2^T} \bmod n$ and prove that (T, n, u, w) is in $\mathcal{L}_{\mathsf{EXP}}$ via Π_{EXP}. Then given w, verifiers can quickly solve the puzzle and verify the correctness of the solution s. The description of Π_{ACorSol} between a prover P and a verifier V is in the following.

1. P sends $w \leftarrow u^{2^T} \bmod n$ to V.
2. P and V engage in $\mathit{\Pi}_{\mathsf{EXP}}$ for the tuple (T, n, u, w).
3. V outputs accept if the argument $\mathit{\Pi}_{\mathsf{EXP}}$ is accepted and $v = w^n(1+n)^s \bmod n$. Otherwise, V outputs reject.

It is direct to have the theorem below and its proof is given in Appendix C.2.

Theorem 2. *The protocol $\mathit{\Pi}_{\mathsf{ACorSol}}$ is a public-coin honest-verifier argument corresponding to the language $\mathcal{L}_{\mathsf{ACorSol}}$.*

Invalidity This protocol allows a puzzle solver to prove to others that the output of the algorithm Solve for a puzzle $Z = (u, v) \in \mathbb{J}_n \times \mathbb{Z}^*_{n^2}$ is $\perp$, *i.e.*, Z is invalid. The idea is similar to $\mathit{\Pi}_{\mathsf{ACorSol}}$, *i.e.*, the puzzle solver helps verifiers quickly solve the puzzle to know that it is invalid. This protocol is denoted by $\mathit{\Pi}_{\mathsf{AInvalid}}$ and its corresponding language is

$$\mathcal{L}_{\mathsf{AInvalid}} = \left\{(T, n, u, v) : \forall s \in \mathbb{Z}_n, v \neq u^{2^T \cdot n}(1+n)^s \bmod n^2\right\} .$$

The description of $\mathit{\Pi}_{\mathsf{AInvalid}}$ between a prover P and a verifier V is in the following.

1. P sends $w \leftarrow u^{2^T} \bmod n$ to V.
2. P and V engage in $\mathit{\Pi}_{\mathsf{EXP}}$ for the tuple (T, n, u, w).
3. V outputs accept if the argument $\mathit{\Pi}_{\mathsf{EXP}}$ is accepted and n does not divide $([vw^{-n} \bmod n^2] - 1)$. Otherwise, V outputs reject.

Similarly, we have the following theorem.

Theorem 3. *The protocol $\mathit{\Pi}_{\mathsf{AInvalid}}$ is a public-coin honest-verifier argument corresponding to the language $\mathcal{L}_{\mathsf{AInvalid}}$.*

Validity This protocol allows a puzzle generator to show the validity of Z by proving in zero-knowledge the knowledge of the solution and randomness for a puzzle $Z = (u, v) \in \mathbb{J}_n \times \mathbb{Z}^*_{n^2}$, *i.e.*, the solution exists. Inspired by the commitments of integers [6], we follow the approach introduced in Section 1.2 to overcome the hidden-order problem. This protocol is denoted by $\mathit{\Pi}_{\mathsf{AValid}}$ and its corresponding relation is defined as

$$\begin{aligned}\mathcal{R}_{\mathsf{AValid}} = \{(T, n, g, h, u, v) : \exists (r, s) \in \mathbb{Z} \times \mathbb{Z}_n,\\ s.t.\ u = \pm g^r \bmod n \wedge v = h^{r \cdot n} \cdot (1+n)^s \bmod n^2\} .\end{aligned}$$

We relax the requirement of $u = g^r \bmod n$ to $u = \pm g^r \bmod n$. Since $u^{2^T} \equiv (-g^r)^{2^T} \equiv (g^r)^{2^T} \pmod n$, this does not compromise the correctness and security of the protocol. The description of $\mathit{\Pi}_{\mathsf{AValid}}$ between a prover P and a verifier V is presented below.

1. P randomly picks $x \leftarrow_{\$} [0, \lceil n/2 \rceil \cdot 2^{2\kappa}]$ and $t \leftarrow_{\$} \mathbb{Z}_n$, computes $a \leftarrow g^x \bmod n$ and $b \leftarrow h^{x \cdot n}(1+n)^t \bmod n^2$. Then P sends a and b to V.

2. V randomly chooses $e \leftarrow_{\$} [0, 2^{\kappa}]$ and sends it to P.
3. P computes $\alpha \leftarrow re + x$, $\beta \leftarrow se + t \bmod n$.
4. If $\alpha \in [0, \lceil n/2 \rceil \cdot 2^{\kappa} + \lceil n/2 \rceil \cdot 2^{2\kappa}]$, $\beta \in \mathbb{Z}_n$, $g^{\alpha} \equiv u^e a \pmod{n}$, and $h^{\alpha \cdot n}(1+n)^{\beta} \equiv v^e b \pmod{n^2}$ hold, V outputs accept, and otherwise reject.

Theorem 4. *The protocol Π_{AValid} is a public-coin honest-verifier zero-knowledge argument of knowledge corresponding to the relation $\mathcal{R}_{\mathsf{AValid}}$.*

Proof. For the completeness, we have $g^{\alpha} \equiv g^{re+x} \equiv u^e a \pmod{n^2}$ and

$$\begin{aligned} h^{\alpha n}(1+n)^{\beta} &\equiv h^{(re+x)n}(1+n)^{[se+t \bmod n]} \\ &\equiv h^{re \cdot n}(1+n)^{[se \bmod n]} h^{x \cdot n}(1+n)^{[t \bmod n]} \equiv v^e b \pmod{n^2}. \end{aligned}$$

Since $\alpha = re + x$, $r < \lceil n/2 \rceil$, $e < 2^{\kappa}$, and $x < \lceil n/2 \rceil \cdot 2^{2\kappa}$, we have $\alpha \in [0, \lceil n/2 \rceil \cdot 2^{\kappa} + \lceil n/2 \rceil \cdot 2^{2\kappa}]$.

For honest-verifier zero-knowledge, we construct the following simulator $\mathcal{S}$.

1. $\mathcal{S}$ randomly choose a challenge $e \leftarrow_{\$} [0, 2^{\kappa}]$.
2. $\mathcal{S}$ picks random responses $\alpha \leftarrow_{\$} [0, \lceil n/2 \rceil \cdot 2^{2\kappa}]$ and $\beta \leftarrow_{\$} \mathbb{Z}_n$.
3. $\mathcal{S}$ computes $a \leftarrow g^{\alpha} u^{-e} \bmod n$ and $b \leftarrow h^{\alpha \cdot n}(1+n)^{\beta} v^{-e} \bmod n^2$ as the messages sent by the prover in Step 1.

The simulated transcript is $((a, b), e, (\alpha, \beta))$. We now prove that it is statistically indistinguishable from a real transcript by a sequence of hybrids as follows.

Hybrid$_0$ This is the transcript of the real execution. The elements are generated as $x \leftarrow_{\$} [0, \lceil n/2 \rceil \cdot 2^{2\kappa}]$, $t \leftarrow_{\$} \mathbb{Z}_n$, $e \leftarrow_{\$} [0, 2^{\kappa}]$, $a \leftarrow g^x \bmod n$, $b \leftarrow h^{x \cdot n}(1+n)^t \bmod n^2$, $\alpha = re + x$, and $\beta = se + t \bmod n$.

Hybrid$_1$ The elements are generated as $e \leftarrow_{\$} [0, 2^{\kappa}]$, $\alpha \leftarrow_{\$} [re, re + \lceil n/2 \rceil \cdot 2^{2\kappa}]$, $\beta \leftarrow_{\$} \mathbb{Z}_n$, $a \leftarrow g^{\alpha - re} \bmod n$, and $b \leftarrow h^{(\alpha - re) \cdot n}(1+n)^{[\beta - se \bmod n]} \bmod n^2$. It is clear that the distribution of the transcript is identical to that in **Hybrid$_0$**.

Hybrid$_2$ Different from **Hybrid$_1$**, let $a \leftarrow g^{\alpha} u^{-e} \bmod n$, and $b \leftarrow h^{\alpha \cdot n}(1+n)^{\beta} v^{-e} \bmod n^2$. It is easy to see that the distribution of the transcript is still identical to that in **Hybrid$_1$**.

Hybrid$_3$ This is the simulated transcript. Let $e \leftarrow_{\$} [0, 2^{\kappa}]$, $\alpha \leftarrow_{\$} [0, \lceil n/2 \rceil \cdot 2^{2\kappa}]$, $\beta \leftarrow_{\$} \mathbb{Z}_n$, $a \leftarrow g^{\alpha} u^{-e} \bmod n$, and $b \leftarrow h^{\alpha \cdot n}(1+n)^{\beta} v^{-e} \bmod n^2$.

The distance between distributions of **Hybrid$_2$** and **Hybrid$_3$** is equivalent to the distance between distributions of α's in these two hybrids. Let X and Y be the random variables for the distributions of α's in **Hybrid$_2$** and **Hybrid$_3$**, respectively. Then the statistical distance $\Delta(X;Y)$ between X and Y is

$$\begin{aligned} \Delta(X;Y) &= \frac{1}{2} \sum_{\alpha \in [0, re + \lceil n/2 \rceil \cdot 2^{2\kappa}]} |\Pr[X = \alpha] - \Pr[Y = \alpha]| \\ &= \frac{1}{2} \sum_{\alpha=0}^{re-1} \lceil n/2 \rceil^{-1} 2^{-2\kappa} + \frac{1}{2} \sum_{\alpha = \lceil n/2 \rceil \cdot 2^{2\kappa}}^{\lceil n/2 \rceil \cdot 2^{2\kappa} + re - 1} \lceil n/2 \rceil^{-1} 2^{-2\kappa} \\ &= re \lceil n/2 \rceil^{-1} 2^{-2\kappa} \leq \lceil n/2 \rceil 2^{\kappa} \cdot \lceil n/2 \rceil^{-1} 2^{-2\kappa} = 2^{-\kappa}. \end{aligned}$$

The distribution of $\mathbf{Hybrid}_3$ is thus statistically indistinguishable from $\mathbf{Hybrid}_2$, and also $\mathbf{Hybrid}_0$. The honest-verifier zero-knowledge property then follows.

We now focus on the witness-extended emulation property. Suppose that verifiers interact with P^* and output accept with non-negligible probability ε. We construct an emulator that runs P^* as a subroutine. After receiving a and b from P^*, the emulator gives $e_1 \leftarrow_\$ [0, 2^\kappa]$ to P^*. If the output α_1 and β_1 from P^* consists of an accepting transcript, the emulator needs to use an extractor $\mathcal{E}$ to extract the witness. $\mathcal{E}$ rewinds P^* to Step 2 (challenge phase) and runs it again with a new random challenge from $[0, 2^\kappa]$ until having an accepting transcript for a challenge e_2. The expected running time for the rewinding is $1/\varepsilon$, and thus polynomial. Since challenges are randomly sampled from $[0, 2^\kappa]$, we have $e_1 \neq e_2$ except for negligible probability. Let $\mathcal{S}$ abort if $e_1 = e_2$. This does not affect our analysis since it happens with negligible probability. We assume that $e_1 \neq e_2$ in the remaining analysis. We denote the two accepting transcripts by $((a, b), e_1, (\alpha_1, \beta_1))$ and $((a, b), e_2, (\alpha_2, \beta_2))$, satisfying

$$\begin{aligned} g^{\alpha_1} &\equiv u^{e_1} a \pmod{n}\,, \qquad & h^{\alpha_1 \cdot n}(1+n)^{\beta_1} &\equiv v^{e_1} b \pmod{n^2}\,, \\ g^{\alpha_2} &\equiv u^{e_2} a \pmod{n}\,, \qquad & h^{\alpha_2 \cdot n}(1+n)^{\beta_2} &\equiv v^{e_2} b \pmod{n^2}\,. \end{aligned}$$

Without loss of generality, we assume that $e_1 > e_2$. Let $e' \leftarrow e_1 - e_2$, $\alpha' \leftarrow \alpha_1 - \alpha_2$, and $\beta' \leftarrow \beta_1 - \beta_2 \bmod n$. We can easily derive $g^{\alpha'} \equiv u^{e'} \pmod{n}$ and $h^{\alpha' n}(1+n)^{\beta'} \equiv v^{e'} \pmod{n^2}$. We focus on the equation $g^{\alpha'} \equiv u^{e'} \pmod{n}$.

Firstly, we analyze the case that e' divides α'. Let $r \leftarrow \frac{\alpha'}{e'}$. If e' is odd, it is obvious that e' is coprime to $\lambda = 2p'q'$, and thus $u = g^r \bmod n$. Therefore, we extract the witness r. If e' is an even number, it can be expressed as $e' = 2^d \rho$, where $d \geq 1$ and ρ is odd. It is obvious that ρ is coprime to λ. Now we obtain $g^{2^d r} \equiv u^{2^d} \pmod{n}$, and thus $(g^r u^{-1})^{2^d} \equiv 1 \pmod{n}$. We can further simplify the equation as $(g^r u^{-1})^2 \equiv 1 \pmod{n}$. Hence, $g^r u^{-1}$ is a square root of 1.

- If $g^r u^{-1}$ is a nontrivial square root of 1, *i.e.*, $g^r u^{-1} \not\equiv \pm 1 \pmod{n}$, we know that $(g^r u^{-1})^2 - 1 \equiv (g^r u^{-1} - 1)(g^r u^{-1} + 1) \equiv 0 \pmod{n}$, where $(g^r u^{-1} - 1) \not\equiv 0 \pmod{n}$ or $(g^r u^{-1} + 1) \not\equiv 0 \pmod{n}$. Hence, a nontrivial factor of the RSA modulus n can be computed from $\gcd(g^r u^{-1} \pm 1, n)$. Since we assume that factoring n is hard, this happens with negligible probability.
- If $g^r u^{-1} \equiv \pm 1 \pmod{n}$, then we have $g^r \equiv \pm u \pmod{n}$, and thus the witness r is extracted.

Then we analyze the case that e' does not divide α'. Let $\gamma = \gcd(e', \alpha')$, $\tau = \frac{e'}{\gamma}$, and $\omega = \frac{\alpha'}{\gamma}$. Note that $\frac{\omega}{\tau}$ is the irreducible fraction form of $\frac{\alpha'}{e'}$. We now show that we can break the strong RSA assumption if e' does not divide α'.

- If γ is an odd number, since $\gamma < e'$, γ is coprime to λ. Hence, from $g^{\alpha'} \equiv u^{e'} \pmod{n}$, we know that $g^\omega \equiv u^\tau \pmod{n}$. Then, we are able to construct an attacker to break the strong RSA assumption. More concretely, given an RSA challenge (n, g_0), we set $g \leftarrow -g_0^2 \bmod n$ in the public parameter pp. Then, if this is the case, we have $(-g_0^2)^\omega \equiv u^\tau \pmod{n}$.

If ω is an even number, we have $(-g_0^2)^{\omega} \equiv g_0^{2\omega} \equiv u^{\tau} \pmod{n}$. Since ω is an even number, τ must be odd, and thus $\gcd(2\omega, \tau) = 1$. Now given $\gcd(2\omega, \tau) = 1$ and $g_0^{2\omega} \equiv u^{\tau} \pmod{n}$, we can derive the τ-th root of g_0 according to Lemma 1, and thus break the strong RSA assumption.

If ω is an odd number, we have $-g_0^{2\omega} \equiv u^{\tau}$. Since $g_0^{2\omega} \in \mathbb{QR}_n$ and $-1 \in \mathbb{J}_n \backslash \mathbb{QR}_n$ for the strong RSA modulus n, we know that $-g_0^{2\omega} \in \mathbb{J}_n \backslash \mathbb{QR}_n$, and thus τ is also an odd number. This implies that $\gcd(2\omega, \tau) = 1$ and $g_0^{2\omega} \equiv (-u)^{\tau}$. Hence, we can derive the τ-th root of g_0 according to Lemma 1 and break the strong RSA assumption.

– If γ is an even number, we denote γ by $2^d\rho$. Since $\rho < \gamma < e'$, ρ is coprime to λ. Then we know that $g^{2^d\omega} \equiv u^{2^d\tau} \pmod{n}$, and thus $g^{2\omega} \equiv u^{2\tau} \pmod{n}$.

  If $g^{\omega} \not\equiv \pm u^{\tau} \pmod{n}$, we can write $(g^{\omega} + u^{\tau})(g^{\omega} - u^{\tau}) \equiv 0 \pmod{n}$, where $g^{\omega} + u^{\tau} \not\equiv 0 \pmod{n}$ or $g^{\omega} - u^{\tau} \not\equiv 0 \pmod{n}$. Hence, a nontrivial factor of the RSA modulus n can be computed from $\gcd(g^{\omega} \pm u^{\tau}, n)$.

  If $g^{\omega} \equiv \pm u^{\tau} \pmod{n}$, we can construct an attacker to break the strong RSA assumption. Given an RSA challenge (n, g_0), we set $g \leftarrow -g_0^2 \bmod n$ in pp.

  - If $g^{\omega} \equiv u^{\tau} \pmod{n}$, we have $(-g_0^2)^{\omega} \equiv u^{\tau} \pmod{n}$.

    When ω is an odd number, $-g_0^{2\omega} \equiv u^{\tau} \pmod{n}$. Following the same argument as above, τ should be odd, and thus $\gcd(2\omega, \tau) = 1$. Then we have $g_0^{2\omega} \equiv (-u)^{\tau} \pmod{n}$. Hence, we can derive the τ-th root of g_0 according to Lemma 1, and thus break the strong RSA assumption.

    When ω is an even number, we have $g_0^{2\omega} \equiv u^{\tau} \pmod{n}$ and $\gcd(2\omega, \tau) = 1$. Then we can derive the τ-th root of g_0 according to Lemma 1, and thus again break the strong RSA assumption.
  - If $g^{\omega} \equiv -u^{\tau} \pmod{n}$, we have $(-g_0^2)^{\omega} \equiv -u^{\tau} \pmod{n}$.

    When ω is an even number, we know that $g_0^{2\omega} \equiv -u^{\tau} \pmod{n}$ and τ is an odd number. So we have $g_0^{2\omega} \equiv (-u)^{\tau} \pmod{n}$ and $\gcd(2\omega, \tau) = 1$. Hence, we can derive the τ-th root of g_0 according to Lemma 1 and break the strong RSA assumption.

    When ω is an odd number, we have $g_0^{2\omega} \equiv u^{\tau} \pmod{n}$. If τ is an odd number, we have $\gcd(2\omega, \tau) = 1$. Therefore, we can derive the τ-th root of g_0 according to Lemma 1, and thus break the strong RSA assumption. Finally, if τ is an even number, since $\gcd(2\omega, \tau) = 2$, we can find integer $\bar{x}$ and $\bar{y}$, such that $2\omega \cdot \bar{x} + \tau \cdot \bar{y} = 2$. Then we have $g_0^2 \equiv g_0^{2\omega \cdot \bar{x} + \tau \cdot \bar{y}} \equiv u^{\tau \cdot \bar{x}} g_0^{\bar{y} \cdot \tau} \equiv (u^{\bar{x}} g_0^{\bar{y}})^{\tau} \pmod{n}$ Let $\tau' \leftarrow \tau/2$ and $z \leftarrow u^{\bar{x}} g_0^{\bar{y}} \bmod n$. We have $g_0^2 \equiv z^{\tau' \cdot 2} \pmod{n}$, *i.e.*, $(g_0 z^{-\tau'})^2 \equiv 1 \pmod{n}$. Hence, $g_0 z^{-\tau'}$ is a square root of 1 modulo n.

    If $g_0 z^{-\tau'}$ is a nontrivial square root of 1, *i.e.*, $g_0 z^{-\tau' \cdot 2} \not\equiv \pm 1 \pmod{n}$, we can easily derive a nontrivial factor of the RSA modulus n using the same approach as above, *i.e.*, by computing $\gcd(g_0 z^{-\tau'} \pm 1, n)$.

    If $g_0 z^{-\tau'} \equiv \pm 1 \pmod{n}$, *i.e.*, $g_0 \equiv \pm z^{\tau'} \pmod{n}$, we consider two cases. If τ' is an odd number, we have $g_0 \equiv (\pm z)^{\tau'} \pmod{n}$, and thus break the strong RSA assumption. If $\tau' = 2^d\rho$, where ρ is an odd number, we have $g_0 \equiv (\pm z^{2^d})^{\rho} \pmod{n}$, and we again break the strong RSA assumption.

In summary, if e' does not divide α', we can break the strong RSA assumption, with at most negligible probability. Hence, with an overwhelming probability, e'

divides α' and $\mathcal{E}$ can successfully extract r such that $u = \pm g^r \pmod{n}$. We can also compute $x \leftarrow \alpha - re$, which satisfies $a = g^x \bmod n$.

According to [16], there is an isomorphism from $\mathbb{Z}_n^* \times \mathbb{Z}_n$ to $\mathbb{Z}_{n^2}^*$. Since we obtain two accepting transcripts, we know that there exist $t \in \mathbb{Z}_n$ and $s \in \mathbb{Z}_n$, such that $v = h^{[r \bmod \lambda] \cdot n}(1+n)^s \bmod n^2$ and $b = h^{[x \bmod n] \cdot n}(1+n)^t \bmod n^2$, for the same r and x. Hence, we can compute the witness s and t from $\beta_1 \leftarrow e_1 s + t \bmod n$ and $\beta_2 \leftarrow e_2 s + t \bmod n$ for given e_1, e_2, β_1, and β_2.

It is easy to verify that $\mathcal{E}$ runs in expected probabilistic polynomial time, and thus the protocol achieves witness-extended emulation. □

4.3 Protocols for Our Multiplicatively HTLP

Correctness Denote the protocol by $\mathit{\Pi}_{\mathsf{MCorSol}}$ and its corresponding language by

$$\mathcal{L}_{\mathsf{MCorSol}} = \{(T, n, \chi, Z = (u, u', v, \theta), s) : \exists \sigma \in \mathbb{Z}_n,$$
$$s.t.\ v = u^{2^T} \cdot \chi^\sigma s \bmod n \wedge \theta = u'^{2^T \cdot n}(1+n)^\sigma \bmod n^2\}\,.$$

The idea of this protocol is similar to $\mathit{\Pi}_{\mathsf{ACorSol}}$. Its description between P and V, together with the related theorem are in the following.

1. P sends $w \leftarrow u^{2^T} \bmod n$ and $w' \leftarrow u'^{2^T} \bmod n$ to V.
2. P and V engage in $\mathit{\Pi}_{\mathsf{EXP}}$ twice[5] for tuples (T, n, u, w) and (T, n, u', w'), respectively.
3. V computes $\sigma \leftarrow \frac{[\theta w'^{-n} \bmod n^2] - 1}{n}$. Then V outputs accept if the two runs of $\mathit{\Pi}_{\mathsf{EXP}}$ are accepted, $\sigma \in \mathbb{Z}_n$, and $v = w\chi^\sigma s$. Otherwise, V outputs reject.

Theorem 5. *The protocol* $\mathit{\Pi}_{\mathsf{MCorSol}}$ *is a public-coin honest-verifier argument corresponding to the language* $\mathcal{L}_{\mathsf{MCorSol}}$.

Invalidity Denote this protocol by $\mathit{\Pi}_{\mathsf{MInvalid}}$. It is easy to see that a puzzle $Z = (u, u', v, \theta)$ for the MHTLP scheme is invalid if and only if (u', θ) is an invalid puzzle with respect to the additively HTLP scheme in Section 3.1. Therefore, $\mathit{\Pi}_{\mathsf{MInvalid}}$ is exactly the same as $\mathit{\Pi}_{\mathsf{AInvalid}}$ for the pair (u', θ).

Validity A puzzle $Z = (u, u', v, \theta) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^*$ output by Gen is valid if there exists a bit $\sigma \in \{0, 1\}$, such that $\theta = u'^{2^T \cdot n}(1+n)^\sigma \bmod n^2$. This is equivalent to proving the knowledge of $r' \in \mathbb{Z}$, such that $u' = g^{r'} \bmod n$ and meanwhile $\theta = h^{r' \cdot n} \bmod n^2$ or $\theta = h^{r' \cdot n}(1+n) \bmod n^2$. Denote the protocol by $\mathit{\Pi}_{\mathsf{MValid}}$ and its corresponding relation by

$$\mathcal{R}_{\mathsf{MValid}} = \{(T, n, g, h, u, v) : \exists (r', \sigma) \in \mathbb{Z} \times \{0, 1\}, s.t.\ u' = \pm g^{r'} \bmod n$$
$$\wedge\, \theta = h^{r' \cdot n}(1+n)^\sigma \bmod n^2\}\,.$$

[5] These two proofs can be aggregated, see [23] for more information.

We require $u' = \pm g^{r'} \bmod n$ as that in $\mathit{\Pi}_{\mathsf{AValid}}$. The description of the protocol $\mathit{\Pi}_{\mathsf{MValid}}$ between a prover P and a verifier V is presented below.

1. Both P and V set $\theta_0 \leftarrow \theta$, and $\theta_1 \leftarrow \theta/(1+n) \bmod n^2$.
2. For $\sigma \in \{0,1\}$, P chooses $e_{1-\sigma} \leftarrow_{\$} [0, 2^{\kappa}]$, $\alpha_{1-\sigma} \leftarrow_{\$} [0, \lceil n/2 \rceil \cdot 2^{2\kappa}]$. Then P computes $a_{1-\sigma} \leftarrow g^{\alpha_{1-\sigma}} u'^{-e_{1-\sigma}} \bmod n$ and $b_{1-\sigma} \leftarrow h^{\alpha_{1-\sigma} \cdot n} \theta_{1-\sigma}^{-e_{1-\sigma}} \bmod n^2$.
3. P randomly picks $x \leftarrow_{\$} [0, \lceil n/2 \rceil + 2^{2\kappa}]$, computes $a_{\sigma} \leftarrow g^{x} \bmod n$ and $b_{\sigma} \leftarrow h^{x \cdot n} \bmod n^2$. Then P sends a_0, a_1, b_0, and b_1 to V.
4. V randomly chooses $e \leftarrow_{\$} [0, 2^{\kappa}]$ and sends it to P.
5. P computes $e_{\sigma} \leftarrow e \oplus e_{1-\sigma}$. Then P computes $\alpha_{\sigma} \leftarrow r' e_{\sigma} + x$. After that, P sends α_0, α_1, e_0, and e_1 to V.
6. V checks whether $\alpha_i \in [0, \lceil n/2 \rceil \cdot 2^{\kappa} + \lceil n/2 \rceil \cdot 2^{2\kappa}]$, $e = e_0 \oplus e_1$, $g^{\alpha_i} \equiv u'^{e_i} a_i \pmod{n}$, and $h^{\alpha_i \cdot n} \equiv \theta_i^{e_i} b_i \pmod{n^2}$ for both $i \in \{0,1\}$. If they all hold, V outputs accept, and reject otherwise.

We have the theorem for $\mathit{\Pi}_{\mathsf{MValid}}$ below, and its proof is put in Appendix C.3.

Theorem 6. *The protocol $\mathit{\Pi}_{\mathsf{MValid}}$ is an honest-verifier zero-knowledge argument of knowledge corresponding to the relation $\mathcal{R}_{\mathsf{MValid}}$.*

5 Analysis

5.1 Communication Cost

Denote by μ the length of the RSA modulus n in bits. An MHTLP puzzle $Z = (u, u', v, \theta) \in \mathbb{J}_n^3 \times \mathbb{Z}_{n^2}^*$ is represented by 5μ bits. For practical usage, we set $\mu = 2048$, and thus a puzzle only occupies 1.25KB. In Table 1, We summarize the numbers of bits for the six protocols in this paper when they are compiled by the Fiat-Shamir heuristic. It is easy to see that the communication cost of our MHTLP scheme and the protocols for verifiability are satisfactory.

Table 1: Length of the six non-interactive arguments.

	$\mathit{\Pi}_{\mathsf{ACorSol}}$	$\mathit{\Pi}_{\mathsf{AInvalid}}$	$\mathit{\Pi}_{\mathsf{AValid}}$	$\mathit{\Pi}_{\mathsf{MCorSol}}$	$\mathit{\Pi}_{\mathsf{MInvalid}}$	$\mathit{\Pi}_{\mathsf{MValid}}$
Total	$\mu + 2\kappa$	$\mu + 2\kappa$	$2\mu + 3\kappa$	$2\mu + 4\kappa$	$\mu + 2\kappa$	$2\mu + 6\kappa$
$\mu = 2048$, $\kappa = 128$	0.28KB	0.28KB	0.55KB	0.56KB	0.28KB	0.59KB

5.2 Computation Cost

We implement our MHTLP scheme and protocols in `C++` using the `NTL` [20] and `OpenSSL` [17] libraries for the underlying modular arithmetic and hash function, respectively. SHA256 is used to implement the random oracle. Experiments are conducted on a Windows 11 laptop of Windows Subsystem for Linux 2 (WSL2) with AMD Ryzen 9 5900HS, 16GB of RAM using a single thread. Set $\mu = 2048$

Table 2: Mul cost of MHTLP.

# of Mul	Time
$100,000$	0.9576s
$1,000,000$	10.8318s

Table 3: Computation cost of Π_{AValid} and Π_{MValid}.

	Π_{AValid}	Π_{MValid}
Prover Time	18.85ms	37.98ms
Verifier Time	19.14ms	38.60ms

and $\kappa = 128$ in our implementation.[6] Table 2 provides the computation cost of multiplications on secrets within puzzles for our MHTLP scheme. Note that for one multiplication, 3 multiplications over $\mathbb{Z}_n^*$ and 1 multiplication over $\mathbb{Z}_{n^2}^*$ are involved, and they cost only around 0.01ms. Table 3 presents the computation cost of the protocols Π_{AValid} and Π_{MValid}. It is easy to see that both the prover and verifier spends only around 19ms to generate and verify a proof for Π_{AValid} and 38ms for Π_{MValid}. For Π_{EXP}, according to the analysis in [23], the prover takes $\mathcal{O}(T/\log(T))$ group operations based on intermediate values from computing the sequential squaring computation to generate the proof. For practical parameters, the total time to generate a proof in Π_{EXP} is around $1/(20s)T$, where s is the number of cores for a computer. In Table 4, for protocols Π_{ACorSol}, Π_{AInvalid}, Π_{MCorSol}, and Π_{MInvalid}, we provide the time for the verifier, given the representative total time for a prover to solve a puzzle and generate a proof. We can see that all verifications spend around 8.5ms. Therefore, the computation cost of our MHTLP scheme and the protocols for verifiability are satisfactory.

Table 4: Computation cost of Π_{ACorSol}, Π_{AInvalid}, Π_{MCorSol}, and Π_{MInvalid}.

	Π_{ACorSol}	Π_{AInvalid}	Π_{MCorSol}	Π_{MInvalid}
Solve and Prove	16.3072s	16.232s	21.725s	21.738s
Verification	8.319ms	8.301ms	8.673ms	8.766ms

In conclusion, our MHTLP scheme and the protocols demonstrates great efficiency in both communication and computation and will significantly broaden the application scenarios of HTLP. HTLP schemes are thereby more practical.

6 Future Work

We propose the following interesting questions that deserve explorations.

1. Our MHTLP scheme requires to execute two sequential squaring processes in parallel. It will be nice if we can design a specific multiplicatively HTLP scheme with solution space $\mathbb{Z}_n^*$, such that one sequential squaring is enough to solve a puzzle.
2. Some practical homomorphic encryption schemes support a wider range of operations. For example, [2] supports an unlimited number of additions and

[6] The implementation is available at https://github.com/liu-yi/HTLP.

up-to-one multiplication on encrypted values, [11] supports polynomial-size branching programs. It will be interesting if we can extend these results and design a practical HTLP scheme that supports a wider range of operations (richer than addition/multiplication).

3. For some applications, we may need to ensure that the secrets are in a specified range. We know that there are some range-proof protocols for encryption and commitment schemes. How to design range-proof protocols for HTLP schemes is an interesting problem.
4. The solution spaces of the additively HTLP scheme ($\mathbb{Z}_n$) and our multiplicatively HTLP scheme ($\mathbb{Z}_n^*$) are compatible. It seems that this may lead to some interesting applications. For example, can we design a mechanism that can use this fact to support a wider range of homomorphic operations, for example, via switching the underlying HTLP scheme of a time-lock puzzle?

References

1. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10991, pp. 757–788. Springer (2018)
2. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Kilian, J. (ed.) Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3378, pp. 325–341. Springer (2005)
3. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1880, pp. 236–254. Springer (2000)
4. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11892, pp. 407–437. Springer (2019)
5. Chvojka, P., Jager, T., Slamanig, D., Striecks, C.: Versatile and sustainable timed-release encryption and sequential time-lock puzzles (extended abstract). In: Bertino, E., Shulman, H., Waidner, M. (eds.) Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12973, pp. 64–85. Springer (2021)
6. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2501, pp. 125–142. Springer (2002)

7. Dwork, C., Naor, M.: Zaps and their applications. SIAM J. Comput. **36**(6), 1513–1543 (2007)
8. Faust, S., Hazay, C., Kretzler, D., Schlosser, B.: Generic compiler for publicly verifiable covert multi-party computation. IACR Cryptol. ePrint Arch. **2021**, 251 (2021), https://eprint.iacr.org/2021/251
9. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
10. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984)
11. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4392, pp. 575–594. Springer (2007)
12. Knapp, J., Quaglia, E.A.: Fair and sound secret sharing from homomorphic time-lock puzzles. In: Nguyen, K., Wu, W., Lam, K., Wang, H. (eds.) Provable and Practical Security - 14th International Conference, ProvSec 2020, Singapore, November 29 - December 1, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12505, pp. 341–360. Springer (2020)
13. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. SIAM J. Comput. **49**(4) (2020)
14. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology **16**(3), 143–184 (2003)
15. Malavolta, G., Thyagarajan, S.A.K.: Homomorphic time-lock puzzles and applications. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 620–649. Springer (2019)
16. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999)
17. Project, O.: Openssl project, https://www.openssl.org/
18. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. rep., Massachusetts Institute of Technology, USA (1996)
19. Scholl, P., Simkin, M., Siniscalchi, L.: Multiparty computation with covert security and public verifiability. IACR Cryptol. ePrint Arch. **2021**, 366 (2021), https://eprint.iacr.org/2021/366
20. Shoup, V.: Ntl: A library for doing number theory, http://www.shoup.net/ntl
21. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 207–220. Springer (2000)
22. Thyagarajan, S.A.K., Bhat, A., Malavolta, G., Döttling, N., Kate, A., Schröder, D.: Verifiable timed signatures made practical. In: Ligatti, J., Ou, X., Katz, J.,

Vigna, G. (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. pp. 1733–1750. ACM (2020)

23. Wesolowski, B.: Efficient verifiable delay functions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 379–407. Springer (2019)

A Computational Assumptions

Definition 2 (Strong Sequential Squaring Assumption [15]). *Let n be a randomly generated strong RSA modulus based on κ, g be a generator of $\mathbb{J}_n$, and $T(\cdot)$ be a polynomial. There exists ε with $0 < \varepsilon < 1$, such that for all polynomial-size adversaries $(\mathcal{A}_1, \mathcal{A}_2) = \{(\mathcal{A}_1, \mathcal{A}_2)_\kappa\}_{\kappa \in \mathbb{N}}$, where the depth of $\mathcal{A}_2$ is bounded from above by $T^\varepsilon(\kappa)$, we have*

$$\Pr\left[b \leftarrow \mathcal{A}_2(x,y,\tau) : \begin{array}{l}\tau \leftarrow \mathcal{A}_1(n,g,T(\kappa)); x \leftarrow_{\$} \mathbb{J}_n; b \leftarrow_{\$} \{0,1\};\\ \textit{if } b = 0, y \leftarrow_{\$} \mathbb{J}_n; \textit{if } b = 1, y = x^{2^{T(\kappa)}} \bmod n;\end{array}\right] \leq \frac{1}{2}+\mathsf{negl}(\kappa)\ .$$

Definition 3 (Decisional Composite Residuosity Assumption). *Let n be a randomly generated strong RSA modulus based on κ. Then the decisional composite residuosity (DCR) assumption is that for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, we have*

$$\Pr\left[b \leftarrow \mathcal{A}(n,y) : \begin{array}{l}x \leftarrow \mathbb{Z}_n^*; b \leftarrow \{0,1\};\\ \textit{if } b = 0, y \leftarrow \mathbb{Z}_{n^2}^*; \textit{if } b = 1, y \leftarrow x^n \bmod n^2;\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\kappa)\ .$$

Definition 4 (Strong RSA Assumption). *Let n be a randomly generated strong RSA modulus based on κ. For all PPT adversaries $\mathcal{A}$, we have*

$$\Pr\left[e \geq 2 \wedge x^e = y \bmod n : \begin{array}{l}y \leftarrow_{\$} \mathbb{Z}_n^*;\\ (x,e) \leftarrow \mathcal{A}(n,y);\end{array}\right] \leq \mathsf{negl}(\kappa)\ .$$

B Definition of Homomorphic Time-Lock Puzzle Scheme

Definition 5 ([15]). *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be a class of circuits. An HTLP scheme with the solution space $\mathbb{S}$ with respect to $\mathcal{C}$ is a tuple of algorithms* (Setup, Gen, Solve, Eval) *defined as follows.*

- *$pp \leftarrow \mathsf{Setup}(1^\kappa, T)$ a probabilistic algorithm that takes as input the security parameter 1^κ and a time hardness parameter T and outputs the public parameter pp.*
- *$Z \leftarrow \mathsf{Gen}(pp, s)$ a probabilistic algorithm that takes as input pp and a solution $s \in \mathbb{S}$ and outputs a homomorphic time-lock puzzle Z.*

- s / $\perp \leftarrow \mathsf{Solve}(pp, Z)$ *a deterministic algorithm that takes as input pp and a puzzle Z, and outputs a solution $s \in \mathbb{S}$ or an error message $\perp$ indicating that Z is invalid.*
- $Z \leftarrow \mathsf{Eval}(pp, C, Z_1, \ldots, Z_N)$ *an algorithm that takes as input pp, a circuit $C \in \mathcal{C}_\kappa$, and a set of n puzzles $(Z_1, \ldots, Z_N)$ and outputs a puzzle Z. Note that this algorithm defines the homomorphic operations for the HTLP scheme.*

It satisfies the following two properties.

Correctness *The scheme with respect to $\mathcal{C}$ is correct if for all polynomials T in κ, all $C \in \mathcal{C}_\kappa$ and inputs $(s_1, \ldots, s_N) \in \mathbb{S}^n$, we have*

$$\Pr\left[C(s_1, \ldots, s_N) \neq s : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\kappa, T); \\ Z_i \leftarrow \mathsf{Gen}(pp, s_i) \text{ for } i = 1, \ldots, N; \\ Z \leftarrow \mathsf{Eval}(pp, C, Z_1, \ldots, Z_N); s \leftarrow \mathsf{Solve}(pp, Z); \end{array} \right] \leq \mathsf{negl}(\kappa) \ .$$

Compactness *The scheme with respect to $\mathcal{C}$ is compact if for all polynomials T in κ, all $C \in \mathcal{C}_\kappa$, and inputs $(s_1, \ldots, s_N) \in \mathbb{S}^n$, when compute $pp \leftarrow \mathsf{Setup}(1^\kappa, T)$, $Z_i \leftarrow \mathsf{Gen}(pp, s_i)$, and $Z \leftarrow \mathsf{Eval}(pp, C, Z_1, \ldots, Z_N)$, the following three properties are satisfied.*

- *There exists a fixed polynomial p_1, such that the running time of the algorithm $\mathsf{Solve}(pp, Z)$ is bounded by $p_1(\kappa, T)$.*
- *There exists a fixed polynomial p_2, such that the length of Z is bounded by $p_2(\kappa, |C(s_1, \ldots, s_N)|)$, where $|C(s_1, \ldots, s_N)|$ is the length of bits to represent $C(s_1, \ldots, s_N)$.*
- *There exists a fixed polynomial p_3, such that the running time of the algorithm $\mathsf{Eval}(pp, C, Z_1, \ldots, C_N)$ is bounded by $p_3(\kappa, |C|)$, where $|C|$ is the size of the circuit C.*

C Security Proofs

C.1 Proof of Lemma 1

Proof. Since e' is coprime to e, we can efficiently compute integer u and v for the Bézout relation $ue + ve' = 1$. Then we have $x^{ve} \equiv y^{ve'} \equiv y^{1-ue} \pmod{n}$. Therefore, we have $(x^v y^u)^e \equiv y \pmod{n}$, such that $[x^v y^u \bmod n]$ is the e-th root of y modulus n. □

C.2 Proof of Theorem 2

Proof. The completeness of the protocol is direct. Using Π_{EXP}, the prover will prove that $w = u^{2^T}$ holds. Then for a valid puzzle $Z = (u, v)$ with solution s, we have $v \equiv u^{2^T \cdot n}(1+n)^s \equiv w^n(1+n)^s \pmod{n^2}$. Then for the soundness property, based on the soundness of Π_{EXP}, it is guaranteed that $w = u^{2^T}$. Hence, if $v = w^n(1+n)^s \bmod n^2$, we have $v = u^{2^T \cdot n}(1+n)^s \bmod n^2$. □

C.3 Proof of Theorem 6

Proof. Note that $\alpha_{1-\sigma}$, $a_{1-\sigma}$, and $b_{1-\sigma}$ are generated to satisfy the final verification. We can see that $g^{\alpha_\sigma} \equiv g^{r'e_\sigma+x} \equiv u'^{e_\sigma} a_\sigma \pmod{n^2}$ and $h^{\alpha_\sigma n} \equiv h^{(r'e_\sigma+x)n} \equiv h^{r'e_\sigma \cdot n} h^{x \cdot n} \equiv \theta_\sigma^{e_\sigma} b_\sigma \pmod{n^2}$. According to the analysis in the proof of Theorem 4, $\alpha_i \in [0, \lceil n/2 \rceil \cdot 2^\kappa + \lceil n/2 \rceil \cdot 2^{2\kappa}]$ except for negligible probability. Hence, the protocol is complete.

We construct a simulator $\mathcal{S}$ as follows. $\mathcal{S}$ choose $e_0, e_1 \leftarrow_\$ [0, 2^\kappa]$, $\alpha_0, \alpha_1 \leftarrow_\$ [0, \lceil n/2 \rceil \cdot 2^{2\kappa}]$, and sets $e \leftarrow e_0 \oplus e_1$, $a_i \leftarrow g^{\alpha_i} u^{-e_i} \bmod n$ and $b_i \leftarrow h^{\alpha_i \cdot n} \theta^{-e_i} \bmod n^2$ for $i \in \{0, 1\}$. Then $\mathcal{S}$ outputs the simulated transcript $((a_i, b_i, e_i, \alpha_i)_{i \in \{0,1\}}, e)$. The distance between the transcript in a real execution and the simulated transcript is equivalent to the distance between α_σ's in the two transcripts. Following the same analysis as in the proof of Theorem 4, this distance is negligible, and the honest-verifier zero-knowledge property follows.

We now focus on the witness-extended emulation property. Suppose that the verifier interacts with P^* and outputs accept with non-negligible probability ε, we construct an emulator as follows. The emulator invokes P^* and receives a_0, a_1, b_0, and b_1. Then the emulator gives $e \leftarrow_\$ [0, 2^\kappa]$ to P^*. If the output α_i's and e_i's from P^* forms an accepting transcript, the emulator needs to use an extractor $\mathcal{E}$ to extract the witness. $\mathcal{E}$ rewinds P^* to Step 2 (challenge phase) and runs it again with a new random challenge from $[0, 2^\kappa]$ until an accepting transcript with challenge e' is obtained. The expected running time for the rewinding is $1/\varepsilon$, and thus polynomial. Since challenges are randomly sampled from $[0, 2^\kappa]$, we have $e \neq e'$ except negligible probability. For $e \neq e'$, we must have $e_0 \neq e'_0$ or $e_1 \neq e'_1$. Hence, σ is set to the bit that $e_\sigma \neq e'_\sigma$. Without loss of generality, we assume that $e_\sigma > e'_\sigma$. $\mathcal{E}$ can thus extract the witness r, such that $u = g^r \bmod n$, from $g^{\alpha_\sigma} \equiv u'^{e_\sigma} a_\sigma \pmod{n}$ and $g^{\alpha'_\sigma} \equiv u'^{e'_\sigma} a_\sigma \pmod{n}$ using the same approach as that in the proof of Theorem 4. Then from $h^{\alpha_\sigma \cdot n} \equiv \theta_\sigma^{e_\sigma} b_\sigma \pmod{n^2}$ and $h^{\alpha'_\sigma \cdot n} \equiv \theta_\sigma^{e'_\sigma} b_\sigma \pmod{n^2}$, we know that $h^{(\alpha_\sigma - \alpha'_\sigma) \cdot n} \equiv \theta_\sigma^{e_\sigma - e'_\sigma} \pmod{n^2}$. According to the same analysis as that in the proof of Theorem 4, we know that the extracted r also satisfies $h^{r \cdot n} \equiv \theta \pmod{n^2}$. The solution s now can be computed via $s \leftarrow \frac{v}{h^r \chi^\sigma}$. Therefore, the extracted witness is (r, σ), and the protocol achieves witness-extended emulation. □