Analytical Evaluation of Time-Based Cryptography

Mohammed Ramadan, Pranit Gadekar, Veit Hagenmeyer and Ghada Elbez
Institute of Automation and Applied Informatics (IAI), KASTEL Security Research Labs, Karlsruhe Institute of Technology (KIT), Eggenstein-Leopoldshafen, 76344, Germany
{mohammed.ramadan, pranit.gadekar9, veit.hagenmeyer, ghada.elbez}@kit.edu

Keywords: Cryptography, Time-Based Cryptography, Time-Lock Puzzle, Verifiable Delay Function.

Abstract: Recent requirements for secure and timely applications have allowed considerable improvements in time-based cryptographic approaches (TBC) to represent the most critical step toward considering time as an essential factor in modern cryptographic protocols. This paper analyzes the performance and security of TLPs and VDFs, highlighting their trade-offs in efficiency and verifiability, focusing on time-lock puzzles (TLPs) and verifiable delay functions (VDFs). Among all TBC approaches, TLP and VDF are relevant in enforcing timed access and verifiable delay in secure systems due to their resistance against parallel computation and predictable delay. Additionally, we present the security analysis, computational efficiency, and implementation of TLP and VDF basic schemes with practical applications, showing that TLPs are simple but suffer from computation delays. In contrast, VDFs are computationally intensive to be evaluated but efficiently verifiable. Subsequently, we deliver recommendations, analysis, and prospective trend scenarios for assessing security analysis and complexity requirements.

1 INTRODUCTION

The information system employs essential cryptographic techniques to ensure data confidentiality, integrity, and authenticity. With the advancement of technology, there is a growing necessity to develop time-based cryptographic methods that address timing considerations. This paper investigates two fundamental concepts: time-lock puzzles and verifiable delay functions. Each idea provides unique advantages that apply to various scenarios (Parno et al., 2012).

Time-based cryptography involves diverse advanced techniques that incorporate temporal elements into cryptographic protocols. These methodologies address critical challenges, such as ensuring fairness in digital transactions, mitigating the risk of premature access to sensitive information, and facilitating time-based access control mechanisms. By integrating time as a variable, these protocols enhance security and manage the temporal constraints of digital interactions effectively. (Dwork and Naor, 1992). A good example is that time-based cryptography in digital rights management will impose restrictions on the usage of digital content for a certain period of time and hence provide a robust method for controlling content distribution (Boneh et al., 2004). Similarly, time-based cryptographic protocols are essential for synchronizing operations within different nodes in distributed systems to make time-dependent processes occur correctly. As modern digital systems grow in complexity and interconnectivity, the importance of developing and understanding time-based cryptographic solutions has become increasingly recognized (Pietrzak, 2019).

Time-lock puzzles, first proposed in (Rivest et al., 1996), prevent information access for a particular time, meaning no one can decrypt the data before this time elapses. It is helpful in many applications, including sealed bid auctions in which all bids must remain secret until a specific time.

Although VDFs are a more recent development in the field of cryptography, whose primary objective is to generate outputs that require a fixed amount of time for computation while allowing for efficient verification, since VDFs can provide predictable delays and proofs of elapsed time (Pietrzak, 2019), there are very substantial prospects for using VDFs in blockchain technology, decentralized systems, and secure time stamping. Their unique properties make them suitable for scenarios where the integrity and verifiability of delayed outputs are imperative. Some other techniques related to time-based cryptography, which were not fully considered within this broad study due to their greater generality or extending beyond the mainstream of time-based cryptography methods, are

Ramadan, M., Gadekar, P., Hagenmeyer, V. and Elbez, G.
Analytical Evaluation of Time-Based Cryptography.
DOI: 10.5220/0013367400003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 624-634
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

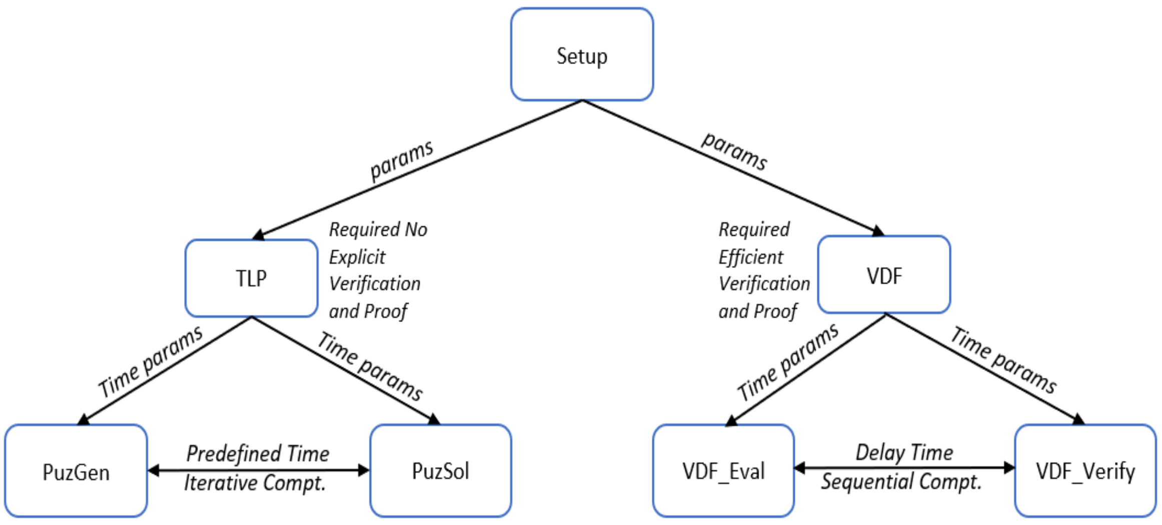
Setup
params
params
TLP
Required No Explicit Verification and Proof
Required Efficient Verification and Proof
VDF
Time params
Time params
Time params
Time params
PuzGen
Predefined Time Iterative Compt.
PuzSol
VDF_Eval
Delay Time Sequential Compt.
VDF_Verify

Figure 1: TBC conceptual framework.

listed below (Pietrzak, 2019) (Schneier, 2007).

- Delay Encryption: Encrypts data with a built-in time delay before it can be accessed.
- Zero-Knowledge Proofs with Time Delays: Uses time delays in zero-knowledge proofs for added security in multi-party computations.
- Blockchain Time-locks: Prevents blockchain actions until a specified time through script-based constraints.
- Proof of Work (PoW) with Delays: Requires completion of work and an enforced delay for certain actions.
- Time-Based One-Time Passwords (TOTP): Generates a short-lived password based on the current time for authentication.
- Timed Commitment Schemes: Binds a value with a time-based release for later disclosure.
- Timed-Release Encryption: Encrypts data to be unlocked only after a specific time, influenced by external factors.
- Time-Based Access Control: Grants or restricts access based on time-specific conditions.

1.1 Related Work

Time-based cryptography (TBC) was first proposed by Timothy et al. (Massias and Quisquater, 1997). This research aimed to discuss the importance of sending encrypted messages in the future and the potential benefits of doing so. This time delay enhances trust by eliminating the need for a trusted third party (TTP) and increasing fairness. Methods for creating the delay include repeated squaring and reducing to modulo in RSA, hash-based techniques, and directed cyclic graphs (DAC). The delay in computation is unavoidable and cannot be bypassed through parallelism mechanisms such as concurrency or multiprocessing power, as the delay construction is based on the concept of sequentiality.

Ronald L. Rivest and Adi Shamir coined the time-lock puzzle (TLP) in (Rivest et al., 1996). The paper states that solving the puzzle to recover the hidden key to decrypt the message should enforce certain time-bound relating it to clock time, not CPU time, through repeated squaring in RSA. The recent idea of batching time lock puzzles delineated in (Abadi and Kiayias, 2021) highlights the importance of opening many puzzles while solving one at any instance by eliminating redundancy and leveraging shared operations among multiple puzzles. The article by Abadi et al. (Abadi et al., 2023) proposed the idea of delegated TLP (DTLP) that allows both the client and the server to delegate their resources and tasks to a helper function, facilitating real-time verification and handling multiple puzzles encompassing the desired variable time. Another vital pragmatic study conducted in (Dujmovic et al., 2024) talks about the multi-instance TLP's where a server is provided with various instances of the puzzle at once; this scheme does not solve all the submitted puzzles immediately but unlocks them at different points of time by chaining them, satisfying the properties of being sequential, restricting parallelism to reduce overhead, and also includes lightweight public verifiable algorithm.

The idea of verifiable delay functions (VDFs) was first formalized by Dan Boneh in (Boneh et al., 2018) based on the idea that it takes several sequential steps for evaluation, but the result can be efficiently verified. The VDF here is constructed using incremental verifiable computation (IVC). In the evaluation phase, the output is produced sequentially. However, the proof is computed to efficiently verify the output by leveraging the power of parallelism in polynomial time. Another prominent idea proposed by Wesolowski (Wesolowski, 2019) adopts the concept of trapdoor VDF, and it can be evaluated by the parties who know the secret. The idea is to keep the trapdoor unknown, and there is no need for a third party to define the setup. The practical study in (Wu et al., 2022) outlines the use cases for VDFs in blockchain applications. It briefly discusses different types of VDFs and their main applications, including timestamping, blockchain technology, randomness beacons, and permission-less mechanisms.

Some critical research works have been proposed in the field of time-based cryptography (Pietrzak, 2019) (Pietrzak, 2019). These approaches have mainly focused on TBCs and their applications from a high-level perspective. However, there is a critical need for formal verification of security proofs to ensure robustness against new attack vectors (Meadows, 2024). Addressing these gaps is crucial for improving time-based cryptographic techniques' efficiency, security, and functionality in real-world scenarios.

1.2 Contributions

This paper presents a comprehensive analytical comparative study of two prominent time-based cryptographic mechanisms, mainly time-lock puzzle (TLP) and verifiable delay function (VDF). The main contributions are stated as follows.

- We provide an in-depth analysis of TLP and VDF, highlighting their underlying principles, construction methods, and operational mechanisms.
- We offer implementation details for TLP and VDF, including pseudocode snippets and performance evaluations.
- We present a comparative performance analysis based on computational complexity, providing insights into the efficiency and feasibility of deploying these mechanisms in real-world scenarios.
- We discuss the security properties of TLP and VDF, analyzing their resistance to various attacks and potential vulnerabilities.
- We explore various application domains for time-based cryptographic systems, emphasizing their relevance and advantages in specific use cases.

Paper Organization. The structure of this paper is as follows: Section 2 delves into the foundational concepts and constructions of basic time-based cryptosystems for both time-lock puzzles (TLP) and verifiable delay functions (VDF). Section 3 details the implementation of TLP and VDF, covering the underlying algorithms and practical considerations. Section 4 offers a comparative performance analysis of TLP and VDF, provides a security analysis, evaluates the strengths and vulnerabilities of both TBC cryptographic approaches, and examines computational complexity. Section 5 presents the results and discussions, highlighting the main findings through comparative analysis, relevant use cases, and future directions. Lastly, Section 6 concludes the paper.

2 TIME-BASED PRIMITIVES

2.1 Time-Lock Puzzle Construction

The implementation of TLP approach is based on repeated squaring modulo an RSA modulus (Rivest et al., 1996). In such schemes, there are three functions: *Setup*, *PuzGen*, and *PuzSol*. The *Setup* accepts the security parameter λ and the desired time T and generates RSA *params* and calculates the time difficulty *t*. *PuzGen* requires *t*, message *m*, and generates a random key *k* fulfilling security requirements and then uses that *k* to encrypt *m* with the *AES* algorithm, and then encrypt *k*, using repeated squaring modulo an RSA modulus, feasibly and optimally through Euler's totient function. The approach of *PuzGen* must be faster than solving the puzzle; the details of how this is achieved will be highlighted in the definition of *PuzGen*. In *PuzSol*, there is a lower bound on *t* to compute the result, as $\phi(n)$ is not efficiently computable in polynomial time, even if n is public, computing $\phi(n)$ from *n* is as hard as finding factors of *n*. The hardness of this problem states that there is no easy and efficient way to solve the puzzle other than solving iterative computations of repeated squaring. However, the number of iterations of squaring can be made dynamic and controlled (Dujmovic et al., 2024).

1. *Setup (λ, T)*: This function inputs the security parameter λ and the Time T, and it generates the RSA *params* for the later puzzle solving.
   - Choose large primes p and q, where $n = p \cdot q$ and $\phi(n) = (p-1)\cdot(q-1)$
   - Compute time difficulty $t = T.S$, and S is the squarings modulo n per sec.
2. *PuzGen (m, t, n)*: This algorithm generates a puzzle. A sender runs this process :
   - Generate key k and hash it to 256 bits using, e.g., SHA-3.

- Message encryption with AES C_m=$Enc_k(m)$
- Key encryption $C_k = k + a^{2^t} \mod n$, where a is random number
- To do this, sender can efficiently compute $C_1 = 2^t \mod \phi(n)$
- Then compute $C_2 = a^{C_1} \mod n$
- Output encrypted key $C_k = (k + C_2) \mod n$
- The sender output (n, a, t, C_m, C_k) as a puzzle.

3. *PuzSol (n, t, a, C_m, C_k)*: to solve the puzzle, the receiver computes the following steps.
   - To recover the key k directly is not feasible; the fastest way known is:
   $$b = a^{2^t} \mod n$$
   - Key k is recovered as $k = (C_k - b) \mod n$
   - Message decryption: M_m=$Dec_k(C_m)$
   - Puzzle is solved, or timeout leads to abort.

2.2 Verifiable Delay Functions Construction

Implementation of VDF, another delay-based primitive, is based on iterated sequential function (ISF) and repeated squaring modulo in RSA (Wesolowski, 2019). The principle behind VDF is sequential, which states that performing t iterations of the computation should be related to clock time, bounding iterations to a time parameter. The VDF defines the key properties of being sequential, efficiently verifiable, and unique. The evaluation takes t iterations of computation, and even with parallelism, there is no possibility of obtaining the result in *t-1* iterations. The VDF algorithm has three steps: *Setup*, which accepts desired time t, security param λ and generates n as composite modulus and hash functions H *and* H_p as public parameters. *Eval* accepts n and message m, t and outputs x, y and proof π. The *Verify* accepts x,t, and π and verifies that the result is valid or invalid. The implementation of VDF is based on efficiently verifying delay functions by using π, and the same π is used for verification rather than computing the y again by the verification algorithm. In this implementation, we adopt Wesolowski's construction of VDF (Wesolowski, 2019).

1. *Setup (λ, T)*: This algorithm inputs the security parameters and the time T.
   - Choose large primes p and q, where $n = p \cdot q$
   - A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{2k}$
   - Function $H_p(m) = \text{next\_prime}(H(m))$
   - Challenge difficulty $t = T \cdot S$, where S is the squaring modulo n per sec.
   - The *Setup* output *params*.
2. *Eval (m, t, n)*: The algorithm inputs n, $m \in \{0,1\}^*$, t and outputs y and π; as follows.
   - Compute $x = H$(m) and assign $y = x$
   - For i = 1 .. t, iteratively compute $y = y^2 \mod n$
   - $l = H_p$(x+y)
   - Compute $\pi = x^{\lfloor 2/l \rfloor} \mod n$
   - The *Eval* returns y and π
3. *Verify (x, l, t, n)*: The algorithm computes the following and verifies the result as valid/invalid.
   - Compute $r = 2^t \mod l$
   - Compute $y = \pi^l \cdot x^r \mod n$
   - If $l = H_p(x+y)$; return $Valid$, else $Invalid$.

3 TBC IMPLEMENTATION

The TBC algorithms are efficiently implemented using Python language and the *pycryptodome* library. This library provides essential functions for generating large prime numbers, AES encryption and decryption, RSA components, and hashing. See the complete code and implementation results with time requirements as well as some screenshots results and appendix data here *Github*. The setup functions for both approaches are also abstracted from pseudocode.

3.1 TLP Implementation

The implementation of time-lock puzzle (TLP) algorithms is divided into three functions. In *Setup*, RSA *params* and the puzzle difficulty t are defined based on the number of iterations corresponding to the desired time. During *PuzGen*, a secret key k is generated randomly. The message m is first encrypted with AES, followed by the efficient encryption of k using the repeated squaring method. The third function, *PuzSol*, involves decrypting k using the repeated squaring method. The puzzle solver must compute t sequential iterations to recover k, highlighting the inherent property that even with multiprocessing capabilities, it is not feasible to decrypt the key in less than T iterations and with $t-1$ iterations. Finally, the decrypted k is used to decrypt the message m using AES. See the result of the TLP implementation in *Github*.

TLP - Pseudo-code:

```
FUNCTION TLP($m, T, \lambda$)
    INPUT: $m$ (message), $T$ (Time), $\lambda$ (security parameter)
    OUTPUT: $C_m$ (encrypted message), $C_k$ (puzzle-encrypted key), $a$ (random number), $t$ (iterations)
                puzzle_gen_time, decrypted_message, puzzle_sol_time

    // PuzzleGen
    START timer
    $a \leftarrow$ Random number in range $[2, n-1]$
    $k \leftarrow$ Generate random 160-bit key
    $C_m \leftarrow$ AES_Encrypt($k, m$)                       // Encrypt the message using AES with key $k$
    $b \leftarrow$ Modular_Exponentiation($a$, $2^t$, $n$)            // Compute $b = a^{2^t} \mod n$
    $C_k \leftarrow (k + b) \mod n$                            // Generate the puzzle-encrypted key
    STOP timer
    puzzle_gen_time $\leftarrow$ End timer - Start timer

    // PuzzleSol
    START timer
    $b \leftarrow a$                                         // Initialize $b$ with the random value $a$
    FOR $i$ FROM 1 TO $t$ DO
        $b \leftarrow (b^2) \mod n$                            // Perform modular squaring $t$ times
    ENDFOR
    $k \leftarrow (C_k - b) \mod n$                            // Recover the key
    TRY
        decrypted_message $\leftarrow$ AES_Decrypt($k, C_m$)      // Decrypt the message using AES
        STOP timer
        puzzle_sol_time $\leftarrow$ End timer - Start timer
    RETURN $\rightarrow C_m, C_k, a$, puzzle_gen_time, decrypted_message, puzzle_sol_time
END FUNCTION
```

VDF - Pseudo-code:

```
FUNCTION VDF($\lambda, T, m$)
    INPUT: $\lambda$ (security parameter), $T$ (delay factor), $m$ (message)
    OUTPUT: $n$ (RSA modulus), $t$ (challenge difficulty), $y$ (result), $h'$ (hashed next prime), $\pi$ (proof), $x$ (hashed input)
            eval_time, verify_time

    // Eval Phase
    START timer
    $x \leftarrow$ Cryptographic hash of $(m, k)$
    $y \leftarrow x$
    FOR $i \leftarrow 1$ TO $t$ DO
        $y \leftarrow (y^2) \mod n$
    ENDFOR
    $h' \leftarrow$ Next prime of $(x + y)$
    $\pi \leftarrow (x^{2/h'}) \mod n$
    STOP timer
    eval_time $\leftarrow$ End timer - Start timer
    RETURN $\rightarrow y, h', \pi, x$, eval_time

    // Verify Phase
    START timer
    $r \leftarrow 2^t \mod h'$
    $y_1 \leftarrow (\pi^{h'}) \mod n$
    $y_2 \leftarrow (x^r) \mod n$
    $y_{res} \leftarrow (y_1 \cdot y_2) \mod n$
    STOP timer
    verify_time $\leftarrow$ End timer - Start timer
    RETURN $\rightarrow h' ==$ Next prime of $(x + y_{res})$, verify_time
END FUNCTION
```

3.2 VDF Implementation

The verifiable delay function (VDF) is also efficiently implemented using Python *pycryptodome* library. The implementation includes a *Setup* function that generates security parameters and defines hash functions. Within the *Eval* algorithm, the message m is processed through the hash function, followed by the application of repeated squaring to the hashed value to produce y. Concurrently, π is generated—despite being resource-intensive, this task can be optimized through parallelism. The calculation of y adheres to the predetermined temporal requirements, while π is calculated efficiently and concurrently. The derivation of π is pivotal to the efficacy of VDF, serving as a proof to validate the correctness of the computed result, thus avoiding the need to calculate the inverse of the function *Eval*. See the result of the VDF implementation in *Github*.

4 PERFORMANCE ANALYSIS

4.1 Complexity Analysis

The experiments were run on an Intel I3 processor with 512GB hard disk drive and 4GB of RAM. The security parameter λ is designated as 2048 bits, ensuring a 112-bit security level, also tested with 3072 bits, providing 128-bit security. The analysis's target duration is 5 seconds (adaptable). Tables 1, 2, and Figure 2 show a detailed evaluation and comparison of the computation complexity of TLP and VDF. In Table 1, R represents the number of runs, and R_{av} is the average of 8 runs, and T is the computational time referencing R_{av}. Since these computational comparisons are among two TBC schemes, VDF and TLP, eight runs are sufficient for an accurate evaluation analysis.

An essential aspect of benchmarking the TBC schemes is the assessment of communication and storage complexities. We evaluated storage and transmission overheads, as detailed in Table 3. In this table, $|params|$ signifies the security parameters, which include $|Z_n|$ as the size of n, while $|Z_p|$ and $|Z_q|$ denote the sizes of the primes p and q, respectively. C is cipher text, t is difficulty/iterations parameter, k is key for TLP. In the case of VDF, π is proof, x is hash function output, and l is a factor for verification. The source code employed for the analysis of communication overhead is available in *Github*.

4.2 Security Analysis

The security of both time-locked puzzles and verifiable delay functions is fundamentally based on the difficulty of sequential computations that cannot be parallelized. Both constructs enforce a computational delay resistant to brute-force and parallelization attacks using the RSA assumption and sequential squaring operations. These properties make TLP and VDF robust tools for applications requiring timed release of information or verifiable computational delays. Refer to (Boneh et al., 2018) (Abram et al., 2024) (Baum et al., 2024) for more details on the security proof and analysis of TBC under various security models and definitions such as one-way, indistinguishability, circuit classes, linearly homomorphic, programmability, and efficiency.

Definition 1. The security of TLPs relies primarily on the infeasibility of parallelizing the puzzle-solving process.

1. Assumptions
   - *RSA Assumption*: The security of TLP relies on the difficulty of factoring the product of two large prime numbers, $n = pq$.
   - *Sequential Squaring*: The time-lock puzzle relies on the difficulty of performing a large number of sequential squaring, which is inherently sequential and cannot be parallelized.
2. Analysis
   - *Puzzle Generation*:
     - The puzzle generation process involves choosing large primes p and q, computing n and $\phi(n)$.
     - The difficulty parameter t is determined by the desired time delay T and the squaring speed S.
   - *Hardness of Puzzle*:
     - The core of the TLP is based on the modular exponentiation $C_k = k + a^{2^t} \mod n$.
     - Solving this puzzle requires computing $a^{2^t} \mod n$, which takes approximately t sequential squarings.
     - Without knowing $\phi(n)$, an adversary cannot speed up this computation, ensuring the delay.
3. Resistance to Attacks
   - *Brute Force*: Due to the sequential nature of the squaring, brute-forcing the solution within the given time frame is computationally infeasible.
   - *Parallelization*: The sequential squaring operation cannot be parallelized, ensuring the delay is enforced even with multiple processors.

Definition 2. The security of VDF squarings is fundamentally based on the guarantee that evaluating the function requires a specific, non-parallelizable sequence of operations.

1. Assumptions
   - *Strong RSA Assumption*: Similar to TLP, the VDF's security relies on the hardness of factoring large composite numbers.
   - *Sequential Work*: The security of VDF relies on the sequential nature of the delay function, which involves repeated squaring or other non-parallelized operations.
2. Analysis
   - *Function Evaluation*:
     – The evaluation function involves computing $y = x^{2^t} \mod n$ for a given input x, which takes t sequential steps.
     – The proof π generated as part of the VDF ensures that the computation was performed correctly.
   - *Hardness of Evaluation*:
     – The evaluation process requires t sequential operations, similar to the squarings in TLP.
     – The hash function H and the next-prime function H_p add complexity to the evaluation, ensuring the delay is enforced.
3. Resistance to Attacks
   - *Parallelization*: Like TLP, the VDF is resistant to parallel attacks due to the inherently sequential nature of the delay function.
   - *Verification*: The output y and proof π can be efficiently verified, ensuring that the delay was correctly applied without needing to repeat the entire computation.

5 RESULTS AND DISCUSSIONS

The experimental results show the practical feasibility and efficiency of both TLP and VDF implementations concerning their computational overhead, security parameters, and the functionality of these cryptographic primitives. The results also show that TLP is relatively easy to deploy and introduces high computational delays since the functions it needs to perform are primarily sequential. VDF allows for efficient verification mechanisms but is resource-intensive during the evaluation phase. TLP ensures resistance to parallelization attacks, while VDF guarantees uniqueness and verifiable delays. In summary, whether to apply TLP or VDF depends on the specific application requirements and resource constraints.

The concept of TLPs described earlier states that a puzzle can be easily constructed, but solving a puzzle takes t sequential steps, and puzzles cannot be solved with the help of multiprocessing or concurrency. The TLP is helpful in applications where the messages can only be recovered after the desired time, such as bit-commitment protocols, auctions, and seal-bid processes. Some application areas where TLP has been deployed and used include time-release TRD, time-release cryptography, and time commitment schemes. Since VDF is inherently sequential, the evaluation phase cannot be parallelized. However, verification is done effectively with the help of proof. This helps reduce trust by using delay-based cryptography and increases fairness. VDFs are based on deterministic delays, which are predictable and unavoidable. Repeated squaring in RSA is the fundamental element used in the security of VDFs. VDF has broad applications in blockchain technology and distributed environments, wherever decision-making applications are concerned. VDF implementations are made by generating a challenge that is hard to verify but easy to verify using the proof-of-work system.

5.1 Complexity Analysis

The computation analysis of TLP and VDF reveals notable differences in efficiency across various stages. Based on the complexity evaluation and comparisons in Table 2 and Figure 2, we notice that TLP is more straightforward to implement but suffers substantial computational delays attributable to sequential operations. The setup time for both TLP and VDF was measured at 1051 *ms*. Generation and evaluation of TLP puzzles required an additional 353 *ms*, and the puzzle solving and verification phase took 5469 *ms*, resulting in a total time of 6873 *ms*. In contrast, the evaluation phase of VDF lasted 5584 *ms*, but its verification process required only 153 *ms*, resulting in a total time of 6788 *ms*.

Based on Figure 2 and the implementation in *Github*, the complexity analysis (running time) of TLP and VDF highlights key differences. TLPs require minimal generation time but are computationally intensive to solve, as evidenced by a 5.66-*second* solution time for repeated squaring operations. VDFs, while taking a similar amount of time for the initial evaluation of 5.49 *seconds*, excel with a much faster verification process, taking only 0.05 *seconds*. This difference makes VDFs more practical for scenarios where swift verification is essential, whereas TLPs are better suited for applications where delayed decryption is prioritized.

Table 1: Notations of operations computation complexity (ms).

$Notations$	$Operations$	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_{av}
T_1	$Setup$	423	437	451	421	467	444	458	417	447
T_2	$C_m : Enc_k(m)$	121	137	135	149	118	154	161	148	141
T_3	$C_k : Enc(k)$	211	198	207	218	196	223	217	198	212
T_4	$F : (T,n) \to S$	1000	1001	981	1002	991	1003	1021	1011	1004
T_5	$k : Dec(C_k)$	5139	5224	5371	5234	5452	5542	5378	5451	5318
T_6	$m : Dec_k(C_m)$	134	156	145	149	153	137	158	161	151
T_7	$H1 : (0,1)^* \to (0,1)^{2k}$	23	31	27	19	24	27	32	22	26
T_8	$H2 : (0,1)^{2k} \to Z_{np}$	14	16	11	18	14	16	11	11	14
T_9	$Eval(n,t,m) \to y,\pi$	5623	5387	5678	5765	5421	5345	5339	5678	5544
T_{10}	$Verify(\pi,t,l,n) \to V/IV$	151	142	137	154	165	152	161	141	153

Table 2: TBC computation efficiency comparisons (ms).

$Algorithms$	TLP	VDF
$Setup$	$T_1 + T_4 = 1051$	$T_1 + T_4 = 1051$
$Gen/Eval$	$T_2 + T_3 = 353$	$T_7 + T_8 + T_9 = 5584$
$Sol/Verify$	$T_5 + T_6 = 5469$	$T_{10} = 153$
$Total$	6873	6788

Table 3: TBC communication efficiency comparisons ($bits$).

$Approach$	$Storage$	$Transmission$	$Total$
TLP	$\|params\| + 2\|C\| + \|a\| + \|t\| + \|k\| = 14992$	$2\|C\| + \|a\| + \|t\| = 7568$	22560
VDF	$\|params\| + \|\pi\| + \|y\| + \|x\| + \|l\| + \|t\| = 18112$	$\|\pi\| + \|x\| + \|l\| = 6976$	25088

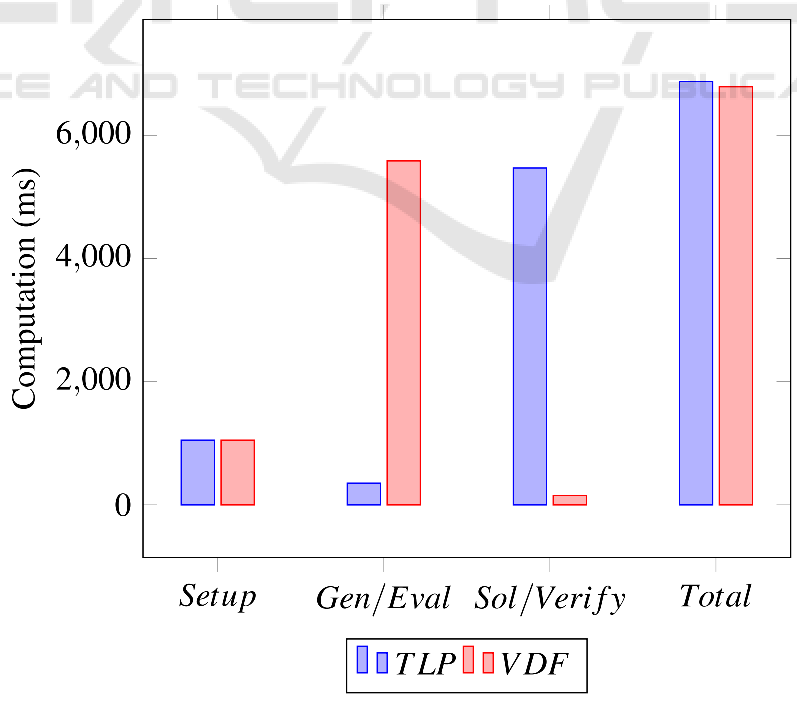

Figure 2: TBC computation efficiency comparisons (ms).

Table 4: Comparison of time-based cryptography techniques.

Aspect	**TLP**	**VDF**
Purpose	Time-delayed data access	Predictable delay with quick verification
Key Features	Sequential, non-parallelizable	Sequential, efficient verification
Complexity	Moderate computational demand	High during evaluation, low in verification
Use Cases	Auctions, time-release encryption	Blockchain, randomness beacons
Advantages	Simple, effective for delays	Fast verification, suitable for decentralized systems
Disadvantages	Vulnerable to parallel attacks	Resource-intensive evaluation
Security	Vulnerable if not implemented properly	More secure, but requires careful design

For the communication and storage overhead, the analysis in Table 3, reveals that the adopted TLP and VDF exhibit minor differences in terms of overhead complexity. TLP consists of a total of 22,560 bits, with a transmission requirement of 7,568 bits. In contrast, VDF encompasses 25,088 bits in total. However, VDFs incur a transmission cost of only 6,976 bits, which enhances their bandwidth efficiency as they primarily transmit proofs instead of complete encrypted data. While TLPs are characterized by a less complex structure, VDFs demonstrate some minor superior efficacy in specific scenarios.

According to the above comparisons, TLP is an efficient algorithm in setup and evaluation but requires longer solving and verification due to its sequential processing nature. VDF, in turn, provides higher verification speed against a longer evaluation phase and is, therefore, suitable for applications requiring a verifiable delay with fast proof verification.

5.2 Use-Case Scenarios

TLPs are mainly applicable for scenarios that require time-bound, such as sealed-bid auctions and time-release encryption. In blockchain and distributed systems where predictable delay and efficient verification are involved, VDFs find their vital applications (Mahmoody et al., 2011). Despite significant advancements in time-based cryptography, notable research gaps persist, particularly in scalability, quantum resistance, and real-world integration. Current implementations of TLPs and VDFs often require substantial computational resources, limiting their scalability and performance in large-scale deployments (Ephraim et al., 2020). Furthermore, with the imminent threat of quantum computing, developing quantum-resistant alternatives to RSA-based schemes is crucial (Baseri et al., 2024). In addition, practical deployment on, for example, blockchain platforms also poses challenges concerning usability and integration with existing infrastructure (Xue et al., 2024). Table 4 summarizes some comparisons between TLP and VDF.

TBC practically permits the implementation of precision time synchronization and packet delivery guarantees for critical infrastructures. A use-case scenario of TBC could be reducing the configuration complexity in time-sensitive networks (TSNs) that provide deterministic communications and synchronization where the time/latency is a storage requirement, e.g., in some high-performance communication systems, TSN will enable ultra-reliable low-latency communication (URLCC) that is essential for applications such as remote surgery, augmented reality, 5G networks, and smart grids (Xue et al., 2024). Also, TSN is specially tailored for Industry 4.0 and the Industrial Internet of Things (IIoT), where machines, robots, and sensors must coordinate their actions with millisecond-level accuracy. Deterministic communication ensures control commands reach machines and sensors without delay, making automated production lines efficient (Ramadan et al., 2016).

A more practical application lies in securely combining TBCs with zero-knowledge proofs (ZKP) to securely hold specific confidential parameters for a predetermined timeframe. This hybrid method effectively counters emerging threats such as forward and backward secrecy and replay attacks, thereby bolstering session security in various applications, including payment systems (Xue et al., 2024). Moreover, long-duration time-lock puzzles, which can span several days, face rising computational costs and the potential obsolescence of cryptographic primitives. To address these challenges, a scalable solution is proposed through hierarchical time-lock puzzles that create shorter puzzles in a layered approach. Alternatively, consistently re-encrypting data with new cryptographic keys can help maintain robust security over prolonged periods (Medley, 2023).

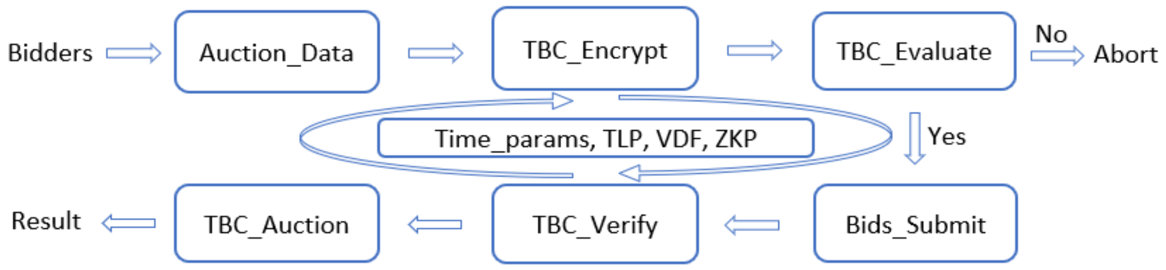

Figure 3: TBC-based auction system - use case.

5.3 Future Directions

The potential future directions of TBC may consider some scenarios that focus on applications related to distributed and decentralized systems that require real-time execution and operations (Pietrzak, 2019). Future directions for TBC may include parallelization of the algorithm for VDF to reduce computational overhead high-security level. This may also be further investigated with partial parallelization and developing special hardware such as ASIC / FPGA for better performance (Ramadan and Raza, 2023). Therefore, integration with blockchain can open up a new horizon for applications in smart contracts and consensus mechanisms. Energy efficiency is another important consideration driving green cryptographic techniques and efficient resource management strategies. Different adversarial model security analyses and formal verification of TBC methods are essential advances that need to be performed to discover vulnerabilities and harden the security of the TBCs.

Recent research regarding post-quantum security has increasingly focused on some alternatives to traditional time-based cryptography. For instance, TBC-based lattice-based constructions that provide secure short signature signatures and guarantee long-term security requirements are being developed alongside the support of industrial standards and other regulations that will allow for broader functionality and interoperability for TBCs (Faleiro et al., 2024). This may also provide viable quantum-resistant solutions for timed-release protocols (TRPs) and verifiable delay functions (VDFs). Nonetheless, such schemes are likely to incur higher time costs. Further investigation is necessary to validate the scalability and practicality of these post-quantum techniques (Shim, 2021).

To address the disadvantages and challenges in TLP and VDF, first, the computation delay in TLP can be avoided by adapting partial parallelization in noncritical parts, optimization of several phases of the sequential squaring process so that efficiency can be improved without compromising the inherent time-lock properties (Rivest et al., 1996). Secondly, resource-intensive VDFs need to adopt specialized hardware that has so far shown great promise in reducing computational resources associated with VDF evaluation (Boneh et al., 2018). Finally, combining VDFs with other TBC techniques, such as zero-knowledge proofs or timed-release encryption, will improve these constructions' security and efficiency. This would allow such a combination to take full advantage of various cryptographic techniques, thereby providing serious machinery for rooting out all the vulnerabilities related to classical and quantum threats. These approaches show the need for further research into post-quantum cryptography and effective formal verification methods to keep TLP and VDF secure and scalable in future cryptographic applications (Medley, 2023).

6 CONCLUSION

In the present paper, we provide an analytically in-depth review of time-based cryptographic mechanisms, namely time-lock puzzles (TLP) and verifiable delay functions (VDFs), regarding their theoretical explanations, implementation details, performance metrics, and security considerations. These results will be helpful as guidance to help future research on the optimization of cryptographic systems and to encourage secure time-bound applications related to blockchain, secure time-stamping, and distributed systems. TLP and VDF have different applications in cryptography, enjoying certain advantages and limitations. TLP is most suitable in cases where the encryption needs to be time-bound. At the same time, VDF is ideal for applications for which verifiable delays and efficient proof mechanisms are necessary. As discussed, future research will focus on analyzing the optimization of computational efficiency in TLP and VDF implementations and exploring new application domains. Improvement in security property analysis and resource reduction overhead remain further areas for investigation.

ACKNOWLEDGEMENTS

This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs (structure 46.23.02).

REFERENCES

Abadi, A. and Kiayias, A. (2021). Multi-instance publicly verifiable time-lock puzzle and its applications. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 541–559. Springer.

Abadi, A., Ristea, D., and Murdoch, S. J. (2023). Delegated time-lock puzzle. *arXiv preprint arXiv:2308.01280*.

Abram, D., Roy, L., and Simkin, M. (2024). Time-based cryptography from weaker assumptions: Randomness beacons, delay functions and more. *Cryptology ePrint Archive*.

Baseri, Y., Chouhan, V., and Hafid, A. (2024). Navigating quantum security risks in networked environments: A comprehensive study of quantum-safe network protocols. *Computers & Security*, page 103883.

Baum, C., David, B. M., Pagnin, E., and Takahashi, A. (2024). Cascade:(time-based) cryptography from space communications delay. In *International Conference on Security and Cryptography for Networks*, pages 252–274. Springer.

Boneh, D., Bonneau, J., Bünz, B., and Fisch, B. (2018). Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer.

Boneh, D., Boyen, X., and Shacham, H. (2004). Short group signatures. In *Annual international cryptology conference*, pages 41–55. Springer.

Dujmovic, J., Garg, R., and Malavolta, G. (2024). Time-lock puzzles with efficient batch solving. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 311–341. Springer.

Dwork, C. and Naor, M. (1992). Pricing via processing or combatting junk mail. In *Annual international cryptology conference*, pages 139–147. Springer.

Ephraim, N., Freitag, C., Komargodski, I., and Pass, R. (2020). Continuous verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–154. Springer.

Faleiro, R., Goulão, M., Novo, L., and Cruzeiro, E. Z. (2024). 1-shot oblivious transfer and 2-party computation from noisy quantum storage. *arXiv preprint arXiv:2410.08367*.

Mahmoody, M., Moran, T., and Vadhan, S. (2011). Time-lock puzzles in the random oracle model. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 39–50. Springer.

Massias, H. and Quisquater, J. J. (1997). Time and cryptography. *US-patent n*, 5:12.

Meadows, C. (2024). Formal verification of cryptosystems. In *Encyclopedia of Cryptography, Security and Privacy*, pages 5–8. Springer.

Medley, L. (2023). A good use of time: Techniques and applications of delay-based cryptography.

Parno, B., Raykova, M., and Vaikuntanathan, V. (2012). How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings 9*, pages 422–439. Springer.

Pietrzak, K. (2019). Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Ramadan, M., Li, F., Xu, C. X., Abdalla, A., and Abdalla, H. (2016). An efficient end-to-end mutual authentication scheme for 2g-gsm system. In *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, pages 1–6. IEEE.

Ramadan, M. and Raza, S. (2023). Secure equality test technique using identity-based signcryption for telemedicine systems. *IEEE Internet of Things Journal*, 10(18):16594–16604.

Rivest, R. L., Shamir, A., and Wagner, D. A. (1996). Time-lock puzzles and timed-release crypto.

Schneier, B. (2007). *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons.

Shim, K.-A. (2021). A survey on post-quantum public-key signature schemes for secure vehicular communications. *IEEE Transactions on Intelligent Transportation Systems*, pages 14025–14042.

Wesolowski, B. (2019). Efficient verifiable delay functions. In *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 38*, pages 379–407. Springer.

Wu, Q., Xi, L., Wang, S., Ji, S., Wang, S., and Ren, Y. (2022). Verifiable delay function and its blockchain-related application: A survey. *Sensors*, 22(19):7524.

Xue, C., Zhang, T., Zhou, Y., Nixon, M., Loveless, A., and Han, S. (2024). Real-time scheduling for 802.1 qbv time-sensitive networking (tsn): A systematic review and experimental study. In *2024 IEEE 30th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 108–121. IEEE.