The Bitcoin Clock for Private Atomic Swaps

Ilya Evdokimov[1,2]
[1]LX Labs LLC,
Delaware, USA
[2]SatsBridge GmbH,
Hannover, Germany
research@satsbridge.com

***Abstract*—Hash Time Locked Contracts (HTLCs) are a fundamental primitive for cross-chain atomic swaps and Lightning Network payment routing, but they depend on a timelock that is typically supplied by the underlying blockchain. In private, zero-knowledge based settings such as Aztec, no such timelock is available inside private circuits, which forces reliance on trusted sequencers or external oracles. This paper proposes a method for constructing verifiable timelocks inside zero-knowledge circuits by reusing Bitcoin's accumulated proof-of-work as a computational reference clock. We describe the protocol, discuss its security margins, and report preliminary benchmarking results for the proving cost of the resulting circuits.**

***Index Terms*—Bitcoin, proof-of-work, hash-time locked contracts, zero knowledge.**

I. Introduction

With the advent of blockchain protocols, a dedicated type of smart contract called the Hash Time Locked Contract (HTLC), or simply "atomic swap," has demonstrated significant adoption among other types of contracts. As a general concept, smart contracts were introduced by Szabo [1], [2] well before their first implementations appeared in decentralized protocols such as Bitcoin [3] and the numerous forks of Bitcoin's codebase. The initial success of HTLC applications was owed to high code compatibility across forks; nowadays they form a basic building block for cross-chain applications and payment routing in the Bitcoin Lightning Network.

In the beginning, atomic swaps did not receive considerable attention. The original definition "atomic transfer" was first introduced by Tier Nolan on the BitcoinTalk forums in 2013 [4]. Nolan described the basic principles of cross-chain cryptocurrency swaps using simple trades across different types of blockchains at that time, primarily forks of the Bitcoin Core codebase. He also pointed to possible Bitcoin Scripts that could facilitate the proposed transactions.

Between 2013 and 2017, during the so-called era of altcoins, it was centralized exchanges which were mainly facilitating trading functions for exchanging cryptocurrencies and tokens. Atomic swaps have not been implemented in practice until September 2017, when Litecoin founder Charlie Lee announced over Twitter [5] the successful execution of an atomic swap between Litecoin and Bitcoin. Litecoin is an "altcoin" and inherits from Bitcoin all its main architectural features, including Unspent Transaction Outputs and tree-like transaction structures with Bitcoin Scripts executing on their branches.

However, the dominance of centralized exchanges over the ways of moving and exchanging value in a decentralized fashion could not lower the importance of HTLC contracts. Our earlier evaluation [6] of HTLC usage in the Bitcoin Lightning Network indicated that this application alone may generate a large amount of traction for this type of contract, not counting Decentralized Finance (DeFi) applications. According to the protocol specification, each channel can have 483 "in-flight" HTLCs [7]. This means that if the number of pending payments equals that amount, the channel may be considered "frozen": it cannot route any new payments until some pending HTLC resolves with a known preimage or timeout. With 74,261 active channels, $74{,}261 \times 483 = 35{,}868{,}063$, or approximately 36 million pending contracts could be enforced by Bitcoin Proof-of-Work at any time.

Besides the well-known "American option" problem that arises when different assets are being exchanged [8], [9], another kind of HTLC failure mode was researched by Zakhary et al. [10]. The authors pointed out an adversarial condition in which one of the participants may not be able to recover their funds in time due to possible persistent failures before the refund window expires. To address this issue, a permissionless open network of witnesses was introduced for coordinating atomic cross-chain transactions. Overall, HTLCs serve as a reliable and well-known secure primitive for various blockchain applications.

A. Problem

To work, the HTLC primitive relies on two conditions: a hash that may be unlocked by a preimage, and a "timelock," which normally relies on the blockchain protocol itself. Self-referential links work equally well across different blockchains because relative timelocks are set by users on each new swap to facilitate refund as a fall-back option. These timelocks underpin the role of regularly produced blocks as internal clock ticks measuring intervals for the appropriate execution flows during the swap. In Proof-of-Work systems such as Bitcoin, "ticking" in terms of physical time is imprecise and prone to drift, because block intervals depend on the available hash power and the current difficulty level [3]. Proof-of-Stake protocols such as Ethereum may be considered as operating in physical time, and they indirectly rely on the Network Time Protocol (NTP) via node operators who must keep their system clocks accurate (usually via NTP). The

consensus algorithm allows minor drift of participants' clocks and imposes penalties to discourage malicious manipulation of block timestamps.

In zero-knowledge based chains, internal clocks may not be provided at all in private chains such as Aztec. The documentation [11] specifically mentions that *... private functions execute on the user's device before the transaction is submitted, so they cannot know which block will include the transaction*. However, similarly to other Ethereum Virtual Machine ZK rollups, on Aztec *... public functions execute on a sequencer who knows the current block's timestamp and number, making these values accessible*, therefore outsourcing the clock function to an external trusted oracle.

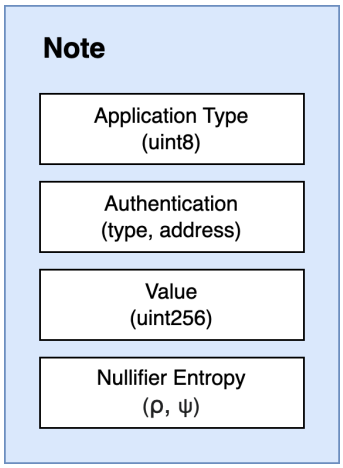

Fig. 1. UTXO note structure [12].

Without timelocks, the HTLC becomes unavailable for ZK-based applications. The diagram in Fig. 1 shows a `Note` structure in a UTXO-based ZK rollup introduced by Moore and Gandhi [12]. For swaps, in addition to the Authentication field, it may contain a Revocation key, but this will potentially lead to a race condition for the entire UTXO proof generated with such a `Note` item (according to [12], it is meant to be a two-input, two-output transaction). As long as the opposite side of the swap supports a timelock, safe margins may be introduced and settlement after submission of the private UTXO proof may be ensured before it expires. However, a race condition remains. Due to latencies, failures, or other adversarial conditions, this issue must be addressed to ensure secure and private decentralized swaps and preserve their atomicity as a whole.

B. Related Work

The sequential nature of blockchains has attracted researchers from different fields. Ladleif and Weske [13] extensively explored how time is represented in blockchain-based process execution for Business Process Management (BPM). They introduced an abstract formal model of a blockchain, analyzed the relations between various internal measures such as timestamp and height, and investigated their properties. They also discussed how the proposed time measures can be used to implement generic temporal constraints found in business process models. At least two groups of authors, Regnath et al. [14] and Fan et al. [15], leveraged blockchain in the Internet of Things (IoT) application field. Regnath et al. [14] focused on using time information in block headers, whereas Fan et al. [15] focused on a blockchain-based protocol for clock synchronization.

An entire field of research, time-release cryptography, focuses on ways to encrypt a message so that it cannot be decrypted until a predetermined amount of time has passed. The problem was introduced by Timothy May in an early cypherpunk research piece from 1993 [16], and Rivest, Shamir, and Wagner later proposed two different solutions to it in 1996 [17]. They introduced the definition of time-lock puzzles and their solutions relied either on repeatable computations or on trusted agents. The authors essentially proposed an adaptive "proof-of-work" and offered to adjust the difficulty of the time-lock puzzle to a desired level. Decades later, Li et al. [18] employed threshold cryptography (i.e., trusted agents) combined with smart contract execution on Arbitrum for time-release cryptography purposes.

After Rivest, Shamir, and Wagner's foundational work, a number of research teams focused exclusively on time-lock puzzles. Bitansky et al. formalized the approach and suggested constructing time-lock puzzles from randomized encodings, demonstrating that time-lock puzzles can be built under a variety of cryptographic assumptions [19]. The authors specifically concluded that their approach also allows the design of proof-of-work puzzles.

Similarly to Rivest, Shamir, and Wagner, Jerschow and Mauve [20] explicitly highlighted that their RSA time-lock puzzle scheme is non-parallelizable. Academic research largely omits direct applications of such time-lock puzzle schemes to so-called Application-Specific Integrated Circuits (ASICs) resistant algorithms and cites other use cases such as e-voting or document certification. A notable example here is the Grin cryptocurrency implementation of Proof-of-Work consensus. It relies on collision-resistant hash functions and graph properties to force the solver to store a large graph in RAM, thereby making the algorithm memory-bound rather than compute-bound. The foundation of Grin's approach traces back to the work of Mahmoody et al. [21], who focused on the construction and formal analysis of publicly verifiable proofs of sequential work. These proofs allow a prover to demonstrate that a certain amount of sequential time has elapsed since receiving a challenge.

The development of Proof-of-Stake blockchains introduced a new kind of problem: trustworthy public randomness generation in a trustless environment. To address this problem in an energy-efficient way, Wesolowski introduced efficient Verifiable Delay Functions (VDFs) [22]. Although the author specifically mentions the limited accuracy of his theoretical model when it comes to accelerating VDFs in real physical time through dedicated hardware, this work marks a shift both in definitions and in the applied space of time-release cryptography.

In the narrow field of atomic swaps, time-lock puzzles have recently found their applications. Tairi et al. introduced an interactive Broadcast Time-Lock Exchange (BTLE) [23]. The authors addressed two major issues of HTLCs: the impossibility of matching two blockchains with different basic hash functions or with no scripting (i.e., no smart contract support), and the lack of support for multiparty exchanges.

When it comes to using Proof-of-Work as a cheap, verifiable way to construct time-lock encryption, Liu et al. introduced definitions and suggested a method to encrypt a message such that it can only be decrypted after a certain deadline has passed, with the key property that even receivers with relatively weak computational resources can decrypt immediately after the deadline, without interaction [24]. Their scheme introduces the concept of computational reference clocks built on top of Bitcoin, combined with Succinct Non-interactive Arguments of Knowledge (SNARKs). The authors researched a witness encryption (WE) scheme, and their work is the closest existing research result to the present paper. We aim to reuse Liu et al.'s definitions and to leverage the existing domain-specific language (DSL) Noir for the purpose of programmable SNARKs and easy demonstration of the developed idea.

II. Noir-Based ZK-Timelocks Proposal

In the present paper we introduce a new way to design timelocks in "private functions" or circuits that can be proven without centralized sequencers or the use of external oracle data. The demo source code provided along with this publication contains Noir source code [25] that can be used primarily when working with the Aztec zero-knowledge (ZK) ecosystem; however, the method should be equally valid for other ZK stacks that support proving SHA-256 hashes in zero knowledge.

A. Proof-of-Work Recap

The Proof-of-Work proposal by Nakamoto [3] relies on the collision-resistant hash function SHA-256 and a difficulty calculation rule. This rule requires the prover to provide only hashes with a given number of leading zeroes as a proof of the resources consumed for calculating that hash. Figure 2 illustrates the most recent block on the right-hand side. The hash of this new block may be accepted only if it contains the required number of leading zeroes for the corresponding difficulty level, which is stored in the *block header* of the preceding block; the preceding block in turn contains the hash of the block before it, and so on.

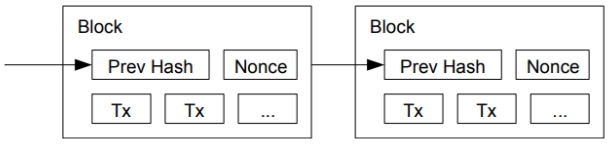

Fig. 2. Diagram of two "chained" neighbouring blocks with verifiable precedence, from Nakamoto [3].

The original whitepaper by Nakamoto [3] does not mention an important balancing component of the Proof-of-Work consensus. The difficulty adjustment mechanism serves to ensure the continued extension of the chain by miners in case finding new hashes becomes too expensive and they shut down their dedicated equipment. It relies on timestamp values in block headers that are allowed to drift within bounds set by the protocol rules. The difficulty adjustment averages timestamp values across 2016 blocks and changes the difficulty parameter so that future expected blocks come at intervals of approximately 10 minutes.

B. Verifiable Accumulated Work

Once the computational effort has been expended to make a block satisfy the difficulty rule, the block cannot be changed without redoing that work. When later blocks are found and extend the tip, the work needed to change a specific block must include redoing all the blocks after it. This constitutes an important rule for resolving potential forks, where competing extensions of the same block are mined by different miners, either intentionally or by chance. In this scenario, the peer-to-peer network decides whether to accept a new block by calculating the greatest total proof-of-work behind it. Honest nodes automatically reorganize to this heavier chain, discarding blocks (orphans) from shorter branches.

This verifiable accumulated work, together with the convergence of the network to a single canonical chain, lays the foundation for Liu et al.'s *computational reference clocks* [24]. As opposed to real physical clocks, which must rely on trusted third parties, Bitcoin's clock has only an implicit connection to physical time and is imprecise (precision is an important property for BPM applications [13]).

The Bitcoin miners perform a series of repeating computations, contributing significant computational resources for the gradual extension of the Bitcoin blockchain, as shown in Fig. 3. As long as the chain keeps extending, the Bitcoin clock can serve to prove an interval of arbitrary length, and the cost associated with rewriting "clock ticks" increases exponentially.

The protocol shown in Fig. 4 may be considered along with the `Note` diagram in Fig. 1: the simplest "time-locked" `Note` would look like a note with $Address = \text{Poseidon}(zero\_block,\ n\_blocks)$. A UTXO circuit would require additional private inputs for the 32-byte SHA-256 hash and a minimal set of block headers that satisfy the swap protocol's security requirements.

Algorithm 6 provides additional detail for the schemes shown in Figs. 4 and 5. The prover commits to an anchor block and a target chain length; the circuit then verifies n_blocks of accumulated proof-of-work extending the anchor, with per-block difficulty constrained to a bounded window Δ_D around the anchor's difficulty.

The minimum-difficulty assertion on the anchor is a sanity check that prevents the prover from selecting an anchor from an unreasonably low-difficulty era. Depending on the desired security profile, the circuit may additionally enforce height and interval checks tied to Bitcoin's 2016-block difficulty

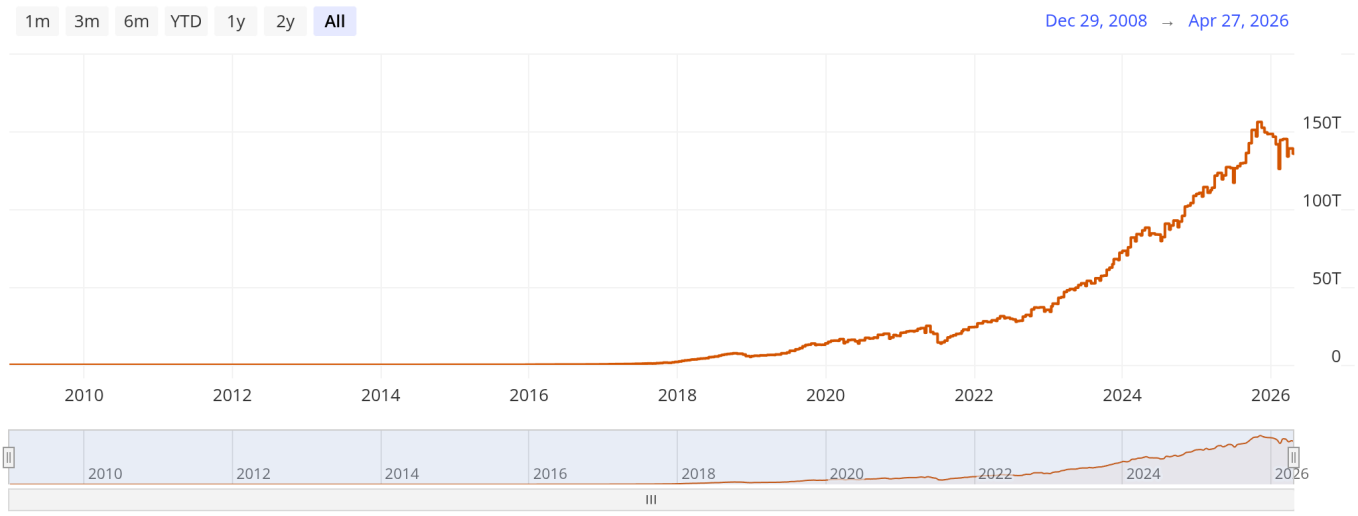

Fig. 3. The Bitcoin network demonstrates an exponential growth in the difficulty measure, which corresponds to an increasing amount of resources deployed by miners [26].

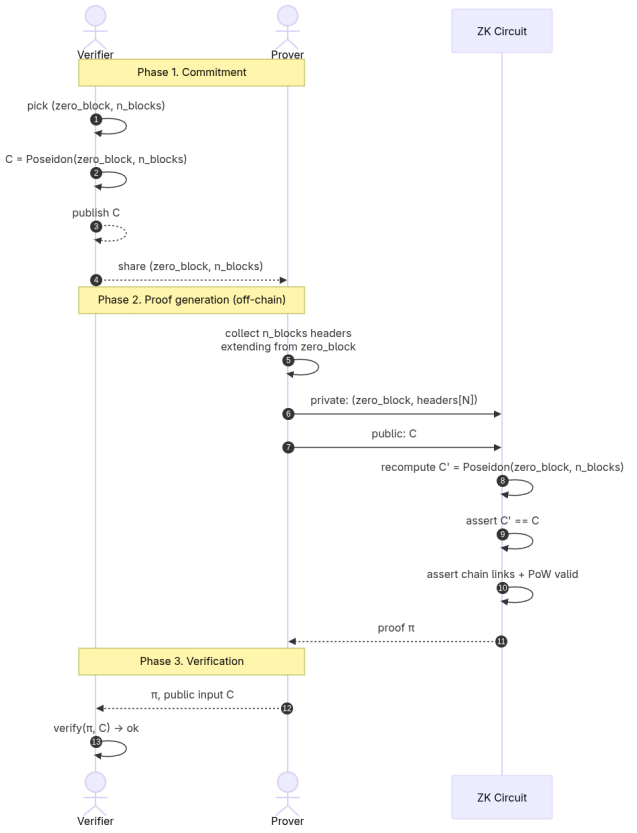

Fig. 4. The atomic swap timelock protocol setup and execution.

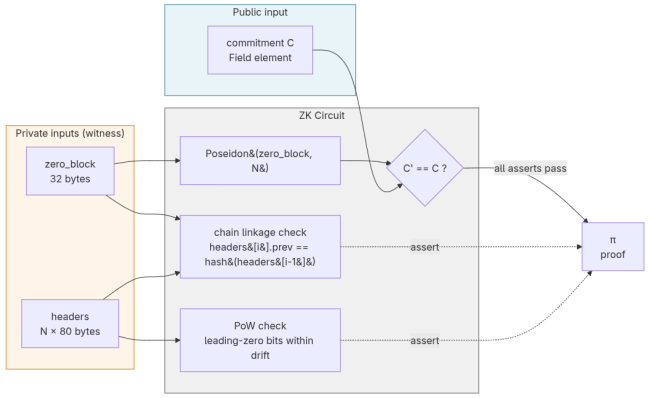

Fig. 5. Required circuit input structure for private atomic swaps.

adjustment period, since a chain segment crossing a retarget boundary can legitimately exhibit a difficulty change of up to a factor of four and would otherwise be rejected unless the circuit itself will allow variability in difficulty values for neighbouring blocks.

Alternatively, the allowed difficulty drift can be omitted entirely by excluding retarget blocks from the proven segment and fixing the difficulty as a protocol parameter agreed upon at swap setup.

III. Benchmarking

Liu et al. [24] highlighted the technical complexity of the proposed scheme and its experimental character. In the present work, the Noir DSL with the Barretenberg backend was used for the implementation of Algorithm 6, which takes around 120 lines of Noir code including comments and compiles into circuits that may be used for proving timelocks. As Liu et al. [24] specifically mention, for an end user, proving a

Require: Public input: $commitment \in \mathbb{F}$
Require: Private inputs: $zero\_block \in \{0,1\}^{256}$, $n\_blocks \in \mathbb{F}$, $headers \in (\{0,1\}^{640})^{N_{\max}}$
Constants: $N_{\max}$, $D_{\min}$, Δ_D

{Anchor difficulty check}
$z_0 \leftarrow \text{LEADINGZEROBITS}(zero\_block)$
assert $z_0 \geq D_{\min}$

{Bind circuit to the agreed commitment}
$h_0 \leftarrow \text{Poseidon2}(zero\_block)$
$c \leftarrow \text{Poseidon2}(h_0, n\_blocks)$
assert $c = commitment$

{Verify accumulated work over n_blocks headers}
$prev \leftarrow zero\_block$
$active \leftarrow$ **true**
for $i = 0$ **to** $N_{\max} - 1$ **do**
  if $i = n\_blocks$ **then**
    $active \leftarrow$ **false**
  end if
  $H_i \leftarrow \text{SHA256}(\text{SHA256}(headers[i]))$
  $z_i \leftarrow \text{LEADINGZEROBITS}(H_i)$
  if $active$ **then**
    assert $\text{PREVHASH}(headers[i]) = prev$
    assert $|z_i - z_0| \leq \Delta_D$
    $prev \leftarrow H_i$
  end if
end for

Fig. 6. Pseudocode representation of Noir circuit for a computational reference clock.

timelock using Bitcoin has negligible costs, and our analysis confirms this. We demonstrate it further using Noir scripts compiled with Barretenberg binaries into circuits and also give an estimation of the circuit parameters depending on the number of Bitcoin block headers being proved and hence the timelock's approximate expiration interval.

A. Benchmark Setup

A project `noir_timechain` proves that a chain of Bitcoin block headers extends a committed anchor block, with PoW difficulty staying within 3 leading zero-bits of the "anchor block."

It uses a Poseidon2 commitment over `(zero_block, n_blocks)` for public input and consumes private inputs that include the anchor block hash, `n_blocks`, and a fixed-length array of `MAX_CHAIN_LEN` 80-byte headers (unused slots may be zero-padded).

Published repo has a constant `MAX_CHAIN_LEN` set to 6, i.e. ≈ 60 mins. Active `n_blocks` for `Prover.toml` is 6, anchored at Bitcoin mainnet height 946920 with the chain extending through height 946926. Headers were fetched from `mempool.space` block explorer but may be also obtained from any explorer or running Bitcoin node.

For the benchmarks, we employed a relatively powerful, consumer-grade personal computer. Its system parameters are given in Table I.

TABLE I
HARDWARE AND SYSTEM PARAMETERS

OS	Linux, x86_64
CPU	AMD Ryzen 7 4700U (8 logical cores)
Memory	62 GiB
nargo	`1.0.0-beta.9`
barretenberg	`1.0.0-nightly.20250723`
scheme	`ultra_honk`
oracle hash	`poseidon2`

B. Proving Costs

Tables II and III demonstrate proving costs and circuit sizes, respectively.

Prove dominates wall-clock cost; the mean prove time of 1.766 s is roughly 4 times longer than witness generation (419 ms) and 45 times longer than verification (39.64 ms). For end-to-end UX, prove is the only number worth tuning, although for most applications timelock proving may be considered insignificant.

Verification is small and roughly constant with regard to the length of the timelock. UltraHonk verifier cost does not depend on circuit size at this scale.

TABLE II
PER-PHASE WALL-CLOCK (5 ITERATIONS, 1 WARMUP)

Phase	Runs	Mean
compile	1	902.13 ms
gates	1	284.31 ms
write_vk	1	759.68 ms
execute	5	419.17 ms
prove	5	1.766 s
verify	5	39.64 ms

TABLE III
CIRCUIT AND ARTIFACT SIZES

item	**value**
Circuit size (UltraHonk gates)	**98 578**
ACIR bytecode (gzipped JSON)	437.5 KiB
Verification key	1.8 KiB
Proof	15.9 KiB
Public inputs	32 B

Static circuit size with `MAX_CHAIN_LEN` set to 6 blocks requires 98 578 gates vs. 52 318 for 3 blocks, which yields predictably linear growth. The dominant cost per slot is the double SHA-256 of the 80-byte header inside the Proof-of-Work validation routine. Per-slot marginal cost is 15k gates, so further raising `MAX_CHAIN_LEN` has a predictable linear effect on prove time. Adaptive selection of the timelock may be important for both business requirements and larger circuits where the timelock is featured among other functions.

Witness generation scales sub-linearly when fewer than `MAX_CHAIN_LEN` blocks are actually provided for proving. The Proof-of-Work validation routine recognizes zero-padded slots and short-circuits the `active` flag; nonetheless, the ACIR opcode count and prove time scale with `MAX_CHAIN_LEN` (the static loop bound), so `n_blocks` does not affect prove cost. Since block headers are provided via private inputs, this may afford additional privacy for the proving side when the general setup allows more expensive circuits.

Proof and verification key sizes are independent of `n_blocks` (15.9 KiB / 1.8 KiB), which is consistent with the proving system's (UltraHonk) properties.

IV. Conclusion

The paper demonstrates how Bitcoin's Proof-of-Work may be re-used for decentralized timelock validation without trusted oracles in private atomic swaps. The basic assumption for the Bitcoin clock is energy expenditure backing existing Proof-of-Work consensus and corresponding verification rules that were partially implemented inside a Noir program published on GitHub [25] along with benchmarking scripts.

References

[1] N. Szabo, "Smart contracts: Building blocks for digital markets," *EXTROPY: The Journal of Transhumanist Thought*, no. 16, 1996, https://www.truevaluemetrics.org/DBpdfs/BlockChain/Nick-Szabo-Smart-Contracts-Building-Blocks-for-Digital-Markets-1996-14591.pdf.

[2] ——, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997, https://firstmonday.org/ojs/index.php/fm/article/view/548.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[4] T. Nolan, "Alt chains and atomic transfers," 2013, https://bitcointalk.org/index.php?topic=193281.msg2003765.

[5] C. Lee, "Litecoin and bitcoin atomic swap announcement," 2017, https://x.com/SatoshiLite/status/911328252928643072.

[6] S. Blog, "The most used smart-contract ever," September 17, 2023, https://blog.satsbridge.com/the-most-used-smart-contract-of-all-times-2f749428adb7.

[7] "Bolt02: Peer protocol," April 28, 2026, https://github.com/lightning/bolts/blob/master/02-peer-protocol.md.

[8] R. Han, H. Lin, and J. Yu, "On the optionality and fairness of atomic swaps," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, ser. AFT '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 62–75. [Online]. Available: https://doi.org/10.1145/3318041.3355460

[9] J. Xu, D. Ackerer, and A. Dubovitskaya, "A game-theoretic analysis of cross-chain atomic swaps with htlcs," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 2021, pp. 584–594.

[10] V. Zakhary, D. Agrawal, and A. El Abbadi, "Atomic commitment across blockchains," *Proc. VLDB Endow.*, vol. 13, no. 9, p. 1319–1331, May 2020. [Online]. Available: https://doi.org/10.14778/3397230.3397231

[11] "Aztec.nr - framework description - global variables," April 28, 2026, https://docs.aztec.network/developers/docs/aztec-nr/framework-description/globals.

[12] "L2 ethereum zk rollup for private and compliant transactions," February 12, 2024 - Draft 0.6.0, https://polybase.github.io/zk-rollup/whitepaper.pdf.

[13] J. Ladleif and M. Weske, "Time in blockchain-based process execution," in *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, 2020, pp. 217–226.

[14] E. Regnath, N. Shivaraman, S. Shreejith, A. Easwaran, and S. Steinhorst, "Blockchain, what time is it? trustless datetime synchronization for iot," in *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, 2020, pp. 1–6.

[15] K. Fan, S. Sun, Z. Yan, Q. Pan, H. Li, and Y. Yang, "A blockchain-based clock synchronization scheme in iot," *Future Generation Computer Systems*, vol. 101, pp. 524–533, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18326657

[16] T. C. May, "Timed-release crypto," February 1993, http://www.hks.net/cpunks/cpunks-0/1460.html.

[17] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Tech. Rep., March 1996.

[18] Z. Li, S. Majumdar, and E. Pournaras, "Send message to the future? blockchain-based time machines for decentralized reveal of locked information," *IEEE Transactions on Network and Service Management*, vol. 22, no. 6, pp. 6112–6127, 2025.

[19] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters, "Time-lock puzzles from randomized encodings," in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ser. ITCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 345–356. [Online]. Available: https://doi.org/10.1145/2840728.2840745

[20] Y. I. Jerschow and M. Mauve, "Offline submission with rsa time-lock puzzles," *2010 10th IEEE International Conference on Computer and Information Technology*, pp. 1058–1064, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:14206230

[21] M. Mahmoody, T. Moran, and S. Vadhan, "Publicly verifiable proofs of sequential work," in *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ser. ITCS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 373–388. [Online]. Available: https://doi.org/10.1145/2422436.2422479

[22] B. Wesolowski, "Efficient verifiable delay functions," Cryptology ePrint Archive, Paper 2018/623, 2018. [Online]. Available: https://eprint.iacr.org/2018/623

[23] F. Barbara, E. Guglielmino, N. Murru, and C. Schifanella, "Btle: Atomic swaps with time-lock puzzles," *Journal of Mathematical Cryptology*, vol. 19, no. 1, p. 20240044, 2025. [Online]. Available: https://doi.org/10.1515/jmc-2024-0044

[24] J. Liu, T. Jager, S. A. Kakvi, and B. Warinschi, "How to build time-lock encryption," Cryptology ePrint Archive, Paper 2015/482, 2015. [Online]. Available: https://eprint.iacr.org/2015/482

[25] I. Evdokimov, "Noir-timechain," 2026, https://github.com/evd0kim/noir-timechain.

[26] "Bitcoin visuals: Bitcoin difficulty," 2026, https://bitcoinvisuals.com/chain-difficulty.