CHAIN: Generalized Escrow-Based High-Throughput Payment Framework: A Hash-Based Approach

Aashish Paliwal

January 2025

1 Introduction

Blockchain protocols for electronic payments offers secure payment transaction. However, their transaction throughput pales in comparison to centralized payment processors like Visa. Additionally, transaction fee associated with each payment represent a significant obstacle to the widespread adoption of blockchain based payments for everyday use, particularly for micropayments or high-frequency interactions due to additional mental overhead.

A major issue is that, despite increased throughput on various chains, all on-chain transactions remain permanently recorded on the blockchain, causing significant clutter. A viable solution to mitigate this issue is by reducing the storage burden of high-frequency transactions on-chain and instead using verifiable payment mechanisms that cryptographically link transactions while recording only the first and last tokens on-chain.

We present a modified PayWord [1] scheme by Rivest and Shamir. It is a credit-based unidirectional system where users deposit funds into an escrow contract. This mechanism enables thousands of off-chain transactions with only two on-chain interactions: the initial deposit and final settlement. Unlike traditional payment channels, this scheme achieves unparalleled efficiency by validating tokens through a single hash operation, making it suitable for both micropayments and high-value transactions.

2 Background

2.1 Hash chain

A hash chain is a cryptographic structure in which each element of the chain is generated by applying a hash function to the previous element. Starting from an initial random value (often referred to as the "seed").

In Figure 1, h^n(seed) represents the trust anchor, which serves as the root of the hash chain. Each subsequent value is cryptographically linked to the trust

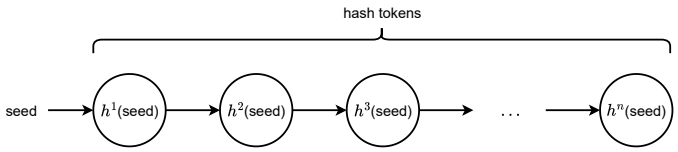

Figure 1: Illustration of a hash chain.

anchor through iterative hashing.

Hash chains ensure tamper-proof verification because modifying any element in the chain results in a mismatch when the final hash is compared to the trust anchor. This cryptographic property guarantees that any alteration of the chain invalidates the entire structure, ensuring its integrity.

2.2 PayWord

PayWord, introduced by Rivest and Shamir, is a credit-based micropayment scheme designed to minimize computational overhead for small transactions. It uses hash chains, where each token in the chain is derived by applying a cryptographic hash function to the subsequent token, starting with a random final value. A user establishes an account with a broker, who issues a digitally signed PayWord certificate, authorizing the user to create and commit hash chains to vendors. The user provides the root of the hash chain (trust anchor) to the vendor, and payments are made by sequentially revealing tokens from the chain. The vendor verifies each token by hashing it iteratively.

3 Generalized Framework: Design and Architecture

The proposed payment framework builds upon the PayWord scheme while extending its functionality to support a wide range of payment types, from micropayments to high-value transactions. At its core, the framework integrates an escrow-based smart contract that ensures secure, trustless, and efficient handling of funds.

3.1 Core components

- **Escrow smart contract**: Act as an secure intermediary which stores funds that can only be redeemed programatically. Funds are controlled by code hence even admin cannot steal funds reducing the trust requirement. It also provides a fallback mechanism for user to claim funds back in case merchant is malicious and refuses to close the channel using time lock.

- **Trust anchor**: Represents the tip of the hash chain. It is created and submitted along with funds at the time of channel creation. It ensures that each token is cryptographically linked to the next one until it reaches trust anchor.

- **Proportional Fund Release Mechanism**: The mechanism validates the submitted final token by iteratively hashing it to derive the trust anchor. The smart contract ensures that the final token submitted by the merchant, when hashed n times (n being the number of tokens used), equals the trust anchor. This guarantees that for each token spent, there is one hash computation on-chain during redemption. Since each token is cryptographically linked in the hash chain, intermediate tokens are not required on-chain. This reduces the storage and computational load while still ensuring trustless verification and fair fund release.

  It determines the payment amount by calculating the ratio of spent tokens (used hashes) to the total chain length.

$$\text{Payment Amount} = \frac{\text{Escrowed Funds} \times \text{Spent Tokens}}{\text{Total Chain Length}}$$

- **Participants**:
  - User: Deposit funds, initialize hash chain, consume goods or service and share hash tokens off-chain.
  - Merchant: Delivers goods or services and submits the final token to claim proportional funds.
  - Escrow smart contract: Automates fund release based on token validation and ratio calculations.

3.2 Workflow

1. Channel setup: The user initiates a payment channel by depositing funds into a smart contract and specifying a trust anchor, the total number of tokens in the hash chain, and a timeout period. The smart contract locks the funds, providing assurance to both parties of secure and conditional payment.

2. Token exchange: Tokens from the hash chain are exchanged off-chain during interactions between the user and merchant. Each token is cryptographically derived from the subsequent token in the chain, ensuring that all tokens are linked back to the trust anchor.

   If multiple tokens are sent in one batch, the user includes the number of tokens k alongside the current token. This allows the merchant to perform iterative hashing k times, ensuring that:

$$h(h(\ldots h(\text{current token}))) = \text{previous token}$$

Such off-chain verification avoids the need to submit intermediate tokens to the blockchain while maintaining efficiency and cryptographic security.

3. Transaction finalization: The merchant submits the final token from the chain, representing the portion of services provided or goods delivered. The smart contract verifies the token by iteratively hashing it and comparing the result to the trust anchor. Based on the verified number of tokens consumed, the contract calculates and disperses the corresponding funds to the merchant, while refunding any remaining balance to the user.

3.3 Example scenario

- Channel creation (*cf.* Figure 2): A user deposits 1 cBTC into the escrow smart contract and establishes a trust anchor for a hash chain of 1000 tokens.

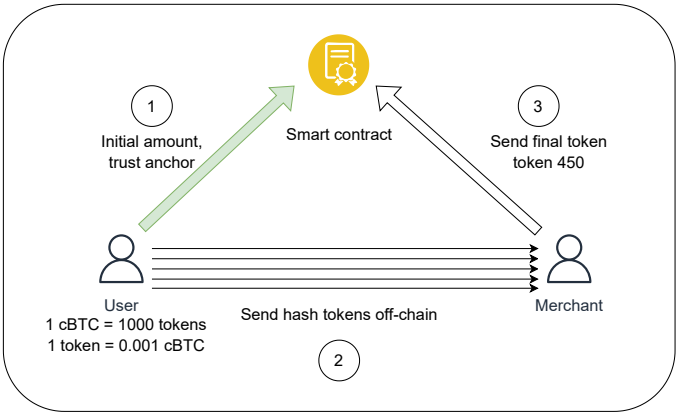

Figure 2: Illustration of the CHAIN payment workflow, where the user commits an initial amount and trust anchor to a smart contract, sending hash tokens off-chain incrementally. Finally, merchant submits final token 450 for verification, representing 0.450 cBTC.

- Token consumption (*cf.* Figure 2): The user interacts with the merchant, consuming three tokens in the hash chain. The remaining seven tokens are unused.

- Finalization (*cf.* Figure **??**): The merchant submits the last token of the consumed portion (third token in the chain) to the escrow contract. The contract verifies the submission by hashing the token three times to match the trust anchor:

$$H_{450} \xrightarrow{\text{hash}} H_{449} \ldots \xrightarrow{\text{hash}} H_1 \xrightarrow{\text{hash}} H_{\text{anchor}}$$

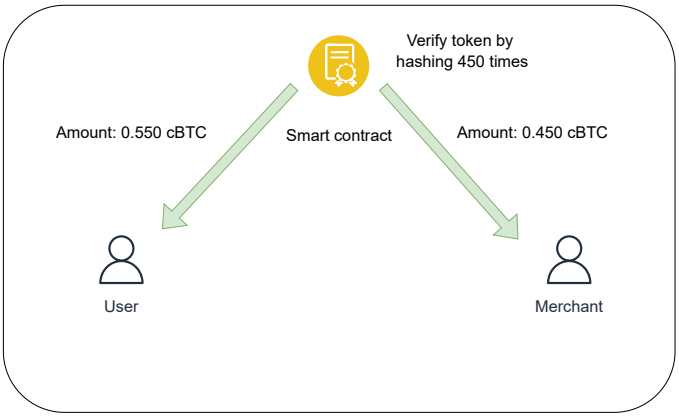

Figure 3: Depiction of the CHAIN redemption process. The smart contract verifies the submitted token by hashing 450 times and calculates proportional payouts: 0.450 cBTC to the merchant and 0.550 cBTC refunded to the user.

Smart contract calculates the payment for merchant: $\frac{450}{1000} \times 1 = 0.45$.
Smart contract calculates the refund to user: $1 - 0.45 = 0.55$.
The funds are dispersed accordingly, and the channel is closed.

3.4 Security mechanisms

- Hash chain verification: The smart contract ensures that the submitted final token is authentic by iteratively hashing it and verifying that it matches the trust anchor.
- Timeout protection: If the merchant does not finalize the transaction by the specified timeout period, the user can reclaim the unspent funds. This acts as a safeguard against inactive or fraudulent merchants.
- Proportional payment assurance: Payments are made proportional to the verified tokens, ensuring fairness.

4 Key Features of the Framework

- Dynamic Payments: Payments are made proportionally to the usage of the hash chain, ensuring fairness for payer and merchant.
- Scalability: Only the first and last tokens are submitted on-chain for verification. All intermediate transactions occur off-chain in verifiable manner, reducing gas costs and latency.

- Trustless operation: Fund release is governed entirely by smart contract logic, removing reliance on intermediaries. Properties of cryptographic hash function guarantees transparency and security.
- Adaptability: Supports a variety of use cases, including incremental payments, subscription models, and large-scale financial transactions.

References

[1] Ronald L. Rivest and Adi Shamir. "PayWord and MicroMint: Two Simple Micropayment Schemes". In: *Security Protocols*. 1997, pp. 69–87.