RBR: Recovering Bitcoin with Rarimo

Rarimo Protocol

Oleksandr Kurbatov, Yaroslav Panasenko, Pavel Kravchenko
ok@distributedlab.com

Mar, 2025

Abstract

This paper outlines the set of approaches for Bitcoin recovery supported by the Rarimo protocol. While this task is resolved for assets on blockchains with smart contracts (through verification proofs of the Rarimo execution correctness), it becomes quite a challenge when we try to support networks that can't verify even SNARK proofs for a reasonable cost or at all. We introduce the **RBR** protocol that allows to realize recovery through a trustless escrow and show how it can be combined with existing Bitcoin recovery options.

1 Introduction

Rarimo proposed a way to recover assets without remembering keys and seed phrases. This became possible because of zero-knowledge proofs for biometry and account abstraction, which allow the implementation of account control and recovery methods that don't rely on keys.

At the same time, realizing the methods mentioned for recovering bitcoins is impossible due to Bitcoin's programmability limitations. This paper proposes a way for the user to program their recovery method on an external blockchain and use a trustless approach with the security deposit provider to recover BTC or withdraw the deposit in the pessimistic scenario.

1.1 Notation

We define a bitcoin transaction as $\mathsf{TX}\{(\mathsf{id}, \mathsf{i}, \mathsf{proof})^{(n)}; (a\ddot{\mathrm{B}}, \mathsf{cond})^{(m)}\}$ with n inputs and m outputs, where id is the reference to the previous transaction, i - output's index, proof - the list of data which is needed to transaction spending, a - the number of coins in the output, cond - scriptPubKey conditions. Additionally, we use $\sigma_{\mathsf{a}}(m)$ notation for the signature that is verified by the public key $\mathsf{P_a}$ according to the message m.

1.2 Requirements

We want to implement a recovery approach with the following properties:

1. The user must control their bitcoins directly, able to spend them at any point of time without additional assumptions (locktime or permissions from the specific party) if the primary key(s) aren't lost.

2. If the user can't access primary key(s), the recovery process should be launched, desirable in the way the trust to recovery provider(s) is minimized (ideally trustless).

Following these points, the final transaction (or precisely UTXO) that supports bitcoin recovery should have the form $\mathsf{TX}\{(...)^{(n)}; (a\ddot{\mathrm{B}}, \mathsf{P} \vee \mathsf{alt})\}$, where P is the owner's public key and alt is alternative spending path depending on the recovery option. Let's note that P can be a single key or any m-of-n multisignature combination, while the whole condition can be compressed to a taproot address.

2 Related work

2.1 Trusted federations

The recovery method can rely on the federative m-of-n approach[KG20; GG20], which consists of creating a UTXO that can be spent either by the coin owner or by a multisignature generated by the trusted quorum. Additionally, the user can set the locktime before which the federation can't spend an output. This approach allows the user to spend the output before locktime and refresh the control of bitcoins. The locking transaction can take the form $\mathsf{TX}\{(...)^{(n)}; (\text{a}\ddot{\text{B}}, \mathsf{P_o} \vee (\mathsf{P_q} \wedge \mathsf{T}))\}$, where $\mathsf{P_o}$ is the user's public key and $\mathsf{P_q}$ is a public key that belongs to the quorum. All conditions can be aggregated to the taproot address, resulting in a single EC point [Max18].

2.2 BitVM2

BitVM2 [Lin24] is the technology for the trust-minimized (1-of-n) verification of program execution on top of Bitcoin. By the programs, we mean rollups, bridges, and other solutions that require operating with native bitcoins. The general idea of BitVM2 is the following:

1. we replace the needed program with SNARK verifier of its execution correctness
2. then we need to divide the program into Bitcoin Script chunks that don't exceed 4MB each
3. setup phase where the operator commits on the intermediate program states, creates, and presigns the challenge transaction with the trusted quorum of signers (n-of-n should be active)
4. if the operator wants to withdraw money, they must publish the program's output
5. anyone can challenge the operator if their output doesn't match with the ones provided by them
6. in the challenge case, the operator must reveal all intermediate states, and then anyone can prove that some of the chunks were executed incorrectly (if they were)

Potentially, the BitVM2 approach can also be used for recovery (by proving the Rarimo account's state through SNARKs). Still, it is accompanied by a much higher cost, potential problems with setup (you need to have all signers active), and many economic issues (operator/challenger games) that must be resolved in advance.

2.3 SPV connector

The SVP connector [Kur24] is a smart contract that synchronizes the whole Bitcoin history to the blockchain when it's deployed. The idea of the connector is to verify all block headers (that any user can provide) and resolve reorganizations by following Bitcoin protocol rules. Using the synchronized SPV contract, the user can prove that a particular transaction was confirmed in the Bitcoin mainnet and trigger the defined action on the targeting blockchain.

3 RBR protocol overview

We propose a protocol combining the SPV connector and the recovery provider's security deposit. This approach allows us to support the following recovery properties (Alice wants to be able to recover BTC with Bob, who is a recovery provider):

1. Alice is ready to use Bob as a recovery provider only if he is ready to put his security deposit (equivalent in USD or wrapped BTC, etc.)
2. Bob is ready to put the security deposit in case he can unlock it if Alice spends her UTXO and leaves the service fee Bob can take
3. Alice should be able to take the security deposit in case Bob steals Alice's BTC

In this case, we need to provide both sides with the ability to prove that the counterparty unlocked the BTC, and we can use the SPV connector for it. We will refer to $\pi_{\mathsf{SPV}}(\mathsf{tx}) \to \mathbb{B}$ as the SPV proof that the particular transaction tx was included in the Bitcoin blockchain.

Algorithm 1 RBR protocol

Condition: Alice sets a recovery option with Bob for her 1₿. Bob provides a security deposit c on Ethereum (for instance) and wants to receive a fee f for his service. $\mathsf{P_a}$ is the Alice's public key and $\mathsf{P_b}$ is Bob's one.

Flow:

Alice creates the transaction TX that sends her 1₿ to the following conditions:

$$\mathsf{TX}\{(-); (1\ddot{\mathrm{B}}, \mathsf{P_a} \vee \mathsf{P_b})\}$$

Alice doesn't sign the transaction but just shares it with Bob to check if it's constructed correctly
Bob deploys the escrow contract on Ethereum and deposits c to it with the following conditions:

1. c can be returned after T_1 (absolute timelock) if Alice doesn't provide $\pi_{\mathsf{SPV}}(\mathsf{TX}) \to 1$ and f to the deposit contract
2. If $\pi_{\mathsf{SPV}}(\mathsf{TX}) \to 1$ and f was provided before T_1, the c is locked before T_2 (absolute timelock, the recovery period that agreed between Alice and Bob)
3. If there is $\pi_{\mathsf{SPV}}(\mathsf{TX_1}\{(\mathsf{TX}, 1, (\mathsf{P_a}, \sigma_\mathsf{a})); (1\ddot{\mathrm{B}}, \mathsf{P_x})\}) \to 1$, Bob can return c to own account
4. There can be a request from Alice to sign the transaction $\mathsf{TX_2}\{(\mathsf{TX}, 1, (\mathsf{P_b}, -); (1\ddot{\mathrm{B}}, \mathsf{P_x})\}$:
   (a) If Bob doesn't respond with the signature σ_b before T_3 (relative timelock) to make a $\mathsf{TX_2}$ valid, Alice can take c
   (b) If Bob responded with σ_b, the contract annulates T_3
   (c) $\mathsf{TX_2}$ can be send by anyone. When it's sent, Bob can produce $\pi_{\mathsf{SPV}}(\mathsf{TX_2}) \to 1$ and take the c and f.

With this approach, we can cover all the requirements we mentioned in section 1.2 and the following potential scenarios:

1. If Alice controls $\mathsf{P_a}$, she can spend her 1₿ at any point of time. When it happens, Bob can return his security deposit and Alice's fee by posting the proof the appropriate transaction was added to the Bitcoin blockchain
2. If Alice loses control of $\mathsf{P_a}$, she can ask Bob (on-chain) to sign the recovery transaction. If Bob satisfies the request – Alice overwrites the bitcoin owner with the new $\mathsf{P_x}$, and Bob returns the security deposit. If Bob doesn't provide the signature (DoS) – Alice can punish him with a deposit.
3. If Bob want to spend 1₿ instead of Alice – she can prove it and take the security deposit
4. If Alice wants to steal security deposit and send her Bitcoin by sending the signature request and simultaneous Bitcoin payment, Bob can prove it and unlock the deposit before

4 Future vectors of improvements

We see two major challenges related to the approach described in the paper:

1. **Locked security deposit is required**. Although Bob can agree on the recovery fee proposed by Alice, it still requires the locked deposit. Using the scheme in the form we described doesn't allow reusing the locked security deposit because Alice and Bob should be able to take it at any time if their counterparty tries to cheat. So, we are investigating different directions that allow us to reuse the locked assets and not introduce additional security issues into the scheme.
2. **Collateral volatility**. The bitcoin price change can lead to a situation where Bob doesn't want to return the bitcoin to Alice because its price exceeds the value of the security deposit and fee. The ideal scheme would provide an ability for Alice to punish Bob with deposit and bitcoins simultaneously in such a case. So, improvement of this scheme can consist of additional guarantees that Alice can return her bitcoins if she doesn't attempt to violate the terms of the agreement.

References

[Max18] Greg Maxwell. "Taproot: Privacy Preserving Bitcoin Transactions". In: (2018). URL: https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.

[GG20] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: *ACM CCS*. 2020. URL: https://eprint.iacr.org/2020/540.pdf.

[KG20] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". In: (2020). URL: https://eprint.iacr.org/2020/852.pdf.

[Kur24] O. Kurbatov. "EXTENDING THE SPV CONTRACT CONCEPT WITH PRIVACY GADGETS". In: *Telecommunication and Information Technologies* 83 (July 2024). DOI: 10.31673/2412-4338.2024.024961.

[Lin24] Robin Linus. "BitVM2: Extending Bitcoin's Computation Model". In: (2024). URL: https://bitvm.org/bitvm2.pdf.