# RBR: RECOVERING BITCOIN WITH RARIMO VIA SPV-VERIFIED CROSS-CHAIN ESCROW AND KEY-LESS ACCOUNT ABSTRACTION

Oleksandr Kurbatov [1], Yaroslav Panasenko [1], Pavlo Kravchenko [1,2], and Volodymyr Dubinin [1]

[1] Distributed Lab, Kyiv, Ukraine
[2] Department of Electronic Computers, Kharkiv National University of Radio ElectronicsKharkiv, Ukraine

***ABSTRACT***

*This paper introduces a series of methodologies for Bitcoin recovery, facilitated by the Rarimo protocol. While existing smart contract platforms are capable of verifyingRarimo's execution proofs, extending this methodology to networks that do not possess cost-effective SNARK verification presents ongoing challenges. We propose the RBR protocol, which facilitates recovery through a trustless escrow mechanism and demonstrates its interoperability with current Bitcoin recovery solutions.*

***KEYWORDS***

*Bitcoin recovery, Trustless escrow, SPV connector, Zero-knowledge proofs, Account abstraction.*

## 1. INTRODUCTION

The Rarimo protocol proposes a method for asset recovery that obviates the need to restore keys or seed phrases. This capability is enabled by zero-knowledge biometric proofs and accountabstraction techniques, which support key-less account control and recovery.

However, implementing these recovery methods on Bitcoin is impeded by the limited expressiveness of the Bitcoin scripting language. We therefore propose a mechanism that lets users specify recovery logic on an external blockchain and employ a trustless security-deposit scheme that either facilitates BTC recovery or, in adversarial scenarios, permits deposit withdrawal.

### 1.1. Notation

We define a bitcoin transaction as

$$TX\{(\mathrm{id}, \mathrm{i}, \mathrm{proof})^{(n)}; (a฿, cond)^{(m)}\},$$

where $n$ is the number ofinputs, $m$ – outputs, $id$is the reference to the previous transaction, $i$– output's index, $proof$– the list of data which is needed for transaction spending, $a$– the number of coins in the output, and $cond$– scriptPubKey conditions.

David C. Wyld et al. (Eds): SIGI, CSTY, AI, NMOCT, BIOS, AIMLNET, MaVaS, BINLP – 2025
pp. 67-72, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151906

Additionally, we use $\sigma_a(m)$ notation for the signature that is verified by the public key $P_a$ according to the message $m$.

### 1.2. Requirements

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

1. The user must retain direct control of their bitcoins and be able to spend them at any timewithout auxiliary assumptions (e.g., locktimes or external permissions), provided the primary key(s) remain available.

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

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

## 2. RELATED WORK

Before detailing our proposal, we survey the main lines of prior work that tackle Bitcoin key recovery and trust-minimised cross-chain interactions. We group these approaches into three categories—trusted federations, BitVM2-style computation verification, and SPV-based connectors—highlighting the design lessons each contributes to RBR.

### 2.1. Trusted Federations

Recovery can employ a federated $m$-of-$n$ approach [1][2], 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 the bitcoins. The locking transaction can take the form$TX\{(\dots)^{(n)}; (a฿, P_o \vee (P_q \wedge T))\}$where $P_o$is the user's public key and $P_q$is a public key that belongs to the quorum. All conditions can be aggregated to the taproot address, resulting in a single EC point [3].

### 2.2. BitVM2

BitVM2 [4] is the technology for the trust-minimized $m$-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. The setup phase, where the operator commits to the intermediate program states, creates, andpresigns 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 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 SPV connector[5] is a smart contract that synchronizes the entirety of Bitcoin block headers 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 the 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.

We will refer to $\pi_{SPV}(TX) \in \{0; 1\}$ as the SPV proof that the particular transaction TXwas included in the Bitcoin blockchain.

## 3. RBR PROTOCOL OVERVIEW

Firstly, let's define some variables that we use in the protocol flow:

| | |
|---|---|
| $P_a, P_b$ | Public keys of Alice and Bob, respectively; |
| $a$ | Alice's funds (number of Bitcoins); |
| $c$ | Bob's security deposit (locked on Ethereum); |
| $f$ | Fixed service fee that Alice pays Bob upon successful recovery; |
| $T_1$ | Absolute timelock after which Bob may reclaim $c$ if Alice never proves she funded the Bitcoins; |
| $T_2$ | Absolute timelock that defines the "recovery window" once Alice has funded the lock; |
| $T_3$ | Relative Bitcoin timelock (CSV) that gives Bob time to supply a signature after Alice requests it; |
| $P_x$ | Alice's new public key that will control the recovered coins. |

Now we propose a protocol that combines an SPV connector with a security deposit furnished by the recovery provider. 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 that 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.

### 3.1. Protocol Flow

Alice creates the transaction $TX$ that sends her $a฿$ to the following conditions:

$$TX\{(-); (a฿, P_a \vee P_b)\}$$

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 $T_1$ if Alice doesn't provide $\pi_{SPV}(TX) \rightarrow 1$ and $f$ to the deposit contract;

2. If $\pi_{SPV}(TX) \rightarrow 1$ and $f$ was provided before $T_1$, the $c$ is locked before $T_2$;

3. If there is $\pi_{SPV}(TX_1\ \{TX,\ 1,\ (P_a, \sigma_a));\ (a฿, P_x)\}) \rightarrow 1$, Bob can return $c$ to his account;

4. There can be a request from Alice to sign the transaction $TX_2\ \{(TX,\ 1,\ (P_b, -));\ (a฿, P_x)\}$

   (a) If Bob doesn't respond with the signature $\sigma_b$ before $T_3$ (relative timelock) to make a $TX_2$ valid, Alice can take $c$;
   (b) If Bob responded with $\sigma_b$, the contract annulates $T_3$;
   (c) $TX_2$ can be sent by anyone. When it's sent, Bob can produce $\pi_{SPV}(TX_2) \rightarrow 1$ 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 $P_a$, she can spend her $a฿$ at any point in time. When it happens, Bob can return his security deposit and Alice's fee by posting the proof that the appropriate transaction was added to the Bitcoin blockchain.

2. If Alice loses control of $P_a$, she can ask Bob (on-chain) to sign the recovery transaction. If Bob satisfies the request, Alice overwrites the bitcoin owner with the new $P_x$, and Bob returns the security deposit. If Bob does not provide the signature (DoS),Alice can punish him by claiming the deposit.

3. If Bob wants to spend $a฿$ instead of Alice – she can prove it and take the security deposit.

4. If Alice wants to steal the security deposit and send her Bitcoin by sending the signature requestand a simultaneous Bitcoin payment, Bob can prove it and unlock the deposit before.

Also, this approach covers other possible risks:

5. If Alice never funds and expects Bob to keep $c$ frozen, the escrow contract automatically refunds $c$ after timelock $T_1$.

6. If chain reorganization happens, funds remain locked until a fresh proof is possible to create.

7. If Alice starts repeatedly requesting a signature,Bob ignores it after supplying $\sigma_b$.

## 4. FUTURE VECTORS OF IMPROVEMENTS

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

1. **A locked security deposit is required**. Although Bob may accept the recovery fee stipulated by Alice, the scheme nevertheless necessitates a 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. Therefore, we are exploring various options that enable us to reuse the locked assets while avoiding the introduction of additional security issues into the scheme.

2. **Collateral volatility**. Volatility in the Bitcoin price may disincentivize Bob from returning the coins to Alice once their value exceeds the combined security deposit and fee. The ideal scheme would provide an ability for Alice to punish Bob with a 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 doesnot attempt to violate the terms of the agreement.

## 5. CONCLUSIONS

This work has presented RBR, a trust-minimised Bitcoin-recovery framework that fuses an SPV-verified proof system with an on-chain security-deposit escrow. By separating the expressive recovery logic to an external smart-contract platform while retaining native Bitcoin custody for the end-user, RBR attains three properties rarely achieved simultaneously: (i) self-custody first– the owner may at any time spend the UTXO unilaterally; (ii) cryptoeconomic deterrence– the recovery provider's worst-case loss always exceeds any potential cheating gain; and (iii) verifiable liveness– all protocol branches are decidable via publicly auditable SPV proofs. Our formal algorithm and threat analysiscollectively demonstrate that each participant's incentives are aligned under realistic market conditions and chain-reorganisation assumptions.

Compared with federation-based recovery and BitVM2-style full-program verification, the proposed design offers a more lightweight trust surface and lower on-chain cost, yet inherits the censorship resistance of Bitcoin's base layer. The remaining challenges– most notably capital inefficiency of locked deposits and exposure to extreme BTC price shocks– outline clear directions for future research, including reusable collateral schemes and oracle-driven, dynamically-adjusted margins. Overall, RBR advances the state of keyless asset recovery by showing that strong user sovereignty and practical, economically secure recovery guarantees are simultaneously attainable without altering the Bitcoin consensus rules.

## REFERENCES

[1] Komlo C., and Goldberg I., (2020)"FROST: Flexible Round-Optimized Schnorr Threshold Signatures", Cryptology ePrintArchive, Paper 2020/852:https://eprint.iacr.org/2020/852.pdf

[2] Gennaro R., and Goldfeder S., (2020) "Fast Multiparty Threshold ECDSA with Fast Trustless Setup", Cryptology ePrintArchive, Paper 2020/540:https://eprint.iacr.org/2020/540.pdf

[3] Maxwell G., (2018) “Taproot: Privacy Preserving Bitcoin Transactions”,https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki
[4] Linus R., (2024)“BitVM2: Extending Bitcoin’s Computation Model”, https://bitvm.org/bitvm2
[5] Kurbatov O., (2024) “Extending the SPV Contract Concept with Privacy Gadgets”, Telecommunication and Information Technologies 83,doi: 10.31673/2412-4338.2024.024961.

## AUTHORS

**Oleksandr Kurbatov** is a PhD Candidate at Karazin Kharkiv National University, Ukraine. His research focuses on public-key infrastructure, blockchain technologies, and anonymous decentralized voting systems. Currently working as the Lead Cryptography Researcher at Distributed Lab.

**Yaroslav Panasenko** is the Chief Technology Officer at Distributed Lab. He holds a B.Sc. in Software Engineering from Kharkiv Polytechnic Institute. His research interests span decentralized anonymous banking systems, blockchain infrastructure, biometric systems, and AI safety and security.

**Pavlo Kravchenko** earned his PhD in Cryptography from Kharkiv National University of Radioelectronics in 2012. He now works as a technology entrepreneur, software architect specializing in cryptographic solutions, and Chief Executive Officer of Distributed Lab. His research interests include public-infrastructure security, blockchain protocols, and privacy-preserving cryptographic systems.

**Volodymyr Dubinin** holds an M. Sc. in Computer Science and is Co-Founder of Distributed Lab. His research interests span Decentralized Systems, Artificial Intelligence, Blockchain Scalability, and Cryptographic Protocol Design.