**Poster Abstract: Multisignatures for Cryptocurrency-Backed Tokens**

Alexei Zamyatin

1 Department of Computing, Imperial College London
2 SBA Research
a.zamyatin@imperial.ac.uk

Despite the influx of new cryptocurrencies and academic research on distributed ledgers, communication between permissionless blockchains is mostly facilitated using centralized liquidity providers, or exchanges.

Recently, XCLAIM, a protocol for issuing, trading and redeeming cryptocurrency-backed tokens was introduced as a mechanism for trustless blockchain interoperability by Zamyatin et al.[7]. Thereby, users lock units of a backing cryptocurrency A, e.g., Bitcoin [5], with an non-trusted and collateralized third party (the Issuer) and create the equivalent amount of tokens $a_b$ on an issuing cryptocurrency B, e.g. Ethereum [4]. To redeem $a_b$ for the corresponding amount of $a$ on chain A, users must destroy or burn the tokens in a publicly verifiable manner on chain B. The scheme leverages hashed time-lock contracts [1] on the backing blockchain, as well as chain relays [6] (e.g. BTC Relay [2]), collateral and smart contracts on the issuing blockchain, i.e., requires (near) Turing complete programming capabilities on chain B. While the Issuer maintains full control of the locked cryptocurrency units $a$ for the duration of the protocol, XCLAIM guarantees that in case of Byzantine or crash failures of the Issuer, the victims will be reimbursed the equivalent monetary value of their loss from the Issuer’s collateral on chain B.

In this extending work, we discuss how multisignatures can be used to further improve the safety properties of the XCLAIM protocol, preventing theft of locked units of the backing cryptocurrency altogether, at the costs of reduced performance. Specifically, instead of cryptographically transferring ownership of units $a$ of the backing cryptocurrency to the Issuer, $a$ is locked using e.g. a multisignature output in Bitcoin [3]. This prevents the Issuer from withdrawing the locked $a$ without the user’s consent (i.e., stealing), while the user still cannot withdraw the funds before burning $a_b$. The improved safety, however, makes trading of $a_b$ more complex, as transfer of token ownership on chain B must now be mirrored on chain A: if Alice wants to transfer $a_b$ to Bob, she must replace herself with Bob in the multisig on A. To this end, she must create and sign a transaction $T_{replace}$ updating the state of the multisignature and present it to both Bob and the Issuer in a publicly verifiable manner, e.g. by uploading the transaction as data on chain B. In the presented poster, we outline the multisignature version of the XCLAIM protocol and discuss possible improvements in terms of performance, cost reduction and privacy. Finally, we discuss the current disadvantages of using multisignatures and give an outlook on their applicability in future extensions of XCLAIM using off-chain payment channels.

* This works extends upon the recently introduced XCLAIM protocol for cryptocurrency-backed tokens [7].
References

Multisignatures for Cryptocurrency-backed Tokens

Alexei Zamyatin

**XCLAIM: Cryptocurrency-backed Tokens**

In three steps to interoperability (e.g., Bitcoin-backed tokens on Ethereum):

1. **Lock with Issuer**
   - (Non-trusted & collateralized 3rd party)

2. **Prove lock to Chain relay**
   - Chain relay verifies TX inclusion proof and informs treasury

3. **Issue tokens**
   - Treasury contract issues Bitcoin-backed ERC20 tokens

**Token States**

- **NONE**: No collateral
- **ISSUED**: Issuer has collateral locked in contract
- **REDEEMED**: Issuer sees lock
- **REIMBURSED**: Issuer sees unlock

**Multisignature Locks: Improving Safety**

Use Bitcoin 2-of-2 multisignatures to make theft by the Issuer impossible

1. Alice generates Ethereum key pair
2. Alice publishes a multisig with the Issuer
3. Alice issues the token in the multisig
4. Chain relay verifies the lock is included in Bitcoin's main chain
5. Alice publishes a transaction for inclusion in the multisig contract
6. Alice commits to unlocking the multisig
7. Alice makes the token transfer to the Issuer

**Challenges**

- Fungibility of tokens cannot be guaranteed
- Substantial amount of data stored on Ethereum
- Fund freeze still possible!

**Protocols**

**Issue**

- Alice generates Ethereum key pair
- Alice publishes a multisig with the Issuer
- Alice issues the token in the multisig
- Chain relay verifies the lock is included in Bitcoin's main chain
- Alice publishes a transaction for inclusion in the multisig contract
- Alice commits to unlocking the multisig
- Alice makes the token transfer to the Issuer

**Trade**

- Alice creates and signs a transaction, which replaces her in the multisig lock
- Bob creates and signs a transaction, which pays him the correct amount of BTC

**Redeem**

- Bob creates and signs a transaction, which pays him the correct amount of BTC
- Bob generates Bitcoin key pair
- Bob replaces Alice in a new multisig with the Issuer, spending from the old multisig output
- State not yet updated publicly.

**Optimizations**

- Reduce Waiting Times
  - Issuer signs all TX only when token is redeemed (P2WSH - BIP141 Segregated Witness required)
- Reduce Costs / Transactions
  - UTXO grouping scheme: optimistic reduction of required TX to O(1). However, interactive protocol!
- Improve Incentives against fund freezing
  - Additional collateral on Bitcoin: $btc_{collateral} = btc_{lock}$