

# XCLAIM: A Framework for Blockchain Interoperability

Alexei Zamyatin<sup>\*†‡</sup>, Dominik Harz<sup>†</sup>, Joshua Lind<sup>†</sup>, Panayiotis Panayiotou<sup>†</sup>, Arthur Gervais<sup>†</sup>, William J. Knottenbelt<sup>†</sup>

<sup>†</sup> Department of Computing, Imperial College London, United Kingdom

<sup>‡</sup> SBA Research, Austria

**Abstract**—Building trustless cross-blockchain trading protocols is challenging. Centralized exchanges thus remain the preferred route to executing transfers across blockchains. However, these services require trust and therefore undermine the very nature of the blockchains on which they operate. To overcome this, several decentralized exchanges have recently emerged which offer support for atomic cross-chain swaps (ACCS). ACCS enable the trustless exchange of cryptocurrencies across blockchains, and are the only known mechanism to do so. However, ACCS suffer significant limitations; they are slow, inefficient and costly, meaning that they are rarely used in practice.

We present XCLAIM: the first generic framework for achieving trustless and efficient cross-chain exchanges using *cryptocurrency-backed assets* (CBAs). XCLAIM offers protocols for issuing, transferring, swapping and redeeming CBAs securely in a non-interactive manner on existing blockchains. We instantiate XCLAIM between Bitcoin and Ethereum and evaluate our implementation; it costs less than USD 0.50 to issue an arbitrary amount of Bitcoin-backed tokens on Ethereum. We show XCLAIM is not only faster, but also significantly cheaper than atomic cross-chain swaps. Finally, XCLAIM is compatible with the majority of existing blockchains without modification, and enables several novel cryptocurrency applications, such as cross-chain payment channels and efficient multi-party swaps.

## I. INTRODUCTION

Blockchain-based cryptocurrencies enable secure and trustless transactions between parties. As a result, they have gained widespread adoption and popularity in recent years; there are currently over 2 000 different cryptocurrencies in operation, with a total market cap of USD 135bn [49]. However, despite a growing and thriving ecosystem, cryptocurrencies continue to operate in complete isolation from one another: blockchain protocols provide no means by which to communicate or exchange data with external systems. As such, achieving interoperability between different cryptocurrencies remains an open challenge.

*Centralized exchanges* thus remain the preferred route to execute fund transfers and exchanges across blockchains. However, these services require trust and therefore undermine the very nature of the cryptocurrencies on which they operate, making them vulnerable to attacks [33], [36], [93], [98]. To overcome this, *decentralized exchanges* [1], [2], [12], [15], [17] (*DEXs*) have recently emerged, removing the need to trust centralized intermediaries for blockchain transfers. However, the vast majority of DEXs only enable the exchange of

*cryptocurrency-assets* within a single blockchain, i.e., they do not operate across blockchains (*cross-chain*). As such, it is only a handful of platforms [16], [30] that actually support cross-chain exchanges through the use of *atomic cross-chain swaps* [4], [34], [72], [110].

Atomic cross-chain swaps (*atomic swaps*) enable secure cross-chain exchanges using *hashed-time-lock contracts* (HTLCs) [5], [48]. At present, they are the only mechanism to do this without necessitating trust. Unfortunately, they require several strong assumptions to maintain security, thus limiting their practicality: they are interactive, requiring all parties to be online and actively monitor all involved blockchains during execution; they require synchronizing clocks between blockchains and rely on pre-established secure out-of-band communication channels. In addition, they also incur long waiting periods between transfers and suffer the limitation that for every cross-chain swap, four transactions need to occur, two on each blockchain. This makes them expensive, slow and inefficient.

We therefore present XCLAIM (pronounced *cross-claim*): the first generic framework for achieving trustless cross-chain exchanges using *cryptocurrency-backed assets*. In XCLAIM, blockchain-based assets can be securely constructed and one-to-one backed by other cryptocurrencies, for example, Bitcoin-backed tokens on Ethereum. Through the secure issuance, swapping, and redemption of these assets, users can perform cross-chain exchanges in a trustless and non-interactive manner, overcoming the limitations of existing solutions.

To achieve this, XCLAIM exploits publicly verifiable *smart contracts* to remove the need for trusted intermediaries and leverages *chain relays* [6], [34], [78], [111] for cross-chain state verification. Using these building blocks, XCLAIM constructs a publicly verifiable audit log of user actions on both blockchains and employs *collateralization* and *punishments* to enforce the correct behavior of participants. Thereby, XCLAIM follows a *proof-or-punishment* approach: instead of relying on reactive fraud proofs, participants must proactively prove adherence to system rules. This allows users to go offline without compromising the safety of their funds and avoids necessity of user synchronization in case of disputes.

Due to its simple and efficient design, XCLAIM enables several novel applications, such as: (i) *cross-chain payment channels*, where users can exchange payments *off-chain* across different blockchains in a trustless manner; (ii) *temporary*

\*Corresponding author. Email: a.zamyatin@imperial.ac.uk.

*transaction offloading*, where users temporarily tokenize their cryptocurrency on other blockchains to overcome network congestion and high fees; and (iii) *N-way and multi-party atomic swaps* allowing efficient and complex atomic swaps. Finally, as XCLAIM maintains compatibility with existing standardized asset interfaces [9], [10], the issued assets are tradeable via existing decentralized exchanges, enabling these exchanges to operate cross-blockchain.

This paper makes the following contributions:

- We formally define the notion of cryptocurrency-backed assets for blockchains and formulate goals for security and functionality (Section III). We then present XCLAIM, a practical and secure system to construct cryptocurrency-backed assets without trusted intermediaries (Section IV-V).
- We provide a formal protocol specification for XCLAIM and analyze in detail the requirements for the underlying blockchains (Section VI). While the blockchain used to issue cryptocurrency-backed assets must support smart contracts, XCLAIM requires only base-ledger functionality on the backing side, supporting practically all cryptocurrencies.
- We implement XCLAIM(BTC,ETH), to the best of our knowledge, the first system for trustlessly issuing, transferring, swapping and redeeming Bitcoin-backed tokens on Ethereum (Section VIII). In our prototype, it costs USD 0.47 to issue, USD 0.04 to transfer, USD 0.19 to atomically swap and USD 0.49 to redeem an arbitrary amount of cross-chain tokens<sup>1</sup>. We compare performance and costs to atomic swaps and show XCLAIM is 95.7% faster and 65.4% cheaper for 1000 swaps.
- Finally, we present and describe several novel applications enabled exclusively by XCLAIM, such as cross-chain payment channels and efficient N-way and multi-party atomic swaps (Section IX).

## II. BACKGROUND

### A. Blockchains and Decentralized Ledgers

Bitcoin [94] allows users to hold and exchange funds in a decentralized manner, without trusting a third party. At its core lies a peer-to-peer replicated state machine that maintains a global append-only ledger, the blockchain. This ledger maintains the history of all transactions and is constructed through a sequence of blocks, chained together via the hashes of their predecessors. The majority of blockchains leverage a random leader election through *Proof-of-Work* [35], [55], where peers must solve a computationally expensive puzzle. Often referred to as *Nakamoto consensus*, this approach achieves agreement in permissionless systems, i.e., where the set of peers is dynamically changing. Inspired by Bitcoin, numerous other blockchain-based cryptocurrencies emerged. Ethereum [47] extends the basic ledger functionality of blockchains by a Turing-complete programming language, and allows users to create so-called *smart contracts*, i.e., programs that are executed and the result of which is agreed upon by all peers. Other systems offer alternative agreement mechanisms, such as

*Proof-of-Stake*, where the rate-limiting resource is the currency itself, rather than computational power [45], [77], [79].

For brevity, we use *blockchain* and *chain* as synonyms in the rest of this paper.

### B. Overlay Protocols, Colored Coins and Tokens

Overlay protocols and colored coins leverage the infrastructure of existing blockchains, extending their functionality with additional features [8], [21], [101]. The necessary requirements for these are provided by most existing blockchains: (i) inclusion of data in blockchain transactions and (ii) eventual agreement on their ordering [120]. As such, overlay protocols enable the creation of cryptocurrencies without bootstrapping a new dedicated blockchain. These are referred to as *cryptocurrency tokens* and can be used to quantify both fungible and non-fungible assets [9], [11]. Profiting from the expressiveness of Ethereum’s scripting language [64], a plethora of such tokens has emerged rivaling cryptocurrencies both in number and market capitalization [49], [117].

### C. Real-World Asset-Backed Cryptocurrencies and Tokens

Cryptocurrencies and, more recently, tokens have been used to represent real-world assets, such as gold [115] and fiat currency [29]. However, since blockchains provide no means to cryptographically verify events in the outside world, issuing and redeeming such tokens is usually handled by a single trusted custodian. Achieving transparency and trustlessness for this process is therefore an open challenge. As a result, many of these existing schemes are susceptible to manipulation [70], or may be halted altogether [27].

## III. SYSTEM OVERVIEW

In this section we present a formal definition of cryptocurrency-backed assets. We next present the system model and actors, and define the network and threat models for XCLAIM. Finally, we present the system goals.

### A. Cryptocurrency-backed Assets (CBA)

**Formal definition.** We define *cryptocurrency-backed assets* (CBAs) as assets deployed on top of a blockchain  $I$  that are backed by a cryptocurrency on blockchain  $B$ . We denote assets in  $I$  as  $i$ , and cryptocurrency on  $B$  as  $b$ . We use  $i(b)$  to further denote when an asset on  $I$  is backed by  $b$ . We extend the definition of assets by Androulaki et al. [32] and describe a CBA through the following fields:

- *issuing blockchain*, the blockchain  $I$  on which the CBA  $i(b)$  is issued.
- *backing blockchain*, the blockchain  $B$  that backs  $i(b)$  using cryptocurrency  $b$ .
- *asset value*, the units of the backing cryptocurrency  $b$  used to generate and represented by  $i(b)$ .
- *asset redeemability*, whether or not  $i(b)$  can be redeemed on  $B$  for  $b$ .
- *asset owner*, the current owner of  $i(b)$  on  $I$ .
- *asset fungibility*, whether or not units of  $i(b)$  are interchangeable.

<sup>1</sup>According to exchange rates as of 30 November 2018.

We define a CBA as *symmetric* if the total amount of backing units  $b$  is equivalent to the total amount of issued units  $i(b)$ , i.e.,  $|b| = |i(b)|$ , and as *asymmetric* if the CBA exhibits an alternate backing rate, i.e.,  $|b| \neq |i(b)|$ . In XCLAIM, we restrict CBAs to be symmetric cryptocurrency-backed assets. Moreover, we allow CBAs to be both *redeemable*, and *fungible* by nature. We defer the analysis and discussion of alternate CBAs, such as asymmetric and non-fungible CBAs, to future work.

### B. System Model and Actors

XCLAIM operates between a backing blockchain  $B$  of cryptocurrency  $b$  and an issuing blockchain  $I$  with underlying CBA  $i(b)$ . To operate CBAs, XCLAIM further differentiates between the following actors in the system:

- **CBA Requester.** Locks  $b$  on  $B$  to request  $i(b)$  on  $I$ .
- **CBA Sender.** Owns  $i(b)$  and transfers ownership to another user on  $I$ .
- **CBA Receiver.** Receives and is assigned ownership over  $i(b)$  on  $I$ .
- **CBA Redeemer.** Destroys  $i(b)$  on  $I$  to request the corresponding amount of  $b$  on  $B$ .
- **CBA Backing Vault (vault).** A (non-trusted) intermediary responsible for fulfilling redeem requests of  $i(b)$  using  $b$  on  $B$ .
- **Issuing Smart Contract (iSC).** A public smart contract responsible for managing the correct issuing and exchange of  $i(b)$  on  $I$ . The iSC further ensures the correct behaviour of the *vault*.

To perform these roles in XCLAIM, actors are identified on a blockchain using their public/private key pairs. As a result, the *requester*, *redeemer* and *vault* must maintain key pairs for both blockchains  $B$  and  $I$ . The *sender* and *receiver* only need to maintain key pairs for the issuing blockchain  $I$ . iSC exists as a publicly verifiable smart contract on  $I$ .

### C. Blockchain Model and Assumptions

XCLAIM establishes communication between two independent blockchains with likely varying consensus mechanisms and trust models. Therefore, should either blockchain  $B$  or  $I$  be compromised by an adversary, the correct functionality of XCLAIM cannot be guaranteed. As such, we assume that the proportion of consensus participants  $f$  (or computational power  $\alpha$  in the case of Nakamoto consensus) corrupted by an adversary for both  $B$  and  $I$  is bounded by the threshold necessary to ensure safety and liveness for the underlying blockchains. For example, in Nakamoto consensus based blockchains, e.g. Bitcoin [94] and Ethereum [47], we assume  $\alpha \leq 33\%$  [62], [68], [102]. In Byzantine fault tolerant settings using e.g. Proof-of-Stake, such as [45], [77], we assume  $f < \frac{n}{3}$  where  $n$  is the total number of consensus participants.

These assumptions guarantee that the number of maliciously generated blocks is upper-bounded by  $\frac{f}{n-f}$  (*chain quality property*) [63]. As such, we assume the probability of blockchain reorganizations therefore drops exponentially with a security parameter  $k \in \mathbb{N}$  (*common-prefix property*) [63], [94],

[99]. We measure  $k$  in blocks and denote  $k^B$  for blockchain  $B$  and  $k^I$  for blockchain  $I$ . Specifically, we say a transaction is *securely*<sup>2</sup> included in the underlying blockchain if, given the current blockchain head at position  $h \in \mathbb{N}$ , the transaction is included in a block at position  $j \in \mathbb{N}$ , such that  $h - j \geq k$ .

For existing blockchains, the delay  $\Delta$  from transaction broadcast to *secure* inclusion depends on the block generation rate  $\tau$ , i.e., the number of blocks created per round (cf. *chain growth property* [63], [99]). Block generation rates are however non-deterministic for most blockchains<sup>3</sup> and differ from system to system. To ensure secure communication between  $B$  and  $I$ , we thus formulate assumptions for both  $\Delta^B$  and  $\Delta^I$ , i.e., argue on a ratio  $r_\tau = \mathbb{E}(\tau^B)/\mathbb{E}(\tau^I)$  between the expected generation rates of  $B$  and  $I$ . To this end, in XCLAIM we assume a known upper bound  $\bar{r}_\tau$ , i.e., that increases and decreases of the speed at which blocks are found in  $B$  remain within a known bound in relation to the rate of  $I$  (and vice-versa).

### D. Threat Model and Network Assumptions

We assume that adversaries may misbehave arbitrarily, e.g. try to delay or censor transactions, read unconfirmed transactions in the network or in mining pool memory, and perform Sybil attacks [54]. However, we assume adversaries are computationally bounded and economically rational, that the cryptographic primitives of  $B$  and  $I$  are secure, and the Diffie–Hellman problem is computationally hard.

For the underlying network, we make the same assumptions as in prior work [61], [77], [81], i.e., we assume (i) honest nodes are well connected and (ii) communication channels between these nodes are synchronous. Specifically, transactions broadcast by users are received by (honest) consensus participants<sup>4</sup> within a known maximum delay  $\Delta_{tx}$  [99], i.e.,  $\Delta_{tx}^B$  for chain  $B$  and  $\Delta_{tx}^I$  for chain  $I$ .

### E. System Goals

XCLAIM has the following goals with respect to the secure and correct functionality of cryptocurrency-backed assets (CBAs):

- 1) **Decentralization.** XCLAIM should not be centralized, have single points of failure, or require absolute trust in any third parties.
- 2) **Transparency.** XCLAIM should allow anyone with read access to the underlying blockchains to verify the correct operation of all actions by all actors.
- 3) **Consistency.** XCLAIM should ensure the total amount of backing cryptocurrency units  $b$  and circulating assets  $i(b)$  are equivalent; that is, XCLAIM should offer symmetric CBAs.
- 4) **Transfer Consistency.** Transfers of ownership of CBAs should be executed consistently on both the issuing blockchain  $I$  and backing blockchain  $B$ .

<sup>2</sup>We note  $k$  is recommended to be set as a function of the transaction value [105].

<sup>3</sup>E.g., for Nakamoto consensus the time between generated blocks is exponentially distributed [52], [82], [94].

<sup>4</sup>E.g. miners in the case of Nakamoto consensus [94]

- 5) **Atomic Swaps.** Users should be able to atomically swap XCLAIM CBAs against other assets on  $I$  or the native currency  $i$ . Changes of ownership during a swap should be reflected on both  $I$  and  $B$  consistently.
- 6) **Censorship Resistance.** No actors or intermediaries in XCLAIM should prevent a user from issuing CBAs.
- 7) **Value Redeemability.** XCLAIM should ensure users can redeem CBAs for the backing currency  $b$ , or be reimbursed with the equivalent monetary value.
- 8) **Scale-out.** The total amount of CBAs available for circulation should increase with the total amount<sup>5</sup> of backing funds locked up in blockchain  $B$ .
- 9) **Generality.** XCLAIM should not rely on a single cryptocurrency implementation or specific features, but should be general to allow deployment between any two blockchains fulfilling the minimum system requirements.
- 10) **Property Equivalence.** CBAs should expose the same properties as the backing cryptocurrency, specifically with respect to *fungibility* and *divisibility*.

#### IV. STRAWMAN SOLUTION AND DESIGN ROADMAP

In this section we present a strawman solution, CENTRALCLAIM, that outlines how a CBA-based system with the actors defined in Section III-B might operate. Next, we use CENTRALCLAIM to highlight the challenges faced by XCLAIM in achieving the goals from III-E. Finally, we lay out a design roadmap for the secure design of XCLAIM. We present XCLAIM in Section V.

##### A. Strawman Solution

CENTRALCLAIM proposes the use of a single trusted intermediary on the backing blockchain  $B$  that takes the role of the *vault*. The iSC is a smart contract deployed on the issuing blockchain  $I$ . The *vault* is registered with the iSC, i.e., the iSC can verify the *vault*'s digital signature and knows the *vault*'s public key. As defined in Section III-B,  $I$  is responsible for managing the correct issuing and exchange of  $i(b)$  on  $I$ .

We assume a user Alice controls units of  $b$  on a blockchain  $B$ , while a user Dave controls units of  $i$  on a blockchain  $I$ . Alice wishes to create  $B$ -backed assets  $i(b)$  and transfer them to Dave on  $I$ . Dave, at some later point in time, wishes to redeem his units of  $i(b)$  for the corresponding amount of  $b$ .

To achieve this, CENTRALCLAIM offers four protocols: *Issue*, *Transfer*, *Swap* and *Redeem*. For simplicity, we omit any processing fees charged by the *vault* or the iSC for the use of the service. We also omit the cost of transaction fees on the underlying blockchains  $B$  and  $I$ .

**Protocol: Issue.** The *requester*, Alice, locks units of  $b$  with the *vault* on  $B$  to create  $i(b)$  on  $I$ :

- 1) *Setup.* First, Alice verifies the iSC smart contract is available on chain  $I$ , i.e., the issuing blockchain, and identifies the single backing intermediary on  $B$ , i.e., the *vault*.

<sup>5</sup>Specifically, locked collateral. To become a backing-intermediary, a user must provide at least a pre-defined minimal amount of collateral; cf. Section V-G.

- 2) *Lock.* Next, Alice generates a new public/private key pair on  $I$  and locks funds  $b$  with the *vault* on  $B$  in a publicly verifiable manner, i.e., by sending  $b$  to the *vault*. As part of locking these funds with the *vault*, Alice also specifies where the to-be-generated  $i(b)$  should be sent, i.e., Alice associates her public key on  $I$  with the transfer of  $b$  to the *vault*.
- 3) *Create.* The *vault* confirms to the issuing smart contract iSC via a signed message that Alice has correctly locked her funds and forwards Alice's public key on  $I$  to the iSC. The iSC verifies the *vault*'s signature, then creates and sends  $i(b)$  to Alice, such that  $|i(b)| = |b|$ .

**Protocol: Transfer.** Alice, the *sender*, transfers  $i(b)$  to Dave, the *receiver*, on  $I$ :

- 1) *Transfer.* Alice notifies the iSC that she wishes to transfer her  $i(b)$  to Dave using Dave's public key on  $I$ . The state of the iSC is updated and Dave becomes the new owner of  $i(b)$ .
- 2) *Witness.* The *vault* witnesses the change of ownership on  $I$  through iSC, and no longer allows Alice to withdraw the associated amount of locked  $b$  on  $B$ . The process for any further transfers from Dave to other users is analogous.

**Protocol: Swap.** Alice, the *sender*, atomically swaps  $i(b)$  against Dave's (*receiver*),  $i$  on  $I$ :

- 1) *Lock.* Alice locks  $i(b)$ , while Dave locks the agreed upon units of  $i$  (or any other asset on  $I$ ) with the iSC.
- 2) *Swap.* The iSC updates the balance of Dave, making him the new owner of  $i(b)$ , and assigns Alice ownership over  $i$ .
- 3) *Witness.* Analogous to the witness step in *Transfer*, i.e., the *vault* witnesses the change of ownership of  $i(b)$  and no longer allows Alice to withdraw the associated amount.

**Protocol: Redeem.** Dave, the *redeemer*, locks  $i(b)$  with the iSC on  $I$  to receive  $b$  from the *vault* on  $B$ ;  $i(b)$  is then destroyed:

- 1) *Setup.* Dave creates a new public/private key pair on  $B$ .
- 2) *Lock.* Next, Dave locks  $i(b)$  with the iSC on  $I$  and requests the redemption of  $i(b)$ . As part of the request, Dave specifies his new public key on  $B$  as the target for the redeem.
- 3) *Release.* The *vault* witnesses the locking and redemption request of  $i(b)$  on  $I$  and releases funds  $b$  to Dave's specified public key on  $B$ , such that  $|b| = |i(b)|$ .
- 4) *Destroy.* Finally, the *vault* confirms with the iSC that  $b$  was redeemed on  $B$ , and the iSC destroys, or *burns*, the locked  $i(b)$  on  $I$ .

##### B. Strawman Limitations

While CENTRALCLAIM, as presented in Section IV-A, already provides sufficient functionality for issuing, transferring, swapping and redeeming CBAs, it does not achieve all the goals defined in Section III-E. Namely, it does not achieve **Decentralization**, **Consistency**, **Transfer Consistency**, **Atomic Swaps**, **Censorship Resistance** or **Value Redeemability**. This is because CENTRALCLAIM is inherently centralized and

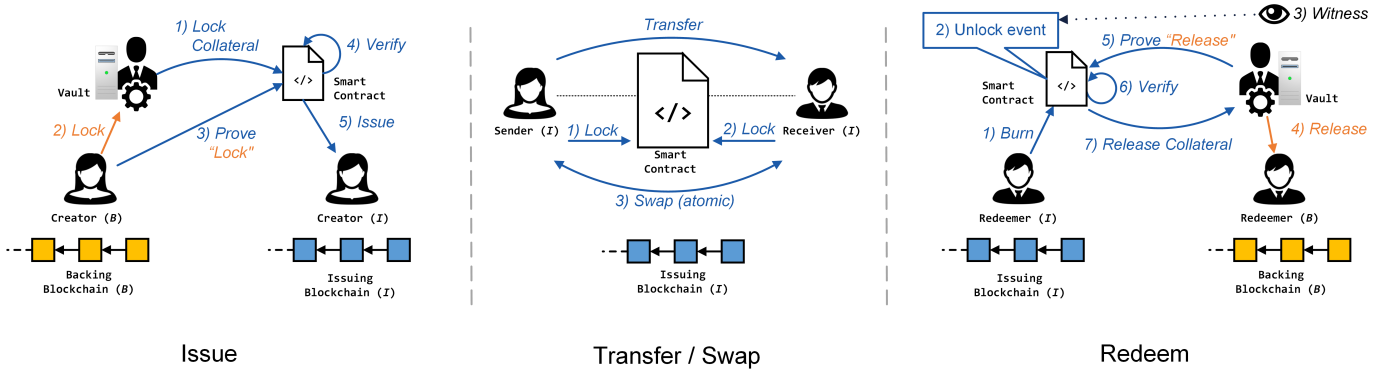


Fig. 1. High-level overview of the *Issue*, *Swap* and *Redeem* protocols in XCLAIM's (under successful execution). All parties interact with the iSC, creating a publicly verifiable audit log. Correct behavior is enforced by (i) over-collateralizing the *vault* and (ii) cross-chain transaction inclusion proofs. When issuing, the *requester* proves correctness of the lock making *Issue* non-interactive. Value redeemability is guaranteed by forcing the *vault* to *proactively* prove correctness of the *Redeem* process. As a result, XCLAIM enforces *Transfer* and *Swap* occur atomically on the backing (*B*) and issuing (*I*) blockchains.

trusts the *vault* to behave correctly. As such, the *vault* can violate these goals by misbehaving. For example, the *vault* is trusted to monitor the backing chain *B* for newly created locks of *b* and notify the iSC via a signed transaction on *I*. Should it fail to do this, it can break both **Consistency** and **Censorship Resistance**. Similarly, the *vault* is trusted to release the correct amount of *b* on *B* when a *redeemer* requests the redemption of *i(b)*. Failing to do this allows the *vault* to break **Value Redeemability**.

CENTRALCLAIM also inherently violates **Decentralization**; it exhibits a single point of failure, as backing-funds are locked with a single intermediary, the *vault*. The *vault* is therefore assumed to be interactive, i.e., always online. As such, even in the case that the *vault* behaves honestly, CENTRALCLAIM is still vulnerable to attacks against liveness, e.g. due to denial-of-service and eclipse attacks [71] on the *vault*. As such, the *vault* affects both the safety and liveness of the system.

Surprisingly however, CENTRALCLAIM already exhibits significant advantages over existing systems offering digital tokens backed by real-world assets, e.g. the US dollar [29] or gold [115]. This is because CENTRALCLAIM achieves **Transparency**: any user with read access to the backing and/or issuing blockchains can audit the actions of the *vault*. As such, dishonest behavior can be detected immediately, and users will cease to trust the misbehaving parties. CENTRALCLAIM also provides **Generality**, as it relies on no implementation-specific features, while **Property Equivalence** of issued CBAs can be achieved by parameterization of the iSC. Finally, it is easy to see CENTRALCLAIM also achieves **Scale-Out**, however, this relies on the availability and honest behavior of the *vault*.

### C. XCLAIM Design Roadmap

To address the security challenges and limitations of CENTRALCLAIM, we outline the design roadmap for XCLAIM and introduce the building blocks used in its construction:

- 1) In Section V-B, we remove the trust required for the *vault* during issuing and make the issuing process non-

interactive, thus achieving **Censorship Resistance**. For this, we use *chain relays* to allow programmatic verification of transaction inclusion proofs for *B* on *I* and require all parties to *proactively* prove correct behavior.

- 2) In Section V-C, we show how to incentivize the correct behaviour of the *vault* during redemption through the introduction of *collateralization* and *punishments*, enforcing a *proof-or-punishment* model. We highlight race condition and griefing vulnerabilities in the *Issue* process due to the use of collateralization, and present two simple but effective mitigations: (i) *deferred collateral withdrawal* and (ii) *collateralized issue commitments*. This way, we achieve **Value Redeemability**, **Transfer Consistency** and **Atomic Swaps** under a fixed exchange rate.
- 3) In Section V-D, we show how to prevent collateral deterioration due to exchange rate fluctuations by introducing (i) *over-collateralization*, (ii) *collateral adjustment* and (iii) *automatic liquidation*. As a result, XCLAIM achieves **Value Redeemability**, **Transfer Consistency** and **Atomic Swaps** under non-constant exchange rates.
- 4) In Section V-E, we discuss challenges of *exchange rate oracles* and outline how XCLAIM enables the use of decentralized exchanges as a more secure data source.
- 5) In Section V-F, we make a first step towards **Decentralization** by introducing *Replace*, a non-interactive atomic cross-chain swap protocol for *vault* replacement, allowing intermediaries to come and go in XCLAIM as they wish without being vulnerable to theft.
- 6) Finally, in Section V-G, we achieve **Decentralization** and **Scale-Out** by removing the single point of failure in CENTRALCLAIM. We do this by making XCLAIM a multi-*vault* system where *any user* can assume the *vault* role. We show how XCLAIM achieves *Sybil-attack resistance* and discuss the resulting open market and collateral rebalancing among *vaults*.

## V. XCLAIM SECURE DESIGN

This section presents the secure design of XCLAIM. We first provide the high-level overview and intuition behind XCLAIM, and then follow the technical roadmap outlined in Section IV to provide a detailed system description.

### A. XCLAIM Overview

XCLAIM overcomes the limitations of CENTRALCLAIM through three primary techniques: (i) constructing secure audit logs on both the backing blockchain  $B$  and the issuing blockchain  $I$  to trace all actions in the system; (ii) transaction inclusion proofs via chain relays to prove correct behaviour on the backing blockchain  $B$  to the iSC; and (iii) collateralization to incentivize correct behaviour through *proof-or-punishment*. We provide a brief overview and intuition for XCLAIM below. Figure 1 illustrates the *Issue*, *Transfer*, *Swap* and *Redeem* protocols in XCLAIM, while the design of the issuing smart contract iSC is shown in Figure 2.

Similar to CENTRALCLAIM, in XCLAIM, funds on the backing blockchain  $B$  are secured by backing intermediaries, *vaults*. The *vaults* store locked coins  $b$  on blockchain  $B$  and handle issue and redeem requests. To avoid necessitating trust in the *vault* however, XCLAIM uses collateral to incentivize behaviour; XCLAIM requires actors, such as the *vault*, to deposit collateral on blockchain  $I$ , owned by the iSC. Every action in XCLAIM is then logged securely via the iSC and misbehaving actors, such as the *vault*, are punished by *slashing* collateral belonging to them, and reimbursing wronged actors. XCLAIM ensures deposited collateral is always sufficient, even in case of exchange rate fluctuations between  $b$  and  $i$ .

For the iSC to ensure that correct behaviours have taken place on blockchain  $B$ , where the iSC does not have direct visibility, XCLAIM uses *chain relays*. Chain relays provide external blockchain data, such as the transactions in blockchain  $B$ , to the iSC executing on  $I$ . As such, the iSC can trace every action by every actor in the system across blockchains. Actors in XCLAIM therefore proactively prove their honest behaviour to the iSC via the chain relay; failure to do so results in punishment. By combining secure audit logs, chain relays, and collateralization in this way, XCLAIM can overcome the limitations of CENTRALCLAIM, and achieve the goals defined in Section III-E.

### B. Chain Relays: Cross-Chain State Verification

As outlined in Section V-A, XCLAIM employs chain relays [6], [78], [111] to provide data from the backing blockchain  $B$  to the iSC on the issuing blockchain  $I$ . We use chain relays to make the issuing of assets  $i(b)$  on  $I$  non-interactive and thus achieve **Censorship Resistance**. For this, XCLAIM introduces a chainRelay component to the smart contract iSC (cf. Figure 2). The chainRelay is capable of interpreting the state of the backing blockchain  $B$  and provides functionality comparable to an SPV or light client [3], [18], [23], [34]. That is, a chainRelay stores and maintains block headers from blocks in  $B$  on  $I$ , and provides two functionalities to the iSC: *Transaction inclusion verification* and *Consensus verification*:

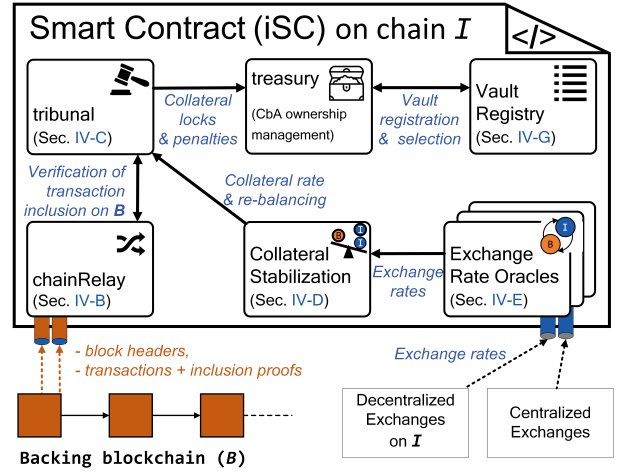


Fig. 2. High level overview of the architecture of the XCLAIM smart contract (iSC) and the interactions between its components. References to sections introducing each component are provided. The treasury refers to the basic ledger functionality of  $I$ .

- **Transaction inclusion verification.** The chainRelay stores every block header in the backing blockchain  $B$  on  $I$ . Each block header in chainRelay contains the root of the Merkle tree [91] containing all transactions (or their identifiers) for that block. To verify the correct inclusion of a transaction in a block in  $B$ , it is sufficient to provide the Merkle tree path from the root to the leaf containing the transaction (identifier) and the transaction data itself. This verification can be then be performed in a non-interactive manner by the chainRelay as part of the iSC.
- **Consensus verification.** The chainRelay can also verify that any given block header is part of the backing blockchain  $B$ , i.e., has been agreed upon by the majority of consensus participants. In XCLAIM, consensus verification depends on the consensus mechanism used by the backing blockchain  $B$ . For Nakamoto consensus [94], the chainRelay must (i) know the difficulty adjustment policy and (ii) verify that the received headers are on the chain with the most accumulated Proof-of-Work [6], [78]. For Proof-of-Stake blockchains, e.g Ouroboros [77], the chainRelay must (i) be aware of the protocol/staking epochs and (ii) verify the signature membership of elected leader(s) for the threshold/multi-signatures of block headers [65]. For permissioned (Proof-of-Authority) systems, the verification is analogous, or simpler, if the consensus participants are pre-defined [113]. We provide a formal definition for the necessary functionality of Proof-of-Work chain relays in Appendix B.

XCLAIM uses the chainRelay to modify the *Issue* protocol presented in CENTRALCLAIM (Section IV-A); after locking funds  $b$  with the *vault*, the *requester* must prove to the chainRelay that funds were locked correctly by presenting the transaction generated when sending  $b$  to the *vault* in  $B$ . The chainRelay can then verify that the given transaction has been securely included in  $B$  and the funds were locked correctly.

If successful, the chainRelay triggers the automatic issuing of the corresponding amount of  $i(b)$  in the iSC.

Similarly, XCLAIM also modifies the *Redeem* protocol; upon a redeem request being made by the user, the *vault* is required to prove that (i) the funds  $b$  were released to the *redeemer* and (ii) the released amount corresponds to the burnt CBA, i.e.,  $|b| = |i(b)|$ . This is done by presenting the chainRelay with the transaction that sends  $b$  to the *redeemer* within a maximum delay  $\Delta_{redeem}^I$ . Should the *vault* fail to comply, the *vault* incurs a financial penalty and the iSC guarantees reimbursement to the *redeemer*: we discuss this later in Section V-C.

*Security Arguments:* We make the following observations to informally argue security: assuming correct functionality of the chainRelay and that our threat models and assumptions hold, i.e., the *vault* cannot tamper with the correctness of the iSC, XCLAIM makes the *Issue* protocol trustless, non-interactive, and achieves **Censorship Resistance**. This is because the *requester* can freely execute the *Issue* protocol using the iSC, without asking permission of the *vault*. Furthermore, it is also easy to see that by removing interaction with the *vault*, the liveness of the *Issue* protocol depends only on the availability of the iSC, which is guaranteed by the properties of the underlying blockchain  $I$ .

For the *Redeem* protocol, XCLAIM forces the *vault* to prove correct behavior, rather than imposing the requirement for reactive fraud-proofs when something goes wrong. We note that for the *vault* to have a realistic time window to provide a proof, we must consider the security parameters for  $B$  and  $I$ , as well as the block generation rates, i.e., it must hold that  $\Delta_{redeem}^I > k^B \cdot \frac{1}{r_r} + k^I$ .

We discuss the security of the chainRelay in Section VII-A.

### C. Tribunal: Incentives via Collateralization

We next modify CENTRALCLAIM by introducing *collateral* as a means to incentivize honest behavior in XCLAIM and impose punishment on misbehaving parties through the iSC. We refer to this component of the iSC as the tribunal (cf. Figure 2). Specifically, we modify CENTRALCLAIM by requiring the *vault* to lock up units of  $i$  as collateral when registering with the iSC, which we denote as  $i_{col}$ . If the *vault* fails to prove correct execution of the *Redeem* protocol, the collateral is automatically used by the iSC to compensate the *redeemer* and to pay an additional punishment fee.

For collateralization of the *vault* to be effective in terms of maintaining incentives to behave honestly, the collateral must be at least equal to the funds locked on backing chain  $B$ . One challenge faced by this approach in XCLAIM is that the *vault*'s collateral is locked in currency  $i$ , while the value it is balanced against is measured in currency  $b$ . To this end, we must ensure  $i_{col} \geq b_{lock} \cdot r_{ex}(b, i)$  holds, where  $b_{lock}$  refers to the units of  $b$  locked with the *vault* on  $B$  and  $r_{ex}(b, i) \in \mathbb{R}_{\geq 0}$  denotes the exchange rate between  $b$  and  $i$ . For ease of explanation, at this point we assume the exchange rate  $r_{ex}(b, i)$  is constant. We discuss challenges of non-constant exchange rates and mitigation thereof in Section V-D.

To ensure **Value Redeemability**, users must only initiate the *Issue* protocol, if sufficient collateral is provided by the *vault* in the iSC. However, the “naïve” *Issue* protocol of CENTRALCLAIM exhibits vulnerabilities to race conditions: (i) the *vault* can attempt to withdraw collateral before the *requester* can finalize the issuing process, i.e., provide the transaction inclusion proof to the chainRelay, and (ii) multiple *requesters* can attempt to simultaneously issue for the same amount of the *vault*'s collateral, triggering a race where the “looser's” locked funds  $b_{lock}$  are not secured by collateral. We hence present two mitigations for the above attacks in XCLAIM:

*Deferred Collateral Withdrawal:* The *vault* may exploit race conditions due to network latency, delays  $\Delta^B$  and  $\Delta^I$  or DoS attacks against the *requester* to attempt collateral withdrawal *after* a lock on  $B$  is executed, committing unpunished theft. We derive a simple announce-delay-withdraw scheme to prevent such attacks. Specifically, we require the *vault* to announce collateral withdrawal publicly via the iSC. The iSC allows users to finalize (in theory also to initiate new) issue process within a delay  $\Delta_{withdraw}$ , after which the *vault* may withdraw the remaining unused collateral. Thereby, the lower bound for  $\Delta_{withdraw}$  is the upper bound on transaction inclusion proofs  $\bar{\Delta}_{relay}$  defined in Section V-B, i.e.,  $\Delta_{withdraw} > \bar{\Delta}_{relay}$ .

*Collateralized Issue Commitments:* To prevent multiple *requesters* from locking funds  $b$  for the same amount of the *vault*'s collateral, we introduce a registration step to the setup phase of the *Issue* protocol. Specifically, a *requester* must register an issue request for  $i(b)$  with the iSC, which temporarily locks the corresponding amount of the *vault*'s collateral. Within the following delay  $\Delta_{commit} > \bar{\Delta}_{relay}$  the *requester* can then safely execute the remaining steps of the *Issue* protocol. The iSC thereby only accepts pre-registered issuing attempts. To avoid *griefing* attacks by malicious *requesters*, i.e., continuous locks of the *vault*'s collateral, we require the *requester* to commit to issuing by providing collateral herself. The latter is used to reimburse the *vault* in case of failure. We note that multiple collateralized commitments can be created in parallel, of which only a single one will be accepted by the iSC, on a first-come-first-served basis. In this worse-case scenario, the losses faced by *requesters* are hereby limited to a transaction fee on  $I$ .

*Security Arguments:* By introducing collateralization in XCLAIM we ensure that an economically rational *vault* has no incentive to misbehave. Deferred collateral withdrawal and collateralized issue commitments thereby prevent the *vault* from exploiting network related race conditions to defraud users. Assuming a constant exchange rate  $r_{ex}(b, i)$ , participating in XCLAIM is a hence *zero-sum-game* [112] with a Nash Equilibrium [96] where both the users (*requester*, *redeemer*, *sender* and *receiver*) and the *vault* adhere to protocol rules. Specifically, the *vault* has no financial gain from misbehaving, as the iSC will reimburse any potential victims using  $i_{col}$  and possibly incur additional penalties. As such, XCLAIM achieves **Value Redeemability** under constant  $r_{ex}(b, i)$ . In

addition, since the *Issue* protocol is now non-interactive (see Section V-B), the iSC is guaranteed to execute correctly due to our threat model and blockchain assumptions, and the *vault* is incentivized to behave correctly during the *Redeem* protocol, XCLAIM also achieves **Transfer Consistency** and **Atomic Swaps**.

#### D. Mitigating Exchange Rate Fluctuations

Until now, we have assumed the both the exchange rate  $r_{ex}(b, i)$  and the collateral  $i_{col}$  provided by the *vault* remain unchanged. However, real world observations have shown the exchange rate  $r_{ex}(b, i) \in \mathbb{R}_{\geq 0}$  between the two cryptocurrencies may be susceptible to strong fluctuations. To ensure **Value Redeemability** under non-constant exchange rates, we hence (i) *over-collateralize* the *vault*, (ii) enable adjustment of the *vault*'s locked collateral and (iii) introduce automatic liquidation to prevent financial loss in case of extreme devaluation of  $i$ .

*Over-Collateralization*: To prevent failure due to sudden drops of  $r_{ex}(b, i)$ , we over-collateralize the *vault* by a factor  $r_{col} \in \mathbb{R}_{\geq 1}$  (i.e.,  $r_{col} \geq 1.0$ ), creating a buffer to account for possible exchange rate fluctuations. For secure operation, the following must hold for the lifecycle of XCLAIM:

$$i_{col} \geq b_{lock} \cdot (r_{col} \cdot r_{ex}(b, i)) \geq b_{lock} \quad (1)$$

As a result, the over-collateralization factor  $r_{col}$  becomes a security parameter in XCLAIM. The combination of  $r_{col}$  with the exchange rate  $r_{ex}(b, i)$  then defines how many units of the backing cryptocurrency  $b$  a *requester* can safely lock with the *vault*, i.e., the maximum amount of safely issuable  $i(b)$ :

$$\max(i(b)) = \frac{i_{col}}{r_{col} \cdot r_{ex}(b, i)} \quad (2)$$

For clarity, we denote *blocked* collateral, i.e., already used to securely issue  $i(b)$ , as  $i_{col}^- = i(b) \cdot r_{col} \cdot r_{ex}(b, i)$  and *free* collateral as  $i_{col}^+ = i_{col} - i_{col}^-$ .

*Collateral Adjustment and Automatic Liquidation*: Over-collateralization helps mitigate extreme fluctuations in the short term, however may be insufficient to securely handle long-term issuing. To this end, we enable the adjustment of the *vault*'s collateral and introduce the notion of automatic liquidation of  $i(b)$  by the iSC. We derive a simple multi-stage system for collateral  $i_{col}$ . The latter defines the behavior of the iSC, based on the observed collateral rate  $r_{col}^* = \frac{i_{col}^- + i_{col}^+}{b_{lock} \cdot r_{ex}(b, i)}$  and the (parameterized) ideal rate  $r_{col}$ . Specifically, we introduce thresholds  $r_{col} > r_{col}^{liq} > 1.0$ . For ease explanation, we assume an exemplary collateral rate  $r_{col} = 2.0$ .

- *Secure Operation* : The *vault* has locked more collateral than necessary to ensure value redeemability in XCLAIM, i.e., new  $i(b)$  can be issued. Alternatively, the available free collateral  $i_{col}^+$  can be withdrawn by the *vault*, as long as  $r_{col}^* \geq r_{col}$  holds
- *Buffered Collateral*: The collateral rate  $r_{col}^*$  has dropped below ideal rate  $r_{col}$ , however there is sufficient buffer to ensure secure operation of XCLAIM. However, as defined in Eq. 2, no new  $i(b)$  can be issued.

- *Liquidation*: The collateral rate is critically close to the lower bound of 1.0 (e.g.  $r_{col}^{liq} = 1.05$ ). If the *vault* does not re-balance  $r_{col}^*$  by increasing  $i_{col}$ , the iSC automatically initiates *Redeem* for all existing  $i(b)$ . The remaining collateral buffer  $r_{col}^* - 1.0 > r_{ex}(b, i)$  is thereby used to cover transaction fees. This measure is necessary to prevent users from facing financial loss, should  $r_{col}^*$  drop below 1.0. We note this feature can be opt-in/opt-out, in case users trust in the honest behavior of the *vault*, despite value redeemability not being guaranteed.

*Security Arguments*: By requiring over-collateralization of the *vault*, we ensure there exists a buffer to counter sudden exchange rate drops. Since the *vault* can now update collateral, it can, in the optimistic case, maintain secure operation of XCLAIM. Should the latter fail, the automatic liquidation feature of the iSC guarantees users do not face financial losses, even in case of collateral deterioration. As a result, XCLAIM ensures **Value Redeemability**, **Transfer Consistency** and **Atomic Swaps** under non-constant exchange rates. Note: XCLAIM provides *better security properties than centralized exchanges* even if the observed collateralization rate  $r_{col}^*$  were theoretically to drop below 1.0, as the iSC ensures user will be reimbursed at least partially.

#### E. Exchange Rate Oracles

To correctly parameterize the XCLAIM collateralization rate defined in Section V-C, we assume an oracle  $\mathcal{O}$  which provides the iSC with the  $r_{ex}(b, i)$ . Access to information external to the issuing blockchain  $I$ , however, requires trust in (i) whoever provides the exchange rate to  $\mathcal{O}$  (i.e. the data provider) and (ii) the bias resistance and non-repudiation of  $\mathcal{O}$  itself. While the exact implementation details of  $\mathcal{O}$  lie beyond the scope of this paper, we elaborate on possible approaches. The data provider for  $r_{ex}(b, i)$  could be performed by existing exchanges, as these maintain sources of exchange rate information. To achieve bias resistance and non-repudiation of  $\mathcal{O}$ , one could further rely on TLS session notarization [100], trusted hardware [121] or voting games [31], [107]. Points of centralization, making the system susceptible to denial of service attacks, are avoidable by replicating and deploying multiple oracles for each data source and relying on majority votes to achieve consensus.

XCLAIM enables to bootstrap more secure oracles for  $r_{ex}(b, i)$ . When  $i(b)$  are issued via XCLAIM and traded against  $i$  via decentralize exchanges, the respective rates  $r_{ex}(b, i)$  are inherently published to blockchain  $I$  via the swap transactions. The more  $i(b)$  are issued via XCLAIM and traded on decentralized exchanges, the more reliable the exchange rate data sampled from  $I$ .

*Security Arguments*: Relying on external data providers introduces points of centralization into any blockchain system. Even if oracles were replicated on  $I$ , XCLAIM still suffers this problem. While we must accept this trade-off for data from centralized platforms, exchange rates of decentralized exchanges are indisputably logged in the underlying blockchain  $I$ . Thereby, manipulation attempts of  $r_{ex}(b, i)$ , e.g.



via artificially inflated trading activity or so called *wash trading* [86], are not only detectable but also significantly more expensive than in the case of centralized exchanges (exchange transactions must be included in the underlying blockchain to be considered). As such, once sufficient trade volume is available on  $I$ , the reliability and bias resistance of XCLAIM is significantly improved.

#### F. Atomic vault Replacement

Until now, we have assumed the *vault* remains in XCLAIM until the locked funds it holds are redeemed by users. In a real world scenario, the *vault* may wish to leave and transfer their role to another party earlier. To avoid troublesome out-of-band private key exchanges, we introduce *Replace*, a non-interactive atomic cross-chain swap (ACCS) protocol based on cross-chain state verification.

**Protocol: Replace.** *vault* migrates locked funds  $b$  to *vault'*, who replaces *vault*'s collateral in the iSC.

- 1) *Setup*. The *vault* submits a *replacement* request to the iSC and locks up collateral  $i_{col}^{replace}$ , sufficient to cover costs of a transaction on  $I$ .
- 2) *Lock*. A new candidate *vault'* can lock the corresponding amount of collateral for a pre-defined period  $\Delta_{replace}$  with the iSC on  $I$ , such that  $|i_{col}^{vault}| = |i_{col}^{vault'}|$ , providing their public key on backing chain  $B$ .
- 3) *Migrate*. Within  $\Delta_{replace} > \overline{\Delta}_{relay}$ , the still active *vault* must migrate the locked  $b_{lock}$  to the public key of *vault'* on  $B$  and submit the corresponding transaction inclusion proofs to the chainRelay on  $I$ .
- 4) *Release*. The chainRelay verifies the migration was executed correctly on  $B$  and the iSC releases the old *vault*'s collateral, i.e., both  $i_{col}^{vault}$  and  $i_{col}^{replace}$ . If the *vault* does not execute the migration on  $B$  within  $\Delta_{replace}$ , the iSC releases the new candidate's collateral, while using  $i_{col}^{replace}$  to reimburse wasted transaction fees.

*Security Arguments:* By allowing *vaults* to leave the system, without risking loss of collateral, we make a first step towards **Decentralization** in XCLAIM. Furthermore, this improves **Scale-Out**, as replacement allows *vaults* with larger collateral capabilities to enter the system. The chainRelay enables the iSC to ensure the migration of locked  $b_{lock}$  and replacement of collateral  $i_{col}$  occur atomically. Griefing attacks by the old *vault* are mitigated by the collateral required to initiate the *Replace* protocol.

#### G. Multi-vault System: Removing Single Points of Failure

Until now, we assumed a single *vault*. The design of XCLAIM allows it to be easily extended to a multi-*vault* system. Hence, we now allow *any* user to become a *vault* by registering with the iSC and providing collateral. The list of *vaults* is maintained in a public registry in the iSC. By allowing both *requesters* and *redeemers* to freely choose which *vault* they wish to use for issuing and redeeming, we create a free market driven by charged fees and the observed collateral rate  $r_{col}^*$  of each *vault*. To prevent financial gains from Sybil attacks, where a single user creates multiple low

collateralized *vaults*, the iSC can enforce (i) a minimum collateral amount and (ii) a fee model based on issued volume, rather than “pay-per-issue”. In practice, users can also easily filter out low-collateral *vaults*.

The availability of multiple *vaults* further allows a *redeemer*, upon a failed *Redeem* caused by a *vault*, to choose between (i) being reimbursed from the slashed collateral  $i_{col}$  or (ii) retrying the *Redeem* process with a different *vault*.

*Collateral Balancing:* We further use the *Replace* protocol to enable re-balancing of a *vault*'s collateral. Assume a *vault*'s observed collateralization rate  $r_{col}^*$  has dropped significantly below ideal rate  $r_{col}$ . To prevent automatic liquidation, the *vault* must contribute additional collateral to the iSC on  $I$ . Alternatively, the *vault* can execute *Replace* to migrate a fraction of total locked coins  $b_{lock}$  to *vault'*, so as to re-balance her collateralization rate  $r_{col}^*$ . Should no single *vault* have sufficient free collateral to complete the re-balancing, the *Replace* protocol can be executed iteratively with multiple *vaults*, e.g. until  $r_{col}^* \geq r_{col}$  holds. This procedure is specifically useful if a *vault* cannot provide additional collateral due to insufficient funds on  $I$ .

*Automatic Liquidation:* A challenge that arises in a multi-*vault* system is correct automatic liquidation. Deterioration of collateral of a single *vault* does not affect the entire system, but only the corresponding fraction of issued  $i(b)$ . If collateral re-balancing as described above fails, the iSC must identify which users are to be considered for automatic liquidation. In a first step, the iSC can offer *beneficial* liquidation, i.e., redemption of  $i(b)$  against the corresponding amount of  $b_{lock}$  and an additional small premium in  $i$ , deducted from the *vault*'s available collateral ( $r_{col}^* - 1.0$ ). Should insufficient users wish to execute *Redeem*, the iSC, as a final fallback, equally distributes the liquidation among all users of XCLAIM. Note: tracing CBAs back to the *vault* they were issued with would make the assets non-fungible, which is contrary to XCLAIM's system goals.

*Security Arguments:* By allowing any user to register as a *vault*, we achieve **Decentralization** in XCLAIM. That is, similar to permissionless blockchains, the set of *vaults* can change dynamically and must not be pre-defined. As a result, an adversary attempting a DoS attack on XCLAIM must now not only target all registered *vaults* but also prevent new registrations. The latter, would require a DoS attack on the iSC, which in turn requires a successful DoS attack on the issuing chain  $I$  (cf. Section VII). XCLAIM further achieves improved **Scale-Out** – by opening collateral registration to all users, arguably more units of  $i$  become available, than with a single, centralized *vault*. We note, the identities of *vaults* are Sybil-resistant due to collateralization, while any potential financial gain from Sybil attacks is preventable by an adequate fee model or efficient filtering.

## VI. FORMAL PROTOCOL SPECIFICATION

This section provides a formal specification for XCLAIM. It presents the *Issue*, *Transfer*, *Swap*, *Redeem* and *Replace*

protocols, as well as the requirements XCLAIM imposes on the backing and issuing blockchains.

**Notation.** We differentiate between *state changing* and *non-state changing* operations in XCLAIM; state changing operations result in new transactions ( $T$ ) in the underlying blockchain, while non-state changing operations, such as verifying operations, are “read-only”, returning boolean values ( $\top|\perp$ ). We use  $T_{id}^B$  to refer to a transaction created on chain  $B$  with identifier  $id$  and  $T_{id}^I$  for transactions on  $I$  respectively. The execution of an operation on input  $in$  that produces an output  $out$  is denoted as  $operation(in) \rightarrow out$ .

To indicate that an operation is executed by a user  $user$ , we write  $user.operation$ . We identify a user  $user$  in XCLAIM by her public key  $pk_u^X$  (with corresponding private key  $sk_u^X$ ), where  $X$  can be either blockchain  $B$  or  $I$ . For readability, we often write  $user^X$  when referring to  $pk_u^X$ . We use  $cond$  to refer to locking and unlocking conditions for funds on both  $B$  and  $I$ , e.g. a condition for a transaction may be the digital signature of user  $u$  on chain  $X$  with private key  $sk_u^X$ , denoted as  $\sigma_{user}^X$ .

### A. XCLAIM Operations and Protocols

A summary of the symbols defined thus far is provided in Appendix A. We parameterize XCLAIM, using: (i) the blockchain security parameters  $k^B$  and  $k^I$ ; (ii) block generation rates  $\tau^B$  and  $\tau^I$ ; (iii) collateral rate  $r_{col}$  and the automatic liquidation threshold  $r_{col}^{liq}$ ; and (iv) the delays  $\Delta_{redeem}$ ,  $\Delta_{withdraw}$ ,  $\Delta_{commit}$ ,  $\Delta_{replace}$  used in the *Issue*, *Redeem* and *Replace* protocols. XCLAIM exhibits the following operations on the backing and issuing chains  $B$  and  $I$ .

**Backing Blockchain:** For the backing blockchain  $B$ , operations are executed by the *requester* and the *vault*. We differentiate between the two:

Operations performed by the *requester*:

- $lockB(b, cond) \rightarrow T_{lock}^B$  which locks coins  $b$  on chain  $B$  under conditions  $cond$ .

Operations performed by the *vault*:

- $verifyOp(operation, T_{op}^I, \Delta_{op}) \rightarrow \top|\perp$  which verifies that operation was executed on  $I$ , i.e., that  $T_{op}^I$  is securely included in  $I$  according to  $k^I$  and within optional delay  $\Delta_{op}$  as per XCLAIM parameters.
- $redeemB(b, user^B) \rightarrow T_{redeem}^B$  that releases locked coins  $b$  to  $user^B$ .
- $transferB(b, user^B) \rightarrow T_{transfer}^B$  which transfers ownership of  $b$  to user  $user^B$ .

**Issuing Blockchain:** For the issuing blockchain  $I$ , operations are executed by the iSC:

- $lock(x, cond) \rightarrow T_{lock}^I$  which locks  $x$  under conditions  $cond$  on  $I$ , where  $x$  can be  $i(b)$  or  $i$ .
- $release(x, user^I) \rightarrow T_{release}^I$  that releases asset or coin  $x$  to  $user^I$ , where  $x$  can be  $i(b)$  or  $i$ .
- $slash(x, user^I, user_*^I) \rightarrow T_{slash}^I$  that destroys or slashes collateral funds  $x$  of  $user^I$  and reimburses them to  $user_*^I$ , where  $x$  can be  $i(b)$  or  $i$ .

### Algorithm 1 XCLAIM’s Issue, Transfer, Swap, Redeem and Replace protocols for Cryptocurrency-backed Assets.

```

1: protocol Issue
2:   vault.lock( $i_{col}$ )
3:   requester.commit( $b_{lock}, pk_{requester}^B, pk_{vault}^I, i_{col}^{commit}$ )
4:   requester.lockB( $b_{lock}, \sigma_{vault}^B$ ) /*  $\rightarrow T_{lock}^B$  */
5:   requester submits  $T_{lock}^B$  to iSC calling verifyBOP
6:   if iSC.verifyBOP( $lockB, T_{lock}^B, \Delta_{commit}$ ) =  $\top$  then
7:     iSC.issue( $b_{lock}, pk_{requester}^B$ )
8:     iSC.release( $i_{col}^{commit}, pk_{requester}^I$ )
9:   else
10:    iSC.slash( $i_{col}^{commit}, pk_{requester}^I, pk_{vault}^I$ )

11: protocol Swap
12:   sender.lock( $i(b), \sigma_{SC}^I$ )
13:   receiver.lock( $i, \sigma_{SC}^I$ )
14:   iSC.swap( $i(b), pk_{receiver}^I, i, pk_{sender}^I$ )

15: protocol Transfer
16:   sender calls iSC.transfer( $i(b), pk_{receiver}^I$ )

17: protocol Redeem
18:   redeemer calls iSC.lock( $i(b), pk_{redeemer}^B$ ) /*  $\rightarrow T_{lock}^B$  */
19:   iSC signals  $|b| = |i(b)|$  is to be released to redeemer on  $B$ 
20:   if vault.verifyOp( $lock, T_{lock}^B$ ) =  $\top$  then
21:     vault.redeemB( $b_{lock}, pk_{redeemer}^B$ ) /*  $\rightarrow T_{redeem}^B$  */
22:     vault submits  $T_{redeem}^B$  to iSC calling verifyBOP
23:     if iSC.verifyBOP( $redeemB, T_{redeem}^B, \Delta_{redeem}$ ) =  $\top$  then
24:       iSC.release( $i_{col}, pk_{vault}^I$ )
25:     else
26:       slash( $i_{col}, pk_{vault}^I, pk_{redeemer}^I$ )
27:   iSC.burn( $i(b)$ )

28: protocol Replace
29:   vault.lock( $i_{col}^{replace}, \sigma_{SC}^I$ )
30:   vault'.lock( $i_{col}', \sigma_{SC}^I$ )
31:   vault.transferB( $b_{lock}, pk_{vault'}^B$ ) /*  $\rightarrow T_{transfer}^B$  */
32:   vault submits  $T_{transfer}^B$  to iSC calling verifyBOP
33:   if iSC.verifyBOP( $transferB, T_{transfer}^B, \Delta_{replace}$ ) =  $\top$  then
34:     iSC.release( $i_{col}, pk_{vault}^I$ )
35:     iSC.release( $i_{col}^{replace}, pk_{vault}^I$ )
36:   else
37:     iSC.release( $i_{col}', pk_{vault'}^I$ )
38:     iSC.slash( $i_{col}^{replace}, pk_{vault}^I$ )

```

- $commit(b, user^B, vault, i) \rightarrow T_{commit}^I$  which commits  $user^B$  to calling  $lockB(b, \sigma_{vault}^B)$  within  $\Delta_{commit}$ . Locks  $i$  as collateral conditioned on the iSC’s signature via  $lock(i, \sigma_{SC}^I)$ .
- $verifyBOP(operation, T_{op}^B, \Delta_{op}) \rightarrow \top|\perp$  which verifies operation was executed on  $B$ , i.e., securely included in  $B$  via  $T_{op}^B$  according to  $k^B$  and within optional delay  $\Delta_{op}$  as per XCLAIM parameters.
- $issue(b, user^I) \rightarrow T_{issue}^I$  which creates and allocates  $i(b)$  to  $user^I$ , such that  $|i(b)| = |b|$ .
- $transfer(x, user^I) \rightarrow T_{transfer}^I$  which transfers ownership of  $x$  to user  $user^I$ , where  $x$  can be  $i(b)$  or  $i$ .
- $swap(x, user_x^I, y, user_y^I) \rightarrow T_{swap}^I$  which transfers ownership of  $x$  to  $user_x^I$  and  $y$  to user  $user_y^I$  atomically;  $x$  and  $y$  can be  $i(b)$  or  $i$ .
- $burn(i(b)) \rightarrow T_{burn}^I$  that destroys  $i(b)$ .

The formal protocol specification for *Issue*, *Transfer*, *Swap*, *Redeem* and *Replace* is provided in Algorithm 1. We refer the interested reader to sequential representations of XCLAIM protocols in Appendix C.

## B. Blockchain Requirements

Using the system operations performed by XCLAIM as defined in section VI-A, we derive the requirements for the underlying backing and issuing blockchains  $B$  and  $I$ . We summarize our findings in Table I and provide examples for backing and issuing chains currently supported by XCLAIM. We note that neither  $B$  nor  $I$  requires a Turing-complete instruction set.

**Backing Blockchain ( $B$ ):** On the backing blockchain  $B$  we need to lock and redeem funds based on conditions *cond*, i.e., a user’s digital signature. *Boolean* operations are required to verify *cond* are either true or false. Moreover, conditions for locking and redeeming from different users require (i) a *stack* to store intermediary values, (ii) *read* and *write* operations for the current stack, and (iii) public-key encryption and signature verification. *Flow control* operations do not have to be available to users and can be expressed by stack states (empty / not empty) [42]. In addition, while public-key encryption and signatures can be implemented using basic *arithmetic* and *bitwise* operations, both script complexity and execution cost can be reduced if *cryptographic* operations, including hash and signature verification functions, are supported by the scripting language as dedicated operations. Finally,  $B$  requires a method to store data, necessary to e.g. include the target public key of the *requester* for issuing on  $I$ .

**Issuing Blockchain ( $I$ ):** To support issuing of CBAs in a smart contract, the issuing chain  $I$  requires a method to create custom assets, which are part of the consensus protocol. This can be realized by *permanent storage*, i.e., *storage read* and *storage write* operations. In accordance to our definitions (cf. Section III-A), the CBA attributes, *issuing chain*, *backing chain*, and *asset value* are represented as integers; integer balances are assigned to the *asset owner*’s public keys (or digests thereof). Modification of asset balances can be realized using *arithmetic* operations on integers, and the authorization of changes via *boolean* operations.

For transaction verification, the chainRelay requires (i) *permanent storage* to store block header, transaction, and proof data and (ii) *arithmetic*, *bitwise*, and *boolean* operations for proof verification. Verifying Merkle tree inclusion requires to traverse the data structure (both of block headers and transaction lists), i.e.,  $I$  must support *finite loops or recursion*. If  $I$  supports the same (or super-) set of *cryptographic* operations used on  $B$  (specifically hash functions and signature schemes), the verification may be executed at lesser cost. Next,  $I$  requires more complex conditional locks than  $B$ , i.e., *flow control* must be supported (in addition to *arithmetic* and *boolean* operations). Since *Issue* and *Redeem* require checking delays (measured in blocks or based on time stamps),  $I$  must allow access to XCLAIM parameters via *global parameters* (integers).

## VII. SECURITY ANALYSIS

In this section we provide an informal security analysis to supplement the design choices and security propositions presented in Sections IV and V.

TABLE I  
Required operations on backing ( $B$ ) and issuing ( $I$ ) chains including application candidates.

Requirements	Operations										Examples for supported blockchains
	Arithmetic	Boolean	Bitwise	Cryptographic <sup>†</sup>	Flow control	Local stack	Finite loops or recursion	Permanent storage	Global parameters		
Backing chain $B$	✓	✓ (✓ <sup>‡</sup> )	✓	✓	✗*	✓	✗	✓	✗		Bitcoin [97], ZCash [38], Namecoin [95], Litecoin [88], Monero [19], Ethereum [59]
Issuing chain $I$	✓	✓	✓ (✓)	✓	✓	✓	✓	✓	✓		Ethereum [59], Zilliqa [103], Cardano [7], Neo [20], Ethereum Classic [13], RSK [84]

<sup>†</sup> Not strictly required, but reduces script complexity.

<sup>‡</sup> Not necessary if native support for cryptographic operations (hash and signature verification) is available.

\* Can be represented as stack states (empty / not empty).

## A. Chain Relay Security

**Freshness:** To correctly verify inclusion proofs, the chainRelay must be up to date with the state of the backing chain  $B$ , i.e., must know the most recent block headers.

XCLAIM makes timing assumptions about the delay of transaction inclusion proofs. Hence, we must define an upper bound  $\overline{\Delta}_{relay}$  for the delay between the generation of a block containing a transaction  $T^B$  on  $B$  and the verification of the transaction inclusion proof for  $T^B$ .

Consequently, we define  $\overline{\Delta}_{relay} = k^B + \overline{\Delta}_{submit} + 2 \cdot k^I$ , where  $\overline{\Delta}_{submit}$  is the upper bound for the delay between secure inclusion of a transaction and block in  $B$  and its submission to the chainRelay on  $I$ . Consequently, we say the freshness property holds for a chain relay, if the upper bound  $\overline{\Delta}_{relay}$  holds. Note,  $\overline{\Delta}_{relay}$  is the *lower bound* for the delays specified in the *Issue*, *Redeem* and *Replace* protocols.

In the case that batched submission of  $N$  block headers within a single transaction is possible, the upper bound for the delay is defined as  $\overline{\Delta}_{batchRel} = \frac{\overline{\Delta}_{relay}}{N}$ . A chain relay supporting batching can hence fall behind  $B$  up to  $N$  blocks, while still ensuring  $\overline{\Delta}_{relay}$  holds. The same applies to relays supporting compact proofing techniques, such as NiPoPoWs [76], [78] or FlyClient [89].

**Correctness:** An adversary may attempt to *poison* a chain relay with false information regarding blockchain  $B$ . Such attacks are equivalent to *selfish mining* [62] attacks, as the adversary must produce a conflicting chain of greater length for the chain relay to accept an alternate transaction history. Even though we assume  $f < n/3$  (or  $\alpha < 33\%$ ), a poisoning attack can be successful well below this threshold if the *freshness* property of the relay does not hold. Proof-of-stake blockchains allow for similar attacks in the form of *stake bleeding* [66]. If no honest blocks are submitted to the chain relay, even a small adversary can corrupt the system by submitting block headers referencing transactions conflicting with the actual state of  $B$ , e.g., to issue fake CBAs.

To prevent relay corruption due to temporary freshness failures, we can introduce a *maturity period*  $\Delta_{maturity}$  for newly generated CBAs [34], similar to newly minted coins in PoW blockchains. If a (correct) conflicting proof is submitted within  $\Delta_{maturity}$ , the pending CBAs are not issued.

## B. Permanent Chain Splits and iSC Replication

In our model, we assume both the backing and issuing chain provide only eventual consistency guarantees for transaction inclusion, deriving security parameters  $k^B$  and  $k^I$  [63] to counter accidental reorganizations. However, we must also handle permanent chain splits or *hard forks*, where consensus rules are loosened or conflicting rules are introduced [120], resulting in multiple instances of the same blockchain.

*Permanent Split of Backing Chain:* Chain splits of the backing blockchain should provide *replay protection*, i.e., a mechanism to differentiate between the two resulting chains  $B$  and  $B'$  [90]. In this case, the chainRelay must be updated to verify the replay protection for  $B'$  (or  $B$  respectively). If no replay protection is implemented, the chainRelay, will behave according to the protocol rules of  $B$  for selecting the “main” chain. For example, it will follow the longest chain for blockchains implementing Nakamoto consensus.

*Permanent Split of Issuing Chain:* A permanent fork on the issuing blockchain resulting in two chains  $I$  and  $I'$ , as well as two instances of the same contract, iSC, identified by the same public key. To prevent an adversary exploiting this to issue multiple  $i(b)$  for the same locked  $b$ , we first require the *requester* to include the public key of the iSC (or digest thereof) in the locking transaction on  $B$ . Next, we identify two possibilities to synchronize  $i(b)$  balances on  $I$  and  $I'$ : (i) deploy a chain relay for  $I$  on  $I'$  and vice-versa to continuously synchronize iSC [24] and iSC' states or (ii) redeploy the iSC on both chains and require users and *vaults* to re-issue  $i(b)$ , explicitly selecting  $I$  or  $I'$ .

*Securing iSC Replication:* In theory, the iSC can be replicated on chain  $I$ . Therefore, the mitigations introduced above can be used to prevent an adversary from exploiting the vulnerability introduced by a *vault* who (accidentally) registers with multiple co-existing instances iSC and iSC' with the same public key. If these mitigations are not enforced, an adversary could issue multiple  $i(b)$  for the same amount of fund  $b$  locked on  $B$ , violating consistency.

## C. Denial-of-Service Attacks

XCLAIM is decentralized by design, thus making denial-of-service (DoS) attacks difficult. Given that any user with access to  $B$  and  $I$  can become a *vault*, an adversary would have to target all *vaults* simultaneously. Where there are a large number of *vaults*, this attack would be impractical and expensive to perform. Alternatively, an attacker may try to target the iSC. However, performing a DoS attack against the iSC is equivalent to a DoS attack against the entire issuing blockchain or network, which conflicts with our assumptions of a resource bounded adversary and the security models of  $B$  and  $I$ . Moreover, should an adversary perform a sybil attack and register as a large number of *vaults* and ignore service requests to perform a DoS attack, the adversary would be required to lock up a large amount of collateral to be effective. This would lead to the collateral being slashed by the iSC, making this attack expensive and irrational.

## VIII. XCLAIM (BTC,ETH) IMPLEMENTATION AND EVALUATION

We instantiate XCLAIM as XCLAIM (BTC,ETH) to issue Bitcoin-backed tokens on Ethereum. Ethereum’s virtual machine provides Turing completeness [64], fulfilling the requirements (Section VI-B) for the issuing chain  $I$ . Bitcoin, due to the limited operation set of its Script language [42] only qualifies as a backing blockchain (Section VI-B). Both Bitcoin and Ethereum use ECDSA with the secp256k1 Koblitz curve [43], [80], providing native support for the corresponding cryptographic operations. Ethereum further makes the SHA-256 and RIPEMD-160 hash functions, which are used in Bitcoin, available as pre-compiled contracts [64].

We implement the iSC smart contract on Ethereum in Solidity v0.4.24 [26] in approximately 820 lines of code. On Bitcoin we use regular P2PKH [41] transactions. We use the existing Serpent [28] implementation of BTCRelay [6] for the chainRelay component of the iSC. Bitcoin-backed tokens per our implementation are compliant with the ERC20 token standard [9], and hence usable in most services on Ethereum, including *decentralized exchanges*. To evaluate cost and performance of XCLAIM (BTC,ETH), we deploy the iSC on the Ethereum Ropsten test network [14].

For evaluations, we assume a *vault* is registered with the iSC and has locked in collateral on Ethereum.

### A. Protocol Execution Costs

We define on-chain execution costs measured in USD as the amount of Bitcoin and Ethereum transaction fees required to execute each of the protocols: *Issue*, *Transfer*, *Swap*, *Redeem* and *Replace*. The costs are calculated using current conversion rates<sup>6</sup>. In Bitcoin, transaction fees are calculated based on the transaction size, i.e., the number of consumed inputs and generated outputs. To ensure transactions are included in the next generated block without delays, we calculate fees with 40 Satoshis ( $10^{-8}$  BTC) per byte [57]. In Ethereum, transaction fees are measured in *gas*, depending both on the transaction size and the cost of executed smart contract operations [64]. We assume a gas cost of 9 Gwei based on current network fees for fast transaction inclusion [58].

We summarize our measurement results in Table II. In our measurements, we refer to the complete process of issuing (*Issue*), executing a trade (*Swap*) and then redeeming (*Redeem*) an arbitrary amount of Bitcoin-backed tokens on Ethereum as a *round*. Our experiments show a full protocol execution round only costs USD 1.15. The main cost factor thereby are Bitcoin transaction fees (53.9%). As such, transferring ownership over units of Bitcoin via XCLAIM Bitcoin-backed tokens on Ethereum costs only USD 0.04, in contrast to USD 0.31 if the transfer is executed natively on Bitcoin (87.1% cheaper). We note, additional costs are incurred for keeping BTCRelay up to date with the current state of the Bitcoin blockchain, amounting to approximately USD 27 per day

<sup>6</sup>Storage and execution costs are in USD as per exchange rates of 30 Nov. 2018: BTC/USD 3717.38 and ETH/USD 105.71

TABLE II

Overview of execution costs<sup>6</sup> and performance<sup>8</sup> for the *Issue*, *Transfer*, *Swap*, *Redeem* and *Replace* protocols in XCLAIM (BTC,ETH).

Protocols	Transactions	Cost (USD)			Duration (minutes)
		Ethereum	Bitcoin	Total	
<i>Issue</i>	$2^{Eth} 1^{Btc}$	0.16	0.31	0.47	75.98
<i>Swap</i>	$2^{Eth}$	0.19		0.19	5.98
<i>Redeem</i>	$2^{Eth} 1^{Btc}$	0.18	0.31	0.49	75.98
<b>Total</b>	$6^{Eth} 2^{Btc}$	0.53 (46.1%)	0.62 (53.9%)	1.15	157.94
<i>Transfer</i>	$1^{Eth}$	0.04		0.04	2.99
<i>Replace</i> †	$3^{Eth} 2^{Btc+}$	0.13	0.62	0.75	148.97+

†Duration and costs of *Replace* depend on the number of Bitcoin UTXOs which need to be migrated. Provided numbers are lower bounds.

(not included in the table). These costs are fixed and shared among all users of XCLAIM; each user’s share decreases with higher adoption of XCLAIM. We further note this number constitutes an *upper bound* given current prices: (i) the existing implementation of BTCRelay is non-optimal<sup>7</sup> and (ii) our measurements consider the worst case scenario where batched block header submissions are not available (cf. Section V-B).

### B. Performance

We evaluate the performance of XCLAIM (BTC,ETH) with respect to the duration of the *Issue*, *Transfer*, *Swap*, *Redeem* and *Replace* protocols (measurements provided in Table II). Thereby, we adhere to the recommended security parameters regarding transaction confirmations based on our threat model ( $\alpha \leq 33\%$ ):  $k^B = k^{BTC} = 6$  and  $k^I = k^{ETH} = 12$ . Recall, we consider a transaction in block at position  $j$  as securely included when the blockchain tip reaches position  $h$ , with  $h - j \geq k$ . At the time of writing, the block time in Bitcoin amounts to 10 minutes, and in Ethereum to 14 seconds.

One execution round of issuing, atomically swapping and redeeming of Bitcoin-backed token on Ethereum requires 2 Bitcoin transactions (1 user transaction and 1 *vault* transaction) and 6 Ethereum transactions (5 user transactions and 1 *vault* transaction). The end-to-end process is securely completed after 158 minutes; a *Swap* only takes 6 minutes, while *Issue* and *Redeem* account for the greater part of the delay due to Bitcoin transaction processing (96,2%). Note: we can use off-chain payment channels [75], [92] on the issuing chain (Ethereum) to significantly reduce execution cost and make *Swap* real time (see Section IX).

### C. Comparison to Atomic Swaps

We compare the performance and execution costs of XCLAIM (BTC,ETH) to that of Atomic Cross-Chain Swaps (ACCS) based on hashed-timelock contracts [4], [5], [110]. We visualize the results of our experiments in Figure 3.

<sup>7</sup>The Serpent language is not actively maintained (last commit on 1 October 2017) and is not optimized to the current version of the EVM, resulting in higher execution costs. More efficient proofing techniques can further reduce costs [74], [76], [89], [108].

<sup>8</sup>Performance is measured in minutes and includes recommended security parameters: 6 conf. a 10 min for Bitcoin; 12 conf. a 14 sec for Ethereum.

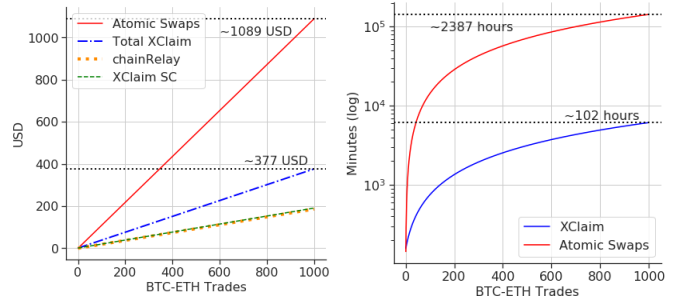


Fig. 3. Comparison of execution costs<sup>6</sup> (Left) and performance<sup>8</sup> (Right, logarithmic y-axis) of BTC-ETH trades with XCLAIM (BTC,ETH) and Atomic Cross-Chain Swaps. For 1000 swaps XCLAIM (BTC,ETH) is 95.7% faster and 65.4% cheaper.

Each interactive atomic swap requires users to create two transactions on both Bitcoin and Ethereum (4 in total). Including a minimum necessary delay to prevent race conditions, an atomic swap takes approximately 146.5 minutes to execute securely. Note: we omit the additional necessary time to establish out-of-band channels and exchange revocation transactions in ACCS; hence the ACCS measurements are *lower bounds*.

In XCLAIM, after an initial *Issue* process, each additional *Swap* requires only 2 Ethereum transactions. As such, using XCLAIM to atomically exchange BTC against ETH is already more efficient than ACCS after the second swap; for 1000 swaps XCLAIM is 95.7% faster.

Similarly, XCLAIM (including BTCRelay fees) outperforms ACCS cost-wise after the second trade; for 1000 trades XCLAIM is 65.4% cheaper than ACCS. Note, that a significant cost factor in XCLAIM (BTC,ETH) are the non-optimized BTCRelay maintenance fees, which e.g. account for 49.3% of the incurred cost for 1000 swaps.

## IX. APPLICATIONS

This section provides a brief overview of several new and novel applications enabled by XCLAIM cryptocurrency-backed tokens. These applications illustrate how XCLAIM paves the way for usable and scalable cross-chain communication.

*Cross-Chain Payment Channels:* Cryptographic payment channels [48], [56], [75], [87], [92] address the performance limitations of blockchain protocols [51], [67] by avoiding the need to publish every transaction on the blockchain. Instead, transactions are executed directly between participants *off-chain*, and only the final balances of the participants are published on the blockchain. Despite improving transaction throughput and latency considerably, payment channels cannot execute payments across different blockchains. As such, users are required to setup and instantiate multiple channels, one for every blockchain they wish to participate in. With XCLAIM, however, payment channels deployed on a blockchain capable of issuing XCLAIM CBAs become cross-blockchain compatible automatically; users can transfer CBAs as per normal in a payment channel network, and later redeem those CBAs for native coins. As such, XCLAIM allows existing issuing blockchains, such as Ethereum, to process transactions of any

backing cryptocurrency off-chain, without requiring changes to the underlying code. XCLAIM can therefore be used to provide novel contributions to state of the art payment channels.

*Temporary Transaction Offloading:* The design of XCLAIM allows for both long-term and short-term issuance of CBAs. As such, during temporary periods of high network congestion [44] or transaction fee spikes [60], e.g. in Bitcoin or Ethereum, XCLAIM CBAs can be used to temporarily switch to another blockchain for secure payment processing. Moreover, users can also temporarily leverage this technique in XCLAIM to exploit features or benefits that may be present in the issuing blockchain, but not on the backing blockchain. For example, Bitcoin users may temporarily switch to Bitcoin-backed tokens on Ethereum to avoid long transaction processing times, or to leverage more complex transaction scripts and smart contracts in Ethereum. Once such periods have passed, users may exchange their CBA back to coins on the native blockchain securely, without requiring trust.

*N-Way and Multi-Party Atomic Swaps:* XCLAIM is more efficient than atomic cross-chain swaps (ACCS), both in terms of performance and cost (see Section VIII). In addition, XCLAIM can be leveraged to perform more complex and intricate swap constructions. For example, XCLAIM enables *N-way atomic swaps*: by extending the *Swap* protocol in XCLAIM, users can swap multiple different units of cryptocurrencies for others, e.g. trading units of cryptocurrency  $x$  and  $y$  against units of  $w$  and  $z$  atomically within a single swap. Comparing this to ACCS, *N-way swaps* are impractical, as they would require the creation of  $N$  locking and spending transactions, while monitoring all involved chains for failures.

In addition, XCLAIM also enables *multi-party atomic swaps*. Assume Alice owns coin  $x$  and wants to acquire coin  $y$  owned by Bob. Bob, however, will only trade  $y$  for coin  $z$  owned by Carol. Attempting to resolve this situation with ACCS would require Alice to separately swap with Carol and then with Bob, i.e., this process would not be atomic, resulting in 8 transactions on 3 different chains [4]. However, with XCLAIM, if  $x$ ,  $y$  and  $z$  are CBAs, Alice can construct a non-interactive multi-party swap via the iSC, where a single transaction can change the ownership of all 3 coins in iSC, atomically.

## X. RELATED WORK

The only existing mechanism to perform a trustless cross-chain transfer today is *Atomic Cross-Chain Swaps* (ACCS) based on time and hash locks [4], [5], [34], [72], [110]. Although ACCS enable trustless cryptocurrency exchanges, they are *interactive*, i.e., they rely on all parties being online and monitoring the blockchain throughout the exchange to ensure security. Each swap thereby incurs long waiting periods to prevent fraud through exploiting blockchain reorganizations, which substantially hinders performance and involves synchronizing clocks between independent decentralized ledgers. Moreover, ACCS are vulnerable to packet and transaction memory-pool sniffing, allowing an adversary to exploit blockchain race conditions to steal funds. Finally, ACCS rely on a pre-established out-of-band communication channel between

parties, required to exchange security-critical *revocation transactions* [39], [109]. XCLAIM not only avoids these problems, as discussed throughout the paper, but is also more efficient in terms of execution costs and time.

Existing approaches towards CBA systems require trust in intermediaries. Liquid [53], RSK [84] and PoA Network [25] use permissioned blockchains on the issuing side, where a pre-defined set of validators is trusted with control over assets on the backing chain. As such, these systems resemble CENTRALCLAIM. In contrast, XCLAIM makes no such trust assumptions and, in fact, can be applied to improve security of CBAs issued on or from permissioned blockchains, just like with CENTRALCLAIM. Bentov et al. describe how to tokenize exiting cryptocurrencies via trusted execution environments (TEEs) [39]. TEEs however, are known to be vulnerable to a wide range of side-channel attacks [69], [114], [118] and require trust in the hardware manufacturer.

Other approaches attempting to achieve blockchain interoperability, such as Polkadot [22], [116], Cosmos [83], AION [106] and COMIT [73], to this date, only achieve communication between instances of their own permissioned blockchains, i.e., they support sharding [50] rather than cross-chain communication. XCLAIM can be used to connect these systems to permissionless blockchains such as Bitcoin and Ethereum.

## XI. CONCLUSION

In this paper, we formalized the notion of cryptocurrency-backed assets. We presented XCLAIM, a system for issuing, trading and redeeming cryptocurrency-backed assets between blockchains, without necessitating trust. We provided a detailed analysis of XCLAIM's design, a formal protocol specification and identified requirements for the underlying blockchains. XCLAIM is general in design and supports many existing cryptocurrencies without modification. We implemented XCLAIM (BTC,ETH) to construct Bitcoin-backed tokens on Ethereum and evaluated the performance and execution costs; XCLAIM achieves a significant improvement over atomic cross-chain swaps. Finally, we outlined several novel and interesting applications enabled by XCLAIM.

## XII. ACKNOWLEDGEMENTS

The authors would like to thank Nicholas Stifter, Pedro Moreno-Sanchez and Peter Smith for helpful comments and insightful discussions. This research was sponsored by Blockchain.com, Outlier Ventures, Bridge 1 858561 SESC, Bridge 1 864738 PR4DLT (all FFG), the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI), and the competence center SBA-K1 funded by COMET.

## REFERENCES

- [1] "0xproject whitepaper," [https://0xproject.com/pdfs/0x\\_white\\_paper.pdf](https://0xproject.com/pdfs/0x_white_paper.pdf), accessed: 2018-05-23.
- [2] "Airswap," <https://www.airswap.io/>, accessed: 2018-07-30.
- [3] "Bitcoin Developer Guide: Simplified Payment Verification (SPV)," <https://bitcoin.org/en/developer-guide#simplified-payment-verification-spv>, accessed: 2018-05-16.

- [4] "Bitcoin Wiki: Atomic cross-chain trading," [https://en.bitcoin.it/wiki/Atomic\\_cross\\_chain\\_trading](https://en.bitcoin.it/wiki/Atomic_cross_chain_trading), accessed: 2018-05-16.
- [5] "Bitcoin Wiki: Hashed Time-Lock Contracts," [https://en.bitcoin.it/wiki/Hashed\\_Timelock\\_Contracts](https://en.bitcoin.it/wiki/Hashed_Timelock_Contracts), accessed: 2018-05-16.
- [6] "Btc relay," <https://github.com/ethereum/btcrelay>, accessed 2018-04-17.
- [7] "Cardano sl," <https://github.com/input-output-hk/cardano-sl>, accessed: 2018-07-30.
- [8] "Counterparty," <https://counterparty.io/>, accessed: 2018-05-03.
- [9] "Erc20: Token standard," <https://github.com/ethereum/EIPs/issues/20>, accessed 2018-06-27.
- [10] "Erc223: Token standard," <https://github.com/ethereum/EIPs/issues/223>, accessed 2018-06-27.
- [11] "Erc721: Non-fungible token standard," <https://github.com/namecoin/namecoin>, accessed 2018-06-27.
- [12] "Etherdelta," <https://etherdelta.com/>, accessed: 2018-07-30.
- [13] "Ethereum Classic," <https://github.com/ethereumproject>, accessed: 2018-05-23.
- [14] "Etherscan - ropsten testnet explorer," <https://ropsten.etherscan.io/>, accessed: 2018-05-23.
- [15] "Index whitepaper," <https://index.market/static/IDEX-Whitepaper-V0.7.5.pdf>, accessed: 2018-05-23.
- [16] "Komodo barterdex," <https://github.com/KomodoPlatform/KomodoPlatform/wiki/BarterDEX-%E2%80%93-93-A-Practical-Native-DEX>, accessed: 2018-11-28.
- [17] "Kyber network whitepaper," <https://home.kyber.network/assets/KyberNetworkWhitepaper.pdf>, accessed: 2018-05-23.
- [18] "Light client protocol," <https://github.com/ethereum/wiki/wiki/Light-client-protocol>, accessed: 2018-11-20.
- [19] "Monero reference implementation," <https://github.com/monero-project/monero>, accessed: 2018-07-30.
- [20] "Neo whitepaper," <http://docs.neo.org/en-us/>, accessed: 2018-07-30.
- [21] "Omnilayer specification," <https://github.com/OmniLayer/spec>, accessed: 2018-05-03.
- [22] "Parity-Bridge," <https://github.com/paritytech/parity-bridge>, accessed: 2018-05-21.
- [23] "The parity light protocol - wiki," [https://wiki.parity.io/The-Parity-Light-Protocol-\(PIP\)](https://wiki.parity.io/The-Parity-Light-Protocol-(PIP)), accessed: 2018-10-30.
- [24] "Peace relay," <https://github.com/loiluu/peacereley>, accessed 2018-04-17.
- [25] "Poa bridge," <https://github.com/poanetwork/poa-bridge>, accessed: 2018-05-23.
- [26] "Solidity programming language," <https://github.com/ethereum/solidity>, accessed: 2018-07-30.
- [27] "Wary of crypto, uk government blocks royal mint's digital gold," *Thomson Reuters*, accessed: 2018-10-30. [Online]. Available: <https://uk.reuters.com/article/us-gold-cryptocurrency-royal-mint/wary-of-crypto-uk-government-blocks-royal-mints-digital-gold-idUKKCN1MZ187>
- [28] "The witness algorithm: Privacy protection in a fully transparent system," <https://gist.github.com/gavofyork/deef3b727f691b381dc>, accessed: 2018-07-30.
- [29] "Tether: Fiat currencies on the bitcoin blockchain," <https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf>, 2016.
- [30] "Decred cross-chain atomic swapping," <https://github.com/decred/atomicswap>, 2017, accessed: 2018-05-16.
- [31] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," *arXiv preprint arXiv:1808.00528*, 2018.
- [32] E. Androulaki, C. Cachin, A. De Caro, and E. Kokoris-Kogias, "Channels: Horizontal scaling and confidentiality on permissioned blockchains," in *European Symposium on Research in Computer Security*. Springer, 2018, pp. 111–131.
- [33] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts," <http://eprint.iacr.org/2016/1007>, Oct 2016, accessed: 2016-11-08. [Online]. Available: <https://eprint.iacr.org/2016/1007.pdf>
- [34] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," <https://blockstream.com/sidechains.pdf>, 2014, accessed: 2016-07-05. [Online]. Available: <https://blockstream.com/sidechains.pdf>
- [35] A. Back *et al.*, "Hashcash-a denial of service counter-measure," <http://www.hashcash.org/papers/hashcash.pdf>, 2002, accessed: 2016-03-09. [Online]. Available: <http://www.hashcash.org/papers/hashcash.pdf>
- [36] C. Baldwin, "Bitcoin worth \$72 million stolen from bitfinex exchange in hong kong," *Reuters*, accessed: 2018-05-23.
- [37] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptol. ePrint Arch., Tech. Rep.*, vol. 46, p. 2018, 2018.
- [38] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 459–474. [Online]. Available: <http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf>
- [39] I. Bentov, Y. Ji, F. Zhang, Y. Li, X. Zhao, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," Cryptology ePrint Archive, Report 2017/1153, 2017, accessed:2017-12-04. [Online]. Available: <https://eprint.iacr.org/2017/1153.pdf>
- [40] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 2012, pp. 326–349.
- [41] "Pay-to-Pubkey Hash," [https://en.bitcoinwiki.org/wiki/Pay-to-Pubkey\\_Hash](https://en.bitcoinwiki.org/wiki/Pay-to-Pubkey_Hash), bitcoin.it, accessed: 2018-11-28.
- [42] "Script," <https://en.bitcoin.it/wiki/Script>, bitcoin.it, accessed: 2018-11-28.
- [43] S. Blake-Wilson and M. Qu, "Standards for efficient cryptography (sec) 2: Recommended elliptic curve domain parameters," *Certicom Research, Oct*, 1999.
- [44] "Bitcoin transaction fees," <https://www.blockchain.com/en/charts/transaction-fees?timespan=all>, Blockchain.com, accessed: 2018-11-28.
- [45] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," [http://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman\\_Ethan\\_201606\\_MAsc.pdf](http://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf), Jun 2016, accessed: 2017-02-06.
- [46] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Bulletproofs: Short Proofs for Confidential Transactions and More*. IEEE, 2018.
- [47] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," <https://github.com/ethereum/wiki/wiki/White-Paper>, 2014, accessed: 2016-08-22. [Online]. Available: <https://github.com/ethereum/wiki/wiki/White-Paper>
- [48] Christian Decker and Roger Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Symposium on Self-Stabilizing Systems*. Springer, 2015, pp. 3–18.
- [49] "Coinmarketcap," <http://coinmarketcap.com/>, CoinMarketCap, accessed 2016-09-10.
- [50] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.
- [51] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. Gün, "On scaling decentralized blockchains," in *3rd Workshop on Bitcoin and Blockchain Research, Financial Cryptography 16*, 2016. [Online]. Available: <http://www.tik.ee.ethz.ch/file/74bc987e6ab4a8478c04950616612f69/main.pdf>
- [52] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10. [Online]. Available: <http://diyhl.us/~bryan/papers2/bitcoin/Information%20propagation%20in%20the%20Bitcoin%20network.pdf>
- [53] J. Dille, A. Poelstra, J. Wilkins, M. Piekarska, B. Gorlick, and M. Friedenbach, "Strong federations: An interoperable blockchain solution to centralized third party risks," *arXiv preprint arXiv:1612.05491*, 2016.
- [54] J. R. Douceur, "The sybil attack," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 251–260. [Online]. Available: <http://www.cs.cornell.edu/people/egs/cs6460-spring10/sybil.pdf>
- [55] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*. Springer, 1992, pp. 139–147. [Online]. Available: <https://web.cs.dal.ca/~abrodsky/7301/readings/DwNa93.pdf>
- [56] S. Dziembowski, L. Eeckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *To appear in the Proceedings of the IEEE ), IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 2019.

- [57] "Predicting bitcoin fees for transactions," <https://bitcoinfees.earn.com/>, earn.com, accessed: 2018-11-28.
- [58] "Ethereum fee estimates," <https://ethgasstation.info/>, ETH Gas Station, accessed: 2018-11-28.
- [59] Ethereum community, "Ethereum: A secure decentralised generalised transaction ledger," <https://github.com/ethereum/yellowpaper>, accessed: 2016-03-30.
- [60] "Ethereum BlockSize History," <https://etherscan.io/chart/blocksize>, Etherscan.io, accessed: 2018-11-28.
- [61] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th USENIX Security Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, Mar 2016. [Online]. Available: <http://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf>
- [62] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454. [Online]. Available: <http://arxiv.org/pdf/1311.0243>
- [63] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol with chains of variable difficulty," <http://eprint.iacr.org/2016/1048.pdf>, 2016, accessed: 2017-02-06. [Online]. Available: <http://eprint.iacr.org/2016/1048.pdf>
- [64] Gavin Wood, "Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dcd - 2017-08-07)," 2017, accessed: 2018-01-03. [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>
- [65] P. Gazi, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *To appear in the Proceedings of the IEEE Symposium on Security & Privacy*. IEEE Computer Society Press, 2019.
- [66] P. Gazi, A. Kiayias, and A. Russell, "Stake-bleeding attacks on proof-of-stake blockchains," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 85–92.
- [67] A. Gervais, G. Karame, S. Capkun, and V. Capkun, "Is bitcoin a decentralized currency?" in *IEEE Security & Privacy*, vol. 12, no. 3, 2014, pp. 54–60. [Online]. Available: <https://eprint.iacr.org/2013/829.pdf>
- [68] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 3–16.
- [69] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel sgx," in *Proceedings of the 10th European Workshop on Systems Security*. ACM, 2017, p. 2.
- [70] J. M. Griffin and A. Shams, "Is bitcoin really un-tethered?" 2018. [Online]. Available: <https://ssrn.com/abstract=3195066>
- [71] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 129–144. [Online]. Available: <https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-heilman.pdf>
- [72] M. Herlihy, "Atomic cross-chain swaps," arXiv:1801.09515, 2018, accessed:2018-01-31. [Online]. Available: <https://arxiv.org/pdf/1801.09515.pdf>
- [73] J. Hosp, T. Hoenisch, and P. Kittiwongsunthorn, "Comit network," <https://www.comit.network/doc/COMIT%20white%20paper%20v1.0.2.pdf>, 2017, accessed: 2018-07-30.
- [74] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proceedings of the 27th USENIX Conference on Security Symposium*. USENIX Association, 2018, pp. 1353–1370.
- [75] R. Khalil and A. Gervais, "Nocust—a non-custodial 2 nd-layer financial intermediary," 2018, <https://eprint.iacr.org/2018/642>.
- [76] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," Cryptology ePrint Archive, Report 2017/963, 2017, accessed:2017-10-03. [Online]. Available: <https://eprint.iacr.org/2017/963.pdf>
- [77] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," Cryptology ePrint Archive, Report 2016/889, 2016, <https://eprint.iacr.org/2016/889>.
- [78] A. Kiayias and D. Zindros, "Proof-of-work sidechains," Cryptology ePrint Archive, Report 2018/1048, 2018, <https://eprint.iacr.org/2018/1048.pdf>.
- [79] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," <https://peercoin.net/assets/paper/peercoin-paper.pdf>, Aug 2012, accessed: 2017-01-07. [Online]. Available: <https://peercoin.net/assets/paper/peercoin-paper.pdf>
- [80] N. Kobitz, "Cm-curves with good cryptographic properties," in *Annual international cryptography conference*. Springer, 1991, pp. 279–287.
- [81] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.
- [82] D. Kraft, "Difficulty control for blockchain-based consensus systems," in *Peer-to-Peer Networking and Applications*, vol. 9, no. 2, 2016, pp. 397–413. [Online]. Available: <https://www.domob.eu/research/DifficultyControl.pdf>
- [83] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," <https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md>, 2015.
- [84] S. D. Lerner, "Rootstock: Bitcoin powered smart contracts," [https://docs.rsk.co/RSK\\_White\\_Paper-Overview.pdf](https://docs.rsk.co/RSK_White_Paper-Overview.pdf), 2015.
- [85] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547. [Online]. Available: [http://www.cs.huji.ac.il/~avivz/pubs/15/inclusive\\_btc\\_full.pdf](http://www.cs.huji.ac.il/~avivz/pubs/15/inclusive_btc_full.pdf)
- [86] T. C. Lin, "The new market manipulation," *Emory LJ*, vol. 66, p. 1253, 2016.
- [87] J. Lind, I. Eyal, F. Kelbert, O. Naor, P. Pietzuch, and E. G. Sirer, "Teechain: Scalable blockchain payments using trusted execution environments," *arXiv preprint arXiv:1707.05454*, 2017.
- [88] Litecoin community, "Litecoin reference implementation," [github.com/litecoin-project/litecoin](https://github.com/litecoin-project/litecoin), accessed: 2017-06-30.
- [89] L. Luu, B. Bueuz, and M. Zamani, "Flyclient super light client for cryptocurrencies," accessed 2018-04-17. [Online]. Available: <https://stanford2017.scalingbitcoin.org/files/Day1/flyclientscalingbitcoin.pptx.pdf>
- [90] P. McCorry, E. Heilman, and A. Miller, "Atomically trading with roger: Gambling on the success of a hardfork," in *CBT'17: Proceedings of the International Workshop on Cryptocurrencies and Blockchain Technology*, Sep 2017. [Online]. Available: <http://homepages.cs.ncl.ac.uk/patrick.mc-corry/atomically-trading-roger.pdf>
- [91] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1987, pp. 369–378.
- [92] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," arXiv:1702.05812, 2017, accessed: 2018-30-22. [Online]. Available: <https://arxiv.org/pdf/1702.05812.pdf>
- [93] Moore, Tyler and Christin, Nicolas, "Beware the middleman: Empirical analysis of bitcoin-exchange risk," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 25–33.
- [94] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," <https://bitcoin.org/bitcoin.pdf>, Dec 2008, accessed: 2015-07-01. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [95] Namecoin community, "Namecoin reference implementation," <https://github.com/namecoin/namecoin>, accessed 2017-06-30.
- [96] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.
- [97] K. Okupski, "Bitcoin protocol specification," <https://github.com/minium/Bitcoin-Spec>, accessed: 2014-10-14.
- [98] J. Pagliery, "Another bitcoin exchange goes down," *CNN Tech*, accessed: 2018-05-23.
- [99] R. Pass, L. Seeman, and a. shelat, "Analysis of the blockchain protocol in asynchronous networks," <http://eprint.iacr.org/2016/454.pdf>, 2016, accessed: 2016-08-01. [Online]. Available: <http://eprint.iacr.org/2016/454.pdf>
- [100] H. Ritzdorf, K. Wüst, A. Gervais, G. Felley *et al.*, "Tls-n: Non-repudiation over tls enabling ubiquitous content signing," in *Network and Distributed System Security Symposium (NDSS)*, 2018.
- [101] M. Rosenfeld, "Overview of colored coins," <https://bitcoil.co.il/BitcoinX.pdf>, 2012, accessed: 2016-03-09. [Online]. Available: <https://bitcoil.co.il/BitcoinX.pdf>
- [102] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 515–532.
- [103] I. Sergey, A. Kumar, and A. Hobor, "Scilla: a smart contract intermediate-level language," arXiv:1801.00687, 2018, accessed:2018-01-08. [Online]. Available: <https://arxiv.org/pdf/1801.00687.pdf>



- [104] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, “Spectre: A fast and scalable cryptocurrency protocol,” Cryptology ePrint Archive, Report 2016/1159, 2016, accessed: 2017-02-20. [Online]. Available: <http://eprint.iacr.org/2016/1159.pdf>
- [105] Y. Sompolinsky and A. Zohar, “Bitcoin’s security model revisited,” <http://arxiv.org/pdf/1605.09193>, 2016, accessed: 2016-07-04. [Online]. Available: <http://arxiv.org/pdf/1605.09193.pdf>
- [106] M. Spoke and Nuco Engineering Team, “Aion: The third-generation blockchain network,” [https://aion.network/media/2018/03/aion.network\\_technical-introduction\\_en.pdf](https://aion.network/media/2018/03/aion.network_technical-introduction_en.pdf), accessed 2018-04-17.
- [107] J. Teutsch and T. Estsblishment, “On decentralized oracles for data availability,” 2017.
- [108] J. Teutsch and C. Reitwiebner, “A scalable verification solution for blockchains,” <https://truebit.io/>, March 2017, accessed:2017-10-06. [Online]. Available: <https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf>
- [109] S. Thomas and E. Schwartz, “A protocol for interledger payments,” URL <https://interledger.org/interledger.pdf>, 2015.
- [110] TierNolan, “Atomic swaps using cur and choose,” <https://bitcointalk.org/index.php?topic=1364951>, 2016, accessed: 2018-05-16.
- [111] V. Buterin, “Chain interoperability,” <https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cft/588680eccd0f68de303349b1/1485209617040/Chain+Interoperability.pdf>, 2016, accessed: 2017-03-25.
- [112] J. Von Neumann and O. Morgenstern, “Theory of games and economic behavior,” 1944.
- [113] M. Vukolić, “The quest for scalable blockchain fabric: Proof-of-work vs. bft replication,” in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125. [Online]. Available: [http://vukolic.com/iNetSec\\_2015.pdf](http://vukolic.com/iNetSec_2015.pdf)
- [114] N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza, “Asyncshock: Exploiting synchronisation bugs in intel sgx enclaves,” in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 440–457.
- [115] “Bullion and bandits: The improbable rise and fall of e-gold,” <https://www.wired.com/2009/06/e-gold/>, Wired, accessed: 2018-11-28.
- [116] G. Wood, “Polkadot: Vision for a heterogeneous multi-chain framework,” *White Paper*, 2015.
- [117] K. Wu, S. Wheatley, and D. Sornette, “Classification of cryptocurrency coins and tokens by the dynamics of their market capitalizations,” *Royal Society Open Science*, vol. 5, no. 9, p. 180381, 2018.
- [118] Y. Xu, W. Cui, and M. Peinado, “Controlled-channel attacks: Deterministic side channels for untrusted operating systems,” in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 640–656.
- [119] Yonatan Sompolinsky and Aviv Zohar, “Phantom: A scalable blockdag protocol,” Cryptology ePrint Archive, Report 2018/104, 2018, accessed:2018-01-31. [Online]. Available: <https://eprint.iacr.org/2018/104.pdf>
- [120] A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W. J. Knottebelt, “(Short Paper) A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice,” in *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security 18 (FC)*. Springer, 2018.
- [121] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, “Town crier: An authenticated data feed for smart contracts,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 270–282. [Online]. Available: <https://eprint.iacr.org/2016/168.pdf>

## APPENDIX A SYMBOLS AND NOTATIONS

Table III provides an overview of symbols and variables used in this paper.

TABLE III  
Summary of used symbols and notations.

Symbol	Description
$B$	Backing blockchain
$b$	Cryptocurrency unit underlying the backing blockchain $B$
$b_{lock}$	Units of $b$ locked by a <i>requester</i> with a <i>vault</i> during <i>Issue</i>
$I$	Issuing blockchain
$i$	Cryptocurrency unit underlying the issuing blockchain $I$
$i_{col}$	Units of $i$ locked by a user with the iSC
$i(b)$	Unit of a CBA on $I$ backed by units of $b$ on $B$
$k^X$	Security parameter of blockchain $X$ . Defines how many confirmations are necessary for secure transaction inclusion
$\tau^X$	Block generation rate of blockchain $X$
$\Delta^X$	Maximum delay from transaction broadcast to secure inclusion in blockchain $X$
$\bar{r}_\tau$	Upper bound for deviations of expected and observed ratio between block generation rates of $B$ and $I$ .
$\Delta_{tx}^X$	Maximum delay from transaction broadcast to receipt by honest consensus participants in $X$
$(pk_u^X, sk_u^X)$	Public / private key pair of user $u$ on blockchain $X$
$\sigma_u^X$	Digital signature of user $u$ on chain $X$ , i.e., via $sk_u^X$
$T_{id}^X$	Transaction created on blockchain $X$ with identifier $id$
operation( $in$ ) $\rightarrow$ $out$	Operation taking $in$ as input an outputting $out$
operation	Short for operation with default inputs and outputs
$cond$	Conditions used to lock coins, e.g. presenting a user’s digital signature $\sigma_{user}$
$\Delta_{id}$	Maximum delay for process $id$
$\bar{\Delta}_{id}$	Pre-defined upper bound for $\Delta_{id}$
$r_{col}$	Ideal (parameterized) collateralization rate of <i>vaults</i>
$r_{col}^*$	Observed collateralization rate
$r_{col}^{iq}$	Collateralization threshold for automatic liquidation

APPENDIX B  
PROOF-OF-WORK CHAIN RELAY MODEL

As discussed in Section V-B, a *chain relay* is a program deployed on a blockchain, capable of reading and verifying the state of some other blockchain, comparable to the notion of SPV-Clients [3]. That is, a chain relay stores and maintains block headers and allows to verify transaction inclusion proofs.

In the following, we provide a formal model for the requirements of a program  $\pi$  to represent a functioning chain relay for a proof-of-work blockchain  $\mathcal{C}$ <sup>9</sup>.

*Notation:* We denote  $H(x)$  as the output of a cryptographic hash function over some input  $x$ . Further,  $\mathcal{B}_i$  shall denote the block header of the block at position  $i$  in the blockchain, represented by the tuple  $\langle S_{i-1}, \mathcal{T}_i, M_i, N_i, t_i \rangle$ , where

- $S_{i-1}$  is the reference to the PoW hash (i.e., solution) of the predecessor of block  $i$ ,
- $\mathcal{T}_i$  is the PoW difficulty target at block  $i$  required by consensus rules,
- $M_i$  is the root hash of the Merkle tree of the hashes of all transactions  $(T_0, T_1, \dots, T_n)$  included in  $i$ ,
- $N_i$  is the random nonce used to generate the PoW solution hash  $S_i = H(\mathcal{B}_i)$ ,
- and  $t_i$  is the timestamp specifying when block  $i$  was generated.

We refer to the header of the first ( $i = 0$ ) or so called *genesis* block as  $\mathcal{G}$ . We assume protocol rules of  $\mathcal{C}$  require that the PoW difficulty target  $T$  is adjusted at some rate  $r$  based on the relation of the time between each two adjustments and some pre-defined number representing the desired block interval. Note, while potentially useful for more extensive block validity checks, for simplification we ignore other information usually included in the block headers. Furthermore, as it is not of greater relevance to our model, we assume the same cryptographically secure hash function  $H()$  is used to calculate both the hashes of block headers and transactions.

*Model:* We require a chain relay program  $\pi$  to support the following functionalities with regards to the state of a Proof-of-Work blockchain  $\mathcal{C}$ :

**Functionality 1 (Difficulty Adjustment).**

Program  $\pi$  has knowledge of the difficulty adjustment rate  $r$  and, given  $\mathcal{B}_i$  and  $\mathcal{B}_{i+r}$ , where  $i \pmod{r} = 0$ , outputs the new difficulty target  $\mathcal{T}_{i+r}$  according to consensus rules of  $\mathcal{C}$ .

**Functionality 2 (Block Validation).**

Program  $\pi$  has a function `checkBlock` which takes as input a block header  $\mathcal{B}_i$  and a PoW solution  $S_i$ , and returns `True` if and only if  $\mathcal{B}_i$  is the pre-image of  $S_i$ ,  $\mathcal{T}_i$  is the difficulty target required at block  $i$  and it holds that  $S_i < \mathcal{T}_i$ .

**Functionality 3 (Chain Validation).**

Program  $\pi$  has a function `checkChain` which takes as input the genesis block  $\mathcal{G}$ , a list of consecutive block headers

$(\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_n)$  and the list of corresponding PoW solutions  $(S_1, S_2, \dots, S_n)$ , and returns `True` if and only if for each tuple  $(\mathcal{B}_i, S_i)$ ,  $i \leq n$ , it holds that `checkBlock` $(\mathcal{B}_i, S_i) = \text{True}$  and for each two consecutive block headers  $\mathcal{B}_i$  and  $\mathcal{B}_{i+1}$  it holds that  $S_i \in \mathcal{B}_{i+1}$ , i.e.,  $\mathcal{B}_i$  is the predecessor of  $\mathcal{B}_{i+1}$ .

**Definition 1 (Valid Chain).**

We define the tuple

$\langle \mathcal{G}, (\mathcal{B}_1, \dots, \mathcal{B}_n), (S_1, \dots, S_n) \rangle$  as a valid chain if `checkChain` outputs `True` given this tuple as input.

**Functionality 4 (Main Chain Detection).**

Program  $\pi$  provides a function denoted `mainChain` which takes as input two valid chains

$\langle \mathcal{G}, (\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{B}_{i+1}, \dots, \mathcal{B}_n), (S_1, \dots, S_i, S_{i+1}, \dots, S_n) \rangle$  and  $\langle \mathcal{G}, (\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{B}'_{i+1}, \dots, \mathcal{B}'_m), (S_1, \dots, S_i, S'_{i+1}, \dots, S'_m) \rangle$  where  $n \neq m$  and for every  $j \geq i$  it holds that  $\mathcal{B}_j \neq \mathcal{B}'_j$ ,  $S_j \neq S'_j$ , and outputs the main chain according to the consensus rules of  $\mathcal{C}$ , e.g., the longest chain in the case of Nakamoto consensus.

**Functionality 5 (Transaction Inclusion Verification).**

Program  $\pi$  provides a function `checkTransaction` which, if given a valid chain  $\langle \mathcal{G}, (\mathcal{B}_1, \dots, \mathcal{B}_n), (S_1, \dots, S_n) \rangle$ , a block header  $\mathcal{B}_i$ , a transaction  $T$  and a Merkle tree path  $p$ , outputs `True` if and only if  $H(T)$  is contained in the Merkle tree with root  $M_i \in \mathcal{B}_i$  at the position defined by  $p$ ,  $\mathcal{B}_i \in (\mathcal{B}_1, \dots, \mathcal{B}_n) | i \leq n$  and the provided chain is the main chain of  $\mathcal{C}$ .

**Definition 2 (Chain Relay).**

A program  $\pi$  is a chain relay of a Proof-of-Work blockchain  $\mathcal{C}$ , if it satisfies Requirements 1-5 with regards to  $\mathcal{C}$ .

*Practicability of Chain relays*

Existing chain relays require to store all block headers of the verified blockchain, which can incur substantial cost. Furthermore, the verification of PoW requires to implement the verification procedure for the respective hash functions. However, if native support for the necessary cryptographic primitives is not provided in the blockchain the relay is deployed on, verification may be infeasible. For example, Ethereum currently only allows feasible chain relays for a subset of existing blockchains, which: (i) use the same PoW algorithm, e.g. Ethereum Classic [24], or (ii) SHA-256 available as pre-compiled contract, such as Bitcoin [6] and Namecoin [95].

The main performance and cost challenges of chain relays as such lie in (i) efficient PoW / hash pre-image verification and (ii) reduction of necessary input data (block headers). The former can in theory be achieved by using non-interactive zero-knowledge proof (ZKP) systems such as SNARKs [40], STARKs [37] or Bulletproofs [46] for hash pre-image verification (assuming the corresponding arithmetic circuits can be efficiently constructed). Alternatively, the verification of PoW can be outsourced to users [108], with disputes handled via interactive games or ZKPs in smart contracts. However, existing schemes have been shown to currently suffer security challenges [74].

<sup>9</sup>This model is also applicable to DAG-based systems [85], [104], [119]. A DAG can e.g. be reduced to a chain, by allowing each vertex to reference a only single predecessor.

New proposals for efficient SPV clients, leveraging concepts such as NiPoPoW [76], [78] or FlyClient [89], can significantly reduce verification costs by reducing the number of stored block headers. Both, however, require additional data to be included in the block headers of the to-be-verified blockchain. While this can be achieved as a velvet fork [76], [120], the performance improvement and security of these schemes rely on the availability of this data.

#### APPENDIX C

#### XCLAIM PROTOCOLS: SEQUENTIAL OVERVIEW

In this section we present sequence diagrams for the *Issue*, *Swap*, *Redeem* and *Replace* protocols in XCLAIM. We omit *Transfer* due to its simplicity. We use the notation defined in Section VI.

Fig. 4. Sequential overview of the *Issue* protocol

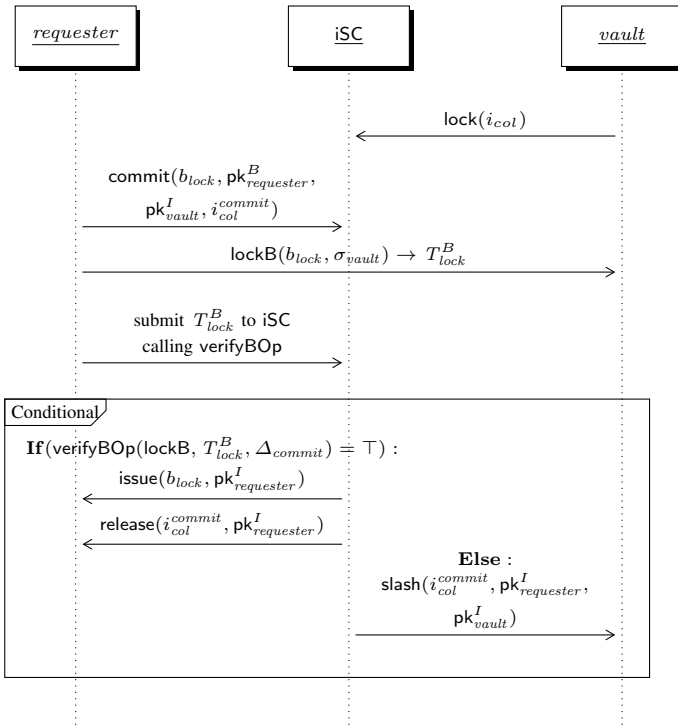


Fig. 5. Sequential overview of the *Swap* protocol

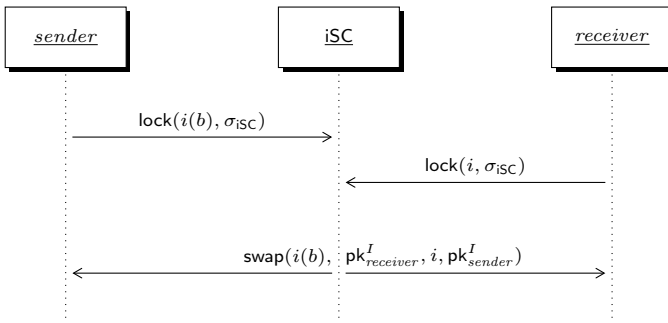


Fig. 6. Sequential overview of the *Redeem* protocol

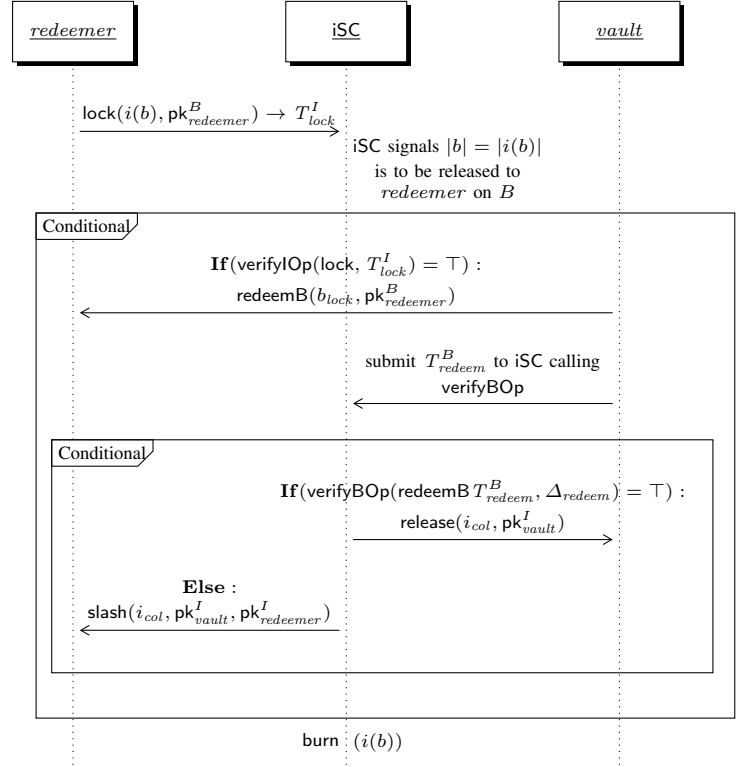


Fig. 7. Sequential overview of the *Replace* protocol

