Privacy-Preserving Asset Swaps via zkBridge

Sriram Sridhar, Wenhao Wang, Yinuo Zhang, Fan Zhang

May 1, 2023

Abstract

This paper proposes a privacy-preserving cross-chain asset transfer protocol, built on top of zkBridge [XZC+22]. We report on a Proof of Concept implementation of the first Zcash-Ethereum bridge that perfectly hides the user identity among all addresses on Zcash (i.e., the anonymity set is the identities on the entire Zcash blockchain).

1 Introduction

Cross-chain bridges received a lot of attention recently due to growing interoperability needs in DeFi and major security flaws in existing protocols that had caused a significant financial loss for their users. Recently works such as zkBridge [XZC+22] improve the safety of cross-chain communication by replacing trusted committees (a single-point failure) with zero-knowledge proofs (ZKP).

While improving safety is crucial, the privacy implication of bridges received much less attention. Most deployed systems do not provide any privacy guarantees whatsoever, which allows an observer to *link* bridged assets to the original ones. Such linkage could affect the fungibility of assets (since minted coins inherit their history from the source chain) and even erode user privacy (e.g., deanonymization attacks on one chain could now impact other chains through bridges).

There are numerous private assets like Dash [Das], Zcash [Zca], and Monero [Mon] that focus on user privacy and use zero-knowledge proofs and other cryptographic techniques to hide transaction details. This paper explores the following question: *Can we securely bridge private assets to a public blockchain (such as Ethereum) with strong privacy guarantees?*

Defining privacy. (Section 2.4) The first contribution is a definition of privacy for asset swaps. Informally, suppose Alice performs a cross-chain swap from chain A to chain B, it is desirable to guarantee that no one on chain A should know that Alice's tokens are being moved to another chain, and no one on chain B must know the origin of the swap on chain A. In addition, an observer that can observe both chains should also learn no information that could link the origin and destination of the moved tokens. Depending on the innate privacy guarantees provided by the chain, we could require stronger notions of privacy as well.

Privacy-preserving cross-chain asset transfers. A common use case of bridges is lock-and-mint cross-chain token transfers (formalized as Single Party Asset Swap in section 2.3) that allow a user to move assets from chain A to chain B. In a lock-and-mint scheme, a user first sends her asset to a *locking* contract on chain A. A bridge, such as zkBridge, then is used to inform the *minting* contract on chain B that v original tokens have been locked, and thus v wrapped tokens can be created. To reverse the process and unlock the original assets, the user first burns v wrapped tokens on chain B, and the bridge is used to inform the locking contract to release v original tokens.

Designing a lock-and-mint protocol between private assets (we consider Zcash specifically) and a public blockchain(e.g., Ethereum) faces three challenges. First, while locking can be implemented by having users send their Zcash to an address that is not controlled by the user, it is unclear how to unlock it — Zcash does not support smart contracts and thus it cannot use zkBridge to verify that wrapped tokens have been burnt. This limitation mandates the reliance on an off-chain party. Second, in the unlocking step, it is unclear how to hide user identity from the said party, since it needs to send the locked Zcash to some user address after

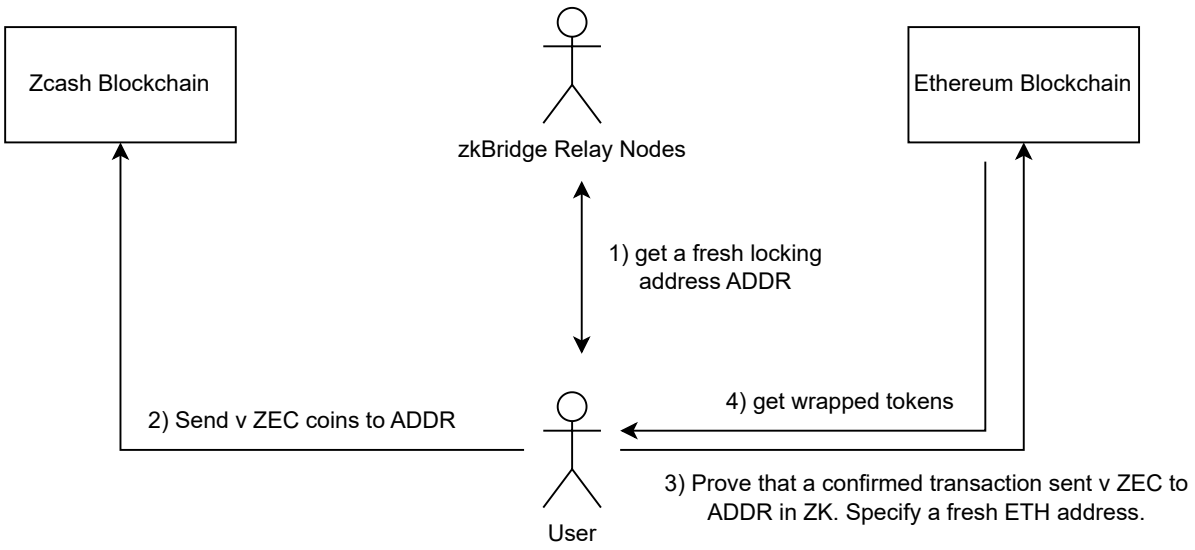

Figure 1: The workflow of our privacy-preserving bridge between Zcash and Ethereum.

all, which necessarily leaks the user address. Third, although introducing a third party seems necessary, we still would like to guarantee that she cannot steal user funds or mint tokens out of thin air.

In our construction, we propose to have the relay nodes in zkBridge act as the said off-chain parties. Since zkBridge already requires a relay network, there is minimal overhead to the system.
The protocol is initiated by a user who wishes to transfer v Zcash to Ethereum. Upon receiving the user request, relay nodes first generate a locking address ADDR that satisfies the following requirements: 1) relay nodes cannot spend money sent to ADDR, 2) each user gets a unique and fresh ADDR for every asset transfer, 3) ADDR does not leak user identity, and 4) the user cannot spend coins from ADDR. The communication can be implemented entirely on Ethereum via smart contracts, since Ethereum is a public chain and the relay nodes can monitor smart contract calls. There is no need for users to ever interact directly with relay nodes, which prevents side-channel leakage of user information (like IP address/location).
Also notice here that the relay nodes are indeed necessary, as there is no way to store private state on Ethereum.

Having generated the locking address ADDR, the user sends v Zcash to ADDR, locking it. She then sends a SNARK proof to the minting contract proving that 1) there exists a Zcash transaction T that sends v coins to ADDR (which leaking any other information about T) and 2) that T has been confirmed in the Zcash blockchain (this step requires block headers provided by zkBridge and a Merkle inclusion proof). To reverse the process, the relay nodes and the user work together to enable the user to retrieve the coins she originally sent to ADDR.

Our construction is presented in Section 4 where ADDR is a (transparent) multisig address supported by ZCash. We are also considering an alternative where ADDR is interactively created by relay nodes and the user, which is work-in-progress for now.

Implementation and demo. To demonstrate the practicality of the protocol, we implemented the SNARK circuit as specified above, as well as the smart contracts that implement the verify, mint and burn functionalities. Our implementation lays the foundation for the first Zcash-Ethereum bridge that perfectly hides the user identity among all addresses on Zcash (i.e., the anonymity set is the identities on the entire Zcash blockchain).

1.1 Contributions

In this paper, we study privacy in cross-chain swaps as defined above. The contributions are as follows:

- We define the notion of privacy for asset swaps as well as the adversarial models that they handle. These are abstracted out to form a useful framework for future protocols (Sec. 2).
- We propose a privacy-preserving cross-chain asset transfer protocol that satisfies the above privacy notion. This protocol will be specific to ZCash being the sender chain (though our approach can be generalized to other private coins such as Monero), whereas only the receiver chain needs to support smart contracts. In our implementation, we use Ethereum as the receiver chain.

1.2 Related work

Privacy for the cross-chain swaps has also been considered in [DH20], where they define multiple privacy notions: *anonymity*, *confidentiality* and *indistinguishability* for asset swap protocols. We also use similar notions in our setting and show that we satisfy these as well.

zkBridge [XZC+22] is a trustless cross-chain bridge that allows two blockchains to securely and efficiently transfer block headers from one chain to another (using a relay network of nodes that are only trusted for liveness, not correctness), with efficient on-chain verification. However, as described in the paper, zkBridge does not provide any privacy to the sender or receiver, as any observer can link the two users across chains, which is undesirable.

We consider the single-party version of the swap, where one party wishes to privately move tokens to another chain without using a counterparty that can potentially break privacy. This is the case in all prior works, where a counterparty needs to exist and agree to swap, and can also break privacy of the other party. We seek to avoid both these issues and only require a smart contract on the receiver chain to facilitate the transfer by verifying certain proofs, in addition to the relay network provided by the zkBridge construction (which also requires a smart contract on the receiver side). These relay network nodes will instead be used to play the role of the other party.

2 Preliminaries

In the construction, we use certain cryptographic objects which we define here in a black-box fashion:

- **Zero-Knowledge proofs**: A Zero-Knowledge argument system for a NP relation $\mathcal{R}$ is a protocol where a Prover convinces a Verifier that there exists a witness w such that $(x, w) \in \mathcal{R}$ without revealing any information about w other than the above fact.
- **Verifiable Random Functions (VRF)**: A VRF is a function parametrised by a (sk, pk) pair, such that on any input x, $VRF_{sk}(x)$ returns a value that is uniquely determined by sk and x, but is indistinguishable from random to anyone without the secret key. For verifiability, there also exists a mechanism to check that the output indeed corresponds to x, without having to know sk.
- **Multisignature scheme**: A multisignature scheme is a digital signature scheme with the following properties. There is a verification key (public key) pk as well as a group of secret signing keys $\{sk_i\}_{i\in[n]}$ which is distributed among n parties. For some message m, it is required that only when each party i signs this message m with its signing key sk_i then one can produce a valid aggregated multisignature on the same message m which can be verified under the public key pk. There are many known constructions based on BLS signatures [BDN18].
- **Multisig Wallet**: A multisignature wallet is an identity on a blockchain which is managed by multiple mutually untrusted parties. In this work we utilize the following property of a multisignature wallet: Let w be a multisignature wallet which is managed collectively by n parties. Let pk be some public key corresponding to a multisignature scheme and $\{sk_i\}_{i\in[n]}$ be these n parties secret signing keys. Then in order to spend the coins contained in such a wallet, it is required that all n parties sign the note using their respective secret signing keys. In some cases we also consider a threshold variant of multisig wallet, where there exists some threshold t such that if $\geq t$ out of n parties sign the note, then the transaction will become valid.

2.1 Zcash

Zcash is a variant of Bitcoin system which additionally ensures strong privacy of each transaction. This is done by using zero-knowledge proof systems which enables revealing no additional secret information from each transaction except for its correctness. We note that Zcash does not support smart contracts.

2.2 zkBridge relay network

zkBridge [XZC+22] is a modular system consisting of a core bridge on top of which applications are built. This functionality is provided by a relay network of nodes that relay block headers of the first chain with proofs of correct updation to an updater contract on the second chain that verifies the proofs and updates its list of block headers.
The relay network is untrusted, except for liveness, so that the transfer does not stall. Applications can now query the updater contract to obtain block headers for their applications.

We note that this relay network only requires the second chain to have scripting/smart contract functionality, which will also be the only requirement for our protocol.

2.3 Asset Swapping Protocol

We define a single-party asset swapping protocol as follows:

Consider two different chains A and B. A single-party asset swap between chain A and B is a protocol where a user desires to swap some tokens from chain A to tokens on chain B.

Single party asset swap (SPAS) For a user u that has identities on two chains A and B, we define a single party asset swap, denoted by $SPAS(u, x)$, as a protocol where the user u wishes to move x tokens from chain A to chain B (for simplicity, we assume that the token conversion rate is $1 : 1$, but this is not necessary for either the protocol or our constructions). $SPAS(u, x)$ can be characterised as two transactions

$$SPAS(u, x) = \langle tx_{burn}(u, x), tx_{mint}(u, x) \rangle$$

where $tx_{burn}(u, x)$ is a transaction from u on chain A that burns x tokens, which means that u can no longer access those tokens once the transaction is complete. $tx_{mint}(u, x)$ is a transaction on chain B that mints x new tokens to the user u upon verifying certain conditions.

We require $SPAS(u, x)$ to satisfy certain notions of privacy as defined below in Sec. 2.4.

Informally, we use *zkBridge* [XZC+22] to implement this by (verifiably) burning tokens on chain A and having a smart contract on chain B that upon input of the updated header and proof from an updater contract (given by *zkBridge*) makes certain checks and mints new tokens to the user on chain B.

Coin Retraction We also require that the SPAS supports coin retraction. Namely after an asset swap has been committed, say SPAS(u, x), the user should be able to retract the asset that has been locked, only if the user can provide a proof that the swapped assets have been burned on chain B. In this case the user u will be able to receive its asset back in the form of native tokens on chain A. In this case we also require $SPAS(u, x)$ to satisfy certain notions of privacy and fairness as defined below in Sec. 2.6.

2.4 Notions of Privacy

In most blockchains, the transaction data are posted in the clear so that the user involved in those transactions, as well as the content of the transactions are revealed to the outsiders. While these blockchains are widely being used, more and more attention has been given to blockchains with some meaningful notion of privacy. In our setting, we consider a meaningful notion of privacy which is necessary for many applications where the user's privacy must be protected.

Following the definition of similar to [DH20], we consider the notion of Unlinkability, which guarantees the following:

Unlinkability: No identity on chain A and chain B should be able to link the users on the two chains that complete the asset swap transactions.

Notice that we do not make any requirement for the relay network nodes between chain A and chain B.:

2.5 Notions of Fairness

We also require SPAS to satisfy the notion of fairness in that the user cannot double spend the swapped assets. More formally, even if the user colludes with part of the relay network, the user should still not be able to spend the locked coins on chain A after its new tokens have been minted on chain B.

2.6 Notions of Security

In terms of security, we do require that if the SPAS has the feature of coin retraction, then no one is able to spend the x amount of coins which the user u locked on chain A in an asset swapping protocol between chain A and chain B. More formally:

Let SPAS(u, x) be an asset swapping protocol from chain A to chain B, and suppose x amount of coins on chain A has been locked by user u in order to swap them onto chain B. Then once the coins have been locked by the user, no entity (including the nodes in the relay network) is able to spend the locked coins on chain A.

3 Setting

We consider building private bridges between Zcash and Ethereum. Importantly, the bridge is asymmetric, where the sender chain is arbitrary (does not need to support smart contracts), while the receiver chain does need to support smart contracts to implement minting functionality for tokens.

We use the relay network (of n nodes) and updater contract from zkBridge as is, to transmit block headers of the sender chain ("chain A") to the updater contract on the receiver chain ("chain B") that stores a list of verified block headers from chain A. We design a application smart contract on chain B that can interact with the updater contract and mint tokens to users upon satisfying certain conditions.

4 Contruction Overview

In the context described above, we present the first SPAS protocol with coin retraction which satisfies the notion of privacy and security as defined above. We first give an overview of this protocol, as described below:

In order for the user to lock its tokens on zcash, we first construct a burning transaction as follows: We will create a recipient address that is being collectively generated and controlled by the relay nodes network. Importantly, the secret key of this user address is distributed across the relay network so that even if the user colludes with part of the relay network nodes, it is still unable to control this address and spend its locked coin. This lock transaction will then be fulfilled with a shielded-to-transparent transaction on Zcash. Due to the privacy guarantee of Zcash, no other identity on chain (except for the relay nodes) will be able to distinguish this transaction from any other transaction.

In order for the user to receive minted tokens on Ethereum, it will provides a proof of this transaction through the relay network, to some smart contract which is located on Ethereum. This smart contract will also have a list of block headers from the zkBridge contract and can then verifies the following:

- The claimed burning transaction is indeed a valid transaction on Zcash, with the recipient address being set to that owned collectively by the relay network (that was generated for this specific user).

Upon checking these properties, the smart contract will then issue a minting transaction and send the minted tokens (for precisely the amount that was locked in the above transaction) to the user designated address on Ethereum.

In order for the user to retract its locked coins, it burns the (same amount of) wrapped tokens on Ethereum (via a smart contract) through the address which previously received the minted tokens. Upon checking that the tokens are indeed burned (this can be done without any interaction from the user, since Ethereum is public), the relay network nodes will collectively release their secret keys to the user u via a transaction on Ethereum that contains the secret keys encrypted with the user's public key (so that nobody else can obtain the funds in the locked address). Upon receiving these keys, the user will then be able to access the tokens previously sent to the corresponding address and then initiates another transaction to move those tokens back to its own address.

5 Construction

5.1 Lock coins – On Zcash

When a user u wants to swap x amount of Zcash into equivalent amount of tokens on Ethereum, she locks coins according to the below protocol:

1. The user informs the relay network of a new asset swap and requests a lock address. Then the relay nodes perform the following:

   (a) Suppose that there are n relay nodes in total in the relay network, these nodes then interact in order to create a multisignature wallet on Zcash.

   Note: There are other techniques to perform the generation of ADDR satisfying the four properties mentioned in Sec 1. Since transparent addresses on Zcash follow Bitcoin specification, the public key is derived from the private key by exponentiation in an elliptic curve group, and the payment address is then derived by hashing the curve point, which is deterministic. Since group exponentiations are homomorphic, we can multiply two public keys to produce a new valid public key which has a secret key that is the sum of the two old keys.

   We can exploit this property to devise a simple MPC protocol where relay nodes sample a random private key and produce the public key corresponding to that key - and simply multiply all the public keys to produce a new key which they send to the user. The user then adds their own public key to this address, which produces a new public key, to which *neither* the relay nodes nor the user know the private key to. This ensures that the coins are cryptographically locked until the relay nodes release their random keys to the user.

   This allows us to obtain a strong security property, where we can show that even if the trust assumption on the relay network is broken, even in the worst case when all relay nodes are corrupt, they will not be able to steal honest users' locked coins. This novel property of the bridge is useful in building trust among users where they can be assured that in the rare case that the assumptions underlying the relay network is broken, they are still safe and their privacy is also maintained.

   Importantly, this also removes the main incentive for relay nodes to collude or external agents to corrupt the relay nodes.

   (b) The relay network forwards the address of this wallet (denote by v) to the user u and store the tuple $(\texttt{user}, \mathsf{ADDR})$.

2. The user then issues a note on Zcash which transfers x amount of tokens to such address v, and creates a proof of the statement above: "There exists a valid transaction on Zcash that belongs to block with block header H that is sent to the address v and transfers x coins."

5.2 Verify and Mint tokens – On Ethereum

The smart contract on Etheurem will then verify the proof and the note and then mint tokens. Upon verification of the proof w.r.t. the block header from the updater contract (to verify inclusion of the note), the smart contract will mint tokens to the address that sent the above proof.

5.3 Coin retraction – on Zcash

Once the burning is done on Ethereum (could be done in multiple ways, including sending it to some public address to which the secret key is unknown, for ex. `0x0`), the relay network will perform the following steps to allow the user u to retract its locked assets on Zcash:

- The relay nodes will each verify that the user has indeed burned v tokens, where the value of tokens corresponds to the previously locked values in the multisig address on Zcash.
- The relay nodes also check that any transaction necessary is confirmed on Ethereum.
- The relay nodes then send their secret signing keys in an encrypted format to the user u (or equivalently to a public address/ bulletin board that the user can monitor).

After obtaining all the secret signing keys $\{sk_i\}_{i\in[n]}$. the user u can then issue a note from the multisig which sends the previously locked tokens back to its own wallet on Zcash.

6 Security Proofs

In this section we provide a high-level security proof of our asset swapping protocol (SPAS). Recall that we require that our SPAS to satisfy the privacy 2.4, fairness 2.5 and security 2.6 definitions.

6.1 Proof of Security

We first start with a proof of security of our SPAS. We show that no identity is allowed to spend the locked coins on Zcash. First notice that all other identities except for the relay node (recipient) can spend the coins. This is guaranteed by the security of Zcash. Next we show that even the relay nodes cannot spend the locked coins. This is due to the ciphertext of the sending note being mal-formed. In particular, since the note is generated via a dummy ciphertext, it contains no valid coin information and thus even the recipient are not allowed to spend such coin.

6.2 Proof of Fairness

We next show that the above SPAS construction satisfies the notion of fairness. In particular, we show that even if the user colludes with all but 1 nodes in the relay network, it is still unable to spend the locked token. This is guaranteed by the security of the multisig wallet, which states that in order to spend the coins contained in such a wallet, it is required that all n parties sign the note using their respective secret signing keys. Thus unless the user receives all secret signing keys from each of the n nodes in the relay network, it is unable to move locked tokens out of the multisig wallet.

Note: To account for any availability issues, we can relax the multisig wallet to be a threshold wallet, where only t out of n parties are required to spend from the address.

6.3 Proof of Privacy

We now show that our SPAS satisfies the privacy. In particular, it satisfies the notion of unlinkability on Zcash due to the underlying privacy guarantee of Zcash.

For unlinkability on Ethereum, notice that for any identity on Etheurem, due to the use of private smart contract, the recipient's address is hidden. Thus no identity is able to link two minting transactions to two different users.

7 Implementation

We implement the required smart contracts on Ethereum that handle proof verification as well as minting and burning tokens. We use the groth16 backend to minimize the on-chain cost of proof verification.
The proof statements and constraint generation was done in Circom and SnarkJS. Furthermore, in the process, we implemented the Blake2b-256 hash function in Circom, which could be useful in other applications as well.

The code is in the GitHub repository https://github.com/bunny-sleepy/private-bridge-zcash-ethereum.

8 Conclusions and Future work

We propose a privacy-preserving protocol for cross-chain asset swaps, which allows for a user to transfer their assets from one blockchain to another. This approach leverages the use of cryptographic techniques, such as verifiable random functions, multisignatures and zero-knowledge proofs to ensure the security and privacy of the transaction. We believe that this solution has applications in decentralised finance and has the potential to significantly improve the liquidity and usability of decentralized assets.

Further improvements include optimisation of the exact ZK statements being proved and provide a competitive implementation of the protocol. In our protocol, we rely on the relay network provided by zkBridge to transmit the block header updates to the other chain, which also requires a smart contract on the receiver end. It is also interesting to consider the case where we do not use smart contracts on the receiver end and instead rely only on some untrusted relay nodes for liveness and/or to verify proofs - this would help in cases where the receiving chain does not have smart contract capabilities (ex: Bitcoin, Zcash).

References

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. Cryptology ePrint Archive, Paper 2018/483, 2018. `https://eprint.iacr.org/2018/483`.

[Das] Dash. `http://dash.org`.

[DH20] Apoorvaa Deshpande and Maurice Herlihy. Privacy-preserving cross-chain atomic swaps. In Matthew Bernhard, Andrea Bracciali, L. Jean Camp, Shin'ichiro Matsuo, Alana Maurushat, Peter B. Rønne, and Massimiliano Sala, editors, *Financial Cryptography and Data Security*, pages 540–549, Cham, 2020. Springer International Publishing.

[Mon] Monero. `http://getmonero.org`.

[XZC+22] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. Zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 3003–3017, New York, NY, USA, 2022. Association for Computing Machinery.

[Zca] Zcash. `http://z.cash`.