Burn-to-Claim: a cross-blockchain asset transfer protocol

B. Pillai, K. Biswas, Z. Hou, V. Muthu

School of ICT, Griffith University, Gold Coast, Australia

March 5, 2021
Overview

1. Blockchain - Interoperability & Challenges

2. Burn-to-Claim - Cross-Blockchain Asset Transfer Protocol

3. Theoretical Analysis - Security, Correctness & Fairness
Information system interoperability refers to the ability to communicate and exchange information/data.

Blockchain aims to share or exchange value or data.

But the design and architecture of the technology:
- limits to the transaction within the network
- independent network has its own state assumptions and
- one network cannot verify the state of another network.

Therefore, interoperability is challenging to reach consensus.
Interoperability challenges

- Technical difficulty to trust and accept external data

Assumptions
- $N_1$ and $N_2$ are secure
- $v$ represent an asset-token
- merge mining and gateway node
Interoperability challenges

- Technical difficulty to trust and accept external data

Cross-blockchain properties

1. Security
2. Correctness
3. Fairness
Public key ($K_p$) and Private key ($K_r$)

genAddrress($K_p$) $\rightarrow$ blockchain address ($K_{adr}$)

$Tx[\text{sender} \rightarrow \text{recipient} : \text{value}]_{sign}$ $Tx[K^S_{adr} \rightarrow K^R_{adr} : v]K_r^S$

**Definition (Burn-address)**

A burn-address given as $\beta$ is an address to which one can send assets, but they can never be recovered because the private key of the corresponding address is not known/ accessible.

$Tx[K^S_{adr} \rightarrow \beta : v]K_r^S$

Karantias et. al., (2020). **Proof-of-burn**. In International Conference on Financial Cryptography and Data Security
**Burn-address**

Public key ($K_p$) and Private key ($K_r$)

$\text{genAddress}(K_p) \rightarrow \text{blockchain address (}K_{adr}\text{)}$

$Tx[\text{Sender } \rightarrow \text{Recipient : value}]_{\text{sign}} \quad Tx[K_{adr}^S \rightarrow K_{adr}^R : v]K_r^S$

**Definition (Burn-address)**

A burn-address given as $\beta$ is an address to which one can send assets, but they can never be recovered because the private key of the corresponding address is not known/ accessible.

$Tx[K_{adr}^S \rightarrow \beta : v]K_r^S \quad Tx[\beta \rightarrow K_{adr} : v]K_r$

Karantias et. al., (2020). **Proof-of-burn.** In International Conference on Financial Cryptography and Data Security
Proposed Burn-to-Claim protocol

Initiate exitTransaction
Generate a proof (burn the asset, time & hash lock)

Initiate reclaim Transaction

if NOT Source network

proof

Initiate entryTransaction
verify the proof & mint the asset

Destination network
Protocol overview

Alice

\[ \text{keygen()} \rightarrow \gamma \]
\[ \text{encrypt} \left( K_p^R, \gamma \right) \rightarrow \Gamma_p \]

Gateway(s)

exitTransaction()

Confirmation period

etit event

Bob

\[ \text{decrypt} \left( K_r^R, \Gamma_p \right) \rightarrow \gamma \]

Gateway(s)

add()

tenTransaction()

Confirmation period

etit event
Protocol overview - commit stage

Alice

\( \text{keygen}() \rightarrow \gamma \)
\( \text{encrypt} (K^R_p, \gamma) \rightarrow \Gamma_p \)

\( \text{exitTransaction}() \)

Gateway(s)

 confirms period

emit event

Bob

\( \text{decrypt} (K^R_r, \Gamma_p) \rightarrow \gamma \)

Gateway(s)

add()

Source

Destination

Time

\( t_0 \)

\( t_1 \)

If \( \text{exportVerifier}() \) is true

\( K^{S}_\text{adr} \rightarrow \beta : v \)

\( \text{timelocked}(v, t) \)

\( T_{xt} \leftarrow (T_x, H(\gamma), \sigma) \)
If importVerifier() is true and \( Tx_t.\text{time-lock} \) under limit \&
\[ H(\text{decrypt}(\gamma, K_p^R)) = Tx_t.H(\gamma) \]
\[ \beta \Rightarrow K_{adr}^R : v \]
Protocol overview - execute stage

1. **Prepare**
   - *keygen()* $\rightarrow \gamma$
   - *encrypt* $(K_p^R, \gamma) \rightarrow \Gamma_p$

2. **Commit**
   - Gateway(s)
   - *decrypt* $(K_r^R, \Gamma_p) \rightarrow \gamma$

3. **Execute**
   - *exitTransaction()*
   - Confirmation period
   - emit event
   - *add()*

4. **Source**
   - *reclaimTransaction()*
   - Confirmation period
   - emit event

5. **Destination**
   - If *reclaimVerifier()* is true and $Tx_t.time - pass limit \& H(decrypt(\gamma, K_p^S)) = Tx_t.H(\gamma)$
   - $\beta \Rightarrow K_{adr}^R : v$
Burn-to-Claim protocol case study

Alice (sender) S
Keygen() → secret-key γ
encrypt(γ, K_p^R) → Γ_p
Prepare

Bob (recipient) R
decrypt(Γ_p, K_p^R) → γ

\( exitTransaction() \) (\( K_{addr}^S \rightarrow β : v \))

\( t_0 \)

emit event

Confirmation period

Gateway node update(event)

\( t_1 \)

emit event

reclaimTransaction() (\( β \rightarrow K_{addr}^S : v \))

Source

Bob (recipient) R

if (R claim v) R reveal γ with in \( t_1 \)

if (not) S reveal γ after \( t_1 \)

Destination

notify

\( v(timeLock - t_1) \rightarrow H(γ) \)

Commit

Gateway node add(event)

Execute

entryTransaction() (\( β \rightarrow K_{addr}^R : v \))

Confirmation period

emit event
The recipient network can rely on the Burn-to-Claim exit-proof guarantee provided by the source network.

- Burn-address $\beta$ is unspendable with respect to a proof-of-burn protocol.
- `exitTransaction` function transfer asset $v$ to a $\beta$, which is unspendable.
The exchange operation only exchange an asset to a correct recipient

- $\beta$ is derived from the recipient’s address $K_{adr}^R$
- Recipient must prove the ownership of address
- Nodes verify signature
- Only the user who owns a private key associated with $K_{adr}^R$ can make the claim for the asset in the destination network
The exchange operation should only yield one of the following outcomes; either the exchange succeeds and the asset is transferred to the recipient; or it failed and the asset must return to the sender.

- For recipient to claim $v$, recipient must reveal $\gamma$ there after $S$ can not claim it because $\gamma$ is known to both the networks.

- If $R$ fails to claim, after the time-lock period $S$ can re-claim $v$.

- With the hash-time-lock mechanism the transfer can be guaranteed atomicity, without a trusted third party, thus the protocol satisfies the fairness property.
Future work

- Implement and test the protocol for an application environment

- Time and cost analysis
  - *time* - transfer time from $N_1$ to $N_2$
  - *cost* - gas cost to perform the transfer

- Develop a threat model - selfish mining and gateways
  - evaluating using formal analysis
thank you...

babu.pillai@griffithuni.edu.au