sigBridge: Cross-chain Bridge for Permissioned Blockchains
and its application to access control

Mahmudun Nabi, Sepideh Avizheh, Preston Haffey, and Reihaneh Safavi-Naini
Computer Science Department
University of Calgary, Alberta, Canada

UNIVERSITY OF CALGARY

1. Cross-chain Bridge

Figure 1: Cross-chain Bridge (basic idea)

- Enables secure and efficient communication between two blockchains BC_1 *and* BC_2
- Provides evidence to applications on BC_2 that a specific event occurred on BC_1 , and vice versa

2. Building on zkBridge (CCS'22)

- .Cross-chain Bridge for a **non-Permissioned**
  - Enables message passing, asset transfer
- Uses *zero-knowledge proofs* (ZKP) to convince an application in BC_2 that certain event took place in BC_1

zkBridge Components:

(Core Bridge functions)

Security guarantees:
- **Correctness**: BC_2 receives correct view of BC_1
- **Liveness**: Data on BC_1 is always available to BC_2

3. *sigBridge*: Signature of consensus bridge

- Cross-chain Bridge in a **Permissioned Setting**
  - Enables message passing, data transfer
- **Uses digital signatures of the consensus protocol of sender's chain on block header and verifies the signatures in receiver's chain for header validation**

sigBridge components:

➢ **Relay network**: a set of nodes that are registered to both blockchains. Each relay node:
- relays block headers from BC_1 (sender chain) to BC_2 (receiver chain)
- relays application–specific messages from BC_1 to BC_2

➢ **Updater contract**:
- stores the consensus rules of the remote chain
- checks the signatures on the block header based on the remote consensus rule
  - If valid, stores the block headers in a header history
  - Else. Rejects the block header
- verifies that the messages (transactions) received by application contracts are correct

➢ **Application contracts**: for resource sharing application
- $obj - dir$: stores resource information
- $SC - \phi$: evaluates resource access policy

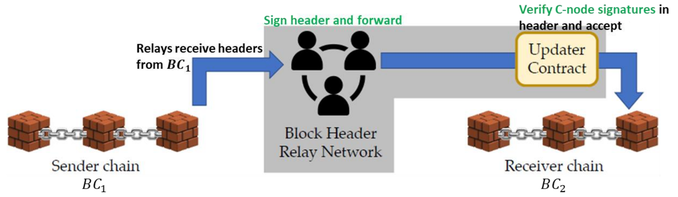


Figure 2: *sigBridge* design. The shaded components belongs to core bridge functionality with green representing our proposed changes in zkBridge.

How it works:
Step 1: BC_1 sends block header to relayers
Step 2: Relayers receive the block header, each sign the block header
Step 3: Each relayer send the block header, signature, and its certificate to BC_2
Step 4: BC_2 mainatiners validate the headers (using updater SC) and add it to their view

Threat Model:
- **Relay nodes that can be fully malicious**. They can-
  - modify, forge, or drop messages
  - try to participate in message-passing without registration

Security assumptions:
- underlying blockchains are secure, i.e., consistent and live
- at least one honest relay node exists in the system
- signature schemes used is unforgeable

Security guarantees:
- **Correctness**: BC_2 accepts wrong transaction of BC_1 with negligible probability.
- **Liveness**: If BC_2 wants to verify a transaction in BC_1, the bridge always provides the necessary information.

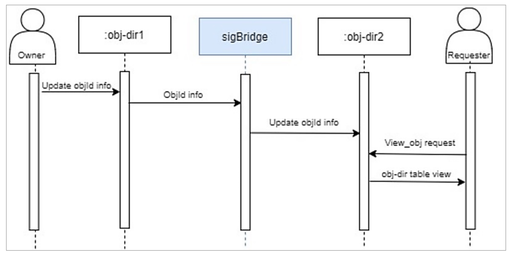
4. sigBridge Application: Cross-chain user-centric access control using *sigBridge*

- Use sigBridge protocol to inter-operate between two chains
- uses smart contracts to share resource information and evaluate access policies

Resource sharing example scenario:

➢ Alice (owner) is $\boldsymbol{BC_1}$ member and shares her resource in $\boldsymbol{BC_1}$
➢ Bob (requester) is a $\boldsymbol{BC_2}$ member. Bob wants to :
- ✓ See available resources in $\mathbf{BC_1}$.
- ✓ Access a specific "Object" information from $\mathbf{BC_1}$.

Browse Available Resources:



Access an Object:

1: $requester \rightarrow$ `obj-dir`$_2$: $Objreq$ ▷
$Objreq = (Fields[$"ObjId", "user attributes", certs$], sig_{pk_B}(Fields), cert_{pk_B})$
2: `obj-dir`$_2$: $RelayTransaction(Objreq)$ ▷ $sigBridge[$`obj-dir`$_2 \rightarrow$ `SC-`$\phi_i](Objreq)$
3: `SC-`ϕ_i evaluates the policy on $Objreq$ and outputs Token
4: `SC-`ϕ_i : $RelayTransaction(Token)$ ▷ $sigBridge[$`SC-`$\phi_i \rightarrow$ `obj-dir`$_2](Token)$
5: `obj-dir`$_2$ $\rightarrow requester : Token$

Threat Model: all users to be potentially malicious
Security assumptions: Both permissioned chains use the same
- cryptographic specifications and request/response formats

Security guarantee:

Cross-chain resource-sharing scheme achieves correctness and security if the sigBridge protocol ensures liveness and correctness and assuming both communicating blockchains ensure trusted execution of the smart contracts.

5. Future Works

- Use proofs of correctness of block headers

Acknowledgment
- In part supported by NSERC-Telus Industrial Research Chair in Information Security.

References

[zkBridge] Xie et. al. "zkBridge: Trustless crosschain bridges made practical" (CCS'22)

Contact:
Mahmudun.nabi1@ucalgary.ca