2025.9.11

Aegis:

Scalable Privacy-preserving CBDC Framework with Dynamic Proof of Liabilities

국가암호공모전'25

Contents

01 Introduction

- CBDC : a digital form of fiat money that is issued by central banks.

Decline in cash usage

Rapid growth
of the virtual asset market

Increasing need for
electronic payment methods

BIS Report (Aug 25, 2022): 91% of central banks actively engaged in CBDC research and pilots.

01 Introduction

- CBDC : a digital form of fiat money that is issued by central banks.

Decline in cash usage

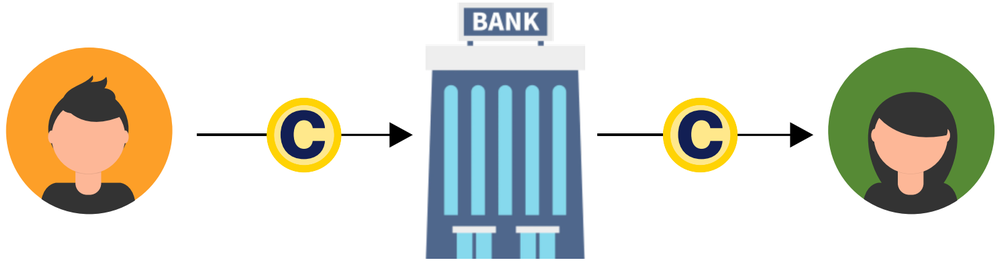
Rapid growth
of the virtual asset market

Increasing need for
electronic payment methods

BIS Report (Aug 25, 2022): 91% of central banks actively engaged in CBDC research and pilots.

- There are two ways to transact CBDC.

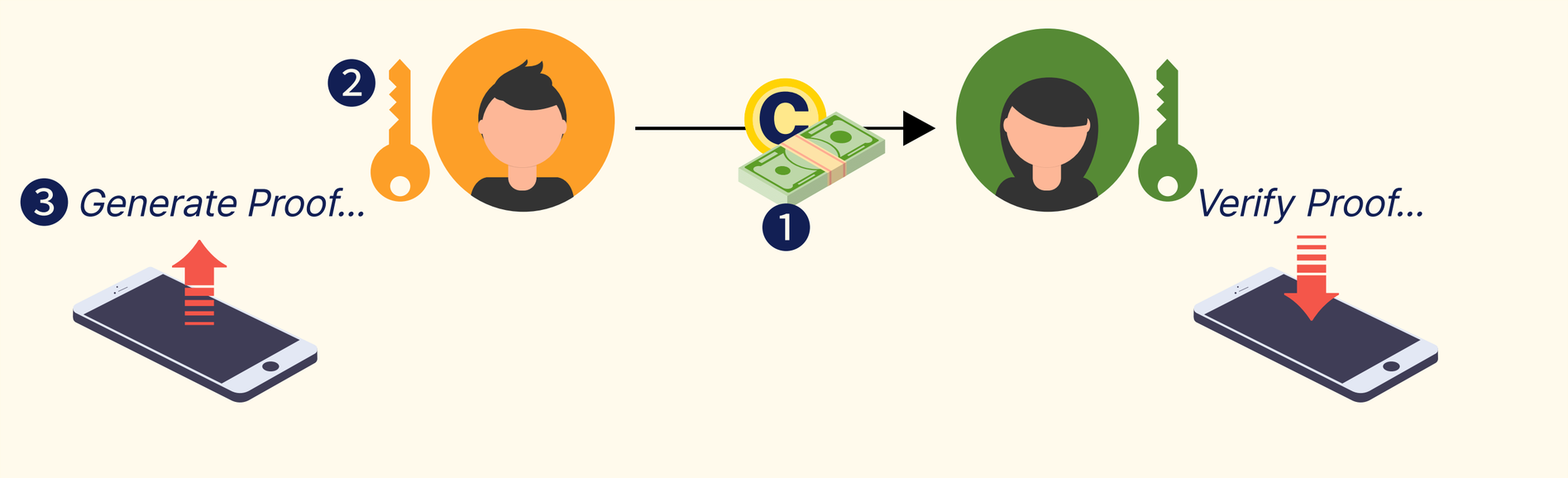
▲ Person-to-Person



▲ Bank-Mediated

01 Introduction

- **Which one is better?**
  - Cons of Person-to-Person CBDC Transactions



❶ Hard to ensure compliance with AML/CFT regulations.

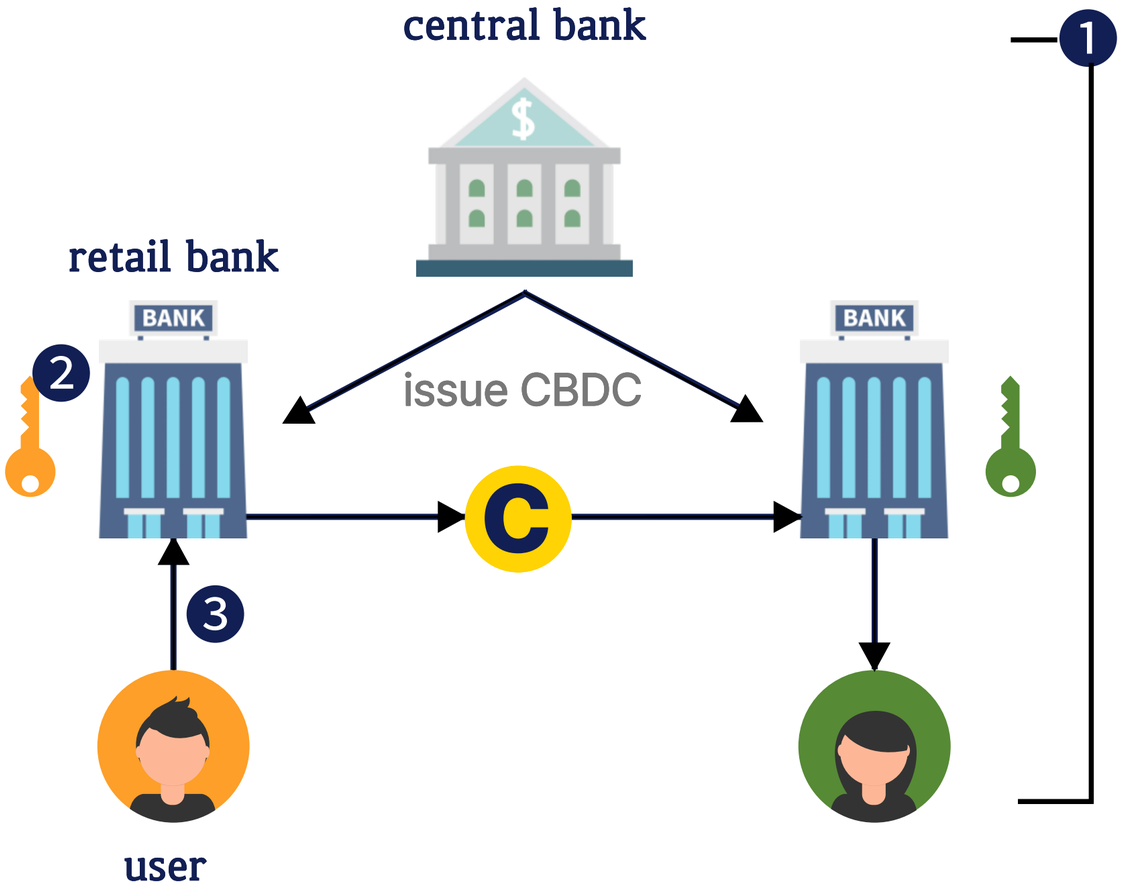
❷ Users must directly manage private keys.

❸ Users must apply cryptographic techniques themselves to validate their transactions.

*AML : Anti-Money Laundering, CFT : Countering the Financing of Terrorism

01 Introduction

- Which one is better?
  - Pros of Bank-Mediated CBDC Transactions



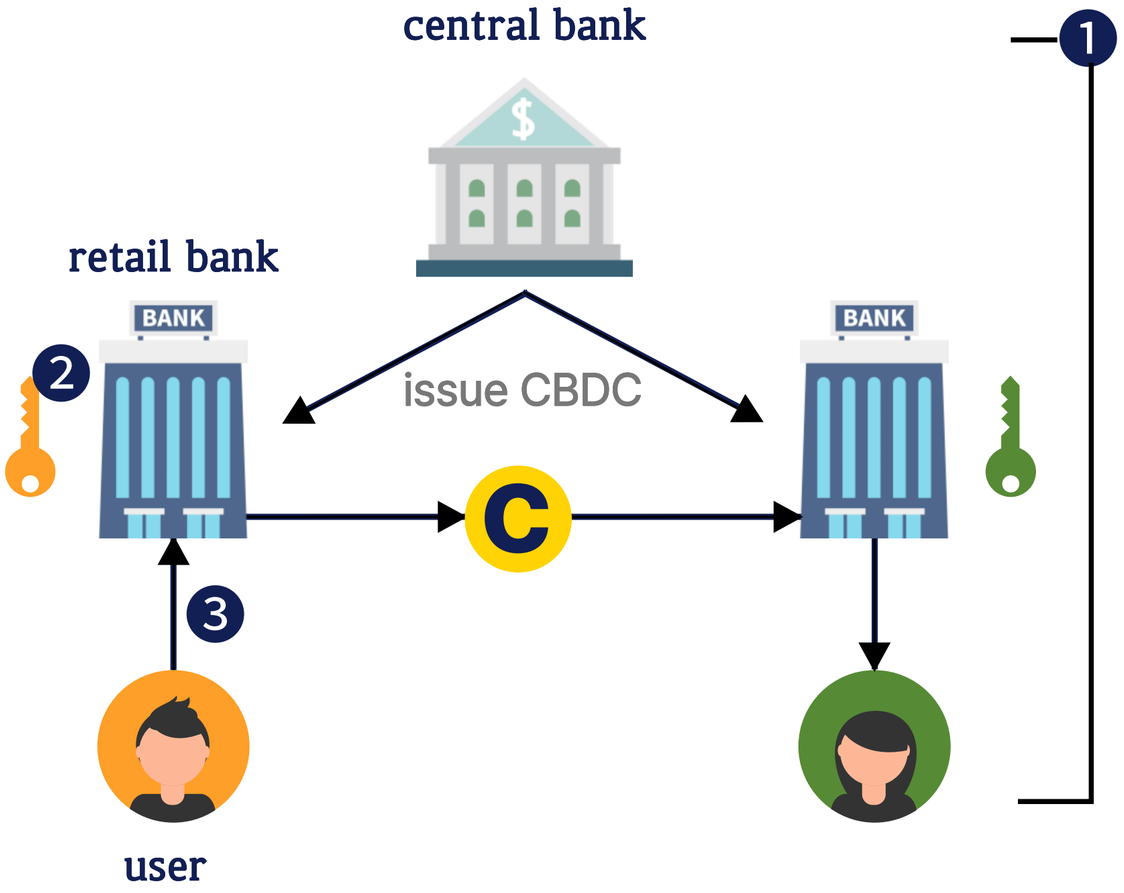
❶ Existing banking infrastructure → User-friendly

❷ Key custody service

❸ Delegated proof generation to bank

01 Introduction

- Which one is better?
  - Pros of Bank-Mediated CBDC Transactions



❶ Existing banking infrastructure → User-friendly

❷ Key custody service

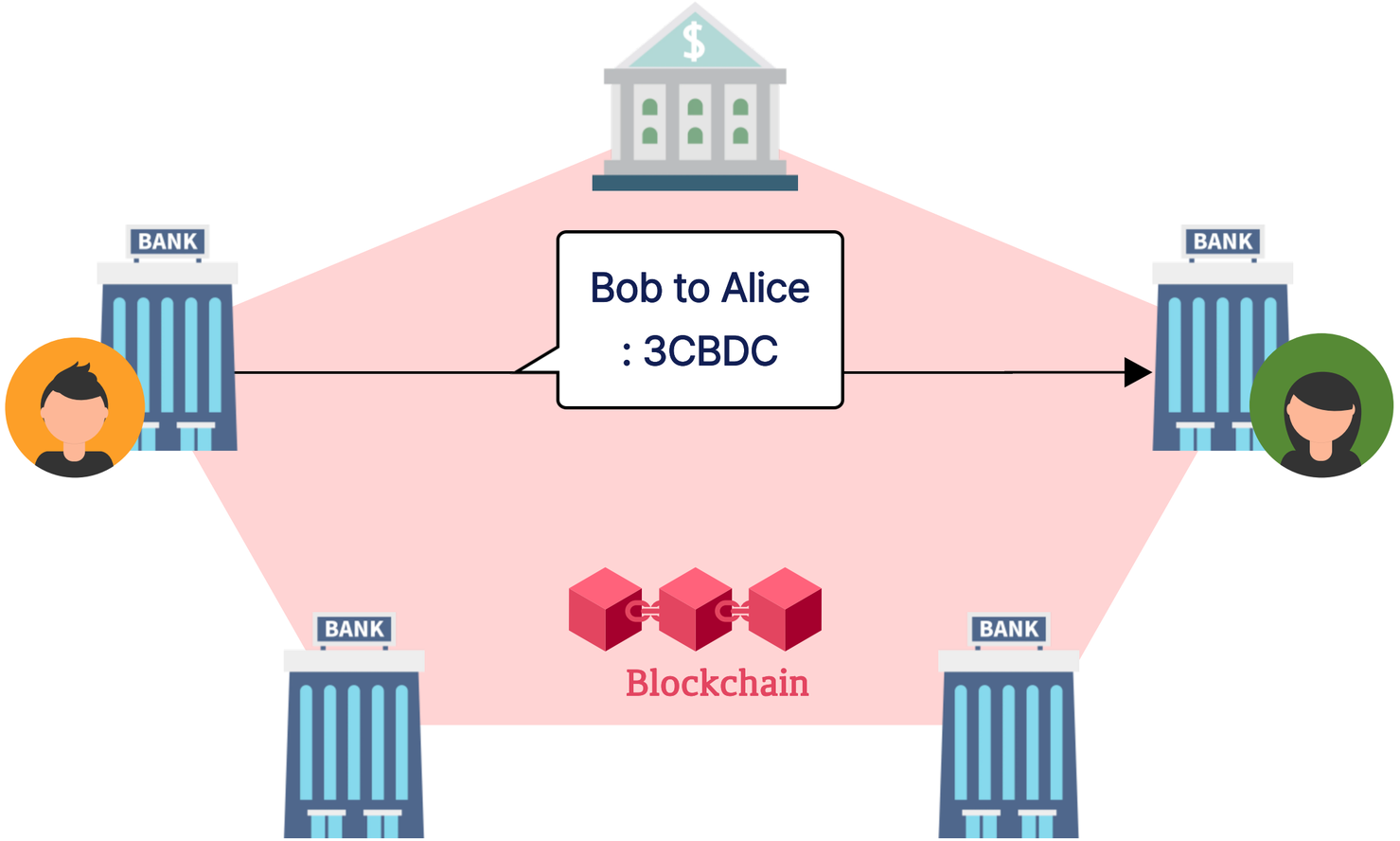
❸ Delegated proof generation to bank

Technologies to Gain Users' Trust?

01 Introduction

1. Blockchain

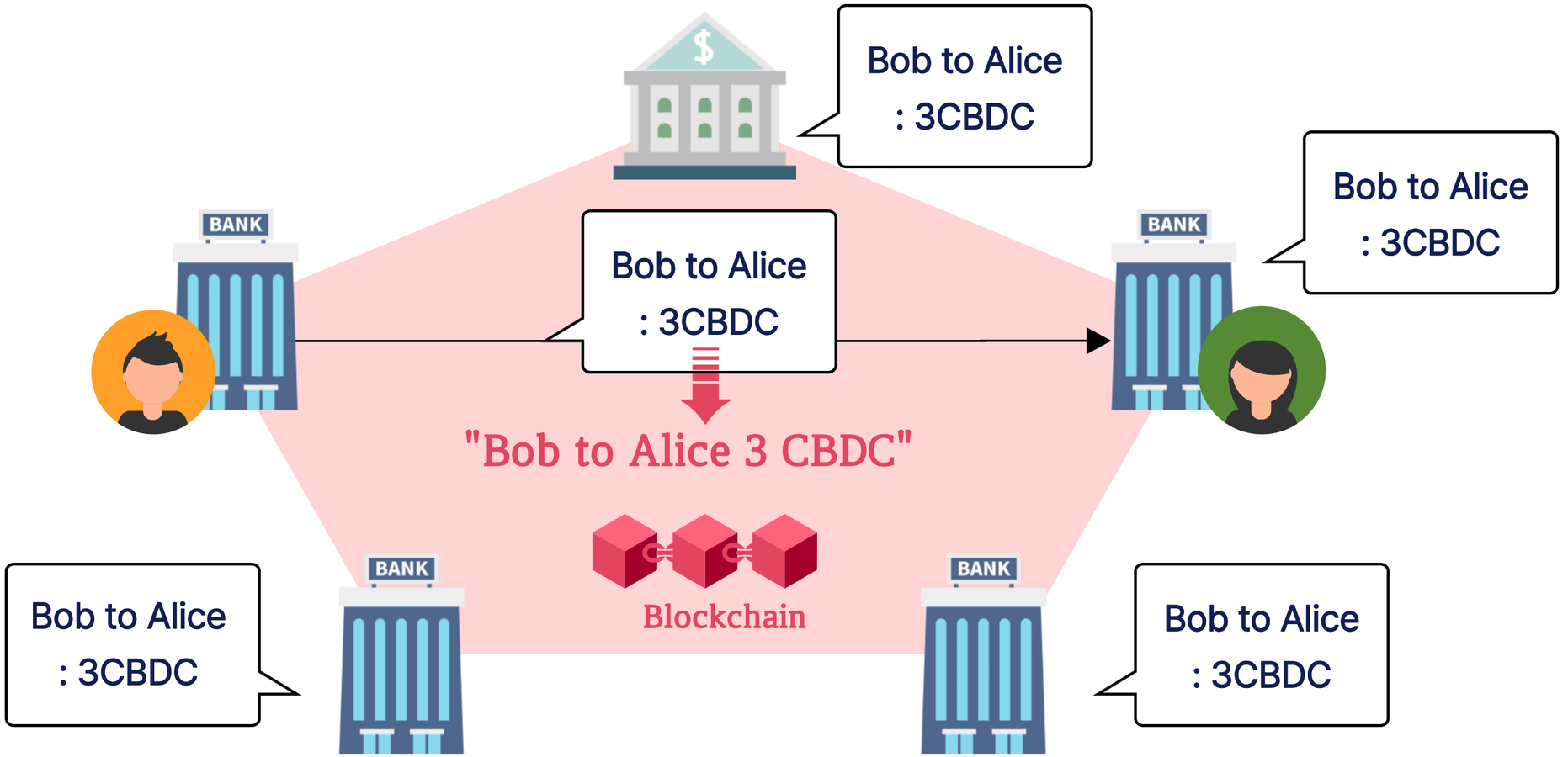
- Adoption of blockchain technology to enhance the trust and transparency of CBDC.



01 Introduction

1. Blockchain

- Adoption of blockchain technology to enhance the trust and transparency of CBDC.

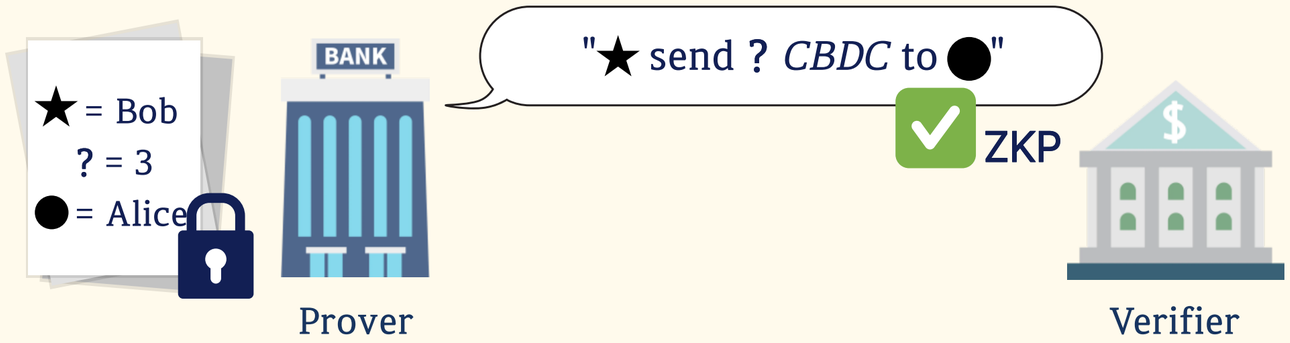


2. Zero-Knowledge Proof

- Blockchain transparency threatens user privacy → Leveraging Zero-Knowledge Proof (ZKP).

Zero-Knowledge Proof

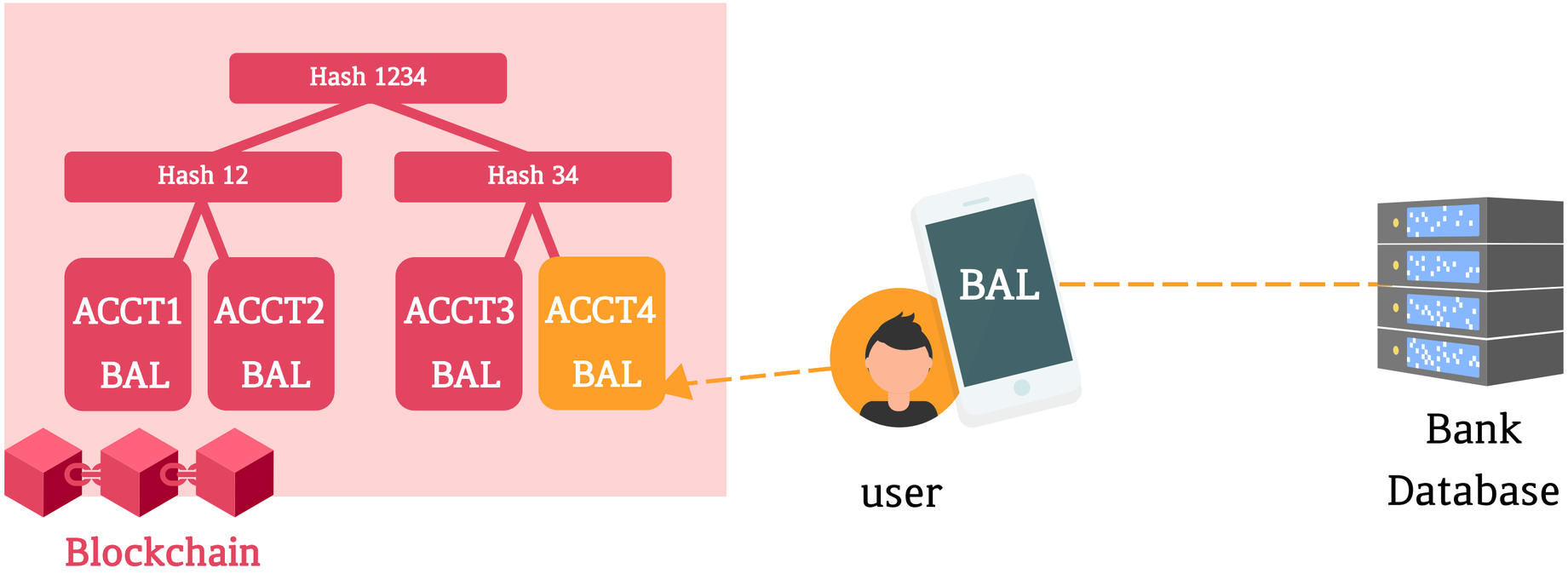
a proof system that can prove the validity of the **statement** without revealing the **witness** values.



A suitable solution for resolving the conflict between *privacy* and *public verifiability.*

3. Proof of Liabilities, PoL

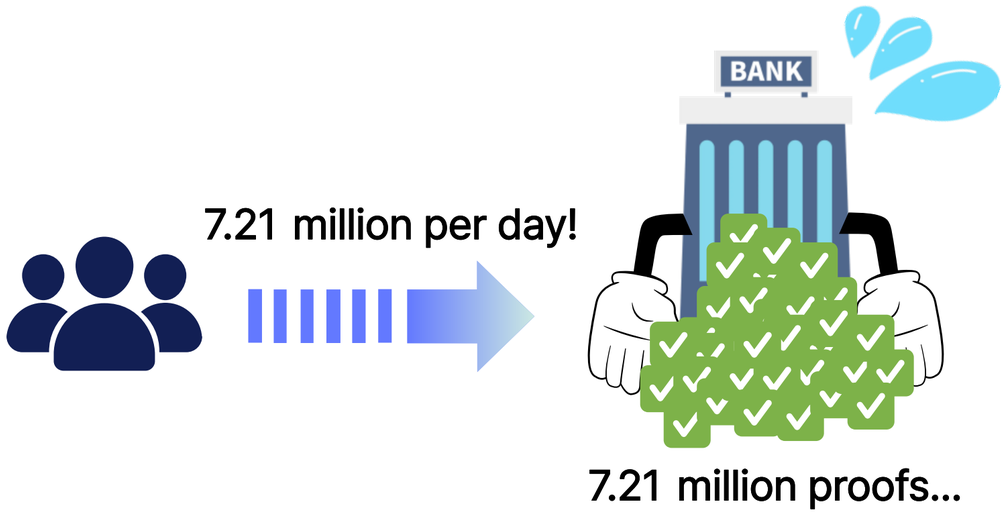
- A cryptographic primitive that allows a party to prove its financial solvency to users.



▲ Common PoL Method(Conducted monthly epoch)

02 Related Work

- The need for optimization for high-volume retail transactions.



▲ Daily average transactions in Korea

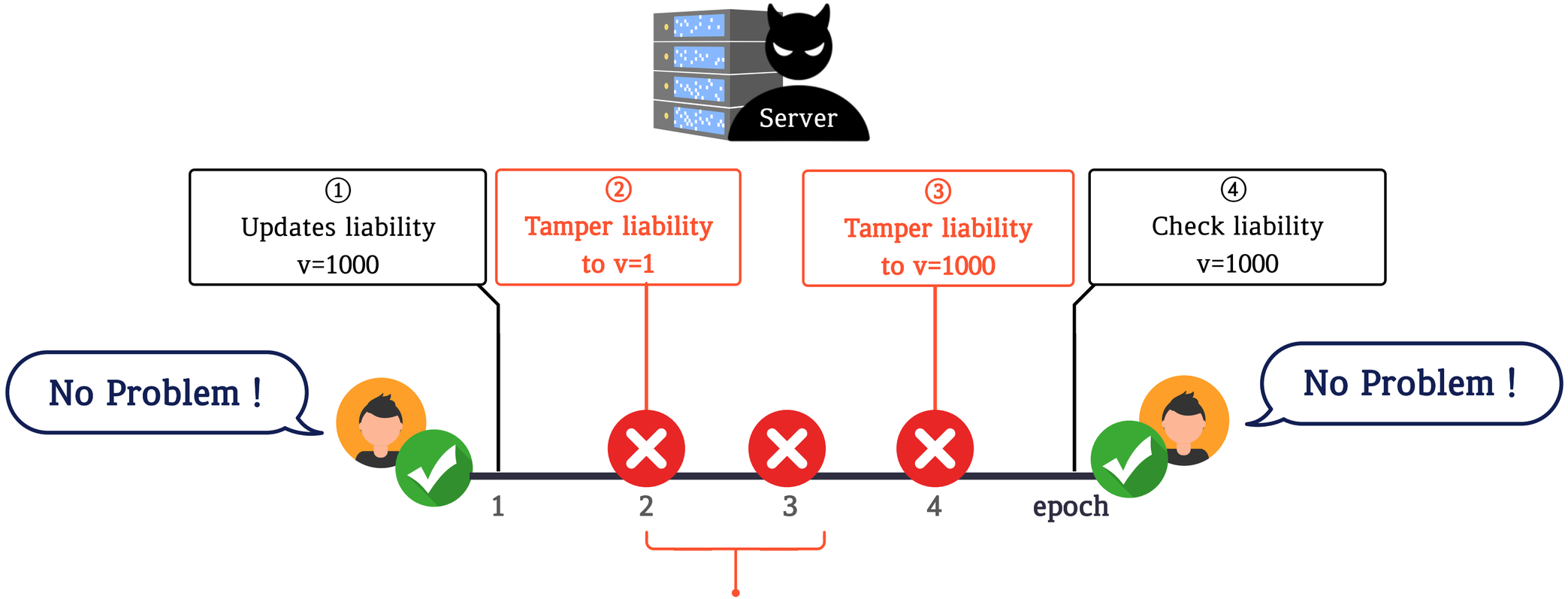
Related Works	1024 Tx (s)
Solidus[Cecchetti et al.,CCS'17]	19.42
zkLedger[Narula et al.,USENIX'18]	221.31

▲ Transaction time(s) about Prior Works

*Bank of Korea Report (Mar 20, 2025), Daily average transactions of simple remittance service in 2024

02 Related Work

- Current PoL: All users must directly verify their records at every epoch.
  - vulnerable to window-opportunity attacks.



03 Our Contribution

- Aegis : Scalable Privacy-preserving CBDC Framework with Dynamic Proof of Liabilities

1. Absence of efficient proof generation and for multiple transactions.

➙ Using a new transaction batching technique,
allow the generation of single proof for large-scale transactions.

2. All users must directly verify their records at every epoch for PoL.

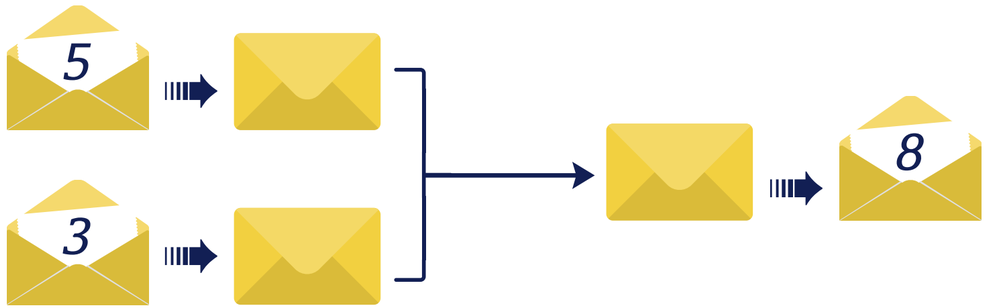
➙ Near-real-time PoL that does not require users to directly verify their records at every epoch.

04 Construction

Building block

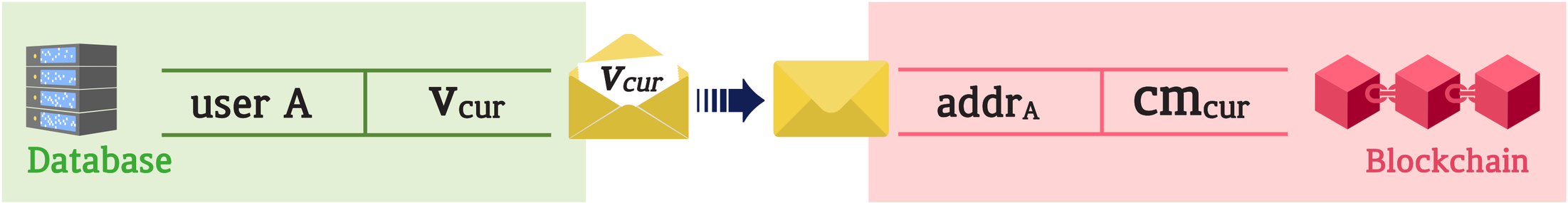
- **Pedersen commitment** supports **additive homomorphism**, enabling the sum of multiple committed values.

Pedersen Commitment



$$\mathrm{cm}_1 \leftarrow \mathrm{Ped.Com}(m_1;\mathrm{o}_1)$$
$$\mathrm{cm}_2 \leftarrow \mathrm{Ped.Com}(m_2;\mathrm{o}_2)$$
$$\mathrm{cm}_1 \uplus \mathrm{cm}_2 = \mathrm{Ped.Com}(m_1+m_2,\ \mathrm{o}_1+\mathrm{o}_2)$$

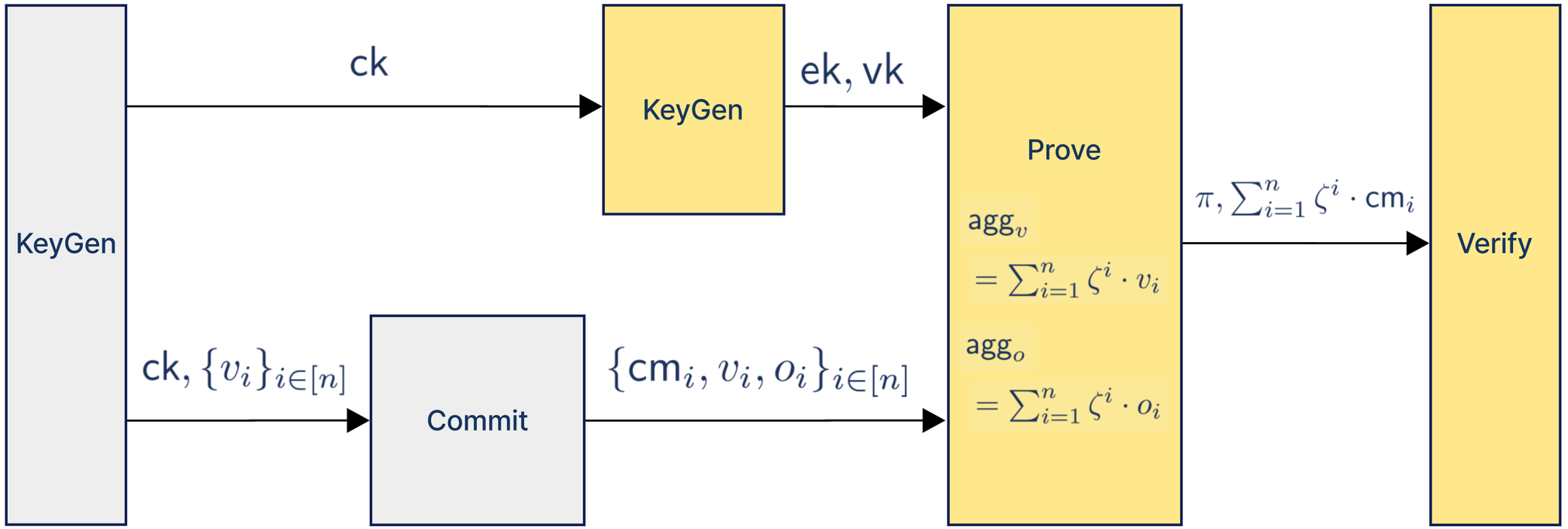
Users' balances will be committed and stored on the blockchain.



04 Construction

Building block

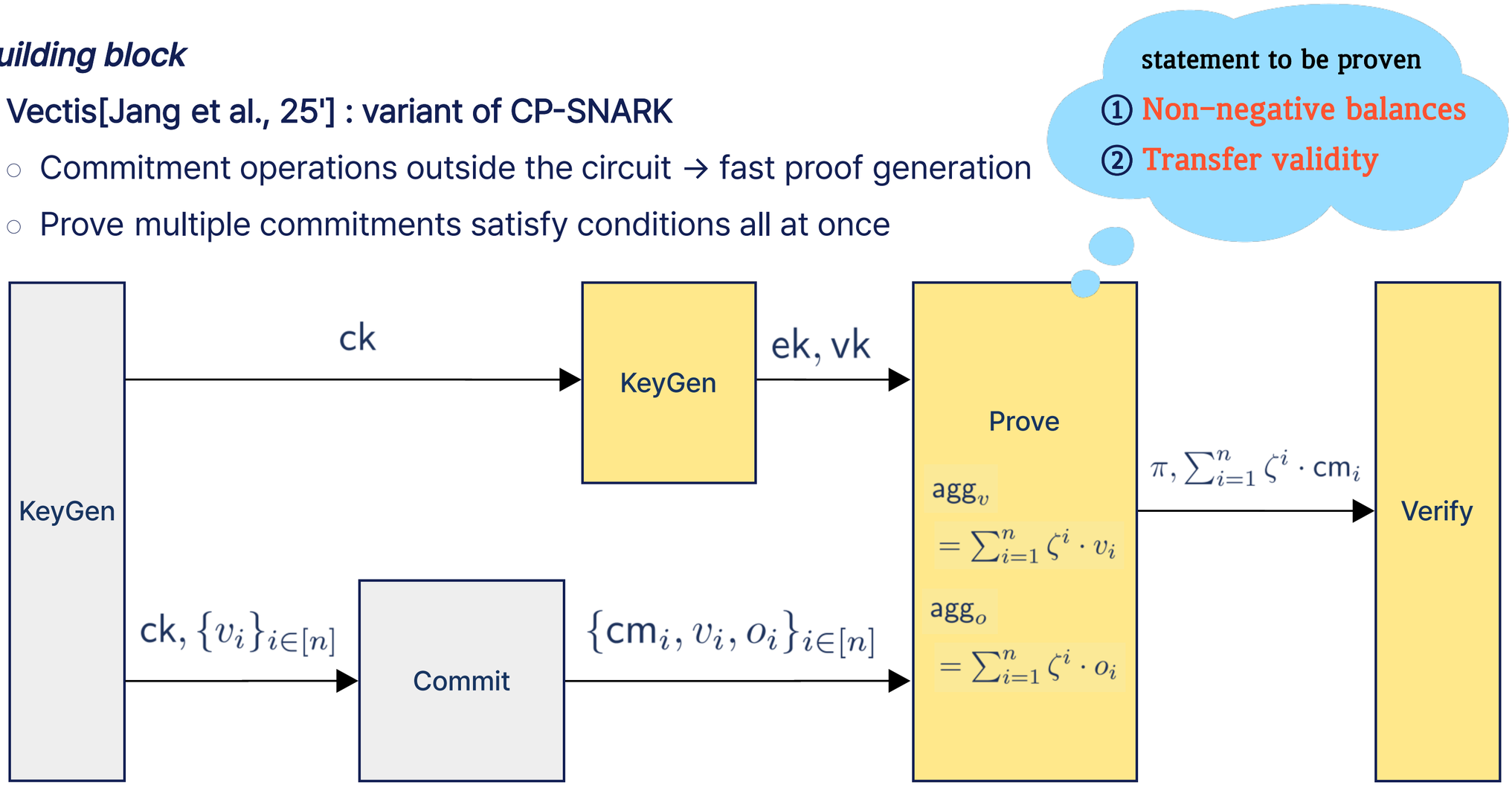
- **Vectis[Jang et al., 25'] : variant of CP-SNARK(Commit and Prove SNARK)**
  - Commitment operations outside the circuit → fast proof generation
  - Prove multiple commitments satisfy conditions all at once



04 Construction

Building block

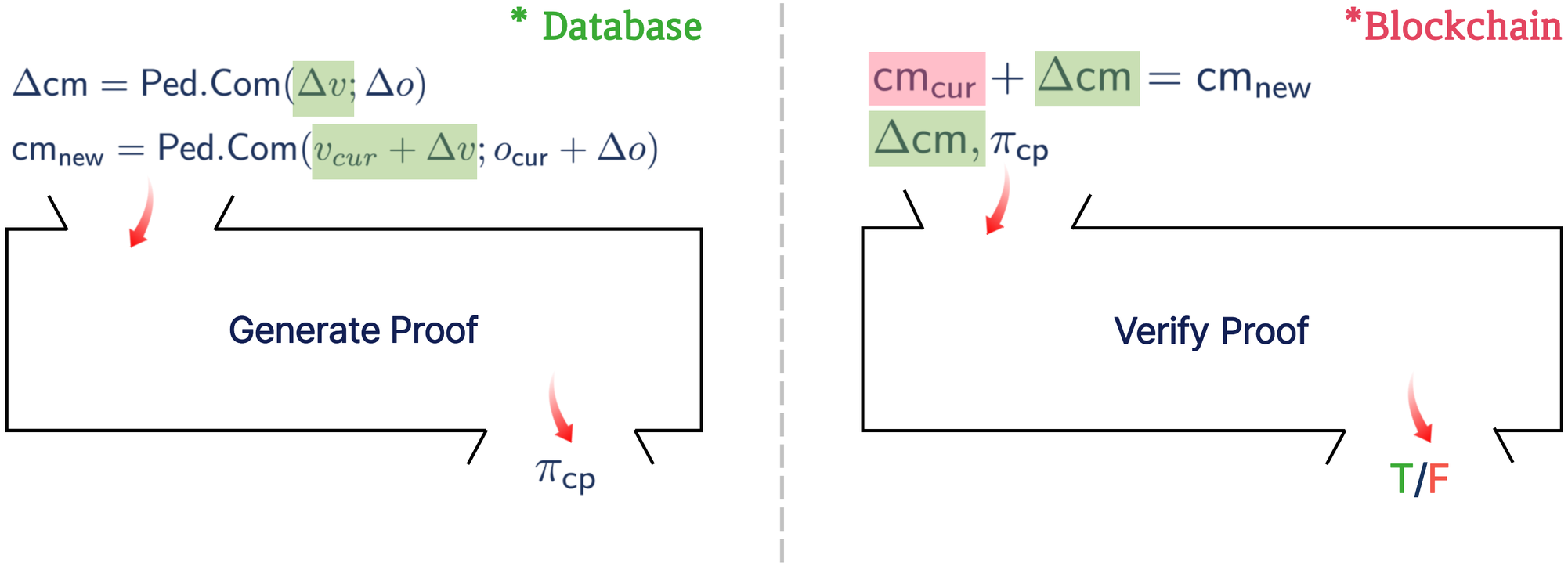
- **Vectis[Jang et al., 25'] : variant of CP-SNARK**
  - Commitment operations outside the circuit → fast proof generation
  - Prove multiple commitments satisfy conditions all at once



04 Construction

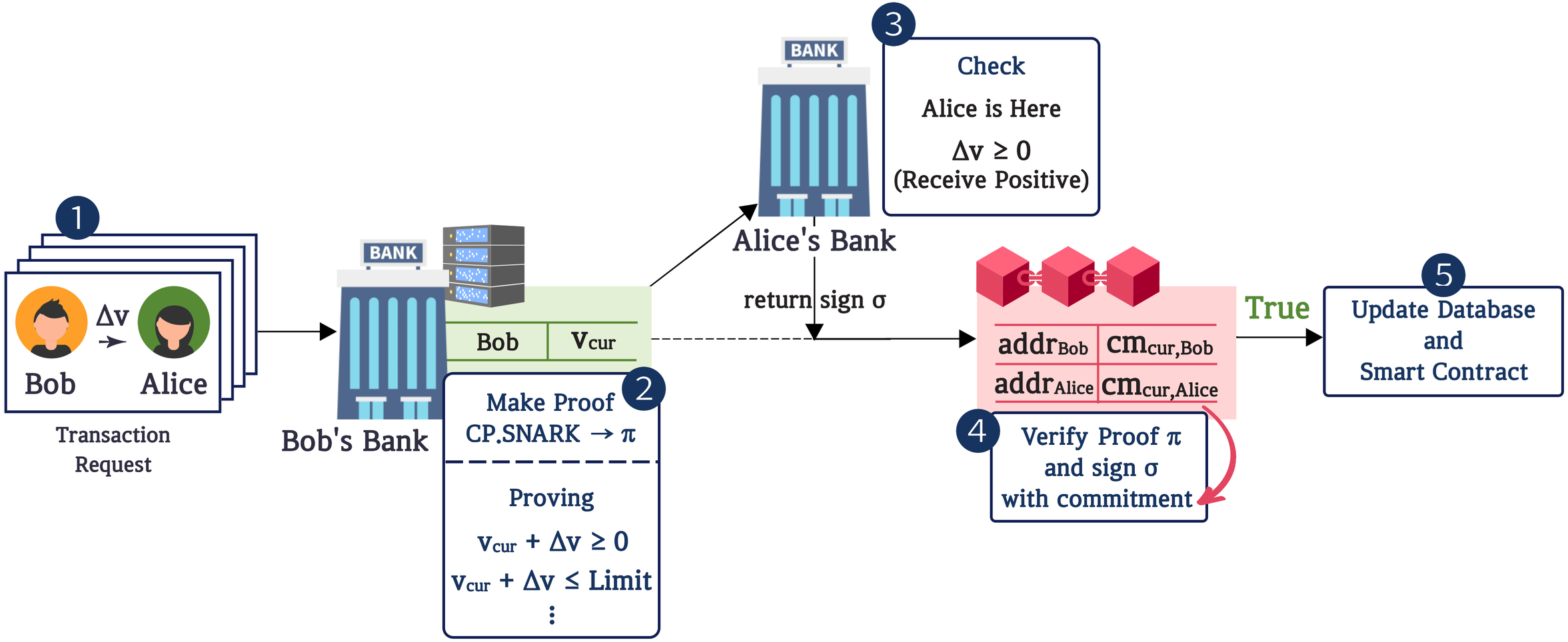
Building block

- **What aspect of CP-SNARK makes PoL possible?**



04 Construction

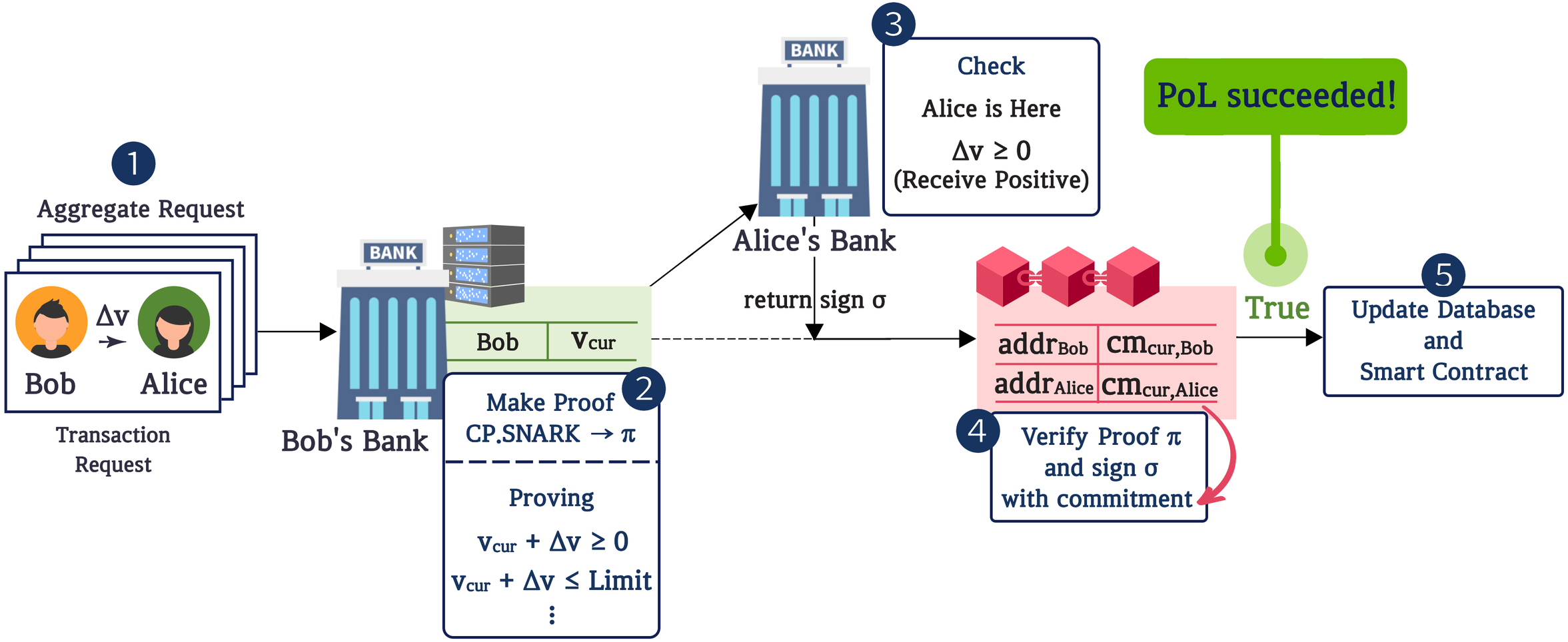
- Overview



▲ Overview of Aegis

04 Construction

- Overview



▲ Overview of Aegis

05 Comparison

Scalability : the generation of single proof for large-scale transactions.

- Implemented in Rust (Arkworks) & TypeScript on Ethereum (Hardhat, Solidity).
- Tested on MacBook Pro M1 (32GB RAM).

size (log)	Setup (s)	Prove (s)	Verify (ms)	Constraints	ek (KB)	vk (B)
3	0.048	0.031	1.264	4.1k	925	
4	0.069	0.054	1.313	8.2k	1,850	
5	0.128	0.099	1.336	16.4k	3,698	
6	0.245	0.191	1.525	32.9k	7,397	
7	0.472	0.316	1.821	65.9k	14,793	296
8	0.896	0.618	2.040	131.8k	29,586	
9	1.781	1.150	2.900	263.6k	59,171	
10	3.469	2.217	4.245	527.3k	118,342	
11	7.046	4.557	6.646	1,054.7k	236,684	
12	14.169	9.571	11.796	2,109.4k	473,367	

▲ zk-SNARK(ZKP) Performance by Batch Size

Tx \ Name	Total Time (s)		
	Aegis	zkLedger	Solidus
4	0.05	0.09	0.89
8	0.08	0.13	1.77
16	0.13	0.36	3.53
32	0.24	0.63	6.95
64	0.39	1.22	14.02
128	0.77	2.50	27.89
256	1.45	4.93	55.57
512	2.84	9.73	111.29
1024	6.07	19.42	221.31
2048	12.67	38.71	449.64

▲ Transaction Scalability(10 threads)

06 Summary

➔ Practical Privacy-Preserving framework for retail payments in the conventional banking system.

➔ One proof for large-scale transactions.

➔ Refine the definition of near-real-time PoL.

➔ Better Auditability & Scalability than in Solidus/zkLedger.

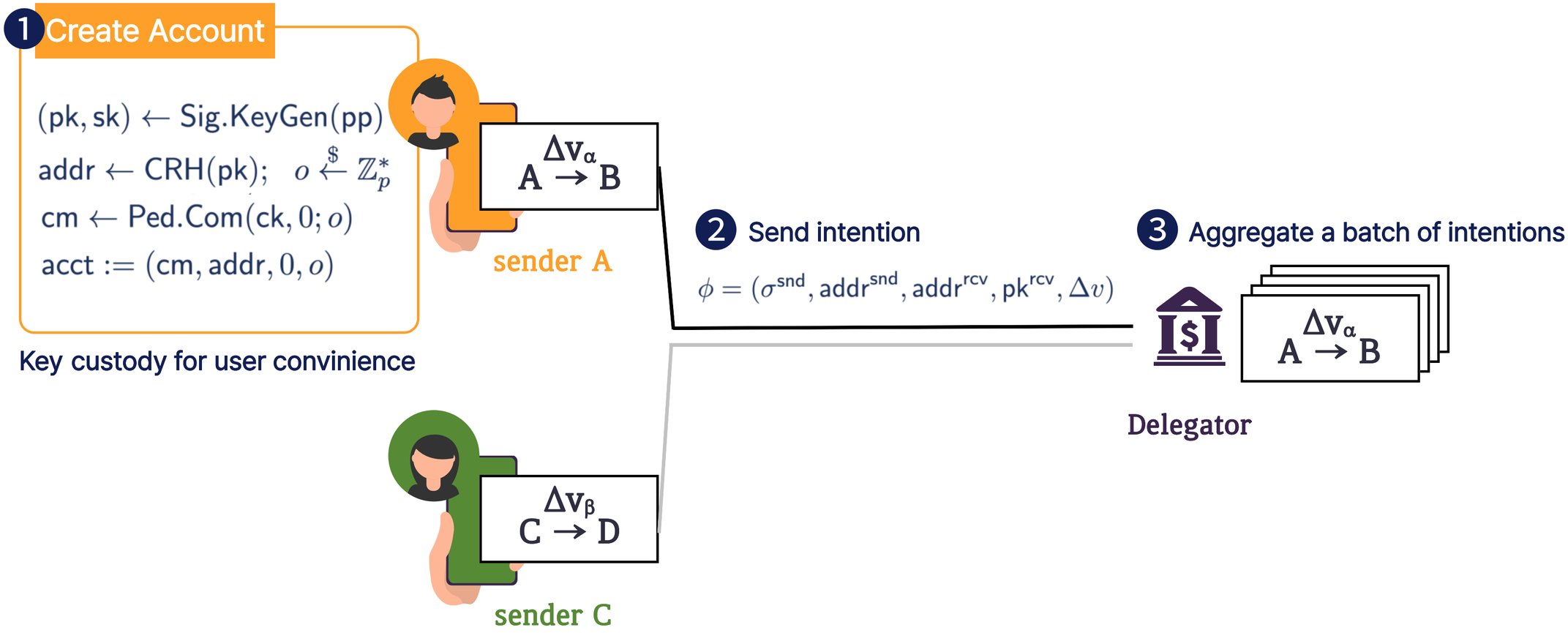
경청해주셔서 감사합니다.

Aegis:
Scalable Privacy-preserving CBDC Framework with Dynamic Proof of Liabilities

부록

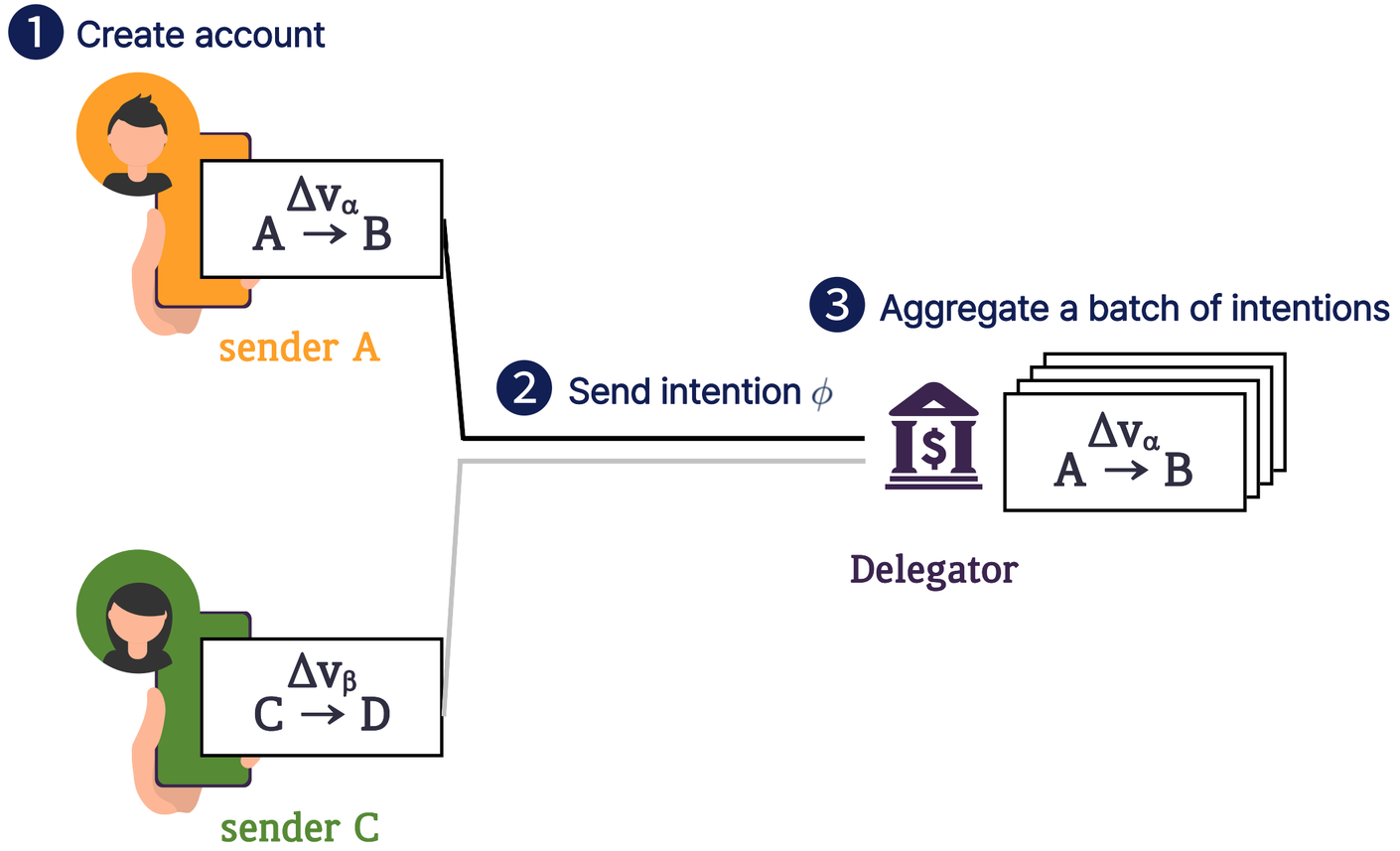
04 Construction

- Step 1. Batching Technique



04 Construction

- Step 1. Batching Technique



key point

1-1. Prepare $N/2$ pairs of

$$\left(v_{\text{cur},j}, o_{\text{cur},j}\right)$$

➙ of senders' from DB.

➙ Also receivers'. (set both 0)

1-2. Applying bijective mapping

$$p(j) = i \quad j \neq i$$

➙ k-anonymity

➙ <u>batch level unlinkability</u>

used as receivers' index :

$$\left(v_{\text{cur},p(j)}, o_{\text{cur},p(j)}\right)$$

04 Construction

- Step 1. Batching Technique

4 Create ZKP

key point

1-3. Calculate updated value of N pair of senders' and receivers'.

sender A 

$$v_{\text{new},j} = v_{\text{cur},j} - \Delta v,\ o_{\text{new},j} = o_{\text{cur},j} - \Delta o$$

receiver B 

$$v_{\text{new},p(j)} = v_{\text{cur},p(j)} + \Delta v,\ \ o_{\text{new},p(j)} = o_{\text{cur},p(j)} + \Delta o$$

1-4. Commit values into Pedersen Commitments.

$$\{\Delta \text{cm}_k := \text{Ped.Com}(\text{ck}, \Delta v_k; \Delta o_k)\}_{k\in[N]}$$

$$\{\text{cm}_{\text{new},k} := \text{Ped.Com}(\text{ck}, v_{\text{new},k}; o_{\text{new},k})\}_{k\in[N]}$$

04 Construction

- Step 1. Batching Technique

4 Create ZKP

key point

1-5. Return ZK-Proof.

$$\vec{x} := \left\{\{\mathsf{cm}_{\mathsf{new},k}\}_{k\in[N]}, \{\Delta cm_k\}_{k\in[N]}\right\}$$
$$\vec{w} := \left\{\{(v_{\mathsf{cur},k}, o_{\mathsf{cur},k})\}_{k\in[N]}, \{(\Delta v_k, \Delta o_k)\}_{k\in[N]}\right\}$$

Prove what ?

① **Non-negative balances**

$$0 \le v_{\mathsf{new}} = v_{\mathsf{cur}} + \Delta c \le \mathsf{Limit}$$

② **Transfer validity**

☞ $\sum_{i=1}^{N} \Delta v_i = 0$

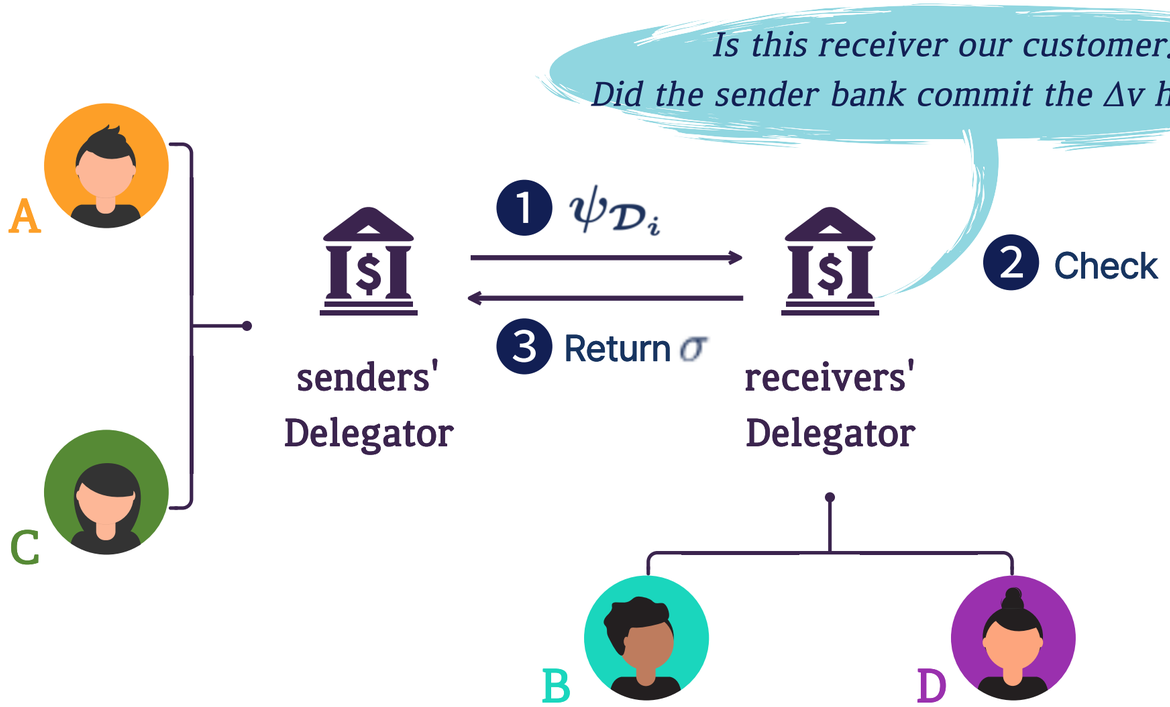
☞ $\mathsf{agg}_v = \sum_{i=1}^{N} \zeta^i \cdot (v_{\mathsf{cur},i} + \Delta v_i) + \zeta^{N+i} \cdot \Delta v_i$

☞ $\mathsf{agg}_o = \sum_{i=1}^{N} \zeta^i \cdot (o_{\mathsf{cur},i} + \Delta o_i) + \zeta^{N+i} \cdot \Delta o_i$

$$\pi_{\mathsf{cp}} \leftarrow \mathbf{\Pi}_{\mathsf{cp}}.\mathsf{Prove}(\mathsf{ek}, \vec{x}; \vec{w})$$

04 Construction

- Step 2. Prevent Cheating Prover



key point

2-1. Check user-specific information

☞ $\Delta v_j \leq 0$

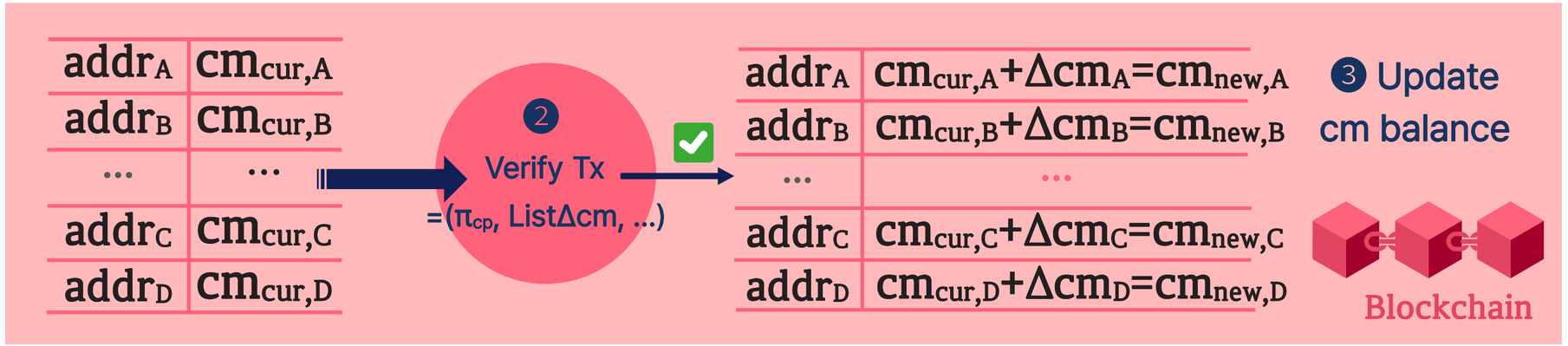
☞ $\mathsf{Ped.VerCom}(\mathsf{ck}, \mathsf{List}_{\Delta cm}[j], \Delta v_j, \Delta o_j) = \mathsf{true}$

2-2. Return signature

$$\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, \pi_{\mathsf{cp}}, \mathsf{List}_{\mathsf{data}})$$

04 Construction

- Step 3. Completion of transaction
  - Verify proof and update smart contract and database.



05 Construction

Auditability : Specification of stepwise audit procedures.

Aggregate

	address	balance	opening
1	0xuser1	30	o_1
2	0xuser2	200	o_2
3	0xuser3	50	o_3

$L = \sum v_j$ $\pi = \sum o_j$

1. Submit the sum of all users' total balances & opening values stored in the database.

2. Aggregate the total commitments in the smart contract.

Blockchain Smart Contract

$addr_{user1}$	$cm_{cur,user1}$
$addr_{user2}$	$cm_{cur,user2}$
$addr_{user3}$	$cm_{cur,user3}$

$cm_L = \sum cm_j$

3. Verify the proof

$cm_L = Ped.Com(L; \pi)$

Complete

Trace Tx logs