k-out-of-n Proofs and Applications to Privacy-Preserving Cryptocurrencies

Min Zhang Yu Chen Xiyuan Fu

Shandong University, China

May 14, 2026

山东大学
SHANDONG UNIVERSITY

https://eprint.iacr.org/2025/884

Outline

Outline

Transaction Privacy in Payment System

cannot learn the transfer amount

?

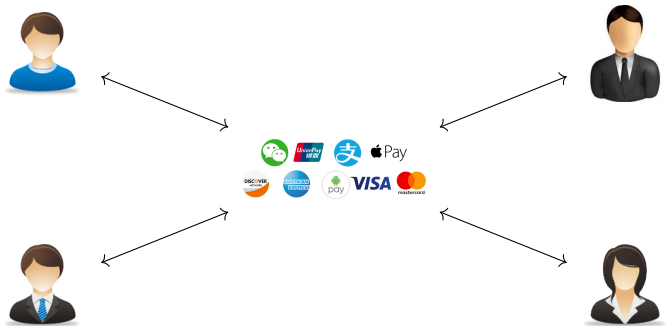
1024

?

?

cannot learn the identities of sender and receiver

Centralized Payment System



- txs are kept on a private ledger only known to the center
- the center is in charge of checking validity as well as protecting privacy

Decentralized Payment System (Blockchain-based Cryptocurrencies)

- txs are kept on a global distributed public ledger — the blockchain
- to ensure public verifiability, Bitcoin (UTXO model) and Ethereum (account-based model) simply expose all tx information in public $\rightsquigarrow$ no privacy

UTXO vs. Account-Based Model

Table: UTXO Model vs. Account-Based Model

Aspect	UTXO Model	Account-Based Model
Concept	track unspent outputs (like cash)	track account balance (like bank)
Privacy	new add. per tx enhance privacy	add. reuse reduce privacy
Scalability	parallel validation	sequential state updates
Functionality	simple value transfers	Turing-complete smart contracts
Complexity	manage multiple UTXOs	manage a single account

- Compared to UTXO model, account-based model offers superior **functionality** (DeFi/NFTs/dApps), but attaining privacy is more **challenging** due to state persistence.

In this work, we focus on account-based cryptocurrencies.

Confidential Transactions

- Chen et al. [CMTA20]: a framework of confidential ABC as well as an efficient instantiation — PGC.
- Bünz et al. [BAZB20]: a concrete confidential ABC scheme — Zether.

Evolution of Privacy-Preserving Account-Based Cryptocurrencies (PPABC)

Confidential Transactions

- Chen et al. [CMTA20]: a framework of confidential ABC as well as an efficient instantiation — PGC.
- Bünz et al. [BAZB20]: a concrete confidential ABC scheme — Zether.

Acquiring Anonymity

- Bünz et al.'s attempt: achieve n-anonymity via adding $n-2$ extra decoy accounts + the Groth-Kohlweiss 1-out-of-n proof [GK15].
- Diamond [Dia21]: 1-out-of-n proof is insufficint $\rightsquigarrow$ develop many-out-of-many proof to build Anonymous Zether with n-anonymity on top of Zether

Other Works

- Madathil and Scafuro [MS23]: PriFHEte with full anonymity based on FHE, but highly impractical due to heavy miner-side computation
- Guo et al. [GKPH24]: PriDe CT supports multi-receiver transfer, but completely compromises sender anonymity

A Closer Look at Anonymous Zether

Anonymous Zether represents the SOTA of PPABC. We sketch it as below:

- Each account is associated with an ElGamal PKE key pair and an encrypted balance, with public key acting as the account address.
- To generate a transaction, the sender:
  1. form an n-sized anonymity set $(pk_1, \ldots, pk_n)$ consisting of the sender's, receiver's and $n-2$ decoys' accounts pk;
  2. decide a vector of transfer amounts $(v_1, \ldots, v_n)$, where v_i denotes the balance change of account pk_i: sender's v_i is non-positive, receiver's v_i is non-negative, and decoys' v_i are 0;
  3. encrypt each transfer amount v_j under public key pk_j with reused randomness ρ;
  4. generate a proof to attest the legality of the encrypted transaction.

pk_1	pk_2	pk_3	pk_4	
$C_1 = (g^\rho, pk_1^\rho g^0)$	$C_2 = (g^\rho, pk_2^\rho g^{-v})$	$C_3 = (g^\rho, pk_3^\rho g^v)$	$C_4 = (g^\rho, pk_4^\rho g^0)$	π
decoy	sender	receiver	decoy	

A Closer Look at Anonymous Zether

Anonymous Zether represents the SOTA of PPABC. We sketch it as below:

- Each account is associated with an ElGamal PKE key pair and an encrypted balance, with public key acting as the account address.
- To generate a transaction, the sender:
  1. form an n-sized anonymity set $(pk_1, \ldots, pk_n)$ consisting of the sender's, receiver's and $n-2$ decoys' accounts pk;
  2. decide a vector of transfer amounts $(v_1, \ldots, v_n)$, where v_i denotes the balance change of account pk_i: sender's v_i is non-positive, receiver's v_i is non-negative, and decoys' v_i are 0;
  3. encrypt each transfer amount v_j under public key pk_j with reused randomness ρ;
  4. generate a proof to attest the legality of the encrypted transaction.

pk_1	pk_2	pk_3	pk_4	
$C_1 = (g^\rho, pk_1^\rho g^0)$	$C_2 = (g^\rho, pk_2^\rho g^{-v})$	$C_3 = (g^\rho, pk_3^\rho g^v)$	$C_4 = (g^\rho, pk_4^\rho g^0)$	π
decoy	sender	receiver	decoy	

However, Anonymous Zether suffers from three shortcomings!

Shortcoming 1: Weak Security Model

Current model fails to capture insider attacks, notably, rogue-key attack (RKA)
$\Leftarrow \mathcal{A}$ is not allowed to register accounts using dishonestly generated public keys

decoy	sender	receiver	decoy
pk_1 $C_1 = (g^\rho, pk_1^\rho g^0)$	pk_2 $C_2 = (g^\rho, pk_2^\rho g^{-v})$	pk_3 $C_3 = (g^\rho, pk_3^\rho g^v)$	pk_4 $C_4 = (g^\rho, pk_4^\rho g^0)$

launch rogue key attacks: sample $\alpha \in \mathbb{Z}_p$, set $pk_1 := pk_i^\alpha$
distinguish sender/receiver from decoy by checking if $C_{1,R} = C_{i,R}^\alpha$

Shortcoming 1: Weak Security Model

Current model fails to capture insider attacks, notably, rogue-key attack (RKA)
$\Leftarrow \mathcal{A}$ is not allowed to register accounts using dishonestly generated public keys

decoy	sender	receiver	decoy
pk_1	pk_2	pk_3	pk_4
$C_1 = (g^\rho, pk_1^\rho g^0)$	$C_2 = (g^\rho, pk_2^\rho g^{-v})$	$C_3 = (g^\rho, pk_3^\rho g^v)$	$C_4 = (g^\rho, pk_4^\rho g^0)$

launch rogue key attacks: sample $\alpha \in \mathbb{Z}_p$, set $pk_1 := pk_i^\alpha$
distinguish sender/receiver from decoy by checking if $C_{1,R} = C_{i,R}^\alpha$

vulnerability stems from randomness reusing $\rightsquigarrow$ crucial for Anon. Zether's design

Shortcoming 1: Weak Security Model

Current model fails to capture insider attacks, notably, rogue-key attack (RKA)
$\Leftarrow \mathcal{A}$ is not allowed to register accounts using dishonestly generated public keys

decoy	sender	receiver	decoy
pk_1	pk_2	pk_3	pk_4
$C_1 = (g^\rho, pk_1^\rho g^0)$	$C_2 = (g^\rho, pk_2^\rho g^{-v})$	$C_3 = (g^\rho, pk_3^\rho g^v)$	$C_4 = (g^\rho, pk_4^\rho g^0)$

launch rogue key attacks: sample $\alpha \in \mathbb{Z}_p$, set $pk_1 := pk_i^\alpha$
distinguish sender/receiver from decoy by checking if $C_{1,R} = C_{i,R}^\alpha$

vulnerability stems from randomness reusing $\rightsquigarrow$ crucial for Anon. Zether's design

Mitigation: require participants to provide PoK of their secret keys during registration

1. incur extra overhead for key verification $\rightsquigarrow$ unsuitable for cryptocurrencies
2. impose artificial restriction on $\mathcal{A} \rightsquigarrow$ undermine the security model
3. fail to defend against a wide range of insider attacks

Shortcoming 1: Weak Security Model

Current model fails to capture insider attacks, notably, rogue-key attack (RKA)
$\Leftarrow \mathcal{A}$ is not allowed to register accounts using dishonestly generated public keys

decoy	sender	receiver	decoy
pk_1 $C_1 = (g^\rho, pk_1^\rho g^0)$	pk_2 $C_2 = (g^\rho, pk_2^\rho g^{-v})$	pk_3 $C_3 = (g^\rho, pk_3^\rho g^v)$	pk_4 $C_4 = (g^\rho, pk_4^\rho g^0)$

launch rogue key attacks: sample $\alpha \in \mathbb{Z}_p$, set $pk_1 := pk_i^\alpha$
distinguish sender/receiver from decoy by checking if $C_{1,R} = C_{i,R}^\alpha$

Technical Challenge 1

How to defend against a wide range of insider attacks (including rogue-key attack)?

Shortcoming 2: Limited Anonymity

In Anonymous Zether, many-out-of-many proof requires that sender index ℓ_0 and receiver index ℓ_1 reside in orbits of opposite parity (i.e., $\ell_0 \not\equiv \ell_1 \bmod 2$).

decoy	sender	receiver	decoy
pk_1 $C_1 = (g^\rho, pk_1^\rho g^0)$	pk_2 $C_2 = (g^\rho, pk_2^\rho g^{-v})$	pk_3 $C_3 = (g^\rho, pk_3^\rho g^v)$	pk_4 $C_4 = (g^\rho, pk_4^\rho g^0)$

Shortcoming 2: Limited Anonymity

In Anonymous Zether, many-out-of-many proof requires that sender index ℓ_0 and receiver index ℓ_1 reside in orbits of opposite parity (i.e., $\ell_0 \not\equiv \ell_1 \bmod 2$).

decoy	sender	receiver	decoy
pk_1 $C_1 = (g^\rho, pk_1^\rho g^0)$	pk_2 $C_2 = (g^\rho, pk_2^\rho g^{-v})$	pk_3 $C_3 = (g^\rho, pk_3^\rho g^v)$	pk_4 $C_4 = (g^\rho, pk_4^\rho g^0)$

Such constraint halves the anonymity guarantee, as it decreases the cardinality of the set of possible sender-receiver pairs from $n \cdot (n-1)$ to $n^2/2$.

- This deficit can be remedied by picking larger anonymity set, but comes with increased computation cost and transaction size.
- Diamond [Dia21] outlines a potential method to eliminate this constraint, but it requires linear proof size and substantial computation costs.

Shortcoming 2: Limited Anonymity

In Anonymous Zether, many-out-of-many proof requires that sender index ℓ_0 and receiver index ℓ_1 reside in orbits of opposite parity (i.e., $\ell_0 \not\equiv \ell_1 \bmod 2$).

decoy	sender	receiver	decoy
pk_1 $C_1 = (g^\rho, pk_1^\rho g^0)$	pk_2 $C_2 = (g^\rho, pk_2^\rho g^{-v})$	pk_3 $C_3 = (g^\rho, pk_3^\rho g^v)$	pk_4 $C_4 = (g^\rho, pk_4^\rho g^0)$

Technical Challenge 2

How to achieve the better anonymity guarantee without sacrificing efficiency?

Shortcoming 3: Inefficient Multi-receiver Transfer

Anonymous Zether only supports single-receiver transaction

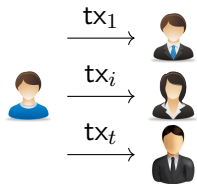
- If a sender want to pay $t > 1$ receivers, he must sequentially executing the basic protocol t times. This incurs prohibitive overhead for large t and reveals transaction patterns (sender lies in intersection!)

$pk_{1,1}$ $(g^{\rho_1}, pk_{1,1}^{\rho_1} g^0)$	pk_s $(g^{\rho_1}, pk_s^{\rho_1} g^{-v_1})$	$pk_{1,3}$ $(g^{\rho_1}, pk_s^{\rho_1} g^{v_1})$	$pk_{1,4}$ $(g^{\rho_1}, pk_{1,4}^{\rho_1} g^0)$
$pk_{2,1}$ $(g^{\rho_2}, pk_{2,1}^{\rho_2} g^0)$	$pk_{2,2}$ $(g^{\rho_2}, pk_{2,2}^{\rho_2} g^0)$	pk_s $(g^{\rho_2}, pk_s^{r_2} g^{-v_2})$	$pk_{2,4}$ $(g^{\rho_2}, pk_{2,4}^{\rho_2} g^{v_2})$
$pk_{3,1}$ $(g^{\rho_3}, pk_{3,1}^{\rho_3} g^{v_3})$	$pk_{3,2}$ $(g^{\rho_3}, pk_{3,2}^{\rho_3} g^0)$	$pk_{3,3}$ $(g^{\rho_3}, pk_{3,3}^{\rho_3} g^0)$	pk_s $(g^{\rho_3}, pk_s^{\rho_3} g^{-v_3})$
pk_s $(g^{\rho_4}, pk_s^{\rho_4} g^{-v_4})$	$pk_{4,2}$ $(g^{\rho_4}, pk_{4,2}^{\rho_4} g^{v_4})$	$pk_{4,3}$ $(g^{\rho_4}, pk_{4,3}^{\rho_4} g^0)$	$pk_{4,4}$ $(g^{\rho_4}, pk_{4,4}^{\rho_4} g^0)$

Shortcoming 3: Inefficient Multi-receiver Transfer

Anonymous Zether only supports single-receiver transaction

- If a sender want to pay $t > 1$ receivers, he must sequentially executing the basic protocol t times. This incurs prohibitive overhead for large t and reveals transaction patterns (sender lies in intersection!)



Technical Challenge 3

How to enable efficient built-in multi-receiver transaction?

Motivation

Motivated by the state of affairs, this work aims to develop a well-rounded PPABC that featuring strong security, rich functionality, and high efficiency.

Motivation

Motivated by the state of affairs, this work aims to develop a well-rounded PPABC that featuring strong security, rich functionality, and high efficiency.

We present the first account-based cryptocurrency simultaneously achieving:
strong privacy + efficient multi-receiver support

Systems	t	Complexity			Security		
		Tx Generation	Tx Verification	Tx size	Insider	C.	A.
Zether	$=1$	$O(\nu)$	$O(\nu)$	$O_\lambda(\log \nu)$	—	✓	✗
PGC	$=1$	$O(\nu)$	$O(\nu)$	$O_\lambda(\log \nu)$	—	✓	✗
Anon.Zether	$=1$	$O(n \log n + \nu)$	$O(n \log n + \nu)$	$O_\lambda(n + \log n + \log \nu)$	✗	✓	✗
PriDe CT	≥ 1	$O(n \log n + n\nu)$	$O(n + n\nu)$	$O_\lambda(n + \log n + \log(n\nu))$	✗	✓	✗
Anon.PGC	≥ 1	$O(n \log n + t\nu)$	$O(n + t\nu)$	$O_\lambda(n + t \log n + \log(t\nu))$	✓	✓	✓

λ is the security parameter, $\nu = \log v_{\mathsf{max}}$ is the bit-length of max amounts (e.g., 32 in practice).
t is the number of intended receivers, and n is the size of the anonymity set.

Outline

Syntax of PPABC

trusted/transparent

$\mathsf{Setup}(1^\lambda) \to pp$

Syntax of PPABC

trusted/transparent

$\mathsf{Setup}(1^\lambda) \to pp$

$\mathsf{CreateAccount}(\tilde{v})$

$\downarrow$

$pk_s, sk_s, \tilde{C}_s = B[pk_s]$

Syntax of PPABC

trusted/transparent

$\mathsf{Setup}(1^\lambda) \to pp$

$\mathsf{CreateAccount}(\tilde{v})$

$\downarrow$

$pk_s, sk_s, \tilde{C}_s = B[pk_s]$

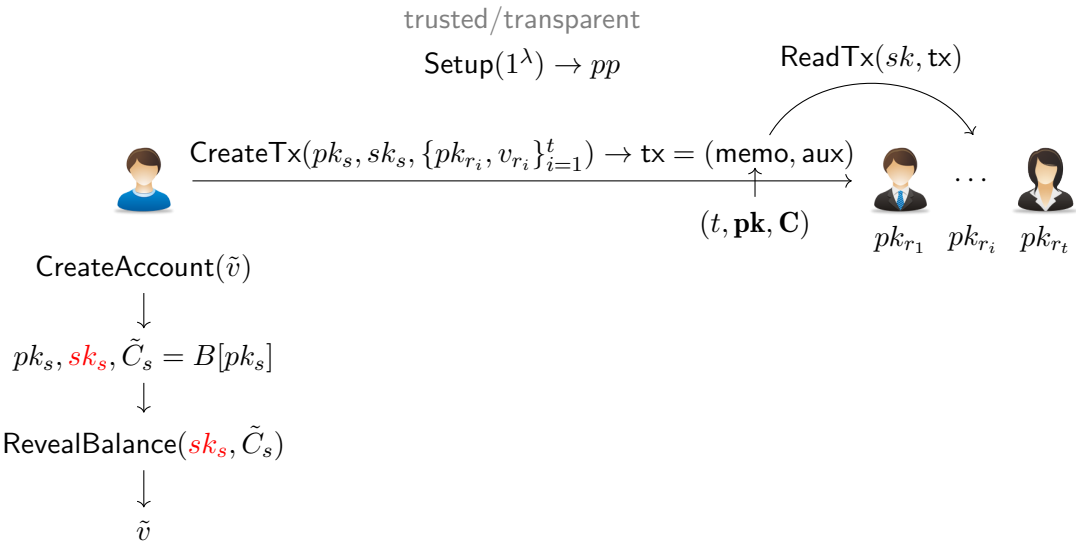
$\downarrow$

$\mathsf{RevealBalance}(sk_s, \tilde{C}_s)$

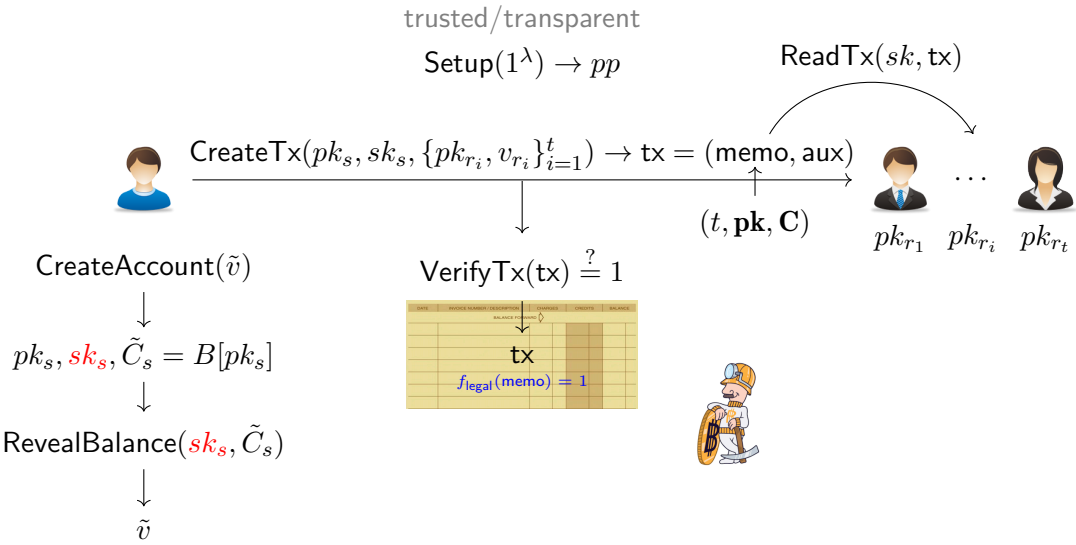
$\downarrow$

$\tilde{v}$

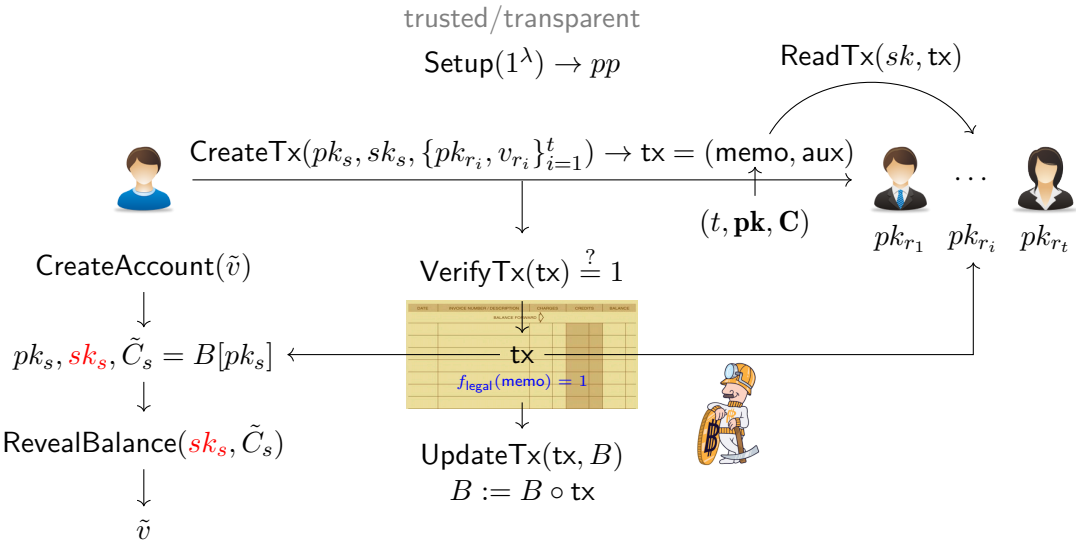
Syntax of PPABC



Syntax of PPABC



Syntax of PPABC



Desired Feature and Security

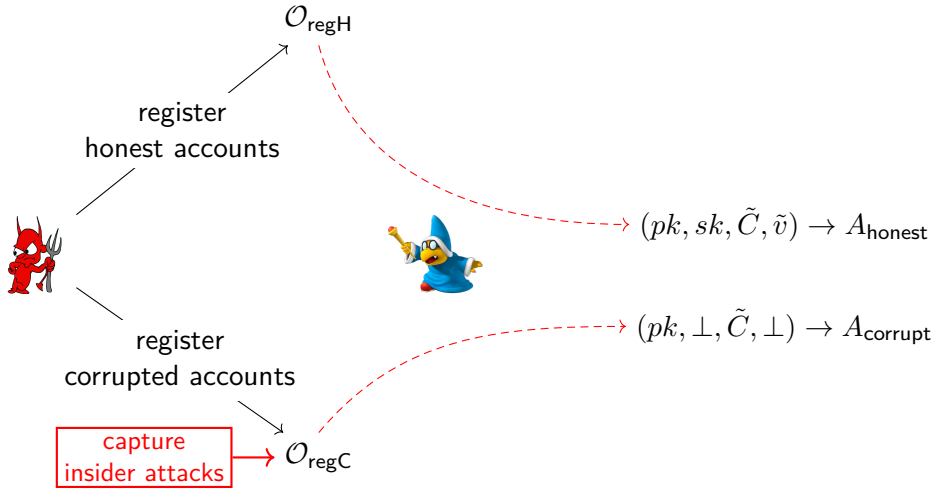
Verifiability	validity of txs are publicly verifiable
Updatability	account states are publicly updatable
Authenticity	only owner can generate tx; nobody else can forge
Soundness	nobody can generate an illegal tx that passes validity check
Confidentiality	external observer does not learn the transfer amount
Anonymity	external observer does not learn the identities of sender and recievers

Security Model of PPABC

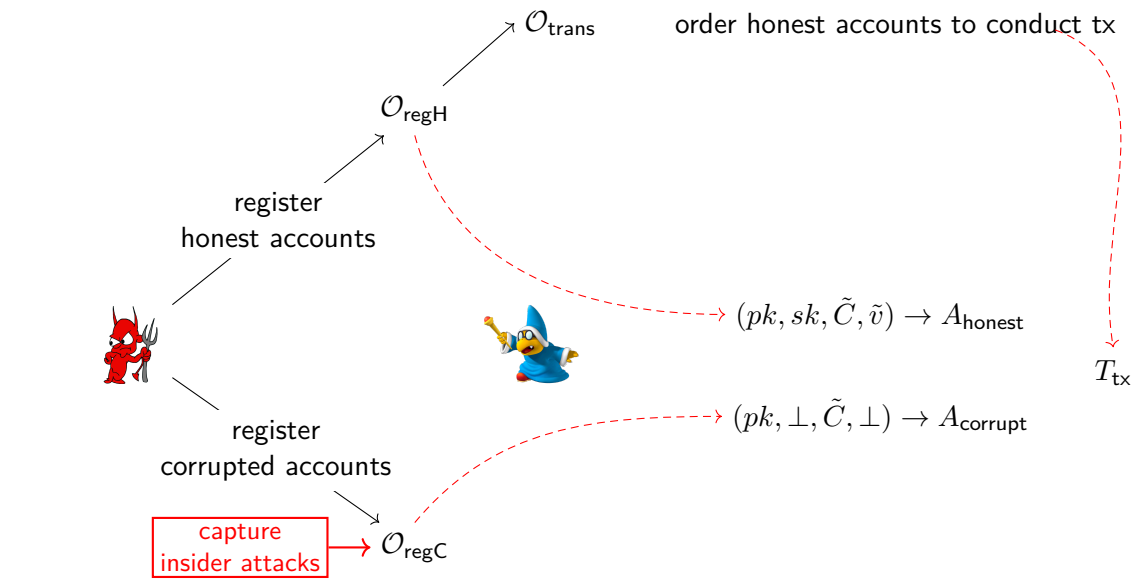
Formalizing security model for PPABC turns out to be tricky

- strong enough to capture all possible real-world attacks
- remain clean and handy to use for security reduction

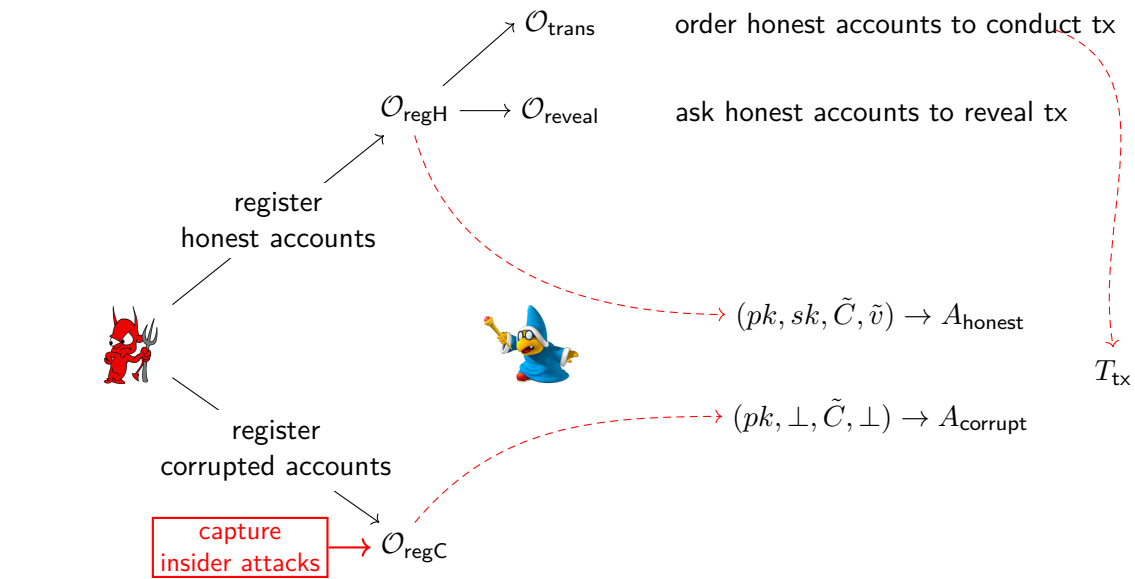
Formal Security Model: Oracles



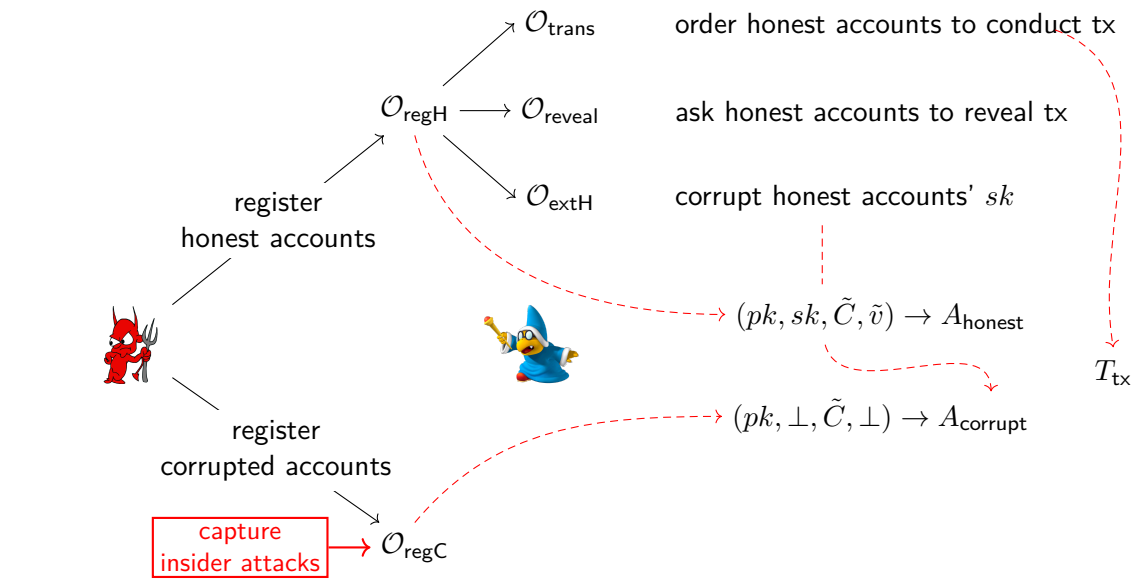
Formal Security Model: Oracles



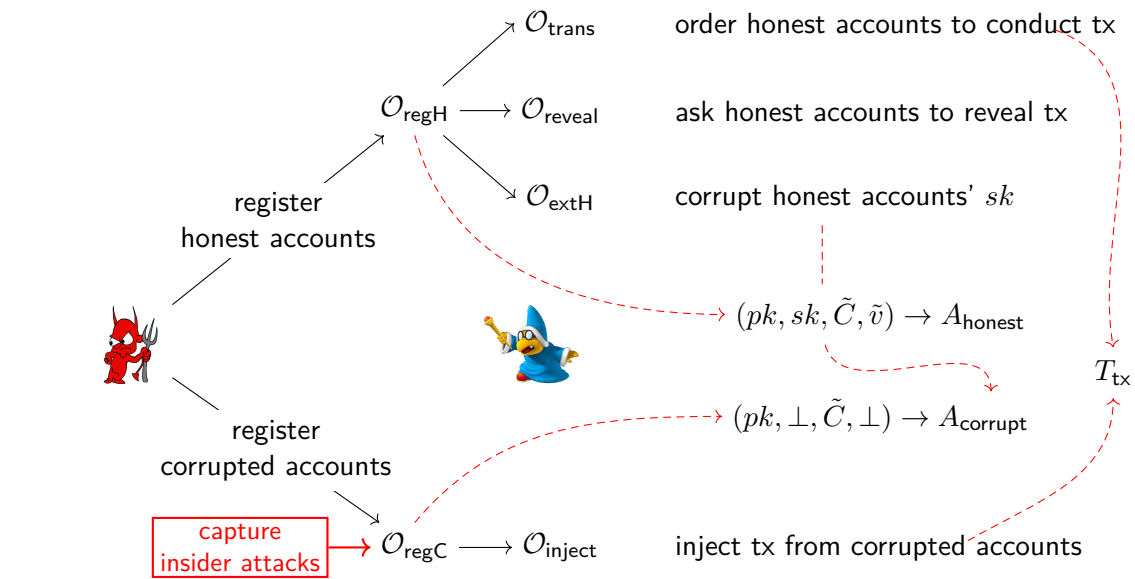
Formal Security Model: Oracles



Formal Security Model: Oracles



Formal Security Model: Oracles



Formal Security Model

We estabilish the security of PPABC upon three orthogonal pillars.

1. Authenticity $\Rightarrow$ Theft-Resistance
   - Only the rightful owners can authorize transactions.
2. Soundness $\Rightarrow$ Policy-Compliance
   - Forcing all transactions to comply with global legality policy.
3. Ledger-Indistinguishability $\Rightarrow$ Privacy
   - Hiding the transaction graph: both edge (sender-receiver) and weight (amounts)

Next, we present the three game-based definitions (giving $\mathcal{A}$ access the above oralces).

Formal Security Model: Authenticity

Capture theft-resistance: Any PPT adversary without the secret key for pk cannot generate a valid transaction with account pk as sender, i.e., stealing pk's coins.

Formal Security Model: Authenticity

Capture theft-resistance: Any PPT adversary without the secret key for pk cannot generate a valid transaction with account pk as sender, i.e., stealing pk's coins.

$\mathcal{O}$

pp

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\mathsf{tx}^* = (\mathsf{memo}^*, \mathsf{aux}^*)$

$\mathsf{memo}^* = (t^*, (pk_1^*, \ldots, pk_n^*), (C_1^*, \ldots, C_n^*))$

$\mathcal{A}$ wins if

$\mathsf{tx}^* \notin T_{\mathsf{tx}} \wedge \mathsf{Verify}(\mathsf{tx}^*) = 1 \wedge \exists \ell_s \in [n]$ s.t.

$pk_{\ell_s}^* \in A_{\mathsf{honest}} \wedge \mathsf{ReadTx}(sk_{\ell_s}^*, C_{\ell_s}^*) < 0$

Formal Security Model: Soundness

Capture policy-compliance: Any PPT adversary (even knowing the secret key of sender) cannot generate a valid transaction with illegal memo.

Formal Security Model: Soundness

Capture policy-compliance: Any PPT adversary (even knowing the secret key of sender) cannot generate a valid transaction with illegal memo.

$\mathcal{O}$

pp

$\mathsf{tx}^* = (\mathsf{memo}^*, \mathsf{aux}^*)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\mathcal{A}$ wins if

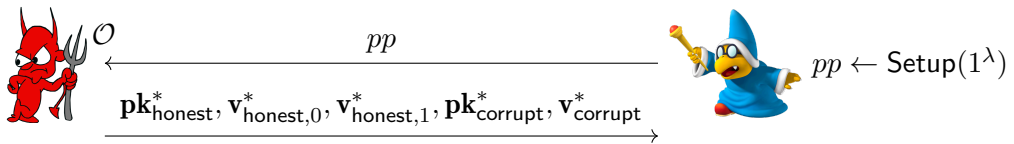
$\mathsf{Verify}(\mathsf{tx}^*) = 1 \wedge f_{\mathsf{legal}}(\mathsf{memo}^*) \neq 1$

Formal Security Model: Ledger-Indistinguishability

Capture both confidentiality and anonymity: Any PPT adversary without secret key of an honest account learns nothing about its transfer amount (revealing account's role).

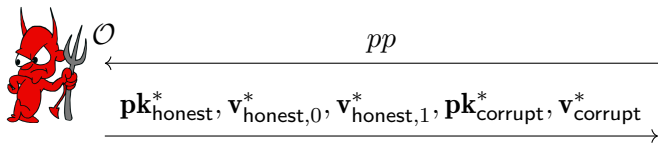
Formal Security Model: Ledger-Indistinguishability

Capture both confidentiality and anonymity: Any PPT adversary without secret key of an honest account learns nothing about its transfer amount (revealing account's role).



Formal Security Model: Ledger-Indistinguishability

Capture both confidentiality and anonymity: Any PPT adversary without secret key of an honest account learns nothing about its transfer amount (revealing account's role).



- $\mathbf{v}_{\text{honest},0}$ and $\mathbf{v}_{\text{honest},1}$ contain exactly one negative element
- denote the corresponding accounts in $\mathbf{pk}^*_{\text{honest}}$ as $pk^*_{s_\beta}$;

Formal Security Model: Ledger-Indistinguishability

Capture both confidentiality and anonymity: Any PPT adversary without secret key of an honest account learns nothing about its transfer amount (revealing account's role).

$\mathcal{O}$

pp

$\mathbf{pk}^*_{\text{honest}}, \mathbf{v}^*_{\text{honest},0}, \mathbf{v}^*_{\text{honest},1}, \mathbf{pk}^*_{\text{corrupt}}, \mathbf{v}^*_{\text{corrupt}}$

$\mathsf{tx}^* \leftarrow \mathsf{CreateTx}^*(t^*, sk^*_{s_\beta}, \mathbf{pk}^*, \mathbf{v}^*_\beta)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\beta \xleftarrow{R} \{0,1\}$

$\mathbf{pk}^* = (\mathbf{pk}^*_{\text{honest}}, \mathbf{pk}^*_{\text{corrupt}})$

$\mathbf{v}^* = (\mathbf{v}^*_{\text{honest},\beta}, \mathbf{v}^*_{\text{corrupt}})$

shuffle both $\mathbf{pk}^*$ and $\mathbf{v}^*$

- $\mathbf{v}_{\text{honest},0}$ and $\mathbf{v}_{\text{honest},1}$ contain exactly one negative element
- denote the corresponding accounts in $\mathbf{pk}^*_{\text{honest}}$ as $pk^*_{s_\beta}$;

Formal Security Model: Ledger-Indistinguishability

Capture both confidentiality and anonymity: Any PPT adversary without secret key of an honest account learns nothing about its transfer amount (revealing account's role).

$\mathcal{O}$

pp

$\mathbf{pk}^*_{\text{honest}}, \mathbf{v}^*_{\text{honest},0}, \mathbf{v}^*_{\text{honest},1}, \mathbf{pk}^*_{\text{corrupt}}, \mathbf{v}^*_{\text{corrupt}}$

$\mathsf{tx}^* \leftarrow \mathsf{CreateTx}^*(t^*, sk^*_{s_\beta}, \mathbf{pk}^*, \mathbf{v}^*_\beta)$

β'

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\beta \xleftarrow{R} \{0,1\}$

$\mathbf{pk}^* = (\mathbf{pk}^*_{\text{honest}}, \mathbf{pk}^*_{\text{corrupt}})$

$\mathbf{v}^* = (\mathbf{v}^*_{\text{honest},\beta}, \mathbf{v}^*_{\text{corrupt}})$

shuffle both $\mathbf{pk}^*$ and $\mathbf{v}^*$

$\mathcal{A}$ wins if $\beta = \beta'$

Formal Security Model: Ledger-Indistinguishability

Capture both confidentiality and anonymity: Any PPT adversary without secret key of an honest account learns nothing about its transfer amount (revealing account's role).

$\mathcal{O}$

pp

$\mathbf{pk}^*_{\text{honest}}, \mathbf{v}^*_{\text{honest},0}, \mathbf{v}^*_{\text{honest},1}, \mathbf{pk}^*_{\text{corrupt}}, \mathbf{v}^*_{\text{corrupt}}$

$\mathsf{tx}^* \leftarrow \mathsf{CreateTx}^*(t^*, sk^*_{s_\beta}, \mathbf{pk}^*, \mathbf{v}^*_\beta)$

β'

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\beta \xleftarrow{R} \{0,1\}$

$\mathbf{pk}^* = (\mathbf{pk}^*_{\text{honest}}, \mathbf{pk}^*_{\text{corrupt}})$

$\mathbf{v}^* = (\mathbf{v}^*_{\text{honest},\beta}, \mathbf{v}^*_{\text{corrupt}})$

shuffle both $\mathbf{pk}^*$ and $\mathbf{v}^*$

$\mathcal{A}$ wins if $\beta = \beta'$

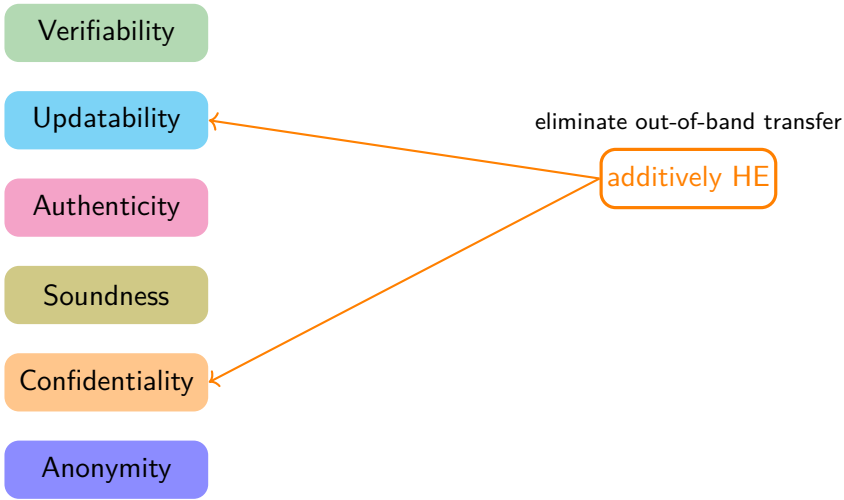
To prevent trivial attacks, $\mathcal{A}$ is subject to the following restrictions:

1. t^* is identical for both $\mathbf{v}^*_0$ and $\mathbf{v}^*_1$ (cause t^* will be public in aux);
2. both $(\mathbf{pk}^*, \mathbf{v}^*_0)$ and $(\mathbf{pk}^*, \mathbf{v}^*_1)$ constitute legal tx
3. for each account $pk_i \in \mathbf{pk}^*_{\text{honest}}$ and both $\mathbf{v}^*_\beta$, there is no overdraft
4. In Phase 2, $\mathcal{A}$ can continue to corrupt $pk_i \in \mathbf{pk}^*_{\text{honest}}$ as long as $v^*_{i,0} = v^*_{i,1}$

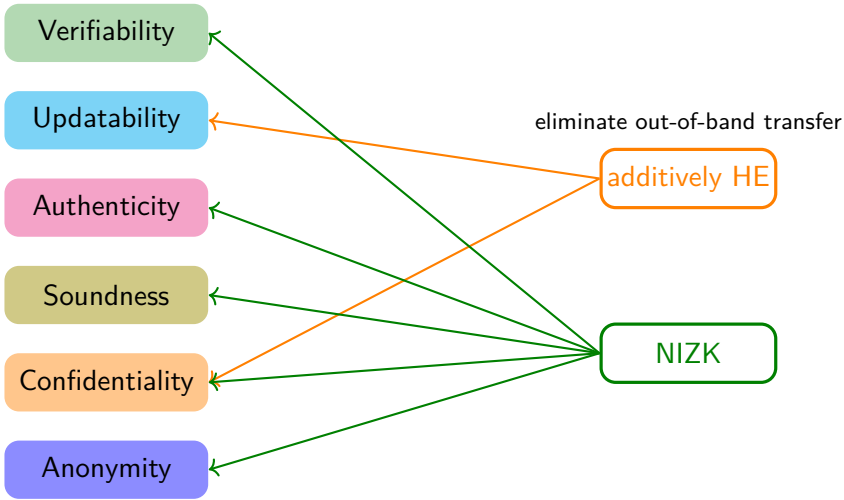
Choice of Building Blocks

- Verifiability
- Updatability
- Authenticity
- Soundness
- Confidentiality
- Anonymity

Choice of Building Blocks



Choice of Building Blocks



Framework of PPABC: Building Blocks

PKE $=$ (Setup, KeyGen, Enc, Dec)

- IND-CPA security (single-user setting directly implies multi-user setting)
- additively homomorphic over $\mathbb{Z}_p$

w.l.o.g. KeyGen algorithm induces a hard relation indexed by pp_{pke}

$$\mathsf{R}_{\mathsf{key}} = \{(pk, sk) \mid pk = \mathsf{PKE.KeyGen}(pp_{\mathsf{pke}}; sk)\}$$

NIZK $=$ (Setup, CRSGen, Prove, Verify)

- labeled: prover and verifier can inject a public "label" (context string) into proof generation and verification algorithms, while adversary is unable to repurpose a proof under a label l to a different lable l'.
- simulation sound extractability
- adaptive zero-knowledge

Generic Construction of PPABC: 1/3

Setup(1^λ): generate pp for PPABC

- $pp_{\mathsf{pke}} \leftarrow \mathsf{PKE.Setup}(1^\lambda)$, $pp_{\mathsf{nizk}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, $crs \leftarrow \mathsf{NIZK.CRSGen}(pp_{\mathsf{nizk}})$;
- pick anonymity level n and the max number of coins v_{max} of the system.

$$\text{output } pp = (pp_{\mathsf{pke}}, pp_{\mathsf{nizk}}, crs, n, v_{\mathsf{max}})$$

CreateAccount($\tilde{v}$): create an account with initial balance $\tilde{v}$

- $(pk, sk) \leftarrow \mathsf{PKE.KeyGen}(pp_{\mathsf{pke}})$, pk serves as account address
- $\tilde{C} \leftarrow \mathsf{PKE.Enc}(pk, \tilde{v}; \rho)$

$(pk, \tilde{C})$ are the public account states, sk is the secret account state

RevealBalance($sk, \tilde{C}$): reveal the balance of an account

- $\tilde{v} \leftarrow \mathsf{PKE.Dec}(sk, \tilde{C})$

Generic Construction of PPABC: 2/3

CreateTx$(sk_s, pk_s, \{pk_{r_i}, v_{r_i}\}_{i=1}^{t})$: transfer $v_{r_i} > 0$ coins from pk_s to $(pk_{r_i}, \ldots, pk_{r_t})$.

1. generate memo $= (t, \mathbf{AnonSet}, \mathbf{C})$
   - randomly picks $(n - t - 1)$ accounts to form $\mathbf{AnonSet} = (pk_1, \ldots, pk_n)$ to include pk_s and $(pk_{r_1}, \ldots, pk_{r_t})$ where $pk_{\ell_s} = pk_s$ and $pk_{\ell_i} = pk_{r_i}$;
   - form corresponding transfer amount vector $\mathbf{v} = (v_1, \ldots, v_n)$ where $v_{\ell_i} = v_{r_i}$, $v_{\ell_s} = -\sum_{i=1}^{t} v_{r_i}$, $v_j = 0$ for $j \neq \{\ell_s, \ell_1, \ldots, \ell_t\}$;
   - encrypt $C_i \leftarrow \mathsf{PKE.Enc}(pk_i, v_i; \rho_i)$ using independent randomness.
2. invoke NIZK.Prove to generate π_{legal} for memo $\in L_{\mathsf{legal}}$ as aux info
3. output tx $=$ (memo, aux)

- Use independent randomness to break correlation among session keys of multi-receiver encryption $\rightsquigarrow$ solves technical challenge 1 about insider attacks
- Admit multi-receivers transaction $\rightsquigarrow$ solves technical challenge 3 about built-in multi-receiver support in syntax sense.

Generic Construction of PPABC: 2/3

$$L_{\mathsf{legal}} := (t, \mathbf{AnonSet}, \mathbf{C}) \mid \exists (sk, \ell_s, R, D, \mathbf{v}, \mathbf{r}) \text{ s.t.}$$

- $\forall i \in [n], C_i = \mathsf{Enc}(pk_i, v_i; \rho_i) \wedge \sum_{i=1}^{n} v_i = 0$ — L_{balanced}: balance
- $\ell_s \notin (R \cup D) \wedge R \cap D = \emptyset$ — L_{disjoint}: set disjointness
- $D \subset [n] \wedge |D| = n - t - 1 \wedge \forall j \in D, v_j = 0$ — L_{zero}: decoy integrity
- $R \subset [n] \wedge |R| = t \wedge \forall j \in S, v_j \in (0, v_{\mathsf{max}}]$ — L_{positive}: receiver validity
- $\ell_s \in [n] \wedge (pk_{\ell_s}, sk) \in \mathsf{R}_{\mathsf{key}} \wedge -v_{\ell_s} \in (0, v_{\mathsf{max}}] \wedge \mathsf{Dec}(sk, \tilde{C}_{\ell_s}) + v_{\ell_s} \in [0, v_{\mathsf{max}}]$ — L_{solvent}: sender solvency

Generic Construction of PPABC: 2/3

$$L_{\mathsf{legal}} := (t, \mathbf{AnonSet}, \mathbf{C}) \mid \exists (sk, \ell_s, R, D, \mathbf{v}, \mathbf{r}) \text{ s.t.}$$

- $\forall i \in [n], C_i = \mathsf{Enc}(pk_i, v_i; \rho_i) \wedge \sum_{i=1}^{n} v_i = 0$ — L_{balanced}: balance
- $\ell_s \notin (R \cup D) \wedge R \cap D = \emptyset$ — L_{disjoint}: set disjointness
- $D \subset [n] \wedge |D| = n - t - 1 \wedge \forall j \in D, v_j = 0$ — L_{zero}: decoy integrity
- $R \subset [n] \wedge |R| = t \wedge \forall j \in S, v_j \in (0, v_{\mathsf{max}}]$ — L_{positive}: receiver validity
- $\ell_s \in [n] \wedge (pk_{\ell_s}, sk) \in \mathsf{R}_{\mathsf{key}} \wedge -v_{\ell_s} \in (0, v_{\mathsf{max}}] \wedge \mathsf{Dec}(sk, \tilde{C}_{\ell_s}) + v_{\ell_s} \in [0, v_{\mathsf{max}}]$ — L_{solvent}: sender solvency

L_{positive} enforces strictly positive transfers to intended receivers

- align with practice $\rightsquigarrow$ diminish empty transfers
- binding property of PKE $\rightsquigarrow$ ℓ_s, R and D must be disjoint $\Rightarrow$ make proof of L_{disjoint} redundant (was exactly the most complex and restricted part in Anonymous Zether) $\rightsquigarrow$ solves technical challenge 2 about limited anonymity

L_{legal} is thus simplified to $L_{\mathsf{balanced}} \wedge L_{\mathsf{zero}} \wedge L_{\mathsf{positive}} \wedge L_{\mathsf{solvent}}$

- When $t = 1$, L_{legal} is further simplified to $L_{\mathsf{balanced}} \wedge L_{\mathsf{zero}} \wedge L_{\mathsf{solvent}}$

Generic Construction of PPABC: 3/3

VerifyTx(tx): on input tx $= (\mathsf{memo}, \mathsf{aux})$

- parses memo $= (t, \mathbf{AnonSet}, \mathbf{C})$ and aux $= \pi_{\mathsf{legal}}$;
- check if tx is fresh;
- check if NIZK.Verify$(crs, \mathsf{memo}, \pi_{\mathsf{legal}}) = 1$.

UpdateTx(tx, B): on input a validated transaction tx $= (\mathsf{memo}, \mathsf{aux})$

- parses memo $= (t, \mathbf{AnonSet}, \mathbf{C})$;
- for each account $pk_i \in$ AnonSet; updates its encrypted balance $\tilde{C}_i = \tilde{C}_i + C_i$.

ReadTx(sk, tx): on input sk and tx $= (\mathsf{memo}, \mathsf{aux})$

- parse memo $= (t, (pk_1, \ldots, pk_n), (C_1, \ldots, C_n))$;
- determine i such that sk is the secret key of pk_i;
- $v \leftarrow$ PKE.Dec(sk, C_i).

Security Proof

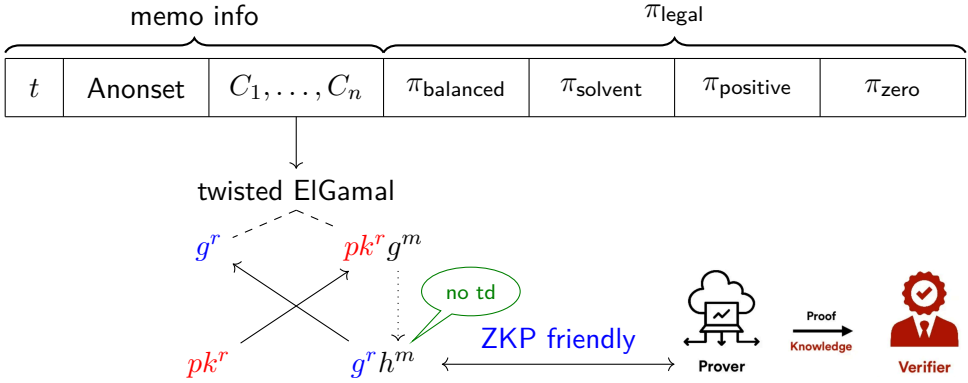
Theorem 1

If the PKE is IND-CPA secure, and the labeled NIZK is simulation sound extractable and adaptively zero-knowledge, then the PPABC framework satisfies authenticity, soundness and ledger-indistinguishability.

- hardness of R_{key} (implied by IND-CPA security of PKE) + simulation sound extractability and adaptive zero-knowledge of NIZK $\Rightarrow$ authenticity
- simulation sound extractability of NIZK $\Rightarrow$ adaptive soundness of NIZK $\Rightarrow$ soundness
- IND-CPA security of PKE + adaptive zero-knowledge of NIZK $\Rightarrow$ ledger indistinguishability

Outline

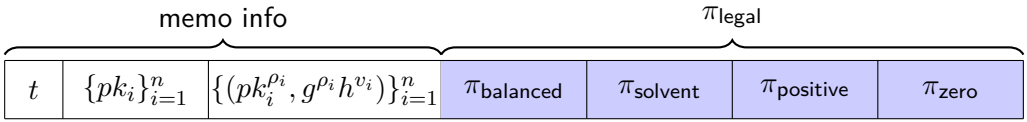
Encryption Component: Twisted ElGamal



- the right half of ciphertext is exactly a Pedersen commitment
- even nice, ciphertexts under different public keys share the same com key

seamlessly soldering with ZKPs that accept Pedersen commitments as instances, notably, the (aggregated) Bulletproof and Σ-protocols for linear relations.

NIZK Component: π_{legal}



$$L_{\mathsf{legal}} := (t, \{pk_i\}_{i=1}^n, \{(pk_i^{\rho_i}, g^{\rho_i} h^{v_i})\}_{i=1}^n) \mid \exists (sk, \ell_s, R, D, v_i, \rho_i) \text{ s.t.}$$

- $\forall i \in [n], C_{i,L} = pk_i^{\rho_i} \wedge C_{i,R} = g^{\rho_i} h^{v_i} \wedge \sum_{i=1}^n v_i = 0$ L_{balanced}
- $D \subset [n] \wedge |D| = n - t - 1 \wedge \forall j \in D, C_{j,R} = g^{\rho_j} h^0$ L_{zero}
- $R \subset [n] \wedge |R| = t \wedge \forall j \in R, C_{j,R} = g^{\rho_j} h^{v_j} \wedge v_j \in (0, v_{\mathsf{max}}]$ L_{positive}
- $\ell_s \in [n] \wedge C_{\ell_s,R} = g^{r_{\ell_s}} h^{v_{\ell_s}} \wedge -v_{\ell_s} \in (0, v_{\mathsf{max}}] \wedge pk_{\ell_s} = g^{sk} \wedge v^* \in [0, v_{\mathsf{max}}]$ L_{solvent}

prove knowledge of openings for some k-subset among n public commitments without revealing which k-subset

NIZK Component: Choices of k-out-of-n Proofs

General-purpose ZKPs (e.g., zk-SNARKs): succinct proof size and fast verification, but incur prohibitive overhead in transforming algebra statement to arithmetic circuit.

Customized k-out-of-n Proofs

- General k-out-of-n Proofs: highly expressive, but the efficiency is poor $\rightsquigarrow$ scales at least linearly with both n and k.

- DL-based k-out-of-n proofs: high efficiency $\Rightarrow$ best match

Prior Works: DL-Based k-out-of-n Proofs

Breakthrough: Logarithmic Proof Size for $k = 1$

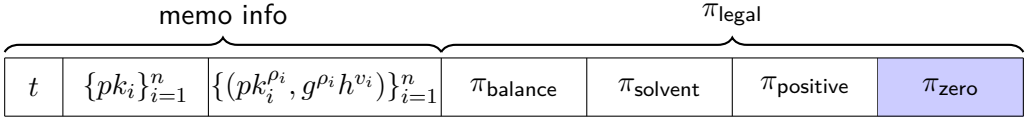
- Groth-Kohlweiss [GK15]: GK 1-out-of-n proof
  (quasi)-linear prover/verifer time, $O(\log n)$ proof size, but restricted to $k = 1$ case

Extensions to General k

- Diamond [Dia21]: many-out-of-many proof
  support $k \geq 1$, but the k secrets have to lie in one orbit of a public permutation
- Attema et al. [ACF21]: ACF k-out-of-n proof
  no restriction for secrets, but the prover/verifier time is $O(n(n-k))$
- Zheng et al. [ZGSX23]: any-out-of-many proof
  $O(n)$ prover/verifier time, $O(\log n)$ proof size, and no restriction for secrets.

Limit — support only homogeneous statements: under the same DL commitment key.

Legality Proof: $\pi_{\mathsf{balance}} \circ \pi_{\mathsf{solvent}} \circ \pi_{\mathsf{positive}} \circ \pi_{\mathsf{zero}}$



$$L_{\mathsf{zero}} := (C_{j,R})_{j=1}^n \mid \exists\ (D, (\rho_j)_{j \in D}) \text{ s.t.}$$

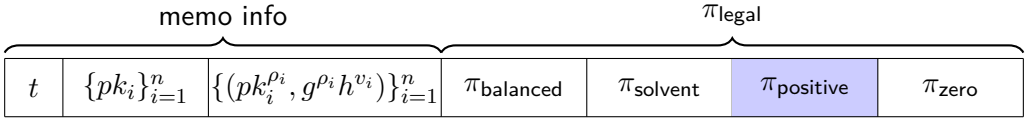
$$D \subset [n] \wedge |D| = n - t - 1 \wedge \forall j \in D, C_{j,R} = g^{\rho_j}$$

exactly $n - t - 1$ of n transfer amounts equal 0

This can be proved via DL-based Homogeneous k-out-of-n Proof

any-out-of-many proof [ZGSX23] is a good choice!

Legality Proof: $\pi_{\text{balance}} \circ \pi_{\text{solvent}} \circ \pi_{\text{positive}} \circ \pi_{\text{zero}}$



$$L_{\text{positive}} := (C_{i,R})_{i=1}^n \mid \exists\ R, (v_j, \rho_j)_{j \in R} \text{ s.t.}$$

$$R \subset [n] \wedge |R| = t \wedge \forall j \in R, C_{j,R} = g^{\rho_j} h^{v_j} \wedge \boxed{v_j \in (0, v_{\max}]}$$

exactly t of n transfer amounts are positive

This should be proved via DL-based Homogeneous k-out-of-n Range Proof:

No such proof exists in the literature $\rightsquigarrow$ requiring new design!

Homogeneous k-out-of-n Range Proof: Technical Challenge

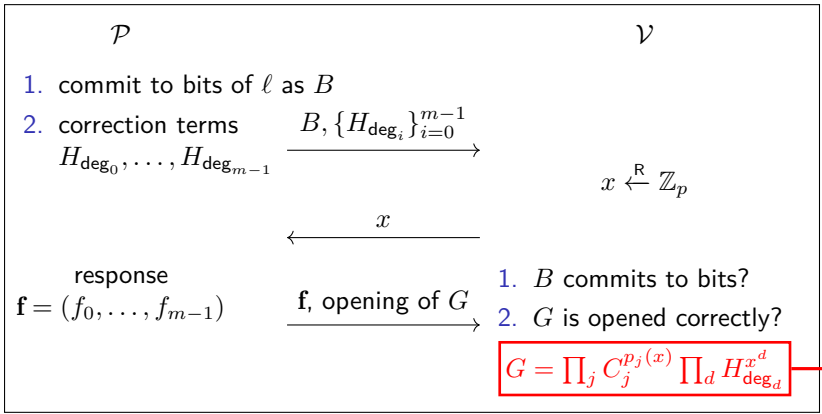
Homogeneous k-out-of-n Range Proof: Technical Challenge

How to feed the hidden commitments to range proof without revealing their indices?

Homogeneous k-out-of-n Range Proof: Starting Point

GK 1-out-of-n Proof [GK15]

$L_{\text{homo-1oon}} := (C_0, \ldots, C_{n-1}) \mid \exists(\ell, r) \text{ s.t. } \ell \in \{0, \ldots, n-1\} \wedge C_\ell = g^r h^0$

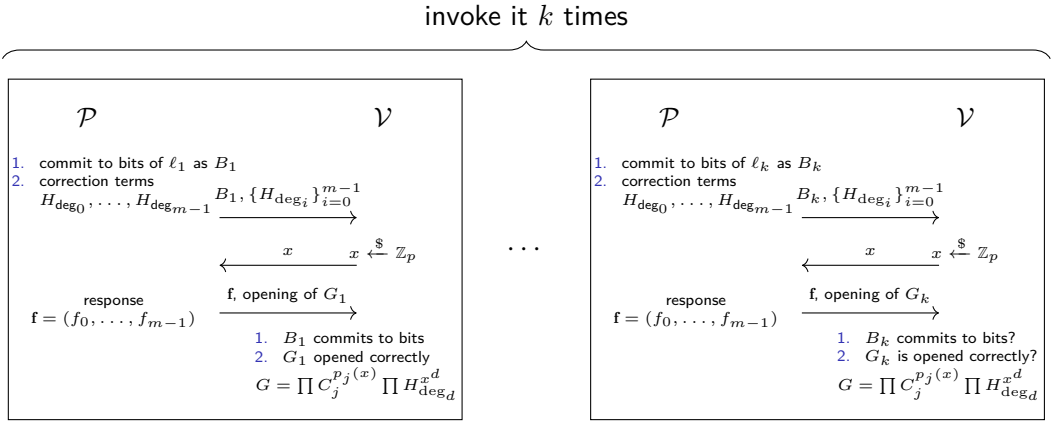


Pick-Out Mechanism
$G^* = G^{x^{-m}}$ is a randomization of C_ℓ
$m = \log n$

$p_j(x) = \delta_{\ell j} x^m + \cdots$
$\deg(p_j) = m$ iff $j = \ell$
$G = C_\ell^{x^m} \cdots$

- Pros: range proof for $C_\ell \Leftrightarrow$ range proof for $G^* \rightsquigarrow$ naturally range proof friendly
- Cons: only support $k = 1$.

Homogeneous k-out-of-n Range Proof: A Naive Method for Extending to $k \geq 1$

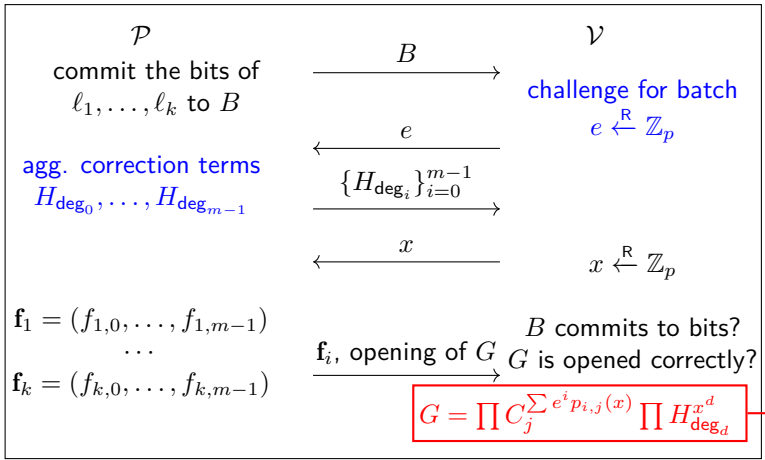


- Efficiency: both communication and computation complexities are linear in k

  prover time $= O(kn \log n)$ verifier time $= O(kn)$ proof size $= O(2k \log n)$
- Security: malicious prover could cheat with duplicated index during k invocations

Homogeneous k-out-of-n Range Proof: Solving Efficiency Issue

Idea: aggregate correction terms via random linear combination.



Batch Pick-Out Mechanism

$G^* = G^{x^{-m}}$ is a randomization of $\prod_i C_{\ell_i}^{e_i}$

$p_{i,j}(x) = \delta_{j\ell_i} x^m + \cdots$

$\mathsf{deg}(p_{i,j}) = m$ iff $j = \ell_i$

$$G = \left(C_{\ell_1}^{e} \cdots C_{\ell_k}^{e^k}\right)^{x^m} \cdots$$

- Efficiency: computation cost is independent of k, communication cost is halved

  prover time $= O(n \log n)$ verifier time $= O(n)$ proof size $= O(k \log n)$

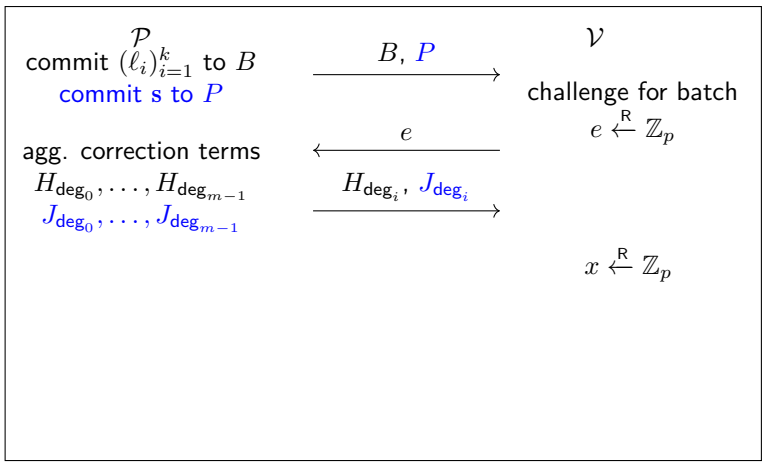
Homogeneous k-out-of-n Range Proof: Solving Security Issue

k-sized multiset I over universe $[n]$
$k = 6$
$(1, 1, 2, 3, 5, 7)$ over $[8]$

I is a set contains distinct elements iff

$\mathbf{s} \in \{0,1\}^n \wedge \langle \mathbf{s}, \mathbf{1}^n \rangle = k$

n-dimension vector $\mathbf{s}_I \in \{0, \ldots, k\}^n$
$s_i := \mathsf{mult}_I(i)$
$(2, 1, 1, 0, 1, 0, 1, 0) \in \{0, \ldots, 6\}^8$



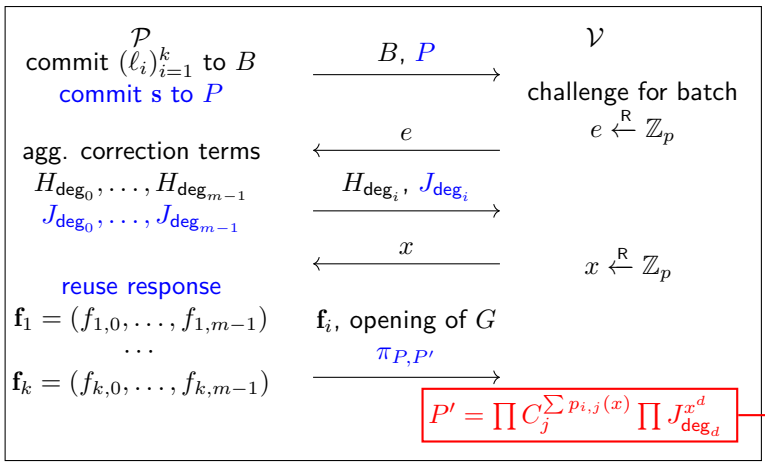
Homogeneous k-out-of-n Range Proof: Solving Security Issue

k-sized multiset I over universe $[n]$
$k = 6$
$(1, 1, 2, 3, 5, 7)$ over $[8]$

I is a set contains distinct elements iff

$\mathbf{s} \in \{0,1\}^n \wedge \langle \mathbf{s}, \mathbf{1}^n \rangle = k$

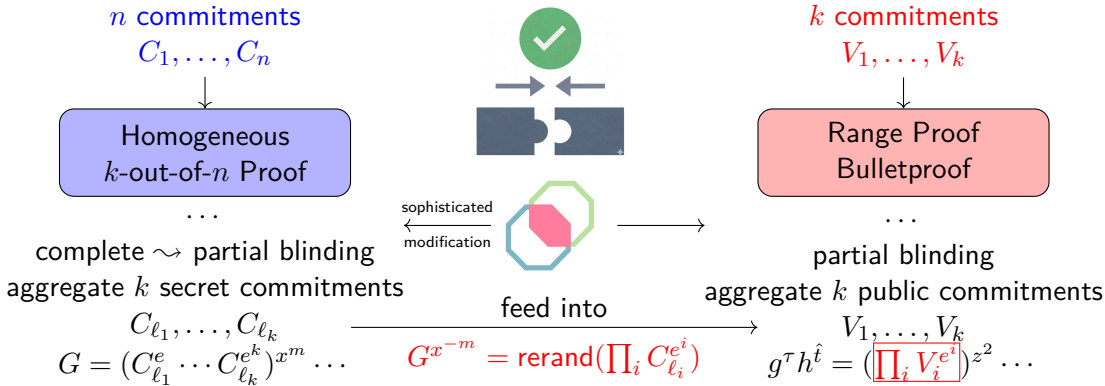
n-dimension vector $\mathbf{s}_I \in \{0, \ldots, k\}^n$
$s_i := \mathsf{mult}_I(i)$
$(2, 1, 1, 0, 1, 0, 1, 0) \in \{0, \ldots, 6\}^8$



use the same polynominal to reconstruct a com P' of $\mathbf{s}_I$
check its consistency with P
P indeed commits to k "1"s

$P' = (g_{\ell_1} \cdots g_{\ell_k})^{x^m} \cdot h^{r^*}$
$P'^{x^{-m}}$: commitment to $\mathbf{s}_I$

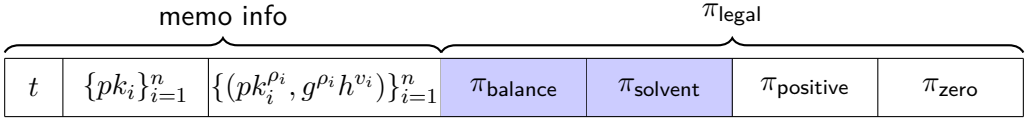
Homogeneous k-out-of-n Range Proof: Seamless Composition



- Generalize the classic GK 1-out-of-n proof to arbitrary $k \geq 1$ case.
- Preserve range proof friendliness $\rightsquigarrow$ efficient homogeneous k-out-of-n range proof

  prover time $= O(n \log n)$ verifier time $= O(n)$ proof size $= O(k \log n)$

Legality Proof: $\pi_{\text{balance}} \circ \pi_{\text{solvent}} \circ \pi_{\text{positive}} \circ \pi_{\text{zero}}$



$$L_{\text{balanced}} := (pk_i, C_{i,L}, C_{i,R})_{i=1}^n \mid \exists (v_i, \rho_i)_{i=1}^n \text{ s.t.}$$
$$C_{i,L} = pk_i^{r_i},\ C_{i,R} = g^{\rho_i} h^{v_i}, \sum v_i = 0$$

the ciphertexts are well-formed and sum of v_j equals zero

$$L_{\text{solvent}} := (pk_i, C_{o,L}, C_{i,R}, \tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n) \mid \exists\ (sk, \ell_s, v^*, v_{\ell_s}, \rho_{\ell_s}) \text{ s.t.}$$
$$\ell_s \in [n] \wedge C_{\ell_s,R} = g^{\rho_{\ell_s}} h^{v_{\ell_s}} \wedge -v_{\ell_s} \in (0, v_{\max}] \wedge$$
$$pk_{\ell_s} = g^{sk} \wedge (\tilde{C}_{\ell_s,R} \cdot C_{\ell_s,R}) = h^{v^*} \left(\tilde{C}_{\ell_s,L} \cdot C_{\ell_s,L}\right)^{sk^{-1}} \wedge v^* \in [0, v_{\max}]$$

one of v_j is negative, and the balance v^* is non-negative

Heterogeneous k-out-of-n Proof: Technical Challenge

We use independent randomness for different public keys, these two proofs requires:

DL-based Heterogeneous k-out-of-n (Range) Proof

prove statements over n public commitments generated under distinct com keys

Heterogeneous k-out-of-n Proof: Technical Challenge

We use independent randomness for different public keys, these two proofs requires:

DL-based Heterogeneous k-out-of-n (Range) Proof
prove statements over n public commitments generated under distinct com keys

Most DL-based homogenerous k-out-of-n proofs [GK15, Dia21] crucially depend on the homomorphism across sub-statements $(C_1, \ldots, C_n)$.

- when commitment keys are different, homomorphism does not hold anymore

extending DL-based homogenerous k-out-of-n proofs to heterogenerous setting is non-trival $\rightsquigarrow$ again requiring new design

Heterogeneous k-out-of-n Proof: Starting Point

ACF k-out-of-n Proof [ACF21]

$L_{\text{hetero-koon}} := (k, C_1, \ldots, C_n) \mid \exists\ (I \subseteq [n], (v_j)_{j \in I})$ s.t. $|I| = k \wedge \forall j \in I : C_j = g^{v_j}$

$\mathcal{P}$ reduce koon to noon $\mathcal{V}$

1. express I via polynomial $p(X) = 1 + \sum_{i=1}^{n-k} a_i x^i$.
2. compute extended witness $\mathbf{y} = (a_1, \ldots, a_{n-k}, p(1)v_1, \ldots, p(n)v_n) \in \mathbb{Z}_p^{2n-k}$.
3. commit to $\mathbf{y}$. $\xrightarrow{Y}$

prove $\mathbf{y}$ satisfy n constraints in batch

$\forall i \in [n] : g^{y_{n-k+i}} = C_i^{p(x)}$

expressed as

$$(C_i^{-i})^{y_1} \cdots (C_i^{-i^{n-k}})^{y_{n-k}} \cdot g^{y_{n-k+i}} = C_i$$

$$\Downarrow \mathbf{M} \boxplus \mathbf{y} = \mathbf{c}$$

$$\left[\begin{array}{ccc|cccc} C_1^{-1} & \cdots & C_1^{-1} & g & 1 & \cdots & \cdots \\ C_2^{-2} & \cdots & C_2^{-2^{n-k}} & 1 & g & 1 & \cdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \\ C_n^{-n} & \cdots & C_n^{-n^{n-k}} & 1 & 1 & \cdots & g \end{array}\right]$$

aggregate n rows: linear combination $\rightsquigarrow O(n(n-k))$

- Pros: does not rely on cross-commitment homomorphism $\rightsquigarrow$ have the potential to be adapted to heterogeneous setting
- Cons: prover and verifier complexities are quadratic $O(n(n-k))$

Heterogeneous k-out-of-n Proof: Improve Efficiency

Insight: ACF protocol uses polynomial to encode the secret index set $I \rightsquigarrow$ overkill

The ACF's

$$\mathbf{y} = (a_1, \ldots, a_{n-k}, p(1)v_1, \ldots, p(n)v_n)$$

$$\forall i \in [n] : (C_i^{-i})^{y_1} \cdots (C_i^{-i^{n-k}})^{y_{n-k}} \cdot g^{y_{n-k+i}} = C_i$$

$$\mathbf{M} \boxplus \mathbf{y} = \mathbf{C}$$

$$\mathbf{M} = \left[\begin{array}{ccc|ccc} C_1^{-1} & \cdots & C_1^{-1} & g & 1 & \cdots \\ C_2^{-2} & \cdots & C_2^{-2^{n-k}} & 1 & g & \cdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ C_n^{-n} & \cdots & C_n^{-n^{n-k}} & 1 & \cdots & g \end{array}\right]$$

dense: n distinct bases per column

aggregate n row vectors of dimension $(2n-k)$

$\rightsquigarrow O(n(n-k))$

Heterogeneous k-out-of-n Proof: Improve Efficiency

Insight: ACF protocol uses polynomial to encode the secret index set $I \rightsquigarrow$ overkill

Idea: drop the polynomial and use indication bit vector to encode I.

k-sized set I over universe $[n]$ $\longleftrightarrow$ bit vector $\mathbf{s} \in \{0,1\}^n \wedge \langle \mathbf{s}, \mathbf{1}^n \rangle = k$

The ACF's

$$\mathbf{y} = (a_1, \ldots, a_{n-k}, p(1)v_1, \ldots, p(n)v_n)$$
$$\forall i \in [n] : (C_i^{-i})^{y_1} \cdots (C_i^{-i^{n-k}})^{y_{n-k}} \cdot g^{y_{n-k+i}} = C_i$$
$$\mathbf{M} \boxplus \mathbf{y} = \mathbf{C}$$

$$\mathbf{M} = \left[\begin{array}{ccc|ccc} C_1^{-1} & \cdots & C_1^{-1} & g & 1 & \cdots \\ C_2^{-2} & \cdots & C_2^{-2^{n-k}} & 1 & g & \cdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ C_n^{-n} & \cdots & C_n^{-n^{n-k}} & 1 & \cdots & g \end{array}\right]$$

dense: n distinct bases per column

aggregate n row vectors of dimension $(2n-k)$

$\rightsquigarrow O(n(n-k))$

Ours

$$\mathbf{y} = (s_1, \ldots, s_n, s_1 v_1, \ldots, s_n v_n)$$
$$\forall i \in [n] : C_i^{-y_i} g^{y_{n+i}} = C_i$$
$$\mathbf{M}^* \boxplus \mathbf{y} = \mathbf{C}$$

$$\mathbf{M}^* = \left[\begin{array}{cccc|ccc} C_1^{-1} & 1 & \cdots & 1 & g & 1 & \cdots \\ 1 & C_2^{-1} & 1 & \cdots & 1 & g & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ 1 & 1 & \cdots & C_n^{-1} & 1 & \cdots & g \end{array}\right]$$

sparse: 1 non-trival base per column

aggregate n row vectors of dimension $(2n-k)$

$\rightsquigarrow O(n)$

Heterogeneous k-out-of-n Proof: Enhance Expressiveness

Our new protocol does not require cross-instance homomorphism $\Rightarrow$ naturally extend to heterogeneous setting just by modifying the bases

$$\mathbf{M} = \left[\begin{array}{cccc|ccc} C_1^{-1} & 1 & \cdots & 1 & g & 1 & \cdots \\ 1 & C_2^{-1} & 1 & \cdots & 1 & g & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ 1 & 1 & \cdots & C_n^{-1} & 1 & \cdots & g \end{array}\right] \quad \mathbf{M}^* = \left[\begin{array}{cccc|ccc} C_1^{-1} & 1 & \cdots & 1 & g_1 & 1 & \cdots \\ 1 & C_2^{-1} & 1 & \cdots & 1 & g_2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ 1 & 1 & \cdots & C_n^{-1} & 1 & \cdots & g_n \end{array}\right]$$

same commitment key g — distinct commitment keys g_i

- Improved efficiency: reduce prover and verifier costs from quadratic complexity $O(n(n-k))$ to optimal linear complexity $O(n)$
- Enhance expressiveness: homogeneous $\rightarrow$ heterogeneous

Heterogeneous k-out-of-n Range Proof

$$L_{\text{hetero-koon}} := (k, C_1, \ldots, C_n) \mid \exists\ (I \subseteq [n], (v_j, r_j)_{j \in I}) \text{ s.t. } |I| = k \wedge \forall j \in I : C_j = g^{v_j} h^{r_j} \wedge v_j \in [0, 2^\nu]$$

Idea: Bridge commitments enable range proofs;
witness reuse ensures value consistency for selected indices.

$\mathcal{P}$ $\mathcal{V}$

1. express I via vector $\mathbf{s}$.
2. generate n bridge commitments $V_1, \ldots, V_n$ with
   V_i commits to v_i using randomness α_i iff $i \in I$; otherwise V_i commits to 0.
3. compute extended witness
   $\mathbf{y} = (s_1, \ldots, s_n, s_1 v_1, s_1 r_1, \ldots, s_n v_n, s_n r_n, s_1 \alpha_1, \ldots, s_n \alpha_n) \in \mathbb{Z}_p^{2n-k}$.
4. commit to $\mathbf{y}$.

$$\xrightarrow{\quad Y \quad}$$

prove $\mathbf{y}$ satisfies n constraints in batch

$$\forall i \in [n] : g^{y_{n+2i-1}} h^{y_{n+2i}} = C_i^{s_i} \wedge g^{y_{n+2i-1}} h^{y_{3n+i}} = V_i^{s_i}$$

Feed all $V_1, \ldots, V_n$ into the range proof

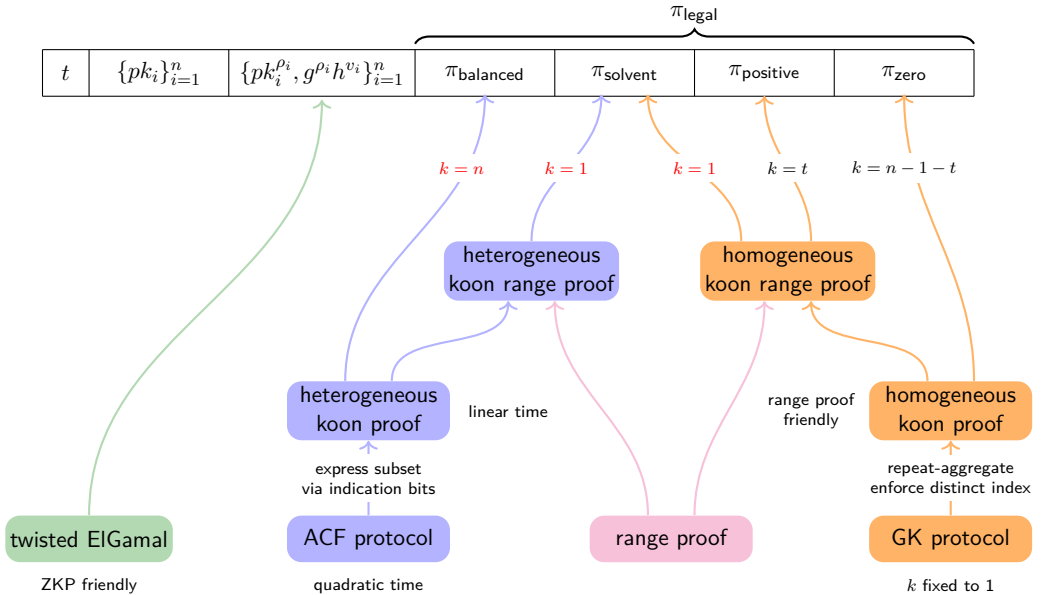
Advancement in k-out-of-n Proofs

Protocol	k	Efficiency			Functionality		
		Prover	Verifier	Size	General	RPF	DGH
[GK15]	$= 1$	$O(n \log n)$	$O(n)$	$O_\lambda(\log n)$	✗	✓	✗
[Dia21]	> 0	$O(n \log^2 n)$	$O(n \log n)$	$O_\lambda(\log n)$	✗	✓	✗
[ACF21]	> 0	$O(n(n-k))$	$O(n(n-k))$	$O_\lambda(\log n)$	✓	✗	✗
[ZGSX23]	> 0	$O(n)$	$O(n)$	$O_\lambda(\log n)$	✓	✗	✗
$\Pi_{\text{homo-koon}}$	> 0	$O(n \log n)$	$O(n)$	$O_\lambda(k \log n)$	✓	✓	✗
$\Pi_{\text{hetero-koon}}$	> 0	$O(n)$	$O(n)$	$O_\lambda(\log n)$	✓	✗	✓

General: arbitrary k-subset RPF: range-proof friendly DGH: heterogeneous commitments

- first DL-based homogeneous k-out-of-n proof with range-proof friendliness
- first DL-based heterogeneous k-out-of-n proof with linear prover/verifier costs

Overview of the Anonymous PGC



Outline

Implementation and Evaluation

Implement and Evaluate

- Homogeneous k-out-of-n proof — $\Pi_{\text{homo-koon}}$
- Homogeneous k-out-of-n range proof — $\Pi^{\text{range}}_{\text{homo-koon}}$
- Heterogeneous k-out-of-n proof — $\Pi_{\text{hetero-koon}}$
- Transaction layer of Anonymous PGC (prototype)

Experimental Setup

- C++ language
- Ubuntu 22.04.05 LTS
- Single-threaded configuration
- Intel(R) Core(TM) i7-11800 CPU operating at 2.3GHz and 16GB of RAM
- Elliptic curve `NIST P-256` with 128 bit security

Homogeneous k-out-of-n (Range) Proof: Performance and Range Proof Friendliness

Evaluate Performance

- Benchmark homogeneous k-out-of-n proof — $\Pi_{\text{homo-koon}}$
- Benchmark homogeneous k-out-of-n range proof — $\Pi^{\text{range}}_{\text{homo-koon}}$

Evaluate Range Proof Friendliness

- Benchmark Bulletproofs range proof — Π_{range}
- Define and compute the metric:

$$rate = \frac{\text{cost}(\Pi^{\text{range}}_{\text{homo-koon}})}{\text{cost}(\Pi_{\text{homo-koon}}) + \text{cost}(\Pi_{\text{range}})}$$

  - Lower $rate$ indicates better range proof friendliness.
  - The case $rate = 1$ means the bridging cost is zero.

Homogeneous k-out-of-n (Range) Proof: Performance and Range Proof Friendliness

Parameters	n	2^5			2^6			2^7		
	k	1	2^2	2^3	1	2^2	2^3	1	2^2	2^3
Prover Time (ms)	$\Pi_{\text{homo-koon}}$	27.01	29.35	37.05	53.40	60.70	67.61	114.3	127.6	142.7
	Π_{range}	11.15	39.45	79.15	10.25	39.55	79.15	10.34	40.15	79.30
	$\Pi^{\text{range}}_{\text{homo-koon}}$	37.90	66.70	109.3	63.41	98.55	142.4	123.5	165.7	219.9
	rate	0.993	0.969	0.941	0.996	0.983	0.970	0.990	0.987	0.990
Verifier Time (ms)	$\Pi_{\text{homo-koon}}$	9.10	10.01	12.75	15.15	17.55	20.02	29.01	31.35	36.05
	Π_{range}	3.35	13.01	25.85	3.32	13.03	25.55	3.12	13.04	25.80
	$\Pi^{\text{range}}_{\text{homo-koon}}$	12.25	22.00	35.65	18.40	29.50	44.00	32.04	44.20	60.95
	rate	0.983	0.956	0.924	0.996	0.965	0.966	0.997	0.995	0.985
Proof Size (bytes)	$\Pi_{\text{homo-koon}}$	1404	1884	2524	1568	2144	2912	1732	2404	3300
	Π_{range}	622	754	820	622	754	820	622	754	820
	$\Pi^{\text{range}}_{\text{homo-koon}}$	1994	2606	3312	2158	2866	3700	2322	3126	4088
	rate	0.984	0.988	0.991	0.985	0.989	0.991	0.986	0.989	0.992

- Under all the parameter choices of (n, k), the cost $rate$ are even below 1.

Our homogeneous k-out-of-n proof integrates range proof seamlessly.

Heterogeneous k-out-of-n Proof: Performance and Comparison

Evaluate Performance

- Benchmark heterogeneous k-out-of-n proof — $\Pi_{\text{hetero-koon}}$ (this work)
- Benchmark k-out-of-n proof [ACF21]

Experimental Details

- Implementation of the ACF protocol is not publicly available.
- We implement the ACF protocol in C++
- The ACF protocol only supports homogeneous statements, we set the homomorphism $F : x \mapsto g^x$ for $\Pi_{\text{hetero-koon}}$ towards a fair comparison

Heterogeneous k-out-of-n Proof: Performance and Comparison with ACF Protocol

Parameters	n	2^5			2^6			2^7		
	k	1	2^2	2^3	1	2^2	2^3	1	2^2	2^3
Prover Time (ms)	[ACF21]	50.00	46.19	42.12	171.3	165.1	156.2	635.5	622.7	605.2
	$\Pi_{\text{hetero-koon}}$	30.16	30.15	30.14	57.17	57.32	57.15	112.5	112.9	113.1
Verifier Time (ms)	[ACF21]	44.45	41.00	36.05	160.2	153.3	144.9	610.5	598.1	580.4
	$\Pi_{\text{hetero-koon}}$	15.20	15.07	15.06	29.03	29.02	29.00	55.80	55.35	55.70
Proof Size (bytes)	[ACF21]	955	955	955	1087	1087	1087	1219	1219	1219
	$\Pi_{\text{hetero-koon}}$	2068	2068	2068	2332	2332	2332	2596	2596	2596

- Compare to the ACF protocol, for $n = \{2^5, 2^6, 2^7\}$ and $k = \{1, 2^2, 2^3\}$:
  proof size doubles
  prover time speeds up $1.4 - 5.6\times$, verifier time speeds up $2.4 - 10.9\times$

Our heterogeneous k-out-of-n proof greatly improves ACF protocol in both functionality and efficiency.

Anonymous PGC: Performance and Comparison with Anonymous Zether

Evaluate Performance

- Benchmark Anonymous PGC (this work)
- Benchmark Anonymous Zether [Dia21]

Experimental Details

- Anonymous Zether is built atop `BN128` in JavaScript.
  - It does not natively support multi-receiver transactions
  - It also includes epoch-handling as a countermeasure against replay attacks, which is not considered in our work.
- We implement the transaction layer of both Anonymous PGC and Anonymous Zether atop `NIST P-256` in C++
  - Dismissing the epoch-handling overhead for both.
  - For Anonymous Zether, we conduct t independent single-receiver transactions to fulfill a t-receiver transaction.

Anonymous PGC: Performance and Comparison with Anonymous Zether

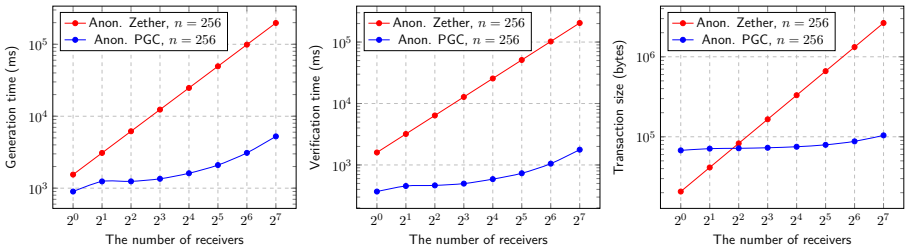
Parameters	n	2^4			2^5			2^6		
	t	1	2^2	2^3	1	2^2	2^3	1	2^2	2^3
Gen Time (ms)	Anon. Zether	35.95	143.8	287.6	62.61	250.4	500.9	136.1	544.3	1088.5
	Anon. PGC	94.42	190.7	271.1	141.8	249.9	331.9	234.5	380.2	461.4
Vrfy Time (ms)	Anon. Zether	21.46	85.84	171.7	45.89	183.6	367.1	121.6	486.2	972.4
	Anon. PGC	38.16	70.74	96.95	59.81	94.19	122.5	101.1	145.2	170.8
Tx. Size (bytes)	Anon. Zether	3514	14056	28112	4898	19592	39184	7338	29352	58704
	Anon. PGC	9896	12996	13640	14152	17512	18284	22104	25724	26624

n is the size of the anonymity set, t is the number of receivers.

- Compared to Anonymous Zether, for 64 size anonymity set and 8 receivers: tx generation speeds up $2.4\times$, tx verification speeds up $5.7\times$, tx size reduces $2.2\times$

In addition to asymptotic improvement and security enhancement, our Anonymous PGC also exhibits superior concrete performance.

Anonymous PGC: Performance and Comparison with Anonymous Zether



- For any fixed anonymity set size n, the costs (tx generation/verification/size) of Anonymous Zether are roughly t times higher than those of our Anonymous PGC

Anonymous PGC exhibits significantly better scalability in multi-receiver setting.

Outline

Summary

Enrich the ZKP Toolbox for Partial Knowledge

1. Extend the GK homogeneous 1-out-of-n proof to general k for the first time
   - overcome the technical hurdle mentioned in [ACF21];
   - construct efficient DL-based homogeneous k-out-of-n (range) proof.
2. Construct the first DL-based heterogeneous k-out-of-n proof based on ACF proof
   - reduce prover and verifier complexity from $O(n(n-k))$ to $O(n)$;
   - expand expressiveness from homogeneous to heterogeneous.

Application: Privacy-Preserving Account-Based Cryptocurrency

1. A generic framework of PPABC from AHE and NIZK: achieving strong security and built-in multi-receiver support
   - resolve open problems in [Dia21] about insider attacks and multi-receiver transfer.
2. Efficient instantiation — Anonymous PGC
   - outperform SOTA Anonymous Zether in both security and efficiency.


(r_2, v_2)
(r_n, v_n)
C_1
C_2
C_3
$\cdots$
C_n
$\mathcal{P}$
$\mathcal{V}$
prove knowledge of k-out-of-n openings

References I


Thomas Attema, Ronald Cramer, and Serge Fehr.
Compressing proofs of k-out-of-n partial knowledge.
In *Advances in Cryptology - CRYPTO 2021*, volume 12828 of *Lecture Notes in Computer Science*, pages 65–91. Springer, 2021.


Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh.
Zether: Towards privacy in a smart contract world.
In *Financial Cryptography and Data Security - FC 2020*, volume 12059, pages 423–443. Springer, 2020.


Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au.
PGC: Pretty Good Confidential Transaction System with Auditability.
In *The 25th European Symposium on Research in Computer Security, ESORICS 2020*, pages 591–610, 2020.
https://eprint.iacr.org/2019/319.

References II


Benjamin E. Diamond.
Many-out-of-many proofs and applications to anonymous zether.
In *42nd IEEE Symposium on Security and Privacy, SP 2021*, pages 1800–1817. IEEE, 2021.


Jens Groth and Markulf Kohlweiss.
One-out-of-many proofs: Or how to leak a secret and spend a coin.
In *Advances in Cryptology - EUROCRYPT 2015*, pages 253–280, 2015.


Yue Guo, Harish Karthikeyan, Antigoni Polychroniadou, and Chaddy Huussin.
Pride CT: towards public consensus, private transactions, and forward secrecy in decentralized payments.
In *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*, pages 3904–3922. IEEE, 2024.

References III


Varun Madathil and Alessandra Scafuro.
Prifhete: Achieving full-privacy in account-based cryptocurrencies is possible, 2023.
https://eprint.iacr.org/2023/710.


Tianyu Zheng, Shang Gao, Yubo Song, and Bin Xiao.
Leaking arbitrarily many secrets: Any-out-of-many proofs and applications to ringct protocols.
In *44th IEEE Symposium on Security and Privacy, SP 2023*, pages 2533–2550. IEEE, 2023.