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

Min Zhang, Yu Chen(✉), and Xiyuan Fu

School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
{zm_min, xiyuanfu}@mail.sdu.edu.cn, yuchen.prc@gmail.com

Abstract

Cryptocurrencies enable transactions among mutually distrustful users. While UTXO-based cryptocurrencies offer mature solutions achieving strong privacy and supporting multi-receiver transfers, account-based cryptocurrencies currently lack practical solutions that simultaneously guarantee these properties.

To close this gap, we propose a generic framework for account-based cryptocurrencies that attains strong privacy and supports multi-receiver transfers, and then give a practical instantiation called *Anonymous PGC*. Our system also outperforms in efficiency: for a 64-sized anonymity set and 8 receivers, Anonymous PGC achieves 2.4× faster transaction generation, 5.7× faster verification, and 2.2× reduction in transaction size compared to state-of-the-art Anonymous Zether (IEEE S&P 2021), which offers only weak privacy and no multi-receiver support.

At the core of Anonymous PGC are two novel zero-knowledge proofs of partial knowledge. First, we generalize the Groth-Kohlweiss (GK) 1-out-of-n proof (EUROCRYPT 2015) to the k-out-of-n case, resolving an open problem regarding its generalization. Particularly, the obtained proof lends itself to seamlessly solder with range proofs, yielding an efficient k-out-of-n range proof that demonstrates k witnesses among n instances lie in specific ranges. Second, we extend the Attema-Cramer-Fehr (ACF) k-out-of-n proof (CRYPTO 2021) to support distinct group homomorphisms, boosting its expressiveness while slashing both prover and verifier complexities from quadratic to linear. We believe these proofs are of independent interest in broader privacy-preserving applications.

Keywords: privacy-preserving cryptocurrency, proofs of partial knowledge, k-out-of-n proofs.

Contents

1 Introduction **3**
- 1.1 Motivations . . . 3
- 1.2 Review of Anonymous Zether . . . 3
- 1.3 Technical Challenges . . . 4
- 1.4 Our Contributions . . . 5
- 1.5 Technical Overview . . . 6
- 1.6 Related Work . . . 12

2 Preliminaries **12**
- 2.1 Public-Key Encryption . . . 12
- 2.2 Zero-Knowledge Proofs . . . 13
- 2.3 A Protocol for Vector Commitments to Bits . . . 15
- 2.4 Bulletproof Compression Technique . . . 15

3 Range Proof Friendly Homogeneous k-out-of-n Proof **18**
- 3.1 Review of the GK 1-out-of-n Proof . . . 18
- 3.2 Generalization of the GK Proof . . . 19
- 3.3 The Homogeneous k-out-of-n Range Proof . . . 24

4 Heterogeneous k-out-of-n Proofs **26**
- 4.1 Review of the ACF k-out-of-n Proof . . . 27
- 4.2 The Heterogeneous k-out-of-n Proof . . . 27
- 4.3 The Heterogeneous k-out-of-n Range Proof . . . 29

5 Privacy-Preserving Account-Based Cryptocurrencies **31**
- 5.1 Definition of PPABC . . . 31
- 5.2 Generic Construction of PPABC . . . 34

6 An Efficient Instantiation: Anonymous PGC **39**
- 6.1 ZKP-Friendly Homomorphic PKE . . . 39
- 6.2 NIZK for Transaction Legality . . . 40
- 6.3 Efficiency Analysis of Anonymous PGC . . . 41

7 Implementation and Evaluation **41**
- 7.1 Experimental Setup . . . 41
- 7.2 Performance of k-out-of-n Proofs . . . 41
- 7.3 Performance of Anonymous PGC . . . 43

1 Introduction

Cryptocurrencies are decentralized payment systems that enable transactions among mutually distrustful parties. These systems are broadly categorized by their transaction models: the unspent-transaction-output (UTXO) model and the account-based model. In the UTXO model (typified by Bitcoin [Nak08]), new transactions consume unspent outputs from previous transactions. In contrast, the account-based model (typified by Ethereum [Woo14]) references the sender's account state, including public keys and balances, as transaction inputs. A critical challenge for cryptocurrencies is achieving transaction privacy, which necessitates both (i) confidentiality, hiding transaction amounts, and (ii) anonymity, concealing participant identities, including the sender and receiver.

1.1 Motivations

While several UTXO-based cryptocurrencies such as ZCash [BCG+14] and Monero [NMT16] achieve strong privacy (i.e., simultaneous confidentiality and anonymity) with practical efficiency, existing account-based cryptocurrencies, despite exhibiting operational advantages, fail to provide strong privacy. We now briefly review account-based cryptocurrencies with emphasis on privacy-preserving techniques.

Account-based cryptocurrencies originated with Ethereum [Woo14], which initially provided no privacy guarantees. In 2020, Chen et al. [CMTA20] proposed an efficient account-based cryptocurrency called PGC, which offers confidentiality but no anonymity. Concurrently, Bünz et al. [BAZB20] introduced another confidential account-based cryptocurrency called Zether, and also outlined how to achieve the so-called n-anonymity by adding $n-2$ extra decoy accounts into a payment transaction with the help of the Groth-Kohlweiss (GK) 1-out-of-n proof [GK15]. Subsequently, Diamond [Dia21] pointed out that Bünz et al.'s approach does not work, as the standard 1-out-of-n proof is insufficient for adding anonymity on top of Zether. Diamond then proposed a variant of 1-out-of-n proof, dubbed many-out-of-many proof, and employed it to build Anonymous Zether. Nevertheless, Anonymous Zether suffers from two deficiencies. First, it fails to provide strong privacy as its security model is weak, yet the obtained n-anonymity is limited. Second, it does not support multi-receiver transactions, which is a vital efficiency feature that avoids per-receiver protocol repetitions. Madathil and Scafuro [MS23] proposed PriFHEte to demonstrate that achieving anonymity within the set of *all accounts* is theoretically possible by leveraging fully homomorphic encryption. However, their system requires miners to perform heavy computation and is unlikely to be practical soon. Recently, Guo et al. [GKPH24] enabled multi-receiver transfer in Anonymous Zether, but completely compromised sender anonymity.

This work focuses on advancing account-based cryptocurrencies by developing a well-rounded system that harmonizes security, functionality, and efficiency. We present the first practical account-based cryptocurrency achieving:

- *Strong privacy:* provides confidentiality and anonymity under a strong security model that captures insider attacks.
- *Efficient multi-receiver transaction:* supports positive transfers to multiple receivers in a single transaction.

1.2 Review of Anonymous Zether

Anonymous Zether [Dia21] represents the current state of the art in privacy-preserving, account-based cryptocurrencies. To provide the necessary background and establish context for our contribution, we first present a detailed review of Anonymous Zether.

In Anonymous Zether, each account is associated with a public/private key pair and an encrypted balance, with the public key acting as the account address. To generate a transaction, the sender proceeds as follows: (1) selects an n-sized anonymity set $(pk_1, \ldots, pk_n)$ consisting of the sender's, receiver's and $n-2$ decoys' accounts; (2) decides a vector of transfer amounts $(v_1, \ldots, v_n)$, where v_j denotes the balance change for account pk_j; specifically, the sender's v_j is non-positive, the receiver's v_j is non-negative (opposite in sign to

the sender's), and the decoys' v_j are equal to 0; (3) encrypts each transfer amount v_j under public key pk_j; (4) generates a proof to attest the legality of the encrypted transaction.

Diamond [Dia21] identified the core challenge in generating *legality proof* is that it entails the fact "not just about two, but also about all of the rest". Namely, given n ciphertexts (generated under distinct public keys), prove that not only two of the plaintexts differ in sign, but also all of the rest are zero. Bünz et al. [BAZB20] suggested a naive approach of invoking Groth-Kohlweiss (GK) 1-out-of-n proofs [GK15] n times, each using one secret index. However, this approach fails to enforce distinctness among the n secret indices (allowing a malicious prover to repeatedly select the same ciphertext), and provides no clear mechanism to prove linear constraints among the plaintexts. Diamond addressed this issue via a *many-out-of-many* proof, enabling a prover to simultaneously demonstrate knowledge of k secrets within a single orbit of a public permutation over n statements and prove these secrets satisfy linear constraints.

Despite these advances, Anonymous Zether suffers from the following deficiencies:

- **Weak security model.** Anonymous Zether captures confidentiality and anonymity via the notion of ledger indistinguishability. However, its security model fails to address a full spectrum of insider attacks, as the adversaries are not allowed to register accounts using dishonestly generated public keys, which is a realistic attack in real world. The vulnerability stems from a core design feature that the sender reuses randomness when encrypting v_j under pk_j. While randomness reusing is fundamental to Anonymous Zether's design (as elaborated in Section 1.3), it also facilitates adversaries to launch *rogue-key* attack (a specific type of insider attacks as described in [Dia21, Section 6.2]). A proposed mitigation would require participants to prove knowledge of their secret keys during registration. However, this solution is not satisfying from two aspects. First, it is merely an ad-hoc countermeasure against rogue-key attack without formal treatment, and will introduce considerable overhead for key management, verification, and storage, making it unsuitable for most cryptocurrencies. Second, it imposes an artificial restriction on the adversary's capabilities, which ultimately undermines the strength of the security model. Diamond left eliminating the proof-of-knowledge requirement as an open problem.

- **Limited anonymity.** In Anonymous Zether, the many-out-of-many proof requires that the sender index l_0 and the receiver index l_1 reside in orbits of opposite parity (i.e., $l_0 \not\equiv l_1 \bmod 2$). This constraint halves the anonymity guarantee, as it decreases the cardinality of the set of possible sender-receiver pairs (l_0, l_1) from $n \cdot (n-1)$ to $n^2/2$. Although this deficit can be remedied by picking larger anonymity sets, it comes at the cost of increased computation cost and transaction size. Diamond [Dia21, Section 6.5.3] outlined a potential method to eliminate the opposite parity constraint, which would also convey an additional benefit of supporting multi-receiver transfers. However, Diamond dismissed this approach, noting that it requires linear proof size and substantial computation costs, thereby representing "an inferior point in the design space". Consequently, It remains an open question whether the anonymity guarantee can be preserved without sacrificing the efficiency.

- **Inability to support multi-receiver transfer.** Anonymous Zether is restricted to single-receiver transaction. As discussed above, although the approach sketched by Diamond [Dia21, Section 6.5.3] theoretically enables multi-receiver transfers, it renders the system impractical due to excessive communication and computation overhead. Alternatively, to transfer amounts to $t \geq 2$ receivers, the sender could sequentially execute the basic protocol t times; however, this naive solution incurs prohibitive overhead for large t and risks revealing transaction patterns (e.g. linking multiple transfers to the same sender). Therefore, enabling efficient multi-receiver support remains a critical challenge for account-based privacy-preserving cryptocurrencies.

In the subsequent sections, we identify the technical challenges underlying these limitations and systematically resolve or bypass them.

1.3 Technical Challenges

Technical challenge 1: heterogeneous k-out-of-n proofs. A natural and provably secure defense against a wide range of insider attacks (including rogue-key attack) is to use distinct randomnesses for en-

crypting each transfer amount. However, this solution creates a new challenge: it results in heterogeneous statement, namely, comprising commitments generated under different commitment keys. Existing k-out-of-n proofs optimized in the discrete logarithm (DL) setting [GK15, ACF21, Dia21, ZGSX23] are built for homogeneous statements, where all commitments share a single commitment key. These proof systems are unable to handle the heterogeneous case because the required homomorphic property does not hold across different commitment keys.[1] Although general k-out-of-n proofs [CDS94, AAB+21, GGHK22, ABFV22, GHKS23] support heterogeneous statements, they incur substantial overhead in both proof size and verification time compared to DL-optimized solutions. Consequently, constructing efficient heterogeneous k-out-of-n proofs remains an open problem.

Technical challenge 2: proofs for "not just two, but all others". As aforementioned, the key difficulty for legality proof lies in demonstrating not only the existence of two indices ℓ_0, ℓ_1 with $v_{\ell_1} = -v_{\ell_0}$ but also that the rest $n-2$ plaintexts $\{v_j\}_{j\in J}$ where $J = [n]\backslash\{\ell_0, \ell_1\}$ satisfy $v_j = 0$. Critically, the proof must enforce disjointness: $\ell_0 \neq \ell_1$ and both $\ell_0, \ell_1 \notin J$.

An obvious approach to proving this statement is translating it into arithmetic circuits and then employing general-purpose zero-knowledge proofs. However, this approach leads to large circuits, which consequently results in extremely inefficient proofs. A more direct approach that avoids generic circuit techniques is therefore preferable. Bünz et al. [BAZB20] suggested a method via repeating the GK 1-out-of-n proof. Unfortunately, Diamond [Dia21] pointed out that the method was problematic cause naive repetition fails to enforce distinctness among indices. Diamond subsequently proposed another method using the *many-out-of-many* proof, which allows proving knowledge of k secrets as well as linear relations among them. However, it imposes restrictive ordering on the secrets, thereby weakens the anonymity guarantees. We also note that the k-out-of-n proof introduced by Attema et al. [ACF21] has no restriction on the k secrets, and can be extended to prove linear relations between them. Nonetheless, it is unable to prove that the secret indices lie outside a specified subset (e.g., $\ell_0, \ell_1 \notin J$). Therefore, a secure and efficient method for proving the statement of "not just two, but all others" remains an open challenge.

Technical challenge 3: k-out-of-n range proofs. To enable positive transfers to $t \geq 2$ receivers, the sender must prove the existence of t positive values among n encrypted transfer amounts, without revealing their indices. This requires efficient k-out-of-n range proofs where $k = t$. However, direct combinations of k-out-of-n proofs with range proofs are infeasible. This stems from the fact that all the existing off-the-shelf range proofs (e.g., Bulletproofs [BBB+18], Bulletproofs++ [EKRN24], Springproofs [ZSLW24]) require the commitments under proven to be explicit input for verification, whereas in k-out-of-n proofs [ACF21, ZGSX23] the target commitments must remain hidden. Thereby, dedicated techniques are needed to build k-out-of-n range proofs.

1.4 Our Contributions

We summarize our contributions as below:

Framework of privacy-preserving account-based cryptocurrencies. As surveyed in Section 1.1, current privacy-preserving account-based cryptocurrencies (PPABC) either fail to offer strong privacy or do not support multi-receiver transfers. In this work, we present a formal definition of PPABC with strong security model and built-in multi-receiver transfers support, then propose a generic construction from homomorphic public-key encryption (PKE) and non-interactive zero-knowledge (NIZK) proofs.

Efficient heterogeneous k-out-of-n proof. We build a k-out-of-n proof for distinct group homomorphisms with logarithmic complexity based on the Attema-Cramer-Fehr (ACF) k-out-of-n proof [ACF21]. While the ACF proof is only applicable to homogeneous statements consisting of commitments under the same commitment key (single group homomorphism that maps witness to instance), our proof can prove

[1] Encrypting n values $(v_1, \ldots, v_n)$ under different public keys $(pk_1, \ldots, pk_n)$ via exponential ElGamal PKE with the same randomness r yields ciphertexts $C_j = (g^r, pk_j^r g^{v_j})$ for $j \in [n]$. The tuples $\{(pk_j, pk_j^r g^{v_j})\}_{j=1}^n$ can be interpreted as n commitments under a common key $(g, h = g^r)$ with distinct randomnesses sk_j, making them compatible with existing homogeneous k-out-of-n proofs. Using distinct randomness for each ciphertext C_j destroys this structure.

heterogeneous statements composed of commitments under different commitment keys (distinct group homomorphisms), capturing more real-world scenarios. This generalized functionality has not been achieved before in the context of logarithmic complexity. As we show below, it turns out to be very useful in the design of privacy-preserving cryptocurrency, and also beyond. More importantly, we reduce both the prover and verifier complexities from $O(n(n-k))$ to $O(n)$ while preserving the logarithmic proof size, rectifying an incorrect $O(n)$ prover complexity claim in [ACF21] (see Section 1.5 for details). Experimental results demonstrate speedups of $1.4-5.6\times$ in proving time and $2.4-10.9\times$ in verification time over ACF, when n ranges from 2^5 to 2^7 and k ranges from 2^0 to 2^3. In sum, our heterogeneous k-out-of-n proof greatly improves ACF proof in both functionality and efficiency.

Range proof friendly homogeneous k-out-of-n proof. We generalize the Groth-Kohlweiss homogeneous 1-out-of-n proof [GK15] to the k-out-of-n case, resolving an open problem left in [ACF21]. Furthermore, the resulting homogeneous k-out-of-n proof is range proof friendly, i.e., it can be seamlessly integrated with off-the-shelf range proofs for Pedersen commitments (e.g., [BBB+18, EKRN24, ZSLW24]). We exemplify this feature by soldering it with Bulletproofs, yielding an efficient homogeneous k-out-of-n range proof, which enables a prover to demonstrate that k committed values hidden among n Pedersen commitments lie in specified range with minor marginal cost, addressing a critical gap in prior work.

Table 1 shows the comparisons between our range proof friendly homogeneous k-out-of-n proof (denoted by $\Pi_{\text{homo-koon}}$), our heterogeneous k-out-of-n proof (denoted by $\Pi_{\text{hetero-koon}}$) and other closely related k-out-of-n proofs in the DL setting.

Table 1: Comparisons between the two k-out-of-n proofs and closely related protocols

Protocol	k	Efficiency			Functionality		
		Prover time	Verifier time	Proof size	Generality	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	$O(n)$	$O(n)$	$O_\lambda(\log n)$	✓	✗	✓

Here, λ denotes the security parameter. "Generality" indicates whether the protocol can prove arbitrary k secrets among n commitments. "RPF" (Range-Proof Friendly) indicates that the protocol can be combined with range proofs without incurring additional bridging costs. "DGH" (Distinct Group Homomorphisms) indicates that the protocol supports heterogeneous statements. Following common practice, we report only the dominant operations when measuring prover and verifier computational costs. In both our protocols and the baseline protocols, the dominant operation is exponentiation in a cryptographic group.

Eventually, by instantiating our PPABC framework with the ZKP-friendly twisted ElGamal PKE and our two novel k-out-of-n proofs, we obtain *Anonymous PGC*, a practical cryptocurrency offering strong privacy and built-in multi-receiver support, thoroughly resolving the open problem left in [Dia21]. As shown in Table 2, compared to existing account-based systems, Anonymous PGC strikes a sweet spot between functionality, efficiency, and security.

1.5 Technical Overview

We first describe our generic framework of privacy-preserving account-based cryptocurrency (PPABC) from homomorphic PKE and NIZK with more details, then discuss how to instantiate the framework with right cryptographic tools.

Generic PPABC Framework. We formalize the definition of PPABC and present a generic construction from homomorphic PKE and NIZK. PPABC maintains a global ledger of *accounts*. Each account is equipped with a homomorphic PKE keypair and an encrypted balance. The public key serves as the account address,

Table 2: Comparisons between Anonymous PGC and other account-based systems

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	$O(n \log n + t\nu)$	$O(n + t\nu)$	$O_\lambda(n + t \log n + \log(t\nu))$	✓	✓	✓

Here, $\nu = \log v_{\max}$ denotes the bit-length of the maximum amounts (e.g., 32 in practice). λ is the security parameter. t is the number of intended receivers in a single transaction, and n is the size of the anonymity set. "Tx" is short for transaction. In both our scheme and the baseline schemes, the dominant operation of transaction generation and verification is exponentiation in a cryptographic group. "Insider" indicates that the security model considers insider attacks. Such attacks are vacuous to PGC and Zether which only provide confidentiality, since the "insiders" (i.e., its sender and receiver) of each transaction already know each other's roles as well as the transfer amount. "C" stands for confidentiality and "A" for anonymity. ✘ indicates limited anonymity.

while the corresponding secret key authorizes transactions and decrypts encrypted balance. To transfer to $t \geq 1$ receivers $pk_{r_1}, \ldots, pk_{r_t}$, the sender pk_s proceeds as below: (1) select n accounts comprising the sender pk_s, the t receivers, and $n - t - 1$ decoy accounts, and shuffle them into an anonymity set $pk_1, \ldots, pk_n$; (2) determine a transfer amounts vector $v_1, \ldots, v_n$, where v_j denotes the balance change associated with account pk_j, i.e., $v_j < 0$ for the sender, $v_j > 0$ for receivers, and $v_j = 0$ for decoys; (3) encrypt each transfer amount v_j under public key pk_j; (4) attach a proof attesting the legality of the transaction. A smart contract verifies the proof and, if valid, homomorphically updates the balances of accounts $pk_1, \ldots, pk_n$. Compared to Anonymous Zether, our PPABC framework offers two key enhancements.

- *Security:* Following Anonymous Zether, PPABC captures confidentiality and anonymity via the notion of ledger-indistinguishability. The enhancement lies in that PPABC considers a wide range of insider attacks, whereas Anonymous Zether just suggests an ad-hoc countermeasure against a specific insider threat — rogue key attack. This strengthening is paramount in the cryptocurrencies domain.

- *Functionality:* PPABC natively supports built-in multi-receiver transfer and mandates positive transfers to receivers (only decoy accounts receive zeros). Anonymous Zether, however, only supports single-receiver transfer and permits zero transfers. This design forces inefficient, repeated workflows for multi-receiver transfers and could compromise anonymity, as the sender must appear in the intersection of multiple anonymity sets. Even worse, it obstructs modular proof design (see Technical Challenge 2 stated in Section 1.3) and is susceptible to empty transaction spam. Our enhancement aligns with practical use cases. It not only prevents empty transaction spam, but also allows complicated legality statements to be intricately decomposed to simpler ones: (i) balance: all ciphertexts are well-formed and $\sum_{j=1}^{n} v_j = 0$; (ii) decoy integrity: $\exists J \subset [n]$, $|J| = n-t-1$ s.t. $v_j = 0$ for all $j \in J$; (iii) receiver validity: $\exists S \subset [n]$, $|S| = t$ s.t. $v_j > 0$ for all $j \in S$; (iv) sender solvency: $\exists\ \ell_s \in [n]$ s.t. $v_{\ell_s} < 0$, meanwhile the sender knows the secret key for pk_{ℓ_s} and its balance is solvent. Such decomposition incidentally circumvents Technical Challenge 2 by disallowing zero transfers, which accounts for the "limited anonymity" hurdle in Anonymous Zether.

An efficient instantiation of our generic PPABC framework entails carefully instantiating the cryptographic ingredients. With the goal to make the legality statement easy to be proved, we choose the twisted ElGamal PKE [CMTA20] to instantiate the PKE component, which is preferable to standard exponential ElGamal PKE due to its ZKP-friendly nature (see Section 6.1 for details). In twisted ElGamal PKE, the right parts of the ciphertexts under different public keys are of the exact shape of Pedersen commitments under the same commitment key (g, h), whose DL relation is unknown to all users. Such structure makes twisted ElGamal PKE particularly compatible with all ZKPs for Pedersen-commitment-based statements, and admits a flexible interpretation of multiple ciphertexts between homogeneous and heterogeneous form.

By plugging twisted ElGamal PKE into our PPABC framework, the four sub-statements of legal statement become concrete: (i) Balance proves knowledge of n plaintexts of n ciphertexts under different public keys and the sum of the n plaintexts is zero. It can be proved via a heterogeneous n-out-of-n proof [2]. (ii) Decoy integrity proves that partial openings of n Pedersen commitments under the same commitment key are zero, which can be proved via homogeneous $(n-t-1)$-out-of-n proof. (iii) Receiver validity proves that partial openings of n Pedersen commitments under the same commitment key lie in specific ranges, which can be proved via homogeneous t-out-of-n range proofs. (iv) Sender solvency proves one opening of n Pedersen commitments under the same commitment key lies in specific range, as well as one opening of n twisted ElGamal ciphertexts under different public keys lies in specific range. The former can be proved via homogeneous 1-out-of-n range proof, and the latter can be proved via heterogeneous 1-out-of-n range proof.

Figure 1 depicts the transaction structure and the underlying protocols for legality proof. In the figure, the special cases highlighted in red (i.e., $k = 1$ and $k = n$) indicate that we use optimized versions of the corresponding protocols rather than directly instantiating the general heterogeneous or homogeneous k-out-of-n (range) proofs. Details of these optimizations are given in Section 3 and Section 4.

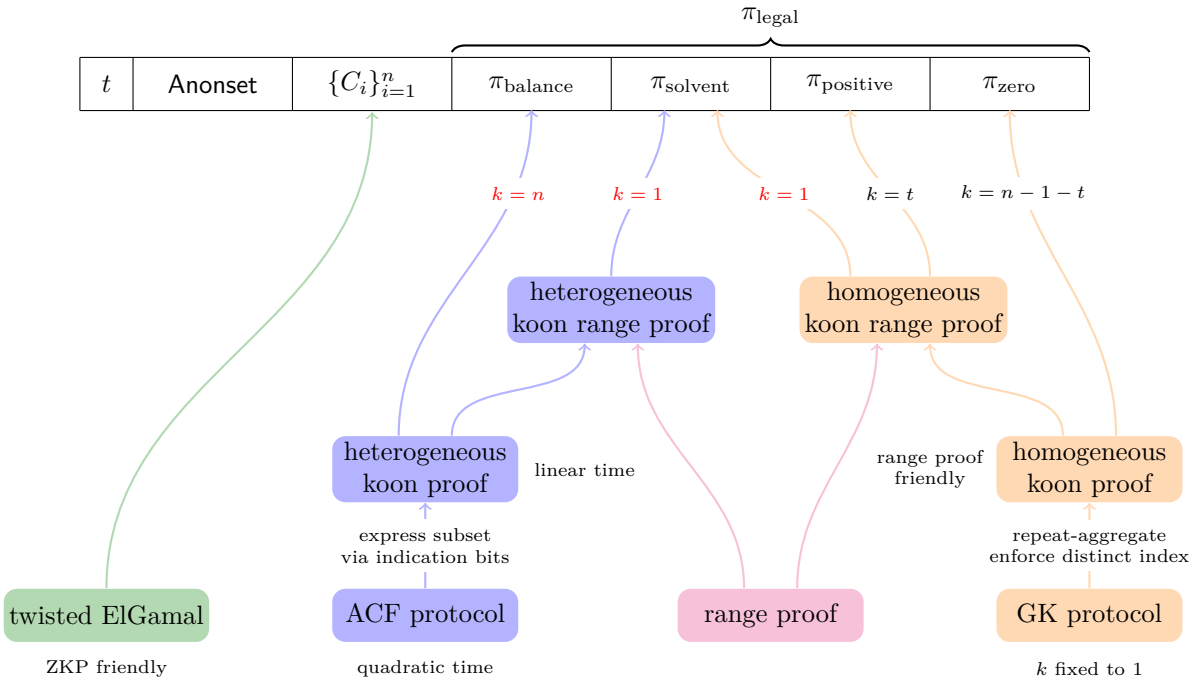

Figure 1: Overview of the transaction structure and the underlying protocols for legality proof.

In the following section, we outline the high-level idea behind the two core k-out-of-n proofs, i.e., the range proof friendly homogeneous k-out-of-n proof and the heterogeneous k-out-of-n proof.

Range proof friendly homogeneous k-out-of-n proof. A k-out-of-n proof enables a prover to prove knowledge of openings of k secret commitments $c_{\ell_1}, \ldots, c_{\ell_k}$ among n public commitments $c_1, \ldots, c_n$. Here, "homogeneous" means that the n commitments are committed under the same commitment key. A k-out-of-n proof system is "range proof friendly" if it can be soldered with off-the-shelf range proofs without

[2] Our protocol retains the extensibility of the original ACF protocol, which is compatible with the circuit ZK technique in [AC20] for proving general linear constraints on the witnesses. Nonetheless, to achieve a more efficient proof specifically for the case where a sum is zero, we leverage the ZKP-friendly structure of the twisted ElGamal PKE and employ a direct sigma protocol.

extra bridging costs. This permits efficient k-out-of-n range proofs, where a prover demonstrates that the k committed values lie within a specific range. Since the GK homogeneous 1-out-of-n proof [GK15] is range proof friendly, it serves as an ideal starting point for our protocol.

Recall that the GK protocol enables a prover to prove knowledge of an opening of a secret commitment c_ℓ to 0 among n public homomorphic commitments $c_1, \ldots, c_n$. The core of the GK protocol is n polynomials $p_1(X), \ldots, p_n(X)$ constructed by the prover using the witness ℓ, of which only $p_\ell(X)$ has a high degree of $\log n$. Through an efficient communication with the prover (i.e., using $O(\log n)$ communication cost), the verifier is able to obtain the n evaluations $p_1(x), \ldots, p_n(x)$ at a random point x chosen by the verifier. By multi-exponentiating $c_1, \ldots, c_n$ with the n evaluations $p_1(x), \ldots, p_n(x)$ and canceling out the lower-order terms via correction terms sent by the prover, the verifier is able to "pick out" a re-randomized commitment c_ℓ^* of c_ℓ (namely c_ℓ^* is also a commitment to 0 but with a different randomness r^*) without knowing the secret index ℓ. Due to this feature, the GK protocol is compatible with range proofs. Specifically, to prove c_ℓ is a commitment to $v \in [a, b]$, it is equivalent to prove c_ℓ^* is a commitment to $v \in [a, b]$. With this in mind, we now present a stepwise bootstrapping of the GK protocol into a k-out-of-n proof, ultimately yielding an efficient k-out-of-n range proof. See Section 3 for the technical details.

Step 1: Repeating the GK protocol in parallel. We begin by executing k parallel instances of the GK 1-out-of-n protocol, where each execution enables the verifier to obtain n evaluations $p_1(x), \ldots, p_n(x)$. Using the n evaluations, the verifier is able to pick out a re-randomized commitment $c_{\ell_i}^*$ of a secret commitment c_{ℓ_i}, such that the prover can prove knowledge of openings of c_{ℓ_i} via revealing openings of $c_{\ell_i}^*$. However, this straightforward approach has two deficiencies. First, the protocol is inefficient due to the need for generating and transmitting correction terms in each execution, incurring $O(k \log n)$ communication complexity, $O(kn \log n)$ prover complexity and $O(kn)$ verifier complexity. Second, the protocol is insecure, as it fails to prevent a malicious prover from reusing index (i.e., $\ell_i = \ell_j$ for $i \neq j$). The subsequent two steps address these deficiencies sequentially.

Step 2: Improving the efficiency via random linear combination. To improve the efficiency, we replace the k lists of individual correction terms with a single list of aggregated correction terms. This optimization simultaneously shrinks proof size, reduces prover time and enables batch verification. Concretely, the group elements contained in the proof are reduced to $O(\log n)$, and thus the proof size is halved (the field elements required are still $O(k \log n)$). The group operations required are reduced to $O(n \log n)$ for the prover, and are reduced to $O(n)$ for the verifier. The verifier is able to pick out the k secret commitments in one shot, i.e., obtaining an aggregated version of k re-randomized commitments $(c_{\ell_1}^*, \ldots, c_{\ell_k}^*)$. The next step aims to enforce the prover to use k distinct indices.

Step 3: Enforcing distinct indices via bit vector commitment. Note that each k-sized set $\{\ell_i\}_{i=1}^k$ where $\ell_i \in [n]$ (which could be a multiset), can be uniquely represented by a vector $\mathbf{s} \in \{0, \ldots, k\}^n$, in which s_j denotes the number of times index j appears in the set $\{\ell_i\}_{i=1}^k$, and thus $\langle \mathbf{s}, \mathbf{1}^n \rangle = k$. Particularly, ℓ_i's are distinct if and only if $\mathbf{s} \in \{0, 1\}^n$. With this property, our high-level idea for enforcing distinct indices is: given a sequence of bases $g_1, \ldots, g_n, h_1, \ldots, h_n, u$ whose DL-relations are unknown to the prover, the verifier is able to "pick out" a vector commitment $P = \mathbf{g}^{\mathbf{s}} \cdot \mathbf{h}^{\mathbf{s}-\mathbf{1}^n} u^{r^*} = \prod_{i=1}^k g_{\ell_i} h_{\ell_i} \cdot \prod_{j=1}^n h_j^{-1} \cdot u^{r^*}$ to $\mathbf{s}$ using the k vectors of polynomials' evaluations $(p_1(x), \ldots, p_n(x))$ generated in the first step and corresponding correction terms (the randomness r^* depends on the randomnesses in these correction terms); the prover then convinces the verifier that all the indices $\{\ell_i\}_{i=1}^k$ are distinct by proving that P is a commitment to a bit vector with Hamming weight k.

Step 4: Soldering with Bulletproofs seamlessly. Range proofs typically require both the prover and the verifier to have explicit access to the commitments to be proved. This poses a challenge when combining them with k-out-of-n proofs, where the k commitments are known only to the prover and must remain hidden from the verifier. As the prover knows both the statements and the witnesses, he is able to generate the proof following the original range proof protocols. The core of the problem is thus enabling the verifier to perform verification using the k commitments without knowing them. Our k-out-of-n proof resolves this problem elegantly by allowing the verifier to obtain an aggregated and re-randomized version of the k secret commitments $(c_{\ell_1}, \ldots, c_{\ell_k})$. This feature makes it compatible with range proofs that support aggregation

and batch verification, such as those in [BBB+18, EKRN24, ZSLW24], as these proofs allow the verifier to incorporate aggregated commitments directly into their verification equations. Specifically, our k-out-of-n proof can be seamlessly integrated with such range proofs, by plugging the re-randomized and aggregated commitments from the k-out-of-n proof into the range proofs' verification equations. Therefore, we obtain efficient homogeneous k-out-of-n range proofs with only minor marginal overhead.

Heterogeneous k-out-of-n proof. A heterogeneous k-out-of-n proof enables a prover to demonstrate knowledge of openings for a subset of k commitments that may be generated under distinct commitment keys, or more generally, distinct group homomorphisms. Existing protocols optimized for the discrete logarithm setting, including the GK protocol [GK15], our generalized GK protocol, and the recent Any-out-of-Many protocol [ZGSX23], fundamentally rely on homomorphic aggregation across statements. Specifically, the verifier must combine the input commitments into a single target value; this operation is mathematically valid only if all commitments share the same commitment key, rendering these protocols inherently inapplicable to the heterogeneous setting. In contrast, the ACF protocol [ACF21] adopts on a different paradigm: it reduces the k-out-of-n relation to an n-out-of-n relation via a witness-masking technique that operates independently of the commitment keys, thereby avoiding the need for using a common commitment key. While the original ACF protocol was described for the single commitment key setting and suffers from quadratic $O(n(n-k))$ prover and verifier time, its independence from cross-commitment homomorphism makes it an ideal starting point for our goals. Below, we first revisit the ACF protocol to identify the source of its inefficiency, then reduce the computational cost to linear in n and extend it to support heterogeneous statements simultaneously.

Given n Pedersen commitments $c_1, \ldots, c_n$, the ACF protocol allows a prover to prove knowledge of a k-sized subset $S \subset [n]$ and a vector $(\mathbf{x}_j)_{j\in S} = (r_j, v_j)_{j\in S}$ satisfying $c_j = g^{r_j} h^{v_j}$ for $j \in S$. At the core of the ACF protocol is a reduction from the k-out-of-n case to the n-out-of-n case, achieved by having the prover "eliminate" the instances for which it does not know a witness. To accomplish this, the protocol employs a secret-sharing mechanism, similar to that used in [CDS94]. In contrast to [CDS94], which shares the verifier's challenge, the ACF protocol shares a default secret 1 instead.

Roughly speaking, the ACF protocol proceeds as below. First, the prover encodes the subset S as an $(n-k)$-degree polynomial $p(X)$ subject to $p(0) = 1$ and $p(j) = 0$ for all $j \notin S$. Next, the prover generates the extended witness $\mathbf{y} = (a_1, \ldots, a_{n-k}, p(1)\mathbf{x}_1, \ldots, p(n)\mathbf{x}_n) \in \mathbb{Z}_p^{3n-k}$, where a_i is the coefficient of the i-th degree term in the polynomial $p(X)$. Then, the prover commits to $\mathbf{y}$ and sends this commitment to the verifier. Finally, the prover demonstrates knowledge of the opening $\mathbf{y}$ and convinces the verifier that the homomorphism relation $H_j(\mathbf{y}) = c_j^{-\sum_i y_i j^i} g^{y_{n-k+2j}} h^{y_{n-k+2j+1}} = c_j$ holds for all $j \in [n]$ (the n-out-of-n case). Here, each $H_j : \mathbb{Z}_p^{3n-k} \mapsto \mathbb{G}$ is a group homomorphism defined by $3n-k$ bases, which consist of $n-k$ index-relevant elements (i.e., $c_j^{-j}, \ldots, c_j^{-j^{n-k}}$), 2 index-irrelevant elements (i.e., g, h), and $2n-2$ identities. Therefore, the n homomorphisms $\{H_j\}_{j=1}^n$ can be defined by a bases matrix $\mathbf{M} \in \mathbb{G}^{n\times(3n-k)}$, whose j-th row is the bases defining H_j:

$$\mathbf{M} = \left[\begin{array}{ccc|cccccc} c_1^{-1} & \cdots & c_1^{-1} & g & h & 1 & 1 & \cdots & \cdots \\ c_2^{-2} & \cdots & c_2^{-2^{n-k}} & 1 & 1 & g & h & \cdots & \cdots \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n^{-n} & \cdots & c_n^{-n^{n-k}} & 1 & 1 & \cdots & \cdots & g & h \end{array}\right]$$

To reduce the proof size, the n homomorphisms are aggregated into a single one using a random linear combination and then proved via the Bulletproof compression technique, resulting in logarithmic proof size. The original ACF protocol claims $O(n)$ prover complexity, but this calculation is incorrect. To generate the aggregated homomorphism $H := \sum_{j=1}^n \rho^j H_j$ for a random challenge ρ, both the prover and the verifier have to perform row-wise random linear combination of the bases matrix $\mathbf{M}$, which requires $O(n(n-k))$ exponentiations due to the $n-k$ columns of index-relevant elements. This is a prohibitive overhead when n is large. Next, we present a method for reducing the computational costs, and then extend it to prove statements involving distinct group homomorphisms. See the detailed technical explanation in Section 4.

Sparsifying the bases matrix. The high computational cost of aggregating the n homomorphisms $\{H_j\}_{j=1}^n$ stems from the structural complexity of each H_j. In particular, each H_j requires a representation involving $n-k$ index-relevant group elements, which makes the bases matrix $\mathbf{M}$ dense. This motivates us to seek a more concise representation of $\{H_j\}_{j=1}^n$ to obtain a sparse bases matrix.

We observe that the representation complexity of H_j roots from the polynomial, which is used to implement secret sharing. We argue, however, the secret sharing mechanism is conceptually excessive, as there is no genuine secret to protect. [3] The polynomial essentially serves as an encoding of the set S. This critical observation opens the possibility for a more direct and efficient approach to enable the same reduction. We are thus motivated to seek for more concise representation of set, yielding a sparser bases matrix.

In this work, we replace the polynomial representation with an indication vector $\mathbf{s} \in \{0, \ldots, k\}^n$. Here, s_j denotes the multiplicity of index j in S, and $\mathbf{s}$ degenerates to a bit vector iff S contains distinct elements. By setting $\mathbf{y} = (s_1, \ldots, s_n, s_1 \cdot \mathbf{x}_1, \ldots, s_n \cdot \mathbf{x}_n)$, we obtain n constraints of the form $H_j(\mathbf{y}) = c_j^{-y_j} g^{y_{n+2j}} h^{y_{n+2j+1}} = 1$ for $j \in [n]$. In this case, each $H_j : \mathbb{Z}_p^{3n} \mapsto \mathbb{G}$ is a group homomorphism defined by $3n$ bases, consisting of only one index-relevant element (i.e., c_j^{-1}), two index-irrelevant group element (i.e., g, h) and $3n-3$ identities. The new bases matrix $\mathbf{M} \in \mathbb{G}^{n \times 3n}$ is as below:

$$\mathbf{M} = \left[\begin{array}{cccc|cccccc} c_1^{-1} & 1 & \cdots & 1 & g & h & 1 & 1 & \cdots & \cdots \\ 1 & c_2^{-1} & 1 & \cdots & 1 & 1 & g & h & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & c_n^{-1} & 1 & 1 & \cdots & \cdots & g & h \end{array}\right]$$

Although this adjustment enlarges the bases matrix $\mathbf{M} \in \mathbb{G}^{n \times 3n}$ a bit, it renders the matrix highly sparse, thereby greatly reducing the cost of computing the aggregated homomorphism $H = \sum_{j=1}^n \rho^j H_j$ to $O(n)$. To ensure the k indices are distinct, the prover has to additionally prove that each s_j is either 0 or 1 and $\langle \mathbf{s}, \mathbf{1}^n \rangle = k$. Thanks to the Bulletproof compression technique, the proof of this part is logarithmic in n. Put it all together, the overall proving and verification complexities are linear in n, and the proof size is logarithmic in n.

Generalizing to distinct group homomorphisms. To further support heterogeneous statements, we exploit the fact that the reduction operates independently across instances. Namely, it does not require a common commitment key across the n Pedersen commitments. Proving knowledge of the opening of each commitment c_j is reduced to a homomorphism constraint involving only c_j and its own commitment key (g, h). This inherent independence allows for a natural extension to heterogeneous settings.

For illustration, consider the following example extended from the above case. Given n Pedersen commitments $c_1, \ldots, c_n$,the prover aims to prove knowledge of a k-sized subset $S \subset [n]$ and a vector $(\mathbf{x}_j)_{j \in S} = (r_j, v_j)_{j \in S}$ satisfying $c_j = g_j^{r_j} h_j^{v_j}$ for each $j \in S$. In this setting, the bases matrix $\mathbf{M} \in \mathbb{G}^{n \times 3n}$ is defined as

$$\mathbf{M} = \left[\begin{array}{cccc|cccccc} c_1^{-1} & 1 & \cdots & 1 & g_1 & h_1 & 1 & 1 & \cdots & \cdots \\ 1 & c_2^{-1} & 1 & \cdots & 1 & 1 & g_2 & h_2 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & c_n^{-1} & 1 & 1 & \cdots & \cdots & g_n & h_n \end{array}\right]$$

In Section 4, we further consider a more general heterogeneous setting. Given n group homomorphisms $\{F_j\}_{j=1}^n$ and n vectors $\mathbf{P}_1, \ldots, \mathbf{P}_n \in \mathbb{G}^s$, the prover demonstrates knowledge of a k-sized subset S and $(\mathbf{x}_j)_{j \in S}$ satisfying $\mathbf{P}_j = F_j(\mathbf{x}_j)$ for each $j \in S$ by defining n homomorphisms H_j as $H_j(\mathbf{y}) = F_j(\mathbf{y}_{j+1}) \mathbf{P}_j^{-y_j}$ for $j \in [n]$, and proving knowledge of $\mathbf{y}$ such that the constraint $H_j(\mathbf{y}) = \mathbf{1}$ holds for all $j \in [n]$. This yields a heterogeneous k-out-of-n proof that supports distinct group homomorphisms while maintaining the computational efficiency of the homogeneous case.

[3] Both [CDS94] and [ACF21] employ secret sharing mechanism to "eliminate" the instances that the prover does not know a witness. Specifically, [CDS94] secret shares the verifier's challenge, while [ACF21] secret shares a default constant 1. We note that they are not secrets.

1.6 Related Work

Cryptocurrencies. In addition to account-based cryptocurrencies introduced in Section 1.1, a significant body of research focuses on enhancing the privacy of UTXO-based cryptocurrencies. In 2015, Maxwell [Max15] first studied the confidentiality of Bitcoin [Nak08], achieving confidentiality by hiding the transfer amount via Pedersen commitments and utilizing range proofs to ensure transaction validity. Mimblewimble [Poe16] and Grin [Gri] further improved Maxwell's protocol by reducing the signature costs. Monero [NMT16] introduces anonymity through linkable ring signatures and stealth addresses, but the transaction size grows linearly with the ring size. ZCash [BCG+14] offers anonymity with sublinear cost by employing zkSNARKs. A series of works, including CoinJoin [Max13], CoinShuffle [RMK14], TumbleBit [HAB+17], Dash [Das], and Mixcoin [BNM+14], achieve anonymity through mixing mechanisms. Several other cryptocurrencies, such as Quisquis [FMMO19], zkLedger [NVV18], and constructions in [WGL+24], are in a hybrid model combining UTXO and account-based systems. While these approaches provide both confidentiality and anonymity, they inherit some of the disadvantages of the UTXO model, such as that all the previous transactions cannot be easily purged without identification.

k-out-of-n proofs. k-out-of-n proofs allow a prover to prove knowledge of k secret openings among a set of n public commitments. It was first introduced by Cramer et al. [CDS94], but the first protocol has a proof size linear to n. In 2015, Groth and Kohlweiss [GK15] proposed a Σ-protocol for 1-out-of-n proof, which has a proof size logarithmic in n but only works on the $k = 1$ case. Bootle et al. [BCC+15] further improved the communication costs of the GK protocol by leveraging Pedersen vector commitments, whereas it is still in the $k = 1$ case. Jivanyan and Mamikonyan [JM20] improved the prover complexity of the GK protocol by proposing a hierarchical approach, at the cost of increasing the communication costs, and is also focusing on the $k = 1$ case. Diamond [Dia21] extended the GK 1-out-of-n proof to the $k \geq 1$ cases, referred to as the many-out-of-many proofs. However, it does not constitute a general k-out-of-n proof, as it requires the k secrets located in one of the orbits of a public permutation of n public commitments. In 2021, Attema et al. [ACF21] proposed the first general k-out-of-n proof (witness can be any k-subset of the n public elements) with a proof size $O(\log n)$, while both its prover and verifier running time are $O(n(n-k))$. Zheng et al. [ZGSX23] presented the any-out-of-many proof, which does not leak the value of k, and can also be modified to the general k-out-of-n proof, but it is unknown how to be extended to meet the specific requirements of account-based cryptocurrencies. In recent years, other general k-out-of-n proofs have emerged [AAB+21, ZLH+22, GGHK22, ABFV22, GHKS23], but most of them focus on proving general threshold relations, and thus are less efficient compared to those tailored for discrete logarithm relations.

2 Preliminaries

Notation. Let $\mathbb{G}$ be a cyclic group of prime order p and $\mathbb{Z}_p$ be the ring of integers modulo p. We use bold font or parentheses to denote vectors, i.e., both $\mathbf{a}, (a_i)_{i=1}^n \in \mathbb{Z}_p^n$ are equal to the vector $(a_1, \ldots, a_n) \in \mathbb{Z}_p^n$. For an integer $c \in \mathbb{Z}_p$, a vector $\mathbf{a} \in \mathbb{Z}_p^n$ and a vector $\mathbf{g} \in \mathbb{G}^n$, denote by $\mathbf{b} = c \cdot \mathbf{a} \in \mathbb{Z}_p^n$ the vector where $b_i = c \cdot a_i$, denote by $\mathbf{g}' = \mathbf{g}^c \in \mathbb{G}^n$ the vector where $g'_i = g_i^c$ and $\mathbf{g}^{\mathbf{a}} \in \mathbb{G}$ the group element $\prod_{i=0}^{n-1} g_i^{a_i}$. For scalars $c, n \in \mathbb{Z}_p$, we denote by $[n]$ the set $\{1, \ldots, n\}$ and $\mathbf{c}^n$ the vector $(c^0, c^1, \ldots, c^{n-1})$. For two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$, let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i \cdot b_i \in \mathbb{Z}_p$ be the inner product of them and $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, \cdots, a_n \cdot b_n) \in \mathbb{Z}_p^n$ be the entry wise multiplication of two vectors. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two Abelian groups, $\mathbb{G}_1^t$ and $\mathbb{G}_2^s$ be vector spaces of dimension t over $\mathbb{G}_1$ and dimension s over $\mathbb{G}_2$, respectively. We denote the set of all group homomorphisms $F : \mathbb{G}_1^t \mapsto \mathbb{G}_2^s$ by $\mathrm{Hom}(\mathbb{G}_1^t, \mathbb{G}_2^s)$.

2.1 Public-Key Encryption

Definition 2.1 (Public-Key Encryption (PKE))**.** A PKE scheme consists of four polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter 1^λ, outputs public parameters pp, which include descriptions of the plaintext space M, ciphertext space C and the randomness space R.

- $\mathsf{KeyGen}(pp; r)$: on input the public parameters pp and randomness r, outputs a public key pk and a secret key sk. This algorithm naturally induces a hard relation $\mathsf{R}_{\text{key}} = \{(pk, (sk, r)) \mid \exists\ r \text{ s.t. } (pk, sk) = \text{PKE.}\mathsf{KeyGen}(pp; r)\}$. Since we always use the randomness as the secret key, w.l.o.g., R_{key} can be simplified to $\mathsf{R}_{\text{key}} = \{(pk, sk) \mid \exists\ sk \text{ s.t. } pk = \text{PKE.}\mathsf{KeyGen}(pp; sk)\}$.
- $\mathsf{Enc}(pk, m)$: on input a public key pk and a plaintext m, outputs a ciphertext c. When emphasizing the randomness r used for encryption, we denote it by $c \leftarrow \mathsf{Enc}(pk, m; r)$.
- $\mathsf{Dec}(sk, c)$: on input a secret key sk and a ciphertext c, outputs a plaintext m.

Correctness. For any $pp \leftarrow \mathsf{Setup}(1^\lambda)$, any key pair $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$ and any message $m \in M$, it holds that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$.

IND-CPA. A PKE scheme is IND-CPA secure in the multi-user setting if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ it holds that

$$\left| \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk_i, sk_i) \leftarrow \mathsf{KeyGen}(pp) \text{ for } i = 1, \ldots, q; \\ (state, \mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}_1(pp, pk_1, \ldots, pk_q); \\ \beta \xleftarrow{\text{R}} \{0,1\}; \\ r_1, \ldots, r_q \xleftarrow{\text{R}} R; \\ C_i \leftarrow \mathsf{Enc}(pk_i, m_{\beta,i}; r_i) \text{ for } i = 1, \ldots, q; \\ \beta' \leftarrow \mathcal{A}_2(state, C_1, \ldots, C_q); \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where $q = \mathsf{poly}(\lambda)$ and $\mathbf{m}_\beta = \{m_{\beta,1}, \ldots, m_{\beta,q}\}$.

Obviously, IND-CPA security in the single-user setting directly implies IND-CPA security in the multi-user setting [BBM00].

Homomorphic PKE. For a PKE scheme, we say it is homomorphic, if for any key pair $(pk, sk) \in \mathsf{R}_{\text{key}}$, any $m_0, m_1 \in M$, and $C_0 \leftarrow \mathsf{Enc}(pk, m_0)$, $C_1 \leftarrow \mathsf{Enc}(pk, m_1)$, it holds that $C_0 \cdot C_1 = \mathsf{Enc}(pk, m_0 + m_1)$. Evidently, it also holds that $\mathsf{Dec}(sk, C_0 \cdot C_1) = m_0 + m_1$.

2.2 Zero-Knowledge Proofs

Let L be an $\mathcal{NP}$ language and R_L be the associated $\mathcal{NP}$ relation. For an instance x, $x \in L$ if and only if there exists a witness w such that $(x, w) \in \mathsf{R}_L$. For two interactive algorithms $\mathcal{P}$ (prover) and $\mathcal{V}$ (verifier), we denote the transcript of their interaction on inputs s and t by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$, and $\mathcal{V}$'s output by $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$, where $b \in \{0, 1\}$ with $b = 1$ indicating acceptance and $b = 0$ rejection. Below, we first recall the definition of arguments of knowledge from [BCC+16, BBB+18], which is defined via computational witness-extended emulation and was first proposed in [Lin03, GI08]. Then we recall the definition of non-interactive zero-knowledge proof from [BFM88, FLS90].

Definition 2.2 (Arguments of Knowledge (AoK))**.** An interactive protocol $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge for an $\mathcal{NP}$ relation R if it satisfies following two conditions:

- **Completeness.** For all $(x, w) \in \mathsf{R}_{pp}$ where $pp \leftarrow \mathsf{Setup}(1^\lambda)$, it holds that $\Pr[\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1] = 1$.
- **Witness-Extended Emulation.** An interactive protocol $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ has computational witness-extended emulation if for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected PPT emulator Ext such that for all interactive adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ it holds that

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}_2(tr) = 1 \ : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathcal{A}_1(pp) \\ tr \leftarrow \langle \mathcal{P}^*(x, w), \mathcal{V}(x) \rangle \end{array} \right] - \\ \Pr\left[\begin{array}{l} \mathcal{A}_2(tr) = 1 \wedge \\ (x, w') \in \mathsf{R}_{pp} \end{array} \ : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathcal{A}_1(pp) \\ (w', tr) \leftarrow \mathsf{Ext}^{\mathcal{O}}(pp, x) \end{array} \right] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathcal{O} = \langle \mathcal{P}^*(x, w), \mathcal{V}(x) \rangle$ permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.

Public-Coin. An argument of knowledge $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is called public-coin if all messages sent from the verifier to the prover are chosen uniformly at random and independently of the messages received from the prover.

Special Honest Verifier Zero-Knowledge (SHVZK). A public-coin argument of knowledge $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for language L is special honest verifier zero-knowledge if there exists a PPT simulator Sim such that given any instance $x \in L$ and any challenge e, the output of $\mathsf{Sim}(x, e)$ is distributed identically to the view of $\mathcal{V}$ in the real interaction between $\mathcal{P}(x, w)$ and $\mathcal{V}(x)$, where $(x, w) \in \mathsf{R}_L$.

Definition 2.3 (Non-Interactive Zero-Knowledge Proof (NIZK))**.** Let L be an $\mathcal{NP}$ language, and let R_L be the corresponding relation. In the common reference string (CRS) model, a NIZK for the language L consists of the following four polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter λ, outputs the public parameter pp.
- $\mathsf{CRSGen}(pp)$: on input the public parameter pp, outputs a common reference string crs.
- $\mathsf{Prove}(crs, x, w)$: on input the common reference string crs and a pair $(x, w) \in \mathsf{R}_L$, outputs a proof π.
- $\mathsf{Verify}(crs, x, \pi)$: on input the common reference string crs, a instance x, and a proof π, outputs "1" (accept) or "0" (reject).

A NIZK must satisfy the following properties:

Completeness. For any $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and any $(x, w) \in \mathsf{R}_L$, it holds that:

$$\Pr\left[\mathsf{Verify}(crs, x, \pi) = 1 : \begin{array}{l} crs \leftarrow \mathsf{CRSGen}(pp); \\ \pi \leftarrow \mathsf{Prove}(crs, x, w); \end{array}\right] = 1.$$

Adaptive Soundness. For any malicious prover Prove^*, it holds that:

$$\Pr\left[\begin{array}{c} x \notin L \wedge \\ \mathsf{Verify}(crs, x, \pi) = 1 \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ crs \leftarrow \mathsf{CRSGen}(pp); \\ (x, \pi) \leftarrow \mathsf{Prove}^*(crs); \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Adaptive Zero-Knowledge. For any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that:

$$\left|\Pr\left[\begin{array}{c} crs \leftarrow \mathsf{CRSGen}(pp); \\ (x, w) \leftarrow \mathcal{A}_1(crs); \\ \pi \leftarrow \mathsf{Prove}(crs, x, w); \\ \mathcal{A}_2(crs, x, \pi) = 1 \end{array}\right] - \Pr\left[\begin{array}{c} (crs, \tau) \leftarrow \mathsf{Sim}_1(pp); \\ (x, w) \leftarrow \mathcal{A}_1(crs); \\ \pi \leftarrow \mathsf{Sim}_2(crs, x, \tau); \\ \mathcal{A}_2(crs, x, \pi) = 1 \end{array}\right]\right| \leq \mathsf{negl}(\lambda).$$

Theorem 2.1 (Fiat-Shamir Transform [FS86, BR93, BCS16])**.** *An interactive public-coin SHVZK argument of knowledge can be converted into a non-interactive protocol that is secure and full zero-knowledge in the random oracle model, through replacing all the challenges by hashes of the transcript up to that point.*

Simulation Sound Extractability. In [BPW12], Bernhard et al. formalized a security notion for zero-knowledge proofs in the random oracle model, called simulation sound extractability. Informally, it captures that if there exists a malicious prover $\mathcal{P}^*$ who is able to output a instance x along with a valid proof π of it after seeing multiple simulated proofs and the pair (x, π) is not one of the simulated proofs, then there exists an extractor Ext that given the pair (x, π) together with the transcripts of random oracle queries and the ability of rewinding $\mathcal{P}^*$, can extract a witness of x with non-negligible probability. What's more, the authors proved that the strong Fiat-Shamir transform [FP01, Gro04] (i.e., include the statement to be proved in the hash input) yields non-interactive proofs that satisfy this security.

2.3 A Protocol for Vector Commitments to Bits

We recall a special honest-verifier zero-knowledge (SHVZK) argument of knowledge, introduced by Bootle et al. [BCC+15], for the language L_{bit} defined as below. This protocol allows a prover to demonstrate that a committed vector is a binary vector (i.e., each element equals 0 or 1).

$$L_{\text{bit}} = \{(\mathbf{g} \in \mathbb{G}^n, h, P \in \mathbb{G}) \mid \exists\ (\mathbf{a} \in \mathbb{Z}_p^n, r \in \mathbb{Z}_p) \text{ s.t. } \mathbf{a} \in \{0,1\}^n \wedge P = \mathbf{g}^{\mathbf{a}} h^r\}.$$

Figure 2 illustrates the protocol $\Pi_{\text{bit-linear}}$, an SHVZK argument of knowledge for L_{bit} with $O(n)$ proof size.

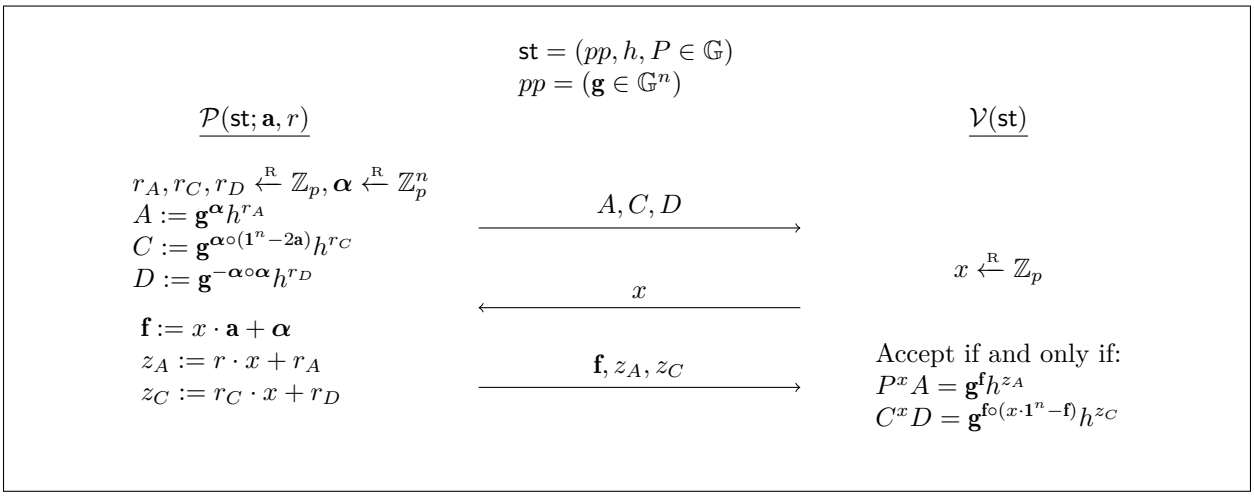

Figure 2: $\Pi_{\text{bit-linear}}$ for L_{bit}: a protocol for vector commitments to bits

2.4 Bulletproof Compression Technique

The Bulletproof protocol, introduced by Bünz et al. [BBB+18], is an efficient range proof that enables a prover to demonstrate a committed value lies within a specified range. At the core of this protocol is a compression technique [BBB+18] that facilitates a proof of knowledge for a witness $\mathbf{a} \in \mathbb{Z}_p^n$ satisfying the multi-exponentiation relation $P = \mathbf{g}^{\mathbf{a}} \in \mathbb{G}$, while achieving $O(\log n)$ communication complexity. The statement can be formalized as the language:

$$L_{\text{exp}} = \left\{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}) \mid \exists\ (\mathbf{a} \in \mathbb{Z}_p^n) \text{ s.t. } P = \mathbf{g}^{\mathbf{a}}\right\}\ .$$

Its key insight is that for a random challenge $x \xleftarrow{R} \mathbb{Z}_p$, the compressed vector $\mathbf{a}' = \mathbf{a}_L + x\mathbf{a}_R \in \mathbb{Z}_p^{n/2}$ is exactly the witness for the compressed statement $P_L^{x^2} P^x P_R = (\mathbf{g}')^{\mathbf{a}'}$, where

$$\begin{aligned}
&P_L = \mathbf{g}_L^{\mathbf{a}_R} \in \mathbb{G}, \quad P_R = \mathbf{g}_R^{\mathbf{a}_L} \in \mathbb{G}, \quad \mathbf{g}' = \mathbf{g}_L^x \circ \mathbf{g}_R \in \mathbb{G}^{n/2}, \\
&\mathbf{a}_L = (a_1, \ldots, a_{n/2}) \in \mathbb{Z}_p^{n/2}, \quad \mathbf{a}_R = (a_{n/2+1}, \ldots, a_n) \in \mathbb{Z}_p^{n/2}, \\
&\mathbf{g}_L = (g_1, \ldots, g_{n/2}) \in \mathbb{G}^{n/2}, \quad \mathbf{g}_R = (g_{n/2+1}, \ldots, g_n) \in \mathbb{G}^{n/2}.
\end{aligned}$$

By recursively applying this compression technique (i.e., $O(\log n)$ iterations), the witness size can be reduced to a scalar. We denote this proof system for L_{exp} by Π_{exp}. We omit the details of this protocol and refer the readers to [BBB+18] for the full construction. Note that the protocol Π_{exp} does not satisfy the zero-knowledge property.

Attema et al. [AC20, ACF21] developed an SHVZK variant of Π_{exp} by integrating it with the classical Σ-protocol for L_{exp}. Specifically, rather than having $\mathcal{P}$ directly send the response $\mathbf{z}$ (which scales linearly

with $|\mathbf{a}|$) to $\mathcal{V}$ for verification of $P^e A = \mathbf{g}^{\mathbf{z}}$ as in the classical Σ-protocol, the protocol instead has $\mathcal{P}$ and $\mathcal{V}$ execute Π_{exp} on input $(\mathbf{g}, P^e A; \mathbf{z})$ to prove $P^e A = \mathbf{g}^{\mathbf{z}}$, where A is $\mathcal{P}$'s initial random commitment and $e \in \mathbb{Z}_p$ is $\mathcal{V}$'s random challenge. We denote this variant by Π_{zkexp} throughout this work and refer to [ACF21] for the details of Π_{zkexp}.

Below, we recall some applications of this technique, particularly those crucial for our design of k-out-of-n proofs.

Inner-product arguments (IPA). An IPA enables a prover to demonstrate knowledge of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ such that $P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$ where $P \in \mathbb{G}$ is a public vector commitment, and $c = \langle \mathbf{a}, \mathbf{b} \rangle$ where $c \in \mathbb{Z}_p$ is a public scalar. Formally, an IPA is an argument of knowledge for the following language:

$$L_{\text{ipa}} = \left\{ (\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p) \mid \exists\ (\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) \text{ s.t. } P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle \right\}.$$

Bünz et al. [BBB+18] constructed an efficient IPA Π_{ipa} with $O(\log n)$ proof size using the compression technique.

Aggregated range proofs. Given a sequence of public commitments $V_1, \ldots, V_k$, a range proof enables a prover to demonstrate knowledge of openings of these commitments and prove that all the committed values lie in a certain range $[0, 2^n - 1]$, where n is the bit length of the range. Formally, a range proof is a protocol for the following language:

$$L_{\text{range}} = \{ (g, h \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^k) \mid \exists\ (\mathbf{v}, \mathbf{r} \in \mathbb{Z}_p^k) \text{ s.t. } \forall i \in [k] : V_i = g^{r_i} h^{v_i} \wedge v_i \in [0, 2^n - 1] \}$$

Bünz et al. [BBB+18] proposed the well-known Bulletproof range proof, via reducing L_{range} to L_{ipa} and using the IPA Π_{ipa} to prove L_{ipa}. The reduction comprises two key steps: (i) Let $\mathbf{a}_i = (a_{i1}, \ldots, a_{in}) \in \{0,1\}^n$ be the bit decomposition of v_i, i.e., $\langle \mathbf{a}_i, \mathbf{2}^n \rangle = v_i$. The prover generates a commitment $A \in \mathbb{G}$ that commits to the vector $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_k) \in \{0,1\}^{kn}$, making the proof for $v_i \in [0, 2^n - 1]$ equivalent to proving the knowledge of $\mathbf{a} \in \mathbb{Z}_p^{kn}$ satisfying:

$$\forall i : \langle \mathbf{a}_i, \mathbf{2}^n \rangle = v_i \qquad \text{and} \qquad \mathbf{a} \circ \mathbf{b} = \mathbf{0}^{kn} \qquad \text{and} \qquad \mathbf{b} = \mathbf{a} - \mathbf{1}^{kn}. \tag{1}$$

(ii) Given random challenges $y, z \in \mathbb{Z}_p$ from $\mathcal{V}$, these constraints can be combined into a single inner-product constraint:

$$\left\langle \mathbf{a} - z \cdot \mathbf{1}^{kn}, \mathbf{y}^{kn} \circ \left(\mathbf{b} + z \cdot \mathbf{1}^{kn} \right) + z^2 \cdot \mathbf{w} \right\rangle = z^2 \cdot \left(\sum_{i=1}^{k} z^{i-1} \cdot v_i \right) + \delta(y, z) \tag{2}$$

where $\mathbf{w} = \left(\mathbf{2}^n, z \cdot \mathbf{2}^n, \ldots, z^{k-1} \cdot \mathbf{2}^n \right) \in \mathbb{Z}_p^{kn}$ and $\delta(y, z) = \left(z - z^2 \right) \cdot \left\langle \mathbf{1}^{kn}, \mathbf{y}^{kn} \right\rangle - \sum_{i=1}^{k} z^{i+2} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle$.

Figure 3 depicts the full Bulletproof range proof Π_{range} [BBB+18].

Range proofs for open interval. Some applications require proving that a committed value lies within an open interval (a, b) where $a < b$ are arbitrary non-negative integers. This can be efficiently achieved by using standard range proof Π_{range} for closed intervals $[0, 2^n - 1]$, where $2^n - 1 \geq b$. Given a commitment $c = g^r h^v$, the prover demonstrates $v \in (a, b)$ through two steps: (i) prove $v - a - 1 \in [0, 2^n - 1]$ using Π_{range} on modified commitment c / g^{a+1} and (ii) prove $v + 2^n - b \in [0, 2^n - 1]$ using Π_{range} on modified commitment $c \cdot g^{2^n - b}$. This technique naturally extends to other interval types, such as range proofs for half-open interval $(a, 2^n - 1]$.

A protocol for vector commitments to bits. The range proof Π_{range} naturally yields a protocol $\Pi_{\text{bit-log}}$ for proving that a committed vector $\mathbf{a} \in \mathbb{Z}_p^n$ is a bit vector, with proof size $O(\log n)$.

To demonstrate $\mathbf{a} \in \{0,1\}^n$, the prover equivalently proves that

$$\mathbf{a} \circ \mathbf{b} = \mathbf{0}^n \qquad \text{where} \qquad \mathbf{b} = \mathbf{a} - \mathbf{1}^n, \tag{3}$$

$\mathsf{st} = (pp, \mathbf{V} \in \mathbb{G}^k, n \in \mathbb{Z}_p)$
$pp = (g, h, u \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^{kn})$

$\mathcal{P}(\mathsf{st}; \mathbf{v}, \mathbf{r})$		$\mathcal{V}(\mathsf{st})$
$\mathbf{a} := (\mathbf{a}_1, \ldots, \mathbf{a}_k) \in \{0,1\}^{kn}$ s.t. $\langle \mathbf{a}_i, \mathbf{2}^n \rangle = v_i$		
$\mathbf{b} := \mathbf{a} - \mathbf{1}^{kn}$		
$r_a, \rho \xleftarrow{R} \mathbb{Z}_p, \boldsymbol{\alpha}, \boldsymbol{\beta} \xleftarrow{R} \mathbb{Z}_p^{kn}$		
$A = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^{r_a}$	$\xrightarrow{A, S}$	
$S = \mathbf{g}^{\boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\beta}} u^{\rho}$		$y, z \xleftarrow{R} \mathbb{Z}_p$
	$\xleftarrow{y, z}$	
Define polynomials as below:		
$\mathbf{l}(X) = (\mathbf{a} - z \cdot \mathbf{1}^{kn}) + \boldsymbol{\alpha} \cdot X$		
$\mathbf{r}(X) = \mathbf{y}^{kn} \circ (\mathbf{b} + z \cdot \mathbf{1}^{kn} + \boldsymbol{\beta} \cdot X) + z^2 \cdot \mathbf{w}$		
where $\mathbf{w}$ is as in Equation (2)		
$t(X) = \langle \mathbf{l}(X), \mathbf{r}(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2$		
Compute as below:		
$\tau_1, \tau_2 \xleftarrow{R} \mathbb{Z}_p$		
$T_1 = g^{\tau_1} h^{t_1}$, $T_2 = g^{\tau_2} h^{t_2}$	$\xrightarrow{T_1, T_2}$	
		$x \xleftarrow{R} \mathbb{Z}_p$
	$\xleftarrow{x}$	
$\mathbf{l} = \mathbf{l}(x)$, $\mathbf{r} = \mathbf{r}(x)$		
$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$		
$\hat{\tau} = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot (\sum_{i=1}^{k} z^i r_i)$		
$\mu = r_a + \rho \cdot x$		
	$\xrightarrow{\hat{t}, \hat{\tau}, \mu, \mathbf{l}, \mathbf{r}}$	
		Compute:
		For $i = 0, \ldots, kn-1$, $h'_i := h_i^{-y^i}$
		$P := A \cdot S^x \cdot \mathbf{g}^{-z \cdot \mathbf{1}^{kn}} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^{kn} + z^2 \cdot \mathbf{w}}$
		Accept if and only if:
		$g^{\hat{\tau}} h^{\hat{t}} = h^{\delta(y,z)} \cdot \mathbf{V}^{z^2 \cdot \mathbf{z}^k} \cdot T_1^x \cdot T_2^{x^2}$
		$P = h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$
		$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$

Figure 3: Π_{range} for L_{range}: the Bulletproof range proof

Similarly, the two constraints can be combined into a single inner-product constraint as below:

$$\langle \mathbf{a} - z \cdot \mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{b} + z \cdot \mathbf{1}^n) \rangle = \delta(y, z) \tag{4}$$

where both $y, z \in \mathbb{Z}_p$ are random challenges from verifier and $\delta(y, z) = \left(z - z^2\right) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle$ can be computed by verifier easily.

Formally, $\Pi_{\text{bit-log}}$ is an SHVZK argument of knowledge for the language L'_{bit} defined as below:

$$L'_{\text{bit}} = \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}) \mid \exists\ (\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n, r \in \mathbb{Z}_p) \text{ s.t. } P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^r \wedge \mathbf{a} \circ \mathbf{b} = \mathbf{0}^n \wedge \mathbf{b} = \mathbf{a} - \mathbf{1}^n\}$$

Note that, while both $\Pi_{\text{bit-linear}}$ (Section 2.3) and $\Pi_{\text{bit-log}}$ are protocols for vector commitments to bits, they handle distinct languages involving different commitment structures. Specifically, $\Pi_{\text{bit-linear}}$ is for language L_{bit}, where the vector commitments have the form $P = \mathbf{g}^{\mathbf{a}} h^r$ with $\mathbf{a} \in \{0, 1\}^n$; $\Pi_{\text{bit-log}}$ is for language L'_{bit}, where vector commitments have the form $P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^r$ with $\mathbf{a} \in \{0, 1\}^n$ and $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$. The choice of protocol depends on which language needs to be proved.

Vector commitments to bits with k "1"s. Obviously, protocol $\Pi_{\text{bit-log}}$ can be easily adapted to a protocol Π_{kbit}, which enables to prove that the bit string $\mathbf{a}$ contains exactly k ones, by additionally proving $\langle \mathbf{a}, \mathbf{1}^n \rangle = k$.

3 Range Proof Friendly Homogeneous k-out-of-n Proof

As discussed in Technical Challenge 3 (Section 1.3), the k-out-of-n range proof is a key component of PPABC for multi-receiver transactions, and constructing a range proof friendly k-out-of-n proof is an effective approach. Here, "range proof friendly" property means the protocol can be seamlessly integrated with range proofs, yielding efficient k-out-of-n range proofs. In this section, we present $\Pi_{\text{homo-koon}}$, a range proof friendly homogeneous k-out-of-n proof generalizing the GK 1-out-of-n protocol [GK15]. It allows a prover to demonstrate knowledge of openings for any k-sized subset of n public commitments without revealing the openings or the subset itself, where the n commitments are generated under the same commitment key. To our knowledge, this is the first GK generalization to arbitrary $k \geq 1$ without restrictions on the subset. For consistency with the original GK technique, we index the statements starting from $j = 0$, i.e., as $(c_0, \ldots, c_{n-1})$.

3.1 Review of the GK 1-out-of-n Proof

The GK proof [GK15] enables a prover to demonstrate that one commitment c_ℓ among n public commitments $(c_0, \ldots, c_{n-1})$ commits to 0, without revealing ℓ or the opening of it. The high-level idea of the GK protocol is enabling the verifier to obtain a re-randomized commitment of c_ℓ without revealing the index ℓ, thereby proving knowledge of the opening of c_ℓ to 0 is reduced to proving knowledge of the opening of the re-randomized commitment c_ℓ^* to 0. Roughly speaking, the GK protocol realizes this high-level idea through two steps.

First, prove knowledge of a secret index $\ell \in \{0, \ldots, n-1\}$. To do this, the prover commits to the bit representation $\boldsymbol{\ell} \in \{0, 1\}^m$ of ℓ, where $m = \log n$. Then, it invokes the Σ-protocol $\Pi_{\text{bit-linear}}$ (see Section 2.3) to show the commitment is indeed to a bit string.

Second, prove knowledge of c_ℓ's opening to 0 via its re-randomization. To enable the verifier to obtain the re-randomized commitment of c_ℓ (denoted as c_ℓ^*), first note a crucial fact: the response $f_b = \ell_b x + \alpha$ in $\Pi_{\text{bit-linear}}$ is actually an evaluation of the one-degree polynomial $f_b(X) = \ell_b X + \alpha$ at the challenge x, where ℓ_b is the b-th bit of ℓ. Given this, the verifier can compute the n evaluations $(p_0(x), \ldots, p_{n-1}(x))$, where the evaluation $p_j(x) = \prod_{b=0}^{m-1} f_{b,j_b}(x)$, j_b denoting the b-th bit of index j, $f_{b,1}(X) = f_b(X)$ and $f_{b,0}(X) = X - f_b(X)$. The polynomial $p_j(X)$ is of degree m and monic if and only if $j = \ell$. Specifically, it is of the form

$$p_j(X) = \delta_{j\ell} X^m + \sum_{d=0}^{m-1} p_{j,d} X^d, \tag{5}$$

where $\delta_{j\ell} = 1$ if $j = \ell$; $\delta_{j\ell} = 0$, otherwise, and $p_{j,d}$ is the coefficient of the d-th term of the polynomial $p_j(X)$. Due to the structure of the polynomials $p_j(X)$, the verifier can *pick out* c_ℓ^* through the following steps: (i) multi-exponentiating $(c_0, \ldots, c_{n-1})$ with the n evaluations $(p_0(x), \ldots, p_{n-1}(x))$; (ii) canceling the lower-order terms via m correction terms $c_{\deg_0}, \ldots, c_{\deg_{m-1}}$. Namely, c_ℓ^* can be computed via the following equation:

$$(c_\ell^*)^{x^m} = \prod_{j=0}^{n-1} c_j^{p_j(x)} \cdot \prod_{d=0}^{m-1} c_{\deg_d}^{-x^d}. \tag{6}$$

It is important to note that the correction terms are randomized through commitments to 0, ensuring that the extracted c_ℓ^* remains a commitment to 0. Since c_ℓ^* is a re-randomized version of c_ℓ, the prover can demonstrate knowledge of c_ℓ's opening to 0 by revealing the opening of c_ℓ^*. Building on this idea, the GK protocol can be readily adapted to prove knowledge of an opening (including both the committed value and the randomness), rather than an opening to 0. This can be achieved by using commitments to random values to construct the m correction terms, so that the picked-out commitment c_ℓ^* becomes a random commitment to a random value. By revealing the opening of c_ℓ^*, the prover thereby demonstrates knowledge of the opening of the secret commitment c_ℓ. In fact, the k-out-of-n proof presented in the following section builds on this generalization, and can be easily adjusted to prove knowledge of openings to 0 if needed.

Range proof friendliness. Building on the "pick out" mechanism, it is straightforward to extend the GK protocol to an efficient 1-out-of-n range proof. To prove both knowledge of an opening and that the committed value lies in a given range, the prover can still use commitments to 0 to generate the correction terms, ensuring that c_ℓ^* remains a commitment to the same value as c_ℓ. Thus, proving c_ℓ commits to a value $v \in [a, b]$ is then equivalent to proving c_ℓ^* commits to a value $v \in [a, b]$. By using c_ℓ^* as a bridge, the 1-out-of-n proof can seamlessly integrate with range proofs, enabling efficient 1-out-of-n range proofs.

3.2 Generalization of the GK Proof

Below, we construct the range proof friendly homogeneous k-out-of-n proof $\Pi_{\text{homo-koon}}$ in three steps. Formally, $\Pi_{\text{homo-koon}}$ is an SHVZK argument of knowledge for the following language:

$$\begin{aligned} L_{\text{homo-koon}} =& \{(g, h \in \mathbb{G}, \mathbf{c} \in \mathbb{G}^n, k \in [n]) \mid \exists\ S \subseteq \{0, \ldots, n-1\}, (v_j, r_j)_{j \in S} \in \mathbb{Z}_p^{2k} \text{ s.t.} \\ & |S| = k \wedge \forall j \in S : g^{r_j} h^{v_j} = c_j\}. \end{aligned}$$

First, prove knowledge of k secret indices $S = \{\ell_1, \ldots, \ell_k\}$ and corresponding openings via running the GK protocol k times in parallel. By doing this, the verifier is able to obtain k responses $\mathbf{f}_1, \ldots, \mathbf{f}_k$, and each $\mathbf{f}_i = (f_{i,0}, \ldots, f_{i,m-1}) \in \mathbb{Z}_p^m$ can be used to compute evaluations of polynomials $(p_{i,0}(X), \ldots, p_{i,n-1}(X))$ at the challenge x, as in the GK protocol. Similarly, each polynomial $p_{i,j}(X)$ is of degree m and monic iff $j = \ell_i$:

$$p_{i,j}(X) = \delta_{j\ell_i} X^m + \sum_{d=0}^{m-1} p_{i,j,d} X^d. \tag{7}$$

Therefore, each set of n evaluations $(p_{i,0}(x), \ldots, p_{i,n-1}(x))$ can be used to *pick out* a re-randomized commitment of c_{ℓ_i} (denoted $c_{\ell_i}^*$) by canceling out the lower-order terms via m randomized correction terms as in the GK protocol. The difference is that, in the GK protocol, the m correction terms are blinded using commitments to 0, ensuring c_ℓ^* remains a commitment to 0. In contrast, in $\Pi_{\text{homo-koon}}$, they are blinded using general commitments, so that $c_{\ell_i}^*$ becomes a random commitment to a random value. Thus, the prover can prove knowing opening of c_{ℓ_i} via directly revealing the opening of $c_{\ell_i}^*$.

As mentioned in Section 1.5, this step introduces two issues that need to be resolved. One is regarding efficiency, generating and transmitting k lists of correction terms (one per execution) incurs significant communication and computation overhead: communication requires $O(k \log n)$ group elements and $O(k \log n)$ field elements; the prover requires $O(kn \log n)$ group operations, and the verifier requires $O(kn)$ group operations. The other is regarding security, there is no guarantee that the prover knows openings of k

distinct secret commitments, as a malicious prover might reuse indices across the k proofs. The following two steps address these issues sequentially.

Second, improve the efficiency via standard random linear combination technique. Instead of generating k independent lists of correction terms, the prover computes a list of aggregated correction terms $C_{(\mathrm{c})\deg_0}, \ldots, C_{(\mathrm{c})\deg_{m-1}}$, such that the verifier is able to *pick out* the k commitments in one shot. Specifically, the verifier obtains the aggregated and re-randomized commitment $G = \left(\prod_{i=1}^{k} \left(c^*_{\ell_i}\right)^{e^{i-1}}\right)^{x^m}$ via the following equation:

$$G = \prod_{j=0}^{n-1} c_j^{\sum_i^k e^{i-1} \cdot p_{i,j}(x)} \cdot \prod_{d=0}^{m-1} C_{(\mathrm{c})\deg_d}^{-x^d}, \tag{8}$$

where e is a random challenge used for random linear combination. It is straightforward to verify that G becomes a commitment to a random value $z_H = x^m \sum e^{i-1} v_{\ell_i} + v^*$, where v^* depends on the random values β_d's from the correction terms. Thus, the prover can demonstrate knowledge of openings of $(c_{\ell_i})_{\ell_i \in S}$ via directly revealing the opening of G. With this technique, both communication and computation complexities are reduced. The group elements in the proof are reduced to $O(\log n)$, halving the proof size (while the number of field elements remains $O(k \log n)$); the prover time is reduced to $O(n \log n)$ and the verifier time is reduced to $O(n)$. The next step addresses the security issue by constructing a proof that the k indices $\{\ell_i\}_{i=1}^k$ are distinct.

Third, prove the k indices are distinct by *picking out* a vector commitment to the indication vector and showing it commits to bits. Recall a crucial fact that each k-sized set $\{\ell_i\}_{i=1}^k$, which may be a multiset, can be uniquely represented by a vector $\mathbf{s} \in \{0, \ldots, k\}^n$, where s_j denotes the number of times index j appears in the set $\{\ell_i\}_{i=1}^k$. We call $\mathbf{s}$ the indication vector of this set and it holds $\langle \mathbf{s}, \mathbf{1}^n \rangle = k$. What's more, if all ℓ_i's are distinct, then $\mathbf{s} \in \{0,1\}^n$. With these in mind, our high-level idea of ensuring the k indices are distinct is that, given a sequence of bases $g_0, \ldots, g_{n-1}, h_0, \ldots, h_{n-1}$ of which the DL-relation is unknown to the prover, the verifier is able to *pick out* the bases g_{ℓ_i}, h_{ℓ_i} via a similar way as before. These bases can be used to yield a commitment to $\mathbf{s}$, i.e., $\prod_{i=1}^k g_{\ell_i} h_{\ell_i} \cdot \prod_{j=0}^{n-1} h_j^{-1} \cdot u^{r^*} = \mathbf{g}^{\mathbf{s}} \mathbf{h}^{\mathbf{s}-\mathbf{1}} u^{r^*}$. Specifically, it can be computed through the following equation:

$$P' = \left(\prod_{j=0}^{n-1} (h_j)^{-x^m}\right) \prod_{i=1}^{k} \prod_{j=0}^{n-1} (g_j h_j)^{p_{i,j}(x)} \cdot \prod_{d=0}^{m-1} C_{(\mathrm{p})\deg_d}^{-x^d}, \tag{9}$$

where $C_{(\mathrm{p})\deg_0}, \ldots, C_{(\mathrm{p})\deg_{m-1}}$ are the correction terms sent by the prover, and r^* is a randomness depend on the randomnesses in these correction terms. Finally, the prover convinces the verifier that all ℓ_i's are distinct by proving the resulting commitment commits to a bit vector, which can be achieved via the protocol $\Pi_{\text{bit-log}}$ (Section 2.4) with $O(\log n)$ proof size.

Figure 4 presents the complete $\Pi_{\text{homo-koon}}$ protocol.

Theorem 3.1. *Protocol* $\Pi_{\text{homo-koon}}$ *as depicted in Figure 4 is an SHVZK argument of knowledge for* $L_{\text{homo-koon}}$.

Proof. We separately argue its completeness, witness-extended emulation and SHVZK properties.

Completeness. As has been noted, $\prod_{b=0}^{m-1} f_{i,b,i_b}$ is a polynomial of the form $p_{i,j}(x) = \delta_{j\ell_i} x^m + \sum_{d=0}^{m-1} p_{i,j,d} x^d$, and thus

$$\sum_{i=1}^{k} p_{i,j}(x) = \sum_{i=1}^{k} \delta_{j\ell_i} x^m + \sum_{i=1}^{k} \sum_{d=0}^{m-1} p_{i,j,d} x^d,$$

$$\sum_{i=1}^{k} e^{i-1} p_{i,j}(x) = \sum_{i=1}^{k} e^{i-1} \delta_{j\ell_i} x^m + \sum_{i=1}^{k} e^{i-1} \left(\sum_{d=0}^{m-1} p_{i,j,d} x^d\right).$$

$\mathsf{st} = (pp, k \in [n], (c_0, \ldots, c_{n-1}) \in \mathbb{G}^n)$
$pp = (g, h, u \in \mathbb{G}, \mathbf{g}' \in \mathbb{G}^{mk}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^n)$

$\mathcal{P}(\mathsf{st}; S, (v_j, r_j)_{j \in S})$		$\mathcal{V}(\mathsf{st})$
Parse $S = \{\ell_1, \ldots, \ell_k\}$		
$\mathbf{L} := (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_k) \in \{0,1\}^{mk}$ s.t. $\forall i: \langle \boldsymbol{\ell}_i, \mathbf{2}^m \rangle = \ell_i$		
$\mathbf{s} \in \{0,1\}^n$ indicating set S		
$r_B, r_P \xleftarrow{R} \mathbb{Z}_p$, $B := \mathbf{g'}^{\mathbf{L}} u^{r_B}$		
$P := \mathbf{g}^{\mathbf{s}} \mathbf{h}^{\mathbf{s}-\mathbf{1}^n} \cdot u^{r_P}$		
$(state_{\text{bit-linear}}, A, C, D) \leftarrow \mathcal{P}_{\text{bit-linear}}(pp, B; \mathbf{L}, r_B)$		
	$\xrightarrow{A, B, C, D, P}$	
	$\xleftarrow{e}$	$e \xleftarrow{R} \mathbb{Z}_p$
$\alpha_0, \beta_0, \rho_0, \ldots, \alpha_{m-1}, \beta_{m-1}, \rho_{m-1} \xleftarrow{R} \mathbb{Z}_p$		
For $d = 0, \ldots, m-1$		
$C_{(\text{c})\deg_d} := \prod_{j=0}^{n-1} c_j^{\sum_{i=1}^k e^{i-1} \cdot p_{i,j,d}} \cdot g^{\alpha_d} h^{\beta_d}$		
$C_{(\text{p})\deg_d} := \prod_{j=0}^{n-1} (g_j h_j)^{\sum_{i=1}^k p_{i,j,d}} \cdot u^{\rho_d}$		
where $p_{i,j,d}$ is as in Equation (7)	$\xrightarrow{(C_{(\text{c})\deg_d}, C_{(\text{p})\deg_d})_{d=0}^{m-1}}$	
	$\xleftarrow{x}$	$x \xleftarrow{R} \mathbb{Z}_p$
$(\mathbf{f}, z_A, z_C) \leftarrow \mathcal{P}_{\text{bit-linear}}(state_{\text{bit-linear}}, x)$		
where $\mathbf{f} = (\mathbf{f}_i)_{i=1}^k \in \mathbb{Z}_p^{mk}$, and $\forall i: \mathbf{f}_i \in \mathbb{Z}_p^m$		
$z_G := \left(\sum_{i=1}^k e^{i-1} \cdot r_{\ell_i}\right) \cdot x^m - \sum_{d=0}^{m-1} \alpha_d \cdot x^d$		
$z_H := \left(\sum_{i=1}^k e^{i-1} \cdot v_{\ell_i}\right) \cdot x^m - \sum_{d=0}^{m-1} \beta_d \cdot x^d$		
$z_P := r_P \cdot x^m + \left(\sum_{d=0}^{m-1} \rho_d \cdot x^d\right)$	$\xrightarrow{\mathbf{f}, z_A, z_C, z_G, z_H, z_P}$	
		$P' := \left(\prod_{j=0}^{n-1} h_j\right)^{-x^m} \prod_{j=0}^{n-1} (g_j h_j)^{\sum_{i=1}^k p_{i,j}} \cdot \prod_{d=0}^{m-1} C_{(\text{p})\deg_d}^{-x^d}$
		$G := \prod_{j=0}^{n-1} c_j^{\sum_{i=1}^k e^{i-1} \cdot p_{i,j}} \cdot \prod_{d=0}^{m-1} C_{(\text{c})\deg_d}^{-x^d}$
		where $p_{i,j}$ is computed using $\mathbf{f}_i$ and index j

Run $\Pi_{\text{bit-log}}$ (Section 2.4) on input $(pp, P; \mathbf{s}, \mathbf{s} - \mathbf{1}^n, r_P)$
// prove P is a commitment to bits

Accept if and only if:
$\mathcal{V}_{\text{bit-linear}}(pp, B, A, C, D, x, \mathbf{f}, z_A, z_C) = 1$
$G = g^{z_G} h^{z_H}$
$P^{x^m} / P' = u^{z_P}$

Figure 4: $\Pi_{\text{homo-koon}}$ for $L_{\text{homo-koon}}$: homogeneous k-out-of-n proof

Let $\boldsymbol{\delta} = (\sum_{i=1}^{k} \delta_{0\ell_i}, \ldots, \sum_{i=1}^{k} \delta_{(n-1)\ell_i})$ (which exactly equals to the indication vector of the set $\{\ell_i\}_{i=1}^{k}$) and $\boldsymbol{\delta_e} = (\sum_{i=1}^{k} e^{i-1}\delta_{0\ell_i}, \ldots, \sum_{i=1}^{k} e^{i-1}\delta_{(n-1)\ell_i})$, then we have

$$P' = \mathbf{g}^{x^m \boldsymbol{\delta}} \mathbf{h}^{(\boldsymbol{\delta}-\mathbf{1}^n)x^m} u^{-\sum_{d=0}^{m-1} \rho_d x^d} = P^{x^m} \cdot u^{-z_P},$$

$$G = \mathbf{c}^{x^m \boldsymbol{\delta_e}} g^{-\sum_{d=0}^{m-1} \alpha_d x^d} h^{-\sum_{d=0}^{m-1} \beta_d x^d} = g^{z_G} h^{z_H}.$$

When ℓ_i's are distinct, P^{-x^m} is a commitment to bit vector. With these in mind, the completeness is straightforward.

Witness-extended emulation. In order to prove witness-extended emulation, we construct an extractor Ext as follows. The extractor runs the prover with k different values of e, $m+1$ different values of x, and additionally it invokes the extractor $\mathsf{Ext}_{\text{bit-log}}$ for the protocol $\Pi_{\text{bit-log}}$ on each of the transcripts. This results in $k \cdot (m+1) \cdot O(\lambda)$ valid proof transcripts.

For each three transcripts with different values of x, Ext runs the extractor $\mathsf{Ext}_{\text{bit-linear}}$ for the protocol $\Pi_{\text{bit-linear}}$ to extract witnesses $\mathbf{L} = (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_k)$ and r_B such that $B = \mathbf{g'}^{\mathbf{L}} u^{r_B}$ and $\forall i \in [k]$, $\boldsymbol{\ell}_i \in \{0,1\}^m$. Then it is easy to compute the committed value $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{Z}_p^{mk}$ in the vector commitment A. Unless the prover breaks the binding property of the vector Pedersen commitment scheme which is based on the DL assumption, it must hold for all challenges x's that the extracted $\mathbf{L}$ and $\mathbf{a}$ are identical, and $\forall i \in [k]$, $\mathbf{f}_i = \boldsymbol{\ell}_i x + \mathbf{a}_i$. For each $i \in [k]$, Ext extracts ℓ_i through computing $\ell_i = \langle \boldsymbol{\ell}_i, \mathbf{2}^m \rangle$. Next, we argue that all $\ell_1, \ldots, \ell_k$ are distinct and show how to extract openings of $c_{\ell_1}, \ldots, c_{\ell_k}$.

Before this, we first note that the form of the $\mathbf{f}_i$'s gives us that $\mathbf{f}_{i,1} = \boldsymbol{\ell}_i x + \mathbf{a}_i$ and $\mathbf{f}_{i,0} = (\mathbf{1}^m - \boldsymbol{\ell}_i)x - \mathbf{a}_i$. For all $i \in [k]$ and $j \in \{0, \ldots, n-1\}$ we therefore get that $p_{i,j}(x) = \delta_{j\ell_i} x^m + \cdots$. Moreover, we have $\sum_{i=1}^{k} p_{i,j}(x) = \sum_{i=1}^{k} \delta_{j\ell_i} x^m + \cdots$ and $\sum_{i=1}^{k} e^{i-1} p_{i,j}(x) = \sum_{i=1}^{k} e^{i-1}\delta_{j\ell_i} x^m + \cdots$. This means we can rewrite P' and G as

$$P' = (\mathbf{g}^{\boldsymbol{\delta}} \mathbf{h}^{\boldsymbol{\delta}-\mathbf{1}^n})^{x^m} \cdot \prod_{d=0}^{m-1} C_{p_d^*}^{x^d} \qquad \text{and} \qquad G = \mathbf{c}^{x^m \boldsymbol{\delta_e}} \cdot \prod_{d=0}^{m-1} C_{c_d^*}^{x^d}$$

where $\boldsymbol{\delta} = (\sum_{i=1}^{k} \delta_{0\ell_i}, \ldots, \sum_{i=1}^{k} \delta_{(n-1)\ell_i})$, $\boldsymbol{\delta_e} = (\sum_{i=1}^{k} e^{i-1}\delta_{0\ell_i}, \ldots, \sum_{i=1}^{k} e^{i-1}\delta_{(n-1)\ell_i})$, and the commitments $(C_{p_d^*})_{d=0}^{m-1}$, $(C_{c_d^*})_{d=0}^{m-1}$ can be computed using the statements and the $(C_{(\text{c})\deg_d}, C_{(\text{p})\deg_d})_{d=0}^{m-1}$.

Now, we argue that all $\ell_1, \ldots, \ell_k$ are distinct. For each $O(\lambda)$ transcripts with same value of e and x, Ext first runs $\mathsf{Ext}_{\text{bit-log}}$ to extract witnesses $\mathbf{s}, r_P$ such that $P = \mathbf{g}^{\mathbf{s}} \mathbf{h}^{\mathbf{s}-\mathbf{1}^n} u^{r_P}$. Unless the prover knows a non-trivial DL-relation between independent generators $\mathbf{g}, \mathbf{h}, u$, it must hold for all different transcripts that the extracted witnesses $\mathbf{s}, r_P$ are identical and $\mathbf{s} \in \{0,1\}^n$. Using the $m+1$ valid transcripts with different challenges $x_1, \ldots, x_{m+1}$, Ext can find a vector $\mathbf{t} = (t_1, \ldots, t_{m+1}) \in \mathbb{Z}_p^{m+1}$ such that $\sum_{o=1}^{m+1} t_o \cdot (x_o^m, \ldots, x_o^0) = (1, 0, \ldots, 0)$. Based on the verification equation of P' and P, we therefore have

$$(\mathbf{g}^{\boldsymbol{\delta}} \mathbf{h}^{\boldsymbol{\delta}-\mathbf{1}^n})/P = \prod_{o=1}^{m+1} (\mathbf{c}^{x_o^m \boldsymbol{\delta_e}} \cdot \prod_{d=0}^{m-1} C_{c_d^*}^{x_o^d} / P^{x_o^m})^{t_o} = u^{-\sum_{o=1}^{m+1} t_o z_{P_o}}.$$

Since $P = \mathbf{g}^{\mathbf{s}} \mathbf{h}^{\mathbf{s}-\mathbf{1}^n} u^{r_P}$, unless the prover knows a non-trivial DL-relation between independent generators $\mathbf{g}, \mathbf{h}, u$, it must hold $\boldsymbol{\delta} = \mathbf{s}$, i.e., $\boldsymbol{\delta} \in \{0,1\}^n$. Furthermore, it is evident that $\langle \boldsymbol{\delta}, \mathbf{1}^n \rangle = \sum_{j=0}^{n-1} \sum_{i=1}^{k} \delta_{j\ell_i} = k$. With these in mind, we can infer that all $\ell_1, \ldots, \ell_k$ are distinct.

Finally, we show how Ext extracts openings of $c_{\ell_1}, \ldots, c_{\ell_k}$. Similar to before, using the $m+1$ valid transcripts with same challenge e, and different challenges $x_1, \ldots, x_{m+1}$, Ext can find a vector $\mathbf{t} = (t_1, \ldots, t_{m+1}) \in \mathbb{Z}_p^{m+1}$ such that $\sum_{o=1}^{m+1} t_o \cdot (x_o^m, \ldots, x_o^0) = (1, 0, \ldots, 0)$. Based on the verification equation of G, we therefore have

$$\prod_{i=1}^{k} c_{\ell_i}^{e^{i-1}} = \mathbf{c}^{\boldsymbol{\delta_e}} = \prod_{o=1}^{m+1} (\mathbf{c}^{x_o^m \boldsymbol{\delta_e}} \cdot \prod_{d=0}^{m-1} C_{c_d^*}^{x_o^d})^{t_o} = g^{\sum_{o=1}^{m+1} t_o z_{G_o}} h^{\sum_{o=1}^{m+1} t_o z_{H_o}}.$$

Thus, for each e_s, $s \in [k]$, we have $\prod_{i=1}^{k} c_{\ell_i}^{e_s^{i-1}} = g^{\sum_{o=1}^{m+1} t_{s,o} z_{G_{s,o}}} h^{\sum_{o=1}^{m+1} t_{s,o} z_{H_{s,o}}}$. Furthermore, since $e_1, \ldots, e_k$ are distinct, Ext can find k vectors $\mathbf{t}_1, \ldots, \mathbf{t}_k \in \mathbb{Z}_p^k$ such that for each $\mathbf{t}_i$, $\sum_{s=1}^{k} t_{i,s} \cdot (e_s^0, \ldots, e_s^{k-1}) = \mathbf{e}_i$, where $\mathbf{e}_i \in \{0,1\}^k$ with only one "1" at the i-th position. Thus, for each c_{ℓ_i} we have

$$c_{\ell_i} = \prod_{s=1}^{k} (c_{\ell_1}^{e_s^0} \cdots c_{\ell_k}^{e_s^{k-1}})^{t_{i,s}} = g^{\sum_{s=1}^{k} t_{i,s} \cdot (\sum_{o=1}^{m+1} t_{s,o} z_{G_{s,o}})} h^{\sum_{s=1}^{k} t_{i,s} \cdot (\sum_{o=1}^{m+1} t_{s,o} z_{H_{s,o}})}.$$

This gives us an extracted opening of c_{ℓ_i} to $v_i = \sum_{s=1}^{k} t_{i,s} \cdot (\sum_{o=1}^{m+1} t_{s,o} z_{H_{s,o}})$ with randomness $r_i = \sum_{s=1}^{k} t_{i,s} \cdot (\sum_{o=1}^{m+1} t_{s,o} z_{G_{s,o}})$.

SHVZK. Given statements $\mathbf{c}$, challenges e, x, challenges $chal_{\text{bit-log}}$ for sub-protocol $\Pi_{\text{bit-log}}$, the simulator Sim works as follows:

1. Pick $B, P \in \mathbb{G}$ at random;
2. Run sub-protocol $\Pi_{\text{bit-linear}}$'s simulator $\mathsf{Sim}_{\text{bit-linear}}$ with input B as statement and x as challenge, obtain $A, C, D \in \mathbb{G}$, $\mathbf{f_1}, \ldots, \mathbf{f}_k \in \mathbb{Z}_p^m$, and $z_A, z_C \in \mathbb{Z}_p$;
3. Run sub-protocol $\Pi_{\text{bit-log}}$'s simulator $\mathsf{Sim}_{\text{bit-log}}$ with input P as statement and $chal_{\text{bit-log}}$ as challenge, obtain $\mathcal{V}$'s views $\text{View}_{\text{bit-log}}$ in $\Pi_{\text{bit-log}}$;
4. Pick z_G, z_H, z_P at random, compute $P' = P^{x^m} / u^{-z_P}$ and $G = g^{z_G} h^{z_H}$;
5. Pick $(C_{(\text{c})\deg_d}, C_{(\text{p})\deg_d})_{d=1}^{m-1} \in \mathbb{G}^{2m-2}$ at random, compute

$$C_{(\text{c})\deg_0} = G^{-1} \cdot \prod_{j=0}^{n-1} c_j^{\sum_{i=1}^{k} e^{i-1} \cdot p_{i,j}} \cdot \prod_{d=1}^{m-1} C_{(\text{c})\deg_d}^{-x^d},$$

$$C_{(\text{p})\deg_0} = (P')^{-1} \cdot \left(\prod_{j=0}^{n-1} h_j \right)^{-x^m} \prod_{j=0}^{n-1} (g_j h_j)^{\sum_{i=1}^{k} p_{i,j}} \cdot \prod_{d=1}^{m-1} C_{(\text{p})\deg_d}^{-x^d};$$

6. Return the simulated transcript $A, B, C, D, P, e, (C_{(\text{c})\deg_d}, C_{(\text{p})\deg_d})_{d=0}^{m-1}$, x, $(\mathbf{f}_i)_{i=1}^{k}$, z_A, z_C, z_G, z_H, z_P and $\text{View}_{\text{bit-log}}$.

Apparently, in both real proofs and simulated proofs, $B, P, (C_{(\text{c})\deg_d})_{d=1}^{m-1}$ and $(C_{(\text{p})\deg_d})_{d=1}^{m-1}$ are uniformly random in $\mathbb{G}$ and z_G, z_H, z_P are uniformly random in $\mathbb{Z}_p$. Moreover, based on the perfect SHVZK property of $\Pi_{\text{bit-linear}}$ and $\Pi_{\text{bit-log}}$, the simulated views $(A, C, D, (\mathbf{f}_i)_{i=1}^{k}, z_A, z_C)$ and $\text{View}_{\text{bit-log}}$ are identically distributed to those of real ones. Furthermore, both in real and in simulated proofs, the verification equations uniquely determine $C_{(\text{c})\deg_0}$ conditioned on $\mathbf{c}, e, (C_{(\text{c})\deg_d})_{d=1}^{m-1}, x, (\mathbf{f}_i)_{i=1}^{k}$, z_G, z_H, and determine $C_{(\text{p})\deg_0}$ conditioned on $\mathbf{g}, \mathbf{h}, u, (C_{(\text{p})\deg_d})_{d=1}^{m-1}, x, (\mathbf{f}_i)_{i=1}^{k}, z_P$. Therefore, the simulated proofs and the real proofs have identical probability distributions.

This proves Theorem 3.1. □

Efficiency analysis. The prover's complexity is dominated by the evaluation of k vectors of polynomials, which requires $O(kn \log n)$ field operations; the computation of m correction terms, each involving a multi-exponentiation of size n; and the computation within the protocol $\Pi_{\text{bit-log}}$, which requires $O(n)$ group operations. The verifier's complexity is dominated by computing all $p_{i,j}$ values, which requires $O(kn \log n)$ field operations in total; the computation of the group elements P' and G, each requiring $O(n)$ group operations; and the computation within the protocol $\Pi_{\text{bit-log}}$, which also requires $O(n)$ group operations. The proof size is dominated by the $2m$ correction terms (i.e., $2m$ group elements), the k vectors $\mathbf{f}_1, \ldots, \mathbf{f}_k$ (i.e., km field elements), and the proof of $\Pi_{\text{bit-log}}$, which consists of $O(\log n)$ group elements. Overall, both the prover's and the verifier's complexities in the k-out-of-n proof are $O(kn \log n)$ field operations and $O(n)$ group operations. The proof size is $O(\log n)$ group elements and $O(k \log n)$ field elements.

An optimization for the case $k = 1$. Our GK-based homogeneous k-out-of-n proof is primarily designed for the setting $k \geq 2$. When $k = 1$, however, there is no need to "pick out" the commitment to the indicator vector or to prove that the committed vector is a bit string containing exactly one "1". As a result, the $k = 1$ case admits a significant simplification, and the homogeneous k-out-of-n proof reduces directly to the original GK proof. The same applies to the homogeneous k-out-of-n range proof introduced in the next subsection, which also simplifies in the case $k = 1$ and coincides with the original GK range proof.

3.3 The Homogeneous k-out-of-n Range Proof

Our homogeneous k-out-of-n proof preserves the range proof friendliness of the GK protocol, as the verifier can *pick out* the k secret commitments $(c_j)_{j\in S}$ (in the aggregated form). This property enables seamless integration with range proofs (which support batch verification) through minor adaptations. Below, we show how to combine our k-out-of-n proof $\Pi_{\text{homo-koon}}$ with the well-known Bulletproof range proof (see Section 2.4), yielding an efficient k-out-of-n range proof $\Pi^{\text{range}}_{\text{homo-koon}}$. Concretely, $\Pi^{\text{range}}_{\text{homo-koon}}$ is an SHVZK argument of knowledge for the following language:

$$\begin{aligned} L_{\text{homo-koon-range}} =& \{(g, h \in \mathbb{G}, \mathbf{c} \in \mathbb{G}^n, k \in [n] \mid \exists\, S \subseteq \{0, \ldots, n-1\}, (v_j, r_j)_{j\in S} \in \mathbb{Z}_p^{2k} \\ & \text{s.t. } |S| = k \wedge \forall j \in S : g^{r_j} h^{v_j} = c_j \wedge v_j \in [0, 2^\nu - 1]\}. \end{aligned}$$

As established earlier, the commitment G computed by the verifier in $\Pi_{\text{homo-koon}}$ aggregates the k secret commitments $(c_j)_{j\in S}$. While the most straightforward approach would be to directly substitute G into Bulletproof's verification equation, a critical limitation is that G is a commitment to a random value z_H (as shown in Figure 4), rather than the target aggregated value $\sum_{i=1}^{k} e^{i-1} v_{\ell_i}$. This occurs because the prover employs a random commitment (i.e., $g^{\alpha_d} h^{\beta_d}$) as blinding commitments in the correction term $C_{(\text{c})\deg_d}$, thereby masking both the values $\sum e^{i-1} v_{\ell_i}$ and the randomnesses $\sum e^{i-1} r_{\ell_i}$. Fortunately, this can be resolved via a slight modification to $\Pi_{\text{homo-koon}}$, as we have mentioned in the GK 1-out-of-n range proof. When generating the correction terms $C_{(\text{c})\deg_d}$, the prover uses commitments to zero (i.e., $g^{\alpha_j} h^0$) rather than commitments to random values as blinding commitments. This modification transforms $G^{x^{-m}}$ into a commitment to $\sum_{i=1}^{k} e^{i-1} v_{\ell_i}$, which can be directly embedded in Bulletproof's verification equation, resulting in an efficient homogeneous k-out-of-n range proof.

Figure 5 shows the complete homogeneous k-out-of-n range proof $\Pi^{\text{range}}_{\text{homo-koon}}$ for $L_{\text{homo-koon-range}}$. Note that the sub-protocol $\Pi_{\text{homo-koon}}$ is the modified version described above.

Theorem 3.2. *Protocol $\Pi^{\text{range}}_{\text{homo-koon}}$ in Figure 5 is an SHVZK argument of knowledge for $L_{\text{homo-koon-range}}$.*

Sketch. The completeness is straightforward, thus we focus on the witness-extended emulation and SHVZK properties below.

Witness-extended emulation. Given the extractor $\mathsf{Ext}_{\text{range}}$ of the protocol Π_{range} ([BBB+18, Theorem 3]) and the extractor $\mathsf{Ext}_{\text{homo-koon}}$ of $\Pi_{\text{homo-koon}}$ (Theorem 3.1), it is relatively straightforward to construct an extractor Ext that can extract the witnesses $\boldsymbol{\ell}, \mathbf{r}, \mathbf{v}$, by making slight modifications to combine $\mathsf{Ext}_{\text{range}}$ and $\mathsf{Ext}_{\text{homo-koon}}$.

Note that Ext needs to run the prover with $k \cdot w$ different values of y, $(k+2)$ different values of z, 3 different values of x_2, $m+1$ different values of the challenge x_1, and additionally invokes the extractor $\mathsf{Ext}_{\text{bit-log}}$ for the protocol $\Pi_{\text{bit-log}}$ on each of the transcripts, resulting in $k\nu \cdot (m+1) \cdot O(\lambda)$ valid proof transcripts in total.

Below, we state the crucial step of combining $\mathsf{Ext}_{\text{range}}$ and $\mathsf{Ext}_{\text{homo-koon}}$ to extract $\mathbf{r}, \mathbf{v}$. First, for given values of y, z, x_1, using 3 transcripts with different x_2's, Ext can compute a v, γ via invoking $\mathsf{Ext}_{\text{range}}$, such that $g^\gamma h^v = G^{-x_1^m}$. Second, for given values of z, x_1, Ext can rewrite $G = (c_{\ell_1}^{z^0} \cdots c_{\ell_k}^{z^{k-1}})^{x_1^m} \cdot \prod_{d=0}^{m-1} C_{c_d^*}^{x_1^d}$, according to the extractor $\mathsf{Ext}_{\text{homo-koon}}$. It means that, for fixed values of z, x_1, we have $(g^\gamma h^v)^{-x_1^m} = (c_{\ell_1}^{z^0} \cdots c_{\ell_k}^{z^{k-1}})^{x_1^m} \cdot \prod_{d=0}^{m-1} C_{c_d^*}^{x_1^d}$. Therefore, via using $m+1$ different challenges x_1 and k different challenges z, Ext can compute $\mathbf{v}, \mathbf{r} \in \mathbb{Z}_p^k$ such that $c_{\ell_i} = g^{r_i} h^{v_i}$ for all $i \in [k]$.

The remaining steps are exactly the same as $\mathsf{Ext}_{\text{range}}$ and $\mathsf{Ext}_{\text{homo-koon}}$.

$\mathsf{st} = (pp, k \in [n], \mathbf{c} \in \mathbb{G}^n)$
$pp = (pp_{\text{koon}}, pp_{\text{range}})$

$\underline{\mathcal{P}(\mathsf{st}; S, (v_j, r_j)_{j \in S})}$ $\underline{\mathcal{V}(\mathsf{st})}$

$(state_{\text{koon}}, A, B, C, D, P) \leftarrow \mathcal{P}_{\text{homo-koon}}(pp_{\text{koon}}; S)$
$(state_{\text{range}}, V, H) \leftarrow \mathcal{P}_{\text{range}}(pp_{\text{range}}; (v_j)_{j \in S})$

$A, B, C, D, P, V, H \longrightarrow$

$y, z \xleftarrow{\text{R}} \mathbb{Z}_p$

$\longleftarrow y, z$

$(state_{\text{koon}}, (C_{(\text{c})\deg_d})_{d=0}^{m-1}, (C_{(\text{p})\deg_d})_{d=0}^{m-1}) \leftarrow \mathcal{P}_{\text{homo-koon}}(z; state_{\text{koon}})$
$(state_{\text{range}}, T_1, T_2) \leftarrow \mathcal{P}_{\text{range}}(y, z; state_{\text{range}})$

$(C_{(\text{c})\deg_d})_{d=0}^{m-1}, (C_{(\text{p})\deg_d})_{d=0}^{m-1}, T_1, T_2 \longrightarrow$

$x_1, x_2 \xleftarrow{\text{R}} \mathbb{Z}_p$

$\longleftarrow x_1, x_2$

$(state_{\text{koon}}, \mathbf{f}, z_A, z_C, z_P) \leftarrow \mathcal{P}_{\text{homo-koon}}(x_1; state_{\text{koon}})$
$(state_{\text{range}}, \mu, \hat{t}, \mathbf{l}, \mathbf{r}) \leftarrow \mathcal{P}_{\text{range}}(x_2; state_{\text{range}})$
$\tau_r = \tau_2 \cdot x_2^2 + \tau_1 \cdot x_2 + z^2 \cdot \gamma$
where $\gamma = \sum_{i=1}^{k} z^{i-1} \cdot r_i - (\sum_{d=0}^{m-1} \alpha_d \cdot x_1^d)/x_1^m$,
and (τ_1, τ_2) can be got from $state_{\text{range}}$
$\alpha_0, \ldots, \alpha_{m-1}$ can be got from $state_{\text{koon}}$

$\mathbf{f}, z_A, z_C, z_P, (\mu, \hat{t}, \mathbf{l}, \mathbf{r}, \tau_r) \longrightarrow$

Compute $G := \prod_{j=0}^{n-1} (c_j)^{\sum_{i=1}^{k} z^{i-1} \cdot p_{i,j}} \cdot \prod_{d=0}^{m-1} C_{(\text{c})\deg_d}^{-x_1^d}$
Check $\Pi_{\text{homo-koon}}$ as in Section 3.2 and
Π_{range} as in Section 2.4, except that
(i) in $\Pi_{\text{homo-koon}}$, ignore checking the opening of G
(ii) in Π_{range}, plug $G^{x_1^{-m}}$ into verification equation,
i.e., check $g^{\tau_r} h^{\hat{t}} \stackrel{?}{=} (G^{x_1^{-m}})^{z^2} \cdot h^{\delta(y,z)} \cdot T_1^{x_2} \cdot T_2^{x_2^2}$
Accept iff all the checks pass

Figure 5: $\Pi_{\text{homo-koon}}^{\text{range}}$ for $L_{\text{homo-koon-range}}$: the homogeneous k-out-of-n range proof

SHVZK. By giving the statement $k \in [n]$, $\mathbf{c} \in \mathbb{G}^n$ and the challenge $y, z, x_1, x_2 \in \mathbb{Z}_p$ and challenges $chal_{\text{bit-log}}$ for sub-protocol $\Pi_{\text{bit-log}}$, the simulator Sim works as follows:

1. Run the simulator $\mathsf{Sim}_{\text{homo-koon}}$ for $\Pi_{\text{homo-koon}}$ on input $(k, \mathbf{c}, z, x_1, chal_{\text{bit-log}})$, obtain A, B, C, D, P, $(C_{(\text{c})\deg_d})_{d=1}^{m-1}, (C_{(\text{p})\deg_d})_{d=0}^{m-1}, \mathbf{f}, z_A, z_C, z_P, \text{View}_{\text{bit-log}}$. As demonstrated in proof for Theorem 3.1, all these values can be simulated without z_G, which is not explicitly given in the transcripts here.

2. Run simulator $\mathsf{Sim}_{\text{range}}$ for protocol Π_{range} on input (y, z, x_2), obtain $V, S, T_2, \mu, \hat{t}, \mathbf{l}, \mathbf{r}$. As demonstrated in proof for [BBB+18, Theorem 3], all these values can be simulated without statements, which is not explicitly given in the scenario here.

3. For the values of $C_{(\text{c})\deg_0}, T_1, \tau_r$, Sim first picks $\tau_r \in \mathbb{Z}_p$ and $T_1 \in \mathbb{G}$ at random, then computes $G := (g^{\tau_r} h^{\hat{t}} \cdot h^{-\delta(y,z)} \cdot T_1^{-x_2} \cdot T_2^{-x_2^2})^{-x_1^m z^2}$, and next computes $C_{(\text{c})\deg_0} = G^{-1} \cdot \prod_{j=0}^{n-1} c_j^{\sum_{i=1}^{k} z^{i-1} \cdot p_{i,j}} \cdot \prod_{d=1}^{m-1} C_{(\text{c})\deg_d}^{-x^d}$.

4. Return the whole transcript.

Based on the perfect SHVZK property of $\Pi_{\text{homo-koon}}$ and Π_{range}, the simulation is perfect.

This proves Theorem 3.2. □

Efficiency analysis. As the protocol $\Pi_{\text{homo-koon}}^{\text{range}}$ is a seamless combination of $\Pi_{\text{homo-koon}}$ and Π_{range}, both the computational cost and the proof size are the sum of that of these two protocols. Overall, the prover's complexity is $O(n \log n + k\nu)$ group operations and $O(kn \log n)$ field operations, the verifier's complexity is $O(n + k\nu)$ group operations and $O(kn \log n)$ field operations, and the proof size is $O(\log n + k\nu)$ group elements and $O(k \log n)$ field elements. Here, the proof size is reduced to $O(\log n + \log k + \log \nu)$ group elements via inner product argument [BBB+18].

k-out-of-n range proof for (half-)open interval. We take a simple example to demonstrate how to extend the homogeneous k-out-of-n range proofs to support proving (half-)open interval, given range proofs for closed interval $[0, 2^\nu - 1]$. For instance, to prove all v_i's lie in the open interval $(a, 2^\nu - 1]$ where a is an arbitrary non-negative integer smaller than $2^\nu - 1$, $\mathcal{P}$ and $\mathcal{V}$ invoke $\Pi_{\text{homo-koon}}^{\text{range}}$ on input $(k, \mathbf{c}; S, \mathbf{r}, \mathbf{v})$ and $(k, \mathbf{c} \cdot h^{-a-1}; S, \mathbf{r}, \mathbf{v} - (a+1) \cdot \mathbf{1}^n)$, respectively. Crucially, to ensure the same $\boldsymbol{\ell}$ is used in both instances, the two invocations must be integrated into a single protocol by sharing the $\Pi_{\text{homo-koon}}$ component. Other similar scenarios can be easily handled with the same technique.

4 Heterogeneous k-out-of-n Proofs

In the previous section, we presented the homogeneous k-out-of-n proof $\Pi_{\text{homo-koon}}$ for statements comprising n Pedersen commitments generated under a common commitment key g, h. We now consider a scenario where statements consist of commitments generated under heterogeneous keys, i.e., each commitment c_j is generated under its own commitment key pair (g_j, h_j). As discussed in Technical Challenge 1 (Section 1.3), supporting such heterogeneous statements is essential for designing PPABC resistant to insider attacks. More generally, we consider a scenario where, given n different group homomorphisms $F_1, \ldots, F_n \in \text{Hom}(\mathbb{Z}_p^t, \mathbb{G}^s)$, n vectors $\mathbf{P}_1, \ldots, \mathbf{P}_n \in \mathbb{G}^s$ and a scalar $k \in [n]$, the prover aims to demonstrate knowledge of k-sized subset $S \subseteq [n]$ and corresponding openings $(\mathbf{x}_j)_{j \in S} \in \mathbb{Z}_p^{tk}$ such that $F_j(\mathbf{x}_j) = \mathbf{P}_j$ holds for each $j \in S$. Note that $L_{\text{homo-koon}}$ is a specific case of this scenario where each F_j is defined as $F_j : (r, v) \mapsto g^r h^v \in \text{Hom}(\mathbb{Z}_p^2, \mathbb{G})$.

4.1 Review of the ACF k-out-of-n Proof

We begin by reviewing the Attema-Cramer-Fehr (ACF) proof [ACF21], which is an SHVZK argument of knowledge for language $L_{\text{homo-koon}}$ (defined in Section 3).

The core innovation of the ACF protocol lies in its reduction of the k-out-of-n case (captured by the language $L_{\text{homo-koon}}$) to the n-out-of-n case (captured by L_{amorhom} defined below). This reduction enables the use of a compressed Σ-protocol Π_{amorhom} for L_{amorhom}, which enables a prover to open n homomorphisms $H_1, \ldots, H_n$ on a single multi-exponentiation P with logarithmic complexity, ultimately yielding a compact k-out-of-n proof for $L_{\text{homo-koon}}$. Formally, the language L_{amorhom} is defined as below:

$$L_{\text{amorhom}} = \{(P, (\mathbf{P}_1, \ldots, \mathbf{P}_n)) \mid \exists\ \mathbf{y} \text{ s.t. } P = \mathbf{g}^{\mathbf{y}}, \mathbf{P}_j = H_j(\mathbf{y}), j \in [n]\}.$$

The reduction proceeds in two phases. First, the prover computes an $(n-k)$-degree polynomial $p(X) = 1 + \sum_{d=1}^{n-k} a_d \cdot X^d$ with roots at $\{j\}_{j \notin S}$, and commits to $\mathbf{y} = (a_1, \ldots, a_{n-k}, p(1)\mathbf{x}_1, \ldots, p(n)\mathbf{x}_n) \in \mathbb{Z}_p^{3n-k}$ where $\mathbf{x}_j = (r_j, v_j) \in \mathbb{Z}_p^2$, by generating $P = \mathbf{g}^{\mathbf{y}} h^{r_P}$. Note that, only the k sub-vectors $(\mathbf{x}_i)_{i \in S}$ and the polynomials coefficients $(a_i)_{i=1}^{n-k}$ are non-zero in $\mathbf{y}$. Second, the prover sends P to the verifier and proves knowledge of $\mathbf{y}$ satisfying n homomorphism equations $H_j(\mathbf{y}) = g^{y_{n-k+2j-1}} h^{y_{n-k+2j}} c_j^{-\sum_i y_i j^i} = c_j$ for $j \in [n]$, thereby reducing $L_{\text{homo-koon}}$ to L_{amorhom}. Each $H_j : \mathbb{Z}_p^{3n-k} \mapsto \mathbb{G}$ is a group homomorphism defined by $3n-k$ bases, i.e., $c_j^{-j}, \ldots, c_j^{-j^{n-k}}$, g, h, and $2n-2$ identities. Thus, the n homomorphisms can be defined by a bases matrix $\mathbf{M} \in \mathbb{G}^{n \times (3n-k)}$ (as presented in Section 1.5).

The protocol Π_{amorhom} executes as follows. First, the prover receives a random challenge $\rho \in \mathbb{Z}_p$ from the verifier. Then, both the prover and the verifier compute the aggregated commitment $G = \sum_{j=1}^{n} c_j^{\rho^j} \in \mathbb{G}$ and the aggregated homomorphism $H = \sum_{j=1}^{n} \rho^j H_j \in \text{Hom}(\mathbb{Z}_p^{3n-k}, \mathbb{G})$. The computation of H is eventually a row random linear combination of the bases matrix $\mathbf{M} \in \mathbb{G}^{n \times (3n-k)}$, which yields $3n-k$ new bases. Finally, the prover applies the Bulletproof compression technique to demonstrate knowledge of $\mathbf{y}, r_P$ satisfying both $P = \mathbf{g}^{\mathbf{y}} h^{r_P}$ and $G = H(\mathbf{y})$.

The computational costs and proof size of the ACF protocol are dominated by the sub-protocol Π_{amorhom}. Regarding proof size, although it is $O(\log(3n-k))$ in theory, the use of the Bulletproof compression technique requires the length of $\mathbf{y}$ to be a power of 2, which restricts the choices of n and k. In practice, $\mathbf{y}$ is typically padded with zeros to the next power of 2, resulting in a proof size of $O(\log n)$. Moreover, both the prover and the verifier incur substantial computational overhead. The dominant cost comes from computing H, which requires $O(n(n-k))$ exponentiations — both $\mathcal{P}$ and $\mathcal{V}$ require computing the $n-k$ bases $\prod_{j=1}^{n} c_j^{\rho^j j^i}$ for $i \in [n-k]$, each of which needs n exponentiations. It is rather inefficient when n is large.

4.2 The Heterogeneous k-out-of-n Proof

Below, we present our heterogeneous k-out-of-n proof $\Pi_{\text{hetero-koon}}$. Formally, $\Pi_{\text{hetero-koon}}$ is an SHVZK argument of knowledge for the following language:

$$\begin{aligned} L_{\text{hetero-koon}} = \{&(F_1, \ldots, F_n \in \text{Hom}(\mathbb{Z}_p^t, \mathbb{G}^s), \mathbf{P}_1, \ldots, \mathbf{P}_n \in \mathbb{G}^s, k \in [n]) \mid \\ &\exists\ S \subseteq [n], (\mathbf{x}_j)_{j \in S} \in \mathbb{Z}_p^{tk} \text{ s.t. } |S| = k \wedge \forall j \in S : F_j(\mathbf{x}_j) = \mathbf{P}_j\}. \end{aligned}$$

As mentioned in Section 1.5, the $\Pi_{\text{hetero-koon}}$ is constructed via two steps.

First, improve the computation complexity of the ACF protocol. This is achieved by expressing the secret set S via the indication vector $\mathbf{s} \in \{0,1\}^n$, where $s_j = 1$ if $j \in S$ and $s_j = 0$ otherwise. In this case, the extended witness $\mathbf{y} = (\mathbf{s}, s_1\mathbf{x}_1, \ldots, s_n\mathbf{x}_n) \in \mathbb{Z}_p^{3n}$ satisfies n simplified homomorphisms constraints: $H_j(\mathbf{y}) = g^{y_{n+2j-1}} h^{y_{n+2j}} c_j^{-y_i} = 1$, $j \in [n]$. The bases matrix $\mathbf{M} \in \mathbb{G}^{n \times 3n}$ of these homomorphisms is much sparser than that in the ACF protocol, thereby reducing the cost of computing $H = \sum_{j=1}^{n} \rho^j H_j$ from $O(n(n-k))$ to $O(n)$.

Second, extend the protocol to support opening distinct group homomorphisms without incurring additional costs. To support distinct group homomorphisms, our key insight is that the reduction from $L_{\text{homo-koon}}$

to L_{amorhom} operates independently across instances. Each commitment c_j is tied only to its corresponding homomorphism H_j, and is independent of the others. This observation enables to extend the protocol to the heterogeneous setting. Concretely, to demonstrate knowledge of a k-sized subset S and $(\mathbf{x}_j)_{j\in S}$ satisfying $\mathbf{P}_j = F_j(\mathbf{x}_j)$ for each $j \in S$, the prover sets the n homomorphisms H_j as $H_j(\mathbf{y}) = F_j(\mathbf{y}_j)\mathbf{P}_j^{-y_j}$, $j \in [n]$. Here, $\mathbf{y}_j = (y_{n+jt-t+1}, \ldots, y_{n+jt})$ is intended to be $s_j\mathbf{x}_j$. Then, the prover invokes the protocol Π_{amorhom} to prove knowledge of $\mathbf{y}$.

Roughly speaking, our heterogeneous k-out-of-n proof for language $L_{\text{hetero-koon}}$ proceeds as below. First, the prover sets the vector $\mathbf{y} = (\mathbf{s}, s_1\mathbf{x}_1, \ldots, s_n\mathbf{x}_n) \in \mathbb{Z}_p^{n+tn}$, and commits to $\mathbf{y}$, $\mathbf{s}$. Then, the prover proves that the committed values $\mathbf{y}$ and $\mathbf{s}$ satisfy the following two conditions:

- $H_j(\mathbf{y}) := F_j(\mathbf{y}_j) \cdot \mathbf{P}_j^{-y_j} = \mathbf{1}^s$, $j \in [n]$;
- $\mathbf{y}_0 = \mathbf{s} \in \{0,1\}^n$ where $\mathbf{y}_0 = (y_i)_{i=1}^n \in \mathbb{Z}_p^n$, and $\langle \mathbf{s}, \mathbf{1}^n \rangle = k$.

Figure 6 shows the complete protocol $\Pi_{\text{hetero-koon}}$.

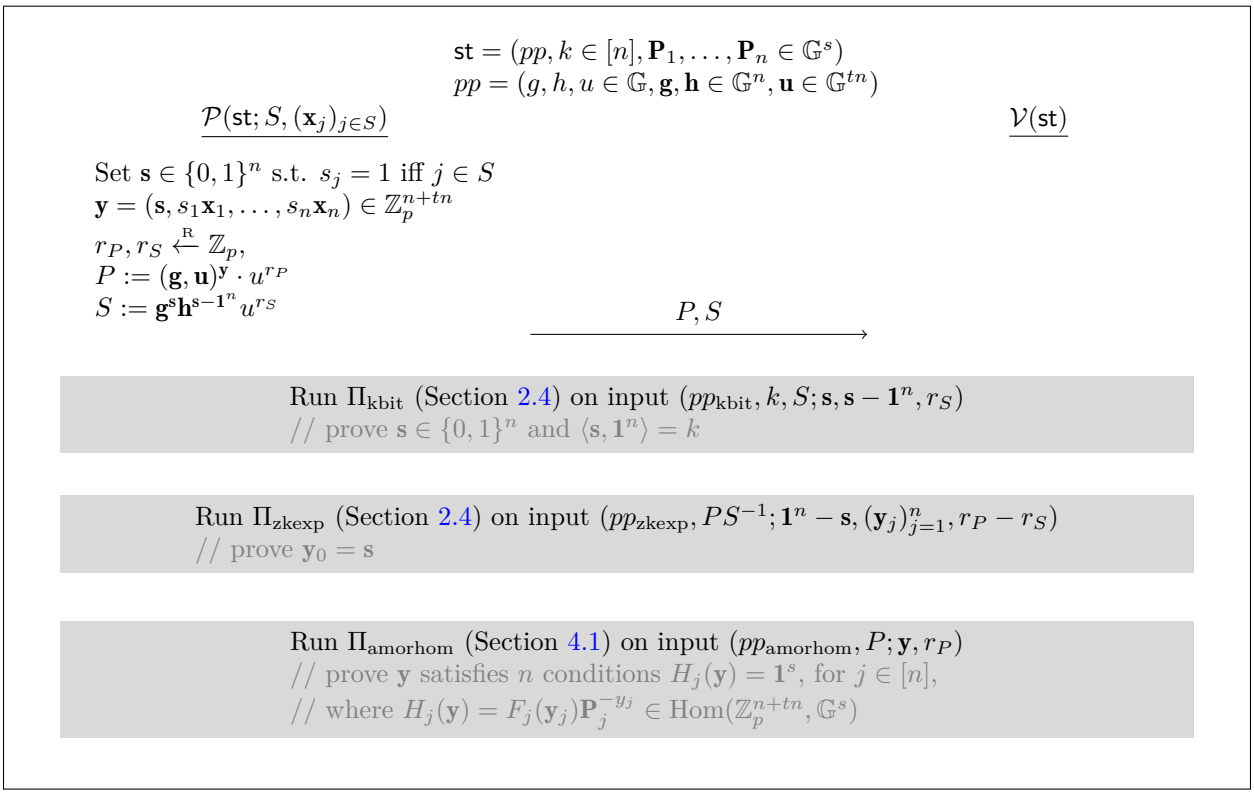

Figure 6: $\Pi_{\text{hetero-koon}}$ for $L_{\text{hetero-koon}}$: the heterogeneous k-out-of-n proof

Theorem 4.1. *Protocol $\Pi_{\text{hetero-koon}}$ in Figure 6 is an SHVZK argument of knowledge for $L_{\text{hetero-koon}}$.*

Proof. We separately argue its completeness, witness-extended emulation and SHVZK properties.

Completeness. The completeness follows straightforwardly from the fact that $\mathbf{P}_j^{s_j} = F_j(s_j \cdot \mathbf{x}_j)$ holds for each $j \in [n]$.

Witness-extended emulation. We construct an extractor Ext by invoking the extractors $\mathsf{Ext}_{\text{kbit}}$, $\mathsf{Ext}_{\text{zkexp}}$ and $\mathsf{Ext}_{\text{amorhom}}$ of the protocol Π_{kbit}, Π_{zkexp} and Π_{amorhom}, respectively.

First, by invoking $\mathsf{Ext}_{\text{amorhom}}$ on input P, Ext extracts $\mathbf{y} = (\mathbf{y}_0, \ldots, \mathbf{y}_n) \in \mathbb{Z}_p^{n+tn}, r_P \in \mathbb{Z}_p$ such that $P = \mathbf{g}^{\mathbf{y}_0}\mathbf{u}^{(\mathbf{y}_1,\ldots,\mathbf{y}_n)}u^{r_P}$ and for each $j \in [n]$, it holds $F_j(\mathbf{y}_j) = \mathbf{P}_j^{y_j}$.

Next, we argue that $\mathbf{y}_0 \in \{0,1\}^n$ and $\langle \mathbf{y}_0, \mathbf{1}^n \rangle = k$. The extractor Ext invokes $\mathsf{Ext}_{\text{kbit}}$ on input (k, S), and obtains $\mathbf{s}, r_S$ such that $S = \mathbf{g}^{\mathbf{s}}\mathbf{h}^{\mathbf{s}-\mathbf{1}^n}u^{r_S}$, $\mathbf{s} \in \{0,1\}^n$ and $\langle \mathbf{s}, \mathbf{1}^n \rangle = k$. By invoking $\mathsf{Ext}_{\text{zkexp}}$ on input $P \cdot S^{-1}$, Ext extracts $\mathbf{a}, r'$ such that $P \cdot S^{-1} = (\mathbf{h}, \mathbf{u})^{\mathbf{a}}u^{r'}$. With these in mind, we have $P \cdot S^{-1} = \mathbf{g}^{\mathbf{y}_0}\mathbf{u}^{(\mathbf{y}_i)_{i\in[1,n]}}u^{r_P} \cdot \mathbf{g}^{-\mathbf{s}}\mathbf{h}^{\mathbf{1}^n-\mathbf{s}}u^{-r_S} = \mathbf{g}^{\mathbf{y}_0-\mathbf{s}}\mathbf{h}^{\mathbf{1}^n-\mathbf{s}}\mathbf{u}^{(\mathbf{y}_i)_{i\in[1,n]}}u^{r_P-r_S} = (\mathbf{h}, \mathbf{u})^{\mathbf{a}}u^{r'}$. Unless the prover knows a non-trivial DL-relation between $\mathbf{g}$, $\mathbf{h}$, $\mathbf{u}$ and u, it holds $\mathbf{y}_0 = \mathbf{s}$, i.e., $\mathbf{y}_0 \in \{0,1\}^n$ and $\langle \mathbf{y}_0, \mathbf{1}^n \rangle = k$.

Finally, Ext computes S by extracting the indices i of which $s_j = 1$. Then, it computes $(\mathbf{x}_j)_{j\in S}$ by setting $\mathbf{x}_j = \mathbf{y}_j$. Ext outputs $(S, (\mathbf{x}_i)_{i\in S})$.

SHVZK. It follows immediately from the fact that P and S are independent and uniformly random in $\mathbb{G}$, and from the corresponding SHVZK property of Π_{kbit}, Π_{zkexp} and Π_{amorhom}.

This proves Theorem 4.1. □

Efficiency analysis. The protocol $\Pi_{\text{hetero-koon}}$ consists of three sub-protocols i.e., Π_{kbit}, Π_{zkexp} and Π_{amorhom}. For Π_{kbit}, the proof size is $O(\log n)$, and both the prover and verifier complexities are $O(n)$. For Π_{zkexp}, the proof size is $O(\log(n+tn))$, and both the prover and verifier complexities are $O(n+tn)$. For Π_{amorhom}, the proof size is $O(s\log(n+tn))$, and both the prover and verifier complexities are dominated by the computation of aggregated homomorphism $H = \sum_{j=1}^{n} \rho^j H_j$. It involves computation of sn bases $(\mathbf{P}_1^{\rho}, \ldots, \mathbf{P}_n^{\rho^n}) \in \mathbb{G}^{sn}$, which requires $O(sn)$ operations, and the computation of homomorphism $\sum_{j=1}^{n} \rho^j F_j$, which requires $O(stn)$ operations. Consequently, the overall proof size of $\Pi_{\text{hetero-koon}}$ is $O(s\log(n+tn))$, and both the prover and verifier complexities are $O(s(n+tn))$.

An optimization for the $k = n$ case. In the case where the prover wishes to open n distinct homomorphisms, i.e., k is publicly equal to n, the protocol $\Pi_{\text{hetero-koon}}$ allows for a further optimization. In this setting, the indication vector $\mathbf{s}$ in the extended vector $\mathbf{y}$ is redundant, as all its entries must be one. Therefore, the protocol can be optimized as below: First, the prover sets $\mathbf{y} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{Z}_p^{tn}$ and commits to $\mathbf{y}$ by computing a commitment P. Then, the prover proves that the committed $\mathbf{y}$ satisfies n homomorphism relations $H_j(\mathbf{y}) := F_j(\mathbf{y}_j) \cdot \mathbf{P}_j^{-1} = \mathbf{1}^s$ for $j \in [n]$. Here, $\mathbf{y}_j = (y_{jt-t+1}, \ldots, y_{jt}) \in \mathbb{Z}_p^t$ is intended to be $\mathbf{x}_j$.

Remark 4.1 (More applications). The heterogeneous k-out-of-n proof serves as a key tool in the construction of multi-receiver PPABC. Its applicability also extends to a wider range of privacy-preserving scenarios beyond cryptocurrencies. A prominent example is building threshold ring signature schemes [GN18] for cross-organizational settings, where users' public keys are generated under distinct bases.

4.3 The Heterogeneous k-out-of-n Range Proof

In the construction of PPABC, the sender must prove that the transfer amounts are within valid ranges. Therefore, it is crucial to explore how to integrate the heterogeneous k-out-of-n proof with range proofs. The GK-based homogeneous k-out-of-n proof (Section 3) inherently supports this integration due to its "pick-out" mechanism, whereas the heterogeneous k-out-of-n proof poses a non-trivial challenge due to its lack of this feature.

In this section, we present a heterogeneous k-out-of-n range proof $\Pi^{\text{range}}_{\text{hetero-koon}}$, which combines the heterogeneous k-out-of-n proof with range proofs, enabling the prover to not only prove knowledge of partial of the openings of n homomorphisms, but also convince the verifier that the opening lies within a specified range. Formally, the heterogeneous k-out-of-n range proof is a protocol for the language $L_{\text{hetero-koon-range}}$ defined as below:

$$L_{\text{hetero-koon-range}} = \{(F_1, \ldots, F_n \in \text{Hom}(\mathbb{Z}_p^t, \mathbb{G}^s), \mathbf{P}_1, \ldots, \mathbf{P}_n \in \mathbb{G}^s) \mid \\ \exists\, S \subseteq [n], (\mathbf{x}_j)_{j\in S} \in \mathbb{Z}_p^{tk} \text{ s.t. } |S| = k \wedge \forall j \in S : F_j(\mathbf{x}_j) = \mathbf{P}_j \wedge x_{j,1} \in [0, 2^{\nu} - 1]\}.$$

Without loss of generality, we describe our protocol by proving that the first element of the opening (i.e., $x_{j,1}$) lies within a specified range. Note that the protocol can be extended to any element in the vector $\mathbf{x}_j$.

Figure 7 depicts the protocol $\Pi^{\text{range}}_{\text{hetero-koon}}$ for $L_{\text{hetero-koon-range}}$. The key innovation lies in the introduction of a list of bridge commitment scheme with corresponding bridging proofs that ensure consistency between the two components. Concretely, compared to the protocol $\Pi_{\text{hetero-koon}}$ (Figure 6), $\Pi^{\text{range}}_{\text{hetero-koon}}$ is updated as follows (the main differences are highlighted in purple in Figure 7):

1. In the initial phase, $\mathcal{P}$ sets $\mathbf{y} = (\mathbf{s}, s_1\mathbf{x}_1, \ldots, s_n\mathbf{x}_n, s_1r_1, \ldots, s_nr_n) \in \mathbb{Z}_p^{2n+tn}$, commits to the element $s_jx_{j,1}$ by computing $V_j = g^{r_j}h^{s_jx_{j,1}}$, and sends the commitments $(V_j)_{j\in[n]}$ to the verifier.
2. $\mathcal{P}$ proves all the committed values $s_jx_{j,1}$ lies in $[0, 2^\nu - 1]$.
3. In sub-protocol Π_{amorhom}, $\mathcal{P}$ additionally proves the committed vector $\mathbf{y}$ satisfies $g^{y_{n+tn+j}}h^{y_{n+jt-t+1}} = V_j^{y_j}, j \in [n]$.

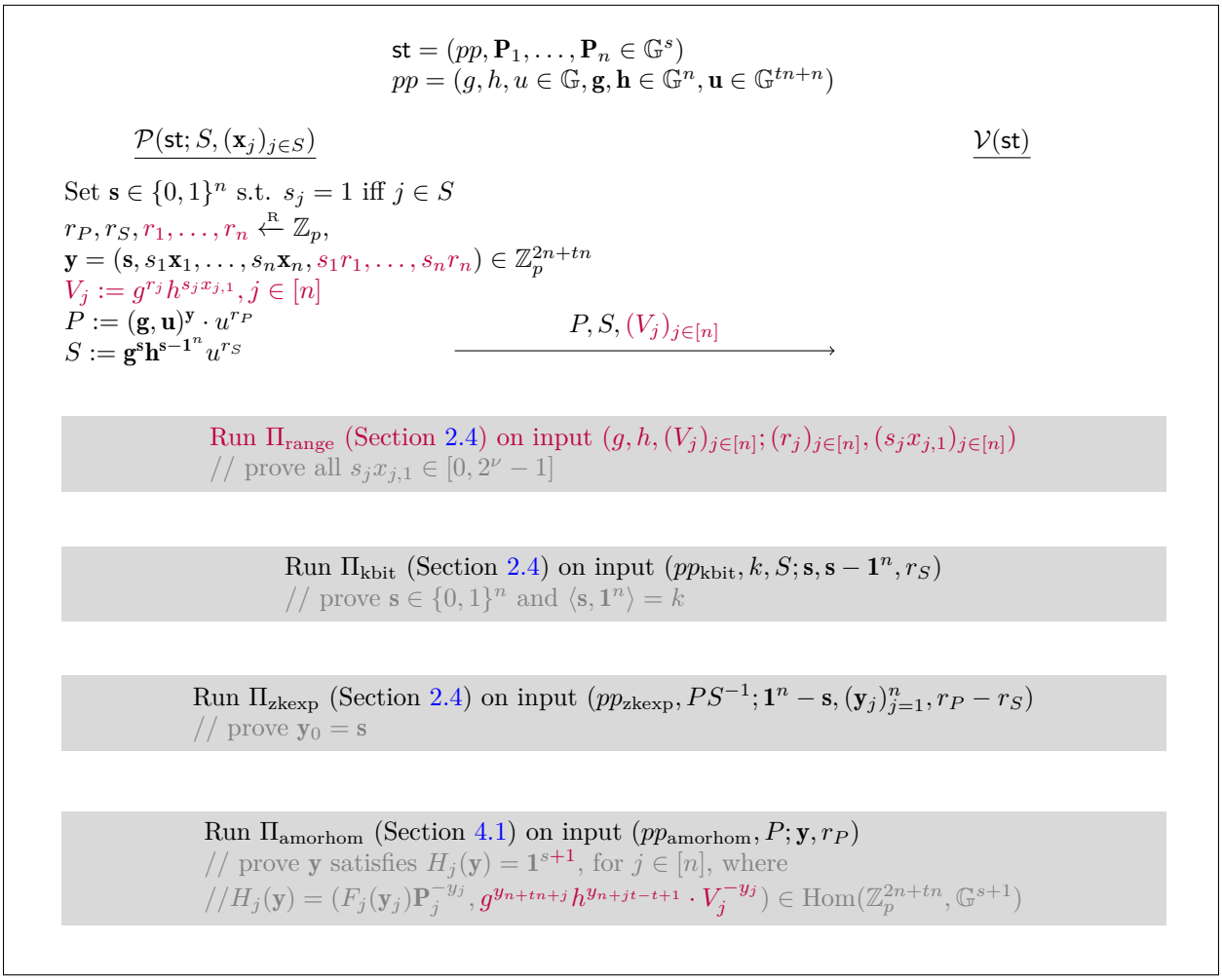

Figure 7: $\Pi^{\text{range}}_{\text{hetero-koon}}$ for $L_{\text{hetero-koon-range}}$: heterogeneous k-out-of-n range proof

Theorem 4.2. *The protocol $\Pi^{\text{range}}_{\text{hetero-koon}}$ in Figure 7 is an SHVZK argument of knowledge for $L_{\text{hetero-koon-range}}$.*

Proof. We separately argue its completeness, witness-extended emulation and SHVZK properties.

Completeness. The completeness follows straightforwardly from the fact that $\mathbf{P}_j^{s_j} = F_j(s_j \cdot \mathbf{x}_j)$ and $V_j^{s_j} = g^{s_jr_j}h^{s_j^2x_{j,1}} = g^{s_jr_j}h^{s_jx_{j,1}}$ hold for each $j \in [n]$.

Witness-extended emulation. We construct an extractor Ext by invoking the extractors $\mathsf{Ext}_{\text{range}}$, $\mathsf{Ext}_{\text{kbit}}$, $\mathsf{Ext}_{\text{zkexp}}$ and $\mathsf{Ext}_{\text{amorhom}}$ of the protocol Π_{range}, Π_{kbit}, Π_{zkexp} and Π_{amorhom}, respectively.

Similar to the proof of Theorem 4.1, Ext first invokes $\mathsf{Ext}_{\text{kbit}}$, $\mathsf{Ext}_{\text{zkexp}}$ and $\mathsf{Ext}_{\text{amorhom}}$ to extract S, $(\mathbf{x}_j)_{j\in S}$ and $(r_j)_{j\in S}$ which satisfy $F_j(\mathbf{x}_j) = \mathbf{P}_j$ and $V_j = g^{r_j}h^{x_{j,1}}$ for $j \in S$. Then Ext invokes $\mathsf{Ext}_{\text{range}}$ to extract $(r'_j, v'_j)_{j\in S}$ satisfying $V_j = g^{r'_j}h^{v'_j}$ and $v'_j \in [0, 2^\nu - 1]$. Unless the prover knows a non-trivial DL-relation between g, h, it must hold that $v'_j = x_{j,1}$, i.e., $x_{j,1} \in [0, 2^\nu - 1]$, for all $j \in S$.

SHVZK. It follows immediately from the fact that P, S and $V_1, \ldots, V_n$ are independent and uniformly random in $\mathbb{G}$, and from the corresponding SHVZK property of Π_{range}, Π_{kbit}, Π_{zkexp} and Π_{amorhom}.

This proves the Theorem 4.2. □

Efficiency analysis. Similar to that of the protocol $\Pi_{\text{hetero-koon}}$ (Figure 6), both of the prover and verifier complexity of $\Pi^{\text{range}}_{\text{hetero-koon}}$ are dominated by the complexities of protocols Π_{zkexp}, Π_{amorhom} and Π_{range}, and thus are $O(s \cdot (2n + tn) + n\nu)$. The proof size is also dominated by that of Π_{kbit}, Π_{zkexp}, Π_{amorhom} and Π_{range}, and thus is $O(s \cdot \log(2n + tn) + \log(n\nu))$.

An optimization for the $k = 1$ case. In the case where the prover wishes to provide range proof for only one opening, i.e., proving an $x_{\ell,1}$ lies within a specific range $[0, 2^\nu - 1]$, the protocol executes as in protocol $\Pi^{\text{range}}_{\text{hetero-koon}}$ except that: (i) instead of generating n commitments, the prover computes only one commitment $V = g^{r_V}h^{x_{\ell,1}}$, sends V to the verifier and proves the committed value lie within the range; (ii) set $\mathbf{y} = (\mathbf{s}, s_1\mathbf{x}_1, \ldots, s_n\mathbf{x}_n, s_1 r_V, \ldots, s_n r_V) \in \mathbb{Z}_p^{2n+tn}$ and $H_j(\mathbf{y}) = (F_j(\mathbf{y}_j)\mathbf{P}_j^{-y_j}, g^{y_{n+tn+j}}h^{y_{n+jt-t+1}} \cdot V^{-y_j}) \in \text{Hom}(\mathbb{Z}_p^{2n+tn}, \mathbb{G}^{s+1})$.

5 Privacy-Preserving Account-Based Cryptocurrencies

In this section, we formally define the notion of multi-receiver privacy-preserving account-based cryptocurrencies (PPABC) with a strong security model, then propose a generic construction from homomorphic PKE and NIZK.

5.1 Definition of PPABC

We first present the formal definition of multi-receiver PPABC, adapted from the definition of single-receiver PPABC [Dia21] with two key enhancements in functionality and security respectively: (1) built-in multi-receiver transaction support; (2) strong security model captures insider attacks.

Definition 5.1. A privacy-preserving account-based cryptocurrency (PPABC) consists of six polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter 1^λ, outputs public parameters pp, which include the maximum transfer amount $v_{\max}$ and the size of the anonymity set n. A trusted party executes this algorithm once-for-all to setup the whole system. pp will be an implicit input on the rest algorithms.
- $\mathsf{CreateAccount}(\tilde{v})$: on input the initial balance $\tilde{v}$, outputs a keypair (pk, sk) and an encoded balance $\tilde{C}$. A user runs this algorithm to create an account.
- $\mathsf{RevealBalance}(sk, \tilde{C})$: on input a secret key sk and an encoded balance $\tilde{C}$, outputs the balance $\tilde{v}$. A user runs this algorithm to reveal the balance.
- $\mathsf{CreateTx}(pk_s, sk_s, (pk_{r_i}, v_{r_i})_{i=1}^{t\le n})$: on input the sender's keypair (pk_s, sk_s), t receiver addresses pk_{r_i} with transfer amounts v_{r_i}, randomly picks $n - k - 1$ decoy addresses pk_{d_j}, shuffles the addresses $(pk_s, \{pk_{r_i}\}_{i=1}^t, \{pk_{d_j}\}_{j=1}^{n-t-1})$ together with transfer amounts to form an anonymity set $\mathsf{AnonSet} = (pk_1, \ldots, pk_n)$ and transfer amount vector $\mathbf{v} = (v_1, \ldots, v_n)$, runs $\mathsf{CreateTx}^*(sk_s, \mathsf{AnonSet}, \mathbf{v})$ to output a transaction $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$, where memo records the number t of intended receivers, the anonymity

set $\mathsf{AnonSet}$, and encoded transfer amounts $\mathbf{C} = (C_1, \ldots, C_n)$; aux records the legality proof to attest the validity of transaction. The transaction legality policy could be application-dependent, which is typically defined via a predicate $f_{\mathsf{legal}}(\cdot)$ specifying the constraints between accounts, balances and transfer amounts. A user runs this algorithm to transfer v_{r_i} coins from account pk_s to account pk_{r_i}.

- $\mathsf{VerifyTx}(\mathsf{tx})$: on input a transaction $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$, if $f_{\mathsf{legal}}(\mathsf{memo}) = 1$ and tx is authenticated, outputs "1"; otherwise, outputs "0". Validators run this algorithm to check the validity of a purported transaction tx.
- $\mathsf{UpdateTx}(\mathsf{tx}, B)$: on input a validated transaction tx, validators run this algorithm to update the blockchain B. Namely, update the encoded balance of involved accounts (i.e., those in $\mathsf{AnonSet}$), and record the transaction tx on B. After this, the sender pk_s's balance decreases by $\sum_{i\in[t]} v_{r_i}$ coins while each receiver pk_{r_i}'s balance increases by v_{r_i} coins.

A PPABC system should satisfy the following properties.

Correctness. An honestly generated transaction will always be accepted and recorded on the blockchain, and the states of involved accounts will be updated properly.

Security. Informally, the security contains three aspects:

- *Authenticity.* Any PPT adversary without the secret key sk for pk cannot generate a valid transaction with account pk as sender, i.e., decreasing pk's balance.
- *Soundness.* Any PPT adversary (even knowing the secret key of sender) cannot generate a transaction $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$ such that $\mathsf{VerifyTx}(\mathsf{tx}) = 1$ but $f_{\mathsf{legal}}(\mathsf{memo}) \neq 1$.
- *Ledger-indistinguishability.* Any PPT adversary without the secret key of an honest account pk cannot learn any information about its transfer amount — neither its exact value nor its sign, which reveals the account's role (i.e., sender, receiver or decoy). This notion captures both confidentiality and anonymity in one shot.

Let $\mathcal{A}$ be an adversary against PPABC. To capture realistic attacks, we first define a list of oracles available to $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{regH}}$: This oracle captures that $\mathcal{A}$ can observe the public information of honest accounts in the system. $\mathcal{CH}$ keeps track of this type of queries by maintaining a list T_{honest}, which is initially empty. Upon receiving a fresh query, $\mathcal{CH}$ responds as below: picks a random value $\tilde{v}$, runs $(pk, sk, \tilde{C}) \leftarrow \mathsf{CreateAccount}(\tilde{v})$, appends $(pk, sk, \tilde{C}, \tilde{v})$ to T_{honest}, sends $(pk, \tilde{C})$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{regC}}$: This oracle captures that $\mathcal{A}$ can dishonestly generate any (possibly malformed) accounts itself. $\mathcal{CH}$ keeps track of this type of queries by maintaining a list T_{corrupt}, which is initially empty. Upon receiving a query $(pk, \tilde{C})$ (which is not allowed to be the one in T_{honest}), $\mathcal{CH}$ records $(pk, \perp, \tilde{C}, \perp)$ to the list T_{corrupt}. We highlight that this oracle empowers an adversary to launch a wide range of powerful insider attacks (including rogue-key attack).
- $\mathcal{O}_{\mathsf{extH}}$: This oracle captures that $\mathcal{A}$ can corrupt some honest accounts. Upon receiving a query $pk \in T_{\text{honest}}$, $\mathcal{CH}$ responds with the sk recorded in T_{honest}, and moves this entry to T_{corrupt}.
- $\mathcal{O}_{\mathsf{trans}}$: This oracle captures that $\mathcal{A}$ can instruct honest accounts to generate specific transactions. $\mathcal{CH}$ keeps track of this type of queries by maintaining a list T_{tx}, which is initially empty. For a query $(pk_s, (pk_{r_i}, v_{r_i})_{i=1}^t)$, where $pk_s \in T_{\text{honest}}$, $\mathcal{CH}$ first checks its legality. If it is illegal, $\mathcal{CH}$ returns $\perp$; else, $\mathcal{CH}$ runs $\mathsf{tx} \leftarrow \mathsf{CreateTx}(sk_s, pk_s, (pk_{r_i}, v_{r_i})_{i=1}^t)$, updates involved accounts, sends tx to $\mathcal{A}$ and appends tx to T_{tx}.

- $\mathcal{O}_{\mathsf{inject}}$: This oracle captures that $\mathcal{A}$ can dishonestly generate any (possible malformed) transaction itself. Upon receiving a query tx, $\mathcal{CH}$ ignores it if $\mathsf{tx} \in T_{\mathrm{tx}}$ and $\mathsf{VerifyTx}(\mathsf{tx}) \neq 1$; otherwise $\mathcal{CH}$ updates the states of associated accounts and records tx to T_{tx}.

Now, we are ready to present formal definitions of authenticity, soundness and ledger-indistinguishability of PPABC. Hereafter, we use $\mathcal{A}^{\mathcal{O}}$ to denote that $\mathcal{A}$ has access to all oracles defined above.

Authenticity. PPABC is authenticated if for any PPT adversary $\mathcal{A}$, it holds that

$$\Pr\left[\begin{array}{l|l} \mathsf{tx}^* \notin T_{\mathrm{tx}} \wedge \mathsf{VerifyTx}(\mathsf{tx}^*) = 1 \ \wedge \ \exists\, \ell_s \in [n] \text{ s.t.} \\ \quad (\mathsf{RevealBalance}(C^*_{\ell_s}) < 0 \ \wedge pk^*_{\ell_s} \notin T_{\mathrm{corrupt}}) \end{array} : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{tx}^* \leftarrow \mathcal{A}^{\mathcal{O}}(pp); \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{tx}^* = (k, (pk^*_1, \ldots, pk^*_n), (C^*_1, \ldots, C^*_n), \mathsf{aux}^*)$.

Soundness. PPABC is sound if for any PPT adversary $\mathcal{A}$, it holds that

$$\Pr\left[\mathsf{VerifyTx}(\mathsf{tx}^*) = 1 \ \wedge f_{\mathsf{legal}}(\mathsf{memo}^*) \neq 1 \ : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{tx}^* \leftarrow \mathcal{A}^{\mathcal{O}}(pp); \end{array}\right] \leq \mathsf{negl}(\lambda),$$

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

Ledger-indistinguishability. PPABC is ledger-indistinguishable if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that

$$\left|\Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (state, \mathbf{PK}^*_{\mathrm{H}}, \mathbf{v}^*_{\mathrm{H},0}, \mathbf{v}^*_{\mathrm{H},1}, \mathbf{PK}^*_{\mathrm{C}}, \mathbf{v}^*_{\mathrm{C}}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pp); \\ \beta \xleftarrow{\mathrm{R}} \{0,1\}; \\ \mathsf{tx}^* \leftarrow \mathsf{CreateTx}^*(sk^*_s, \mathbf{PK}^*, \mathbf{v}^*_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}}(state, \mathsf{tx}^*); \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where $\mathbf{PK}^*_{\mathrm{H}} \subset T_{\mathrm{honest}}$, $\mathbf{PK}^*_{\mathrm{C}} \subset T_{\mathrm{corrupt}}$, $\mathbf{PK}^* = \{\mathbf{PK}^*_{\mathrm{H}}, \mathbf{PK}^*_{\mathrm{C}}\}$, and $\mathbf{v}^*_\beta = (\mathbf{v}^*_{\mathrm{H},\beta}, \mathbf{v}^*_{\mathrm{C}})$. Specifically, $\mathbf{v}^*_{\mathrm{H},\beta}$ is the vector of transfer amounts to the group of honest participants $\mathbf{PK}^*_{\mathrm{H}}$, and $\mathbf{v}^*_{\mathrm{C}}$ is the vector of transfer amounts to the group of corrupted participants $\mathbf{PK}^*_{\mathrm{C}}$. Additionally, both $\mathbf{v}^*_{\mathrm{H},0}$ and $\mathbf{v}^*_{\mathrm{H},1}$ contain exactly one negative element. We denote the corresponding account in $\mathbf{PK}^*_{\mathrm{H}}$ as pk^*_s, and let sk^*_s be the secret key for pk^*_s. The transaction $\mathsf{tx}^* = ((t^*, \mathsf{AnonSet}^*, \mathbf{C}^*), \mathsf{aux}^*)$, where $\mathsf{AnonSet}^* = \mathbf{PK}^*$ and t^* is the number of positive elements in $\mathbf{v}^*_\beta$ (for $\beta \in \{0,1\}$). To avoid trivial definitions, we restrict (i) t^* should be identical for both $\mathbf{v}^*_0$ and $\mathbf{v}^*_1$; (ii) both $(\mathbf{PK}^*, \mathbf{v}^*_0)$, $(\mathbf{PK}^*, \mathbf{v}^*_1)$ constitute valid transactions; (iii) $\mathcal{A}_2$ is not allowed to corrupt accounts in $\mathbf{PK}^*_{\mathrm{H}}$; (iv) for each account $pk_i \in \mathbf{PK}^*_{\mathrm{H}}$, both $\tilde{v}_i + v^*_{\mathrm{H},0,i} + v_{\mathrm{sum},i}$ and $\tilde{v}_i + v^*_{\mathrm{H},1,i} + v_{\mathrm{sum},i}$ should lie in $[0, v_{\max}]$, where $\tilde{v}_i$ is the balance of pk_i before tx^* and $v_{\mathrm{sum},i}$ is the accumulation of the transfer amounts in $\mathcal{A}_2$'s queries to $\mathcal{O}_{\mathsf{trans}}$.

Additional remarks:

- *Insider attacks.* Our notion of ledger-indistinguishability formally captures insider attacks by allowing an adversary to adaptively corrupt a set of accounts denoted by T_{corrupt} and selects $\mathbf{PK}^*_{\mathrm{C}} \subset T_{\mathrm{corrupt}}$ to form the anonymity set in target transaction. Here, T_{corrupt} consists of two types of corrupted accounts: accounts whose secret keys are leaked to the adversary through queries to $\mathcal{O}_{\mathsf{extH}}$; accounts whose public keys are dishonestly generated (and potentially malformed) by the adversary via $\mathcal{O}_{\mathsf{regC}}$.
- *Front-running, replay, and synchronization issues.* Following prior works [FMMO19, CMTA20, Dia21], this work focuses exclusively on the transaction layer, abstracting away front-running, replay, and synchronization issues. Actually, existing methods — such as those described in [BAZB20, Appendix D] — for mitigating these attacks are generic or semi-generic and can be easily adapted to PPABC. For example, the front-running attack can be prevented using the (un)locking and rolling over mechanism employed in [BAZB20, Appendix D]; the replay attack can be defended by employing an epoch-based group element as done in [BAZB20, Appendix D]. For comprehensive treatments of these attacks, we refer to [BAZB20].

- *Sender anonymity.* Guo et al. [GKPH24] argue that sender anonymity is non-essential, justifying this by observing that Ethereum's current implementation reveals sender identities during smart contract invocations. We contend, however, that this exposure is a byproduct of Ethereum's transparent execution model, not an inherent limitation of smart contract systems. Privacy-preserving frameworks such as Hawk [KMS+16], zkHawk [BCT21], and Zexe [BCG+20] demonstrate that private contract execution is feasible. Since this work aims to advance the state-of-the-art in PPABC, we consider sender anonymity an indispensable property.

5.2 Generic Construction of PPABC

Before presenting the generic construction of multi-receiver PPABC from homomorphic PKE and NIZK, we first define the transaction legality policy.

Legality policy. A transaction $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$ is *legal* (i.e., $f_{\mathsf{legal}}(\mathsf{memo}) = 1$) iff it satisfies following four conditions:

1. Balance: the encoded amounts are well-formed and sum of all amounts equals zero;
2. Decoy integrity: exactly $n - t - 1$ encoded transfer amounts must be zero.
3. Receiver validity: exactly t encoded transfer amounts must be positive integers within a specified range.
4. Sender solvency: (i) one encoded transfer amount must be a negative integer within a specified range; (ii) the sender must be authenticated and must have sufficient balance.

As mentioned in Section 1.5, we stipulate that the transfer amount to any intended receiver must be positive. This requirement is reasonable, as it prevents empty transaction spam (i.e., no balances will be changed after the transaction). Importantly, it helps to simplify the design of zero-knowledge proofs for transaction legality, which can be formalized as language L_{legal} and will be defined later.

Now, we show our generic construction of multi-receiver PPABC. Let $\text{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme that is additively homomorphic on message space $\mathbb{Z}_p$ and $\text{NIZK} = (\mathsf{Setup}, \mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be an NIZK proof for language L_{legal}. A PPABC can be constructed as below:

- $\mathsf{Setup}(1^\lambda)$: runs $pp_{\text{pke}} \leftarrow \text{PKE.}\mathsf{Setup}(1^\lambda)$, $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, $crs \leftarrow \text{NIZK.}\mathsf{CRSGen}(pp_{\text{nizk}})$, picks an integer $n \geq 2$, outputs the public parameters $pp = (pp_{\text{pke}}, pp_{\text{nizk}}, crs, n)$.
- $\mathsf{CreateAccount}(\tilde{v})$: runs $(pk, sk) \leftarrow \text{PKE.}\mathsf{KeyGen}(pp_{\text{pke}})$ and $\tilde{C} \leftarrow \text{PKE.}\mathsf{Enc}(pk, \tilde{v}; r)$, outputs $(pk, sk, \tilde{C})$. This algorithm induces a hard relation $\mathsf{R}_{\text{key}} = \{(pk, sk) \mid \exists\ sk \text{ s.t. } pk = \text{PKE.}\mathsf{KeyGen}(pp_{\text{pke}}; sk)\}$.
- $\mathsf{RevealBalance}(sk, \tilde{C})$: runs $\tilde{v} \leftarrow \text{PKE.}\mathsf{Dec}(sk, \tilde{C})$, outputs $\tilde{v}$.
- $\mathsf{CreateTx}(sk_s, pk_s, (pk_{r_i}, v_{r_i})_{i=1}^t)$: queries pk_s's balance ciphertext $\tilde{C}_s$, runs $\tilde{v}_s \leftarrow \text{PKE.}\mathsf{Dec}(sk_s, \tilde{C}_s)$, checks whether all $(v_{r_i})_{i=1}^t, \sum_{i=1}^t v_{r_i}$ lie in $(0, v_{\max}]$ and the updated balance $v^* = \tilde{v}_s - \sum_{i=1}^t v_{r_i}$ lies in $[0, v_{\max}]$, if not, returns $\perp$. Otherwise, it creates a transaction tx via the following steps:
  1. Generate $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}})$: (i) Pick a random anonymity set $\mathsf{AnonSet}$ of size n, where $pk_{\ell_0} = pk_s$ (sender) and $pk_{\ell_i} = pk_{r_i}$ for $i \in [t]$ (receivers). (ii) Sample randomness $\mathbf{r} = (r_1, \ldots, r_n)$ uniformly. Construct transfer amount vector $\mathbf{v} = (v_1, \ldots, v_n)$, where $v_{\ell_0} = -\sum_{i=1}^t v_{r_i}$ (sender's), $v_{\ell_i} = v_{r_i}$ for $i \in [t]$ (receivers') and $v_j = 0$ for $j \notin \{\ell_i\}_{i=0}^t$ (decoys'). Compute transfer amounts ciphertexts $C_j = \text{PKE.}\mathsf{Enc}(pk_j, v_j; r_j)$ for $j \in [n]$. (iii) Fetch balance ciphertext $\tilde{C}_j$ for all accounts pk_j in $\mathsf{AnonSet}$.
  2. Run NIZK.Prove to generate π_{legal} for proving $\mathsf{memo} \in L_{\text{legal}}$, where L_{legal} is defined according to the legality policies.

3. Output a transaction $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$ where $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$ (there is no need to include $\tilde{\mathbf{C}}$ in tx, as it can be queried on demand) and $\mathsf{aux} = \pi_{\text{legal}}$.

- $\mathsf{VerifyTx}(\mathsf{tx})$: on input $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$, parses $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$ and $\mathsf{aux} = \pi_{\text{legal}}$, then extends $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}})$ by querying the balance ciphertexts $\tilde{C}_j$ for each pk_j and verifies that (i) tx does not appear in historical transactions, and (ii) $\text{NIZK.}\mathsf{Verify}(crs, \mathsf{memo}, \pi_{\text{legal}}) = 1$. It outputs "1" if both conditions hold; otherwise, outputs "0".
- $\mathsf{UpdateTx}(\mathsf{tx}, B)$: on input a validated transaction $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$, parses $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$, for each account $pk_j \in \mathsf{AnonSet}$, updates the corresponding balance ciphertext $\tilde{C}_j$ on the blockchain B by setting $\tilde{C}_j = \tilde{C}_j + C_j$.

Definition and decomposition of L_{legal}. According to the legality policy, L_{legal} is defined as:

$$
\begin{aligned}
L_{\text{legal}} = \{(pp, t, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}}) \mid \exists\ (sk, \ell_s, J, S, \mathbf{v}, \mathbf{r}) \text{ s.t.} \\
\quad \forall j \in [n], C_j = \mathsf{Enc}(pk_j, v_j; r_j) \wedge \textstyle\sum_{j=1}^{n} v_j = 0 \ \text{//balance} \\
\wedge\ \ell_s \notin (S \cup J) \wedge S \cap J = \emptyset \ \text{//set disjointness} \\
\wedge\ J \subset [n] \wedge |J| = n - t - 1 \wedge \forall j \in J, C_j = \mathsf{Enc}(pk_j, 0; r_j)\} \ \text{//decoy integrity} \\
\wedge\ S \subset [n] \wedge |S| = t \wedge \forall j \in S, C_j = \mathsf{Enc}(pk_j, v_j; r_j) \wedge v_j \in (0, v_{\max}] \ \text{//receiver validity} \\
\wedge\ \ell_s \in [n] \wedge (pk_{\ell_s}, sk) \in \mathsf{R}_{\text{key}} \wedge -v_{\ell_s} \in (0, v_{\max}] \wedge \mathsf{Dec}(sk, \tilde{C}_{\ell_s}) + v_{\ell_s} \in [0, v_{\max}] \text{//sender solvency}\}
\end{aligned}
$$

Our requirement of positive transfers to intended receivers, combined with the binding property of the PKE scheme, enables elimination of the proof for set disjointness. Since if there exists an $\ell_i \in (S \cup J)$ such that $\ell_i = \ell_s$, then either the sender breaks the binding property of the PKE, or a contradiction arises: $v_{\ell_s} \in [-v_{\max}, 0) \wedge v_{\ell_s} \in [0, v_{\max}]$. The analysis for $S \cap J = \emptyset$ follows analogously. Thus, the complex language L_{legal} can be decomposed into four sub-languages, yielding modular design of PPABC: L_{balance} for balance, L_{zero} for decoy integrity, L_{positive} for receiver validity and L_{solvent} for sender solvency. What's more, for single-receiver transactions ($t = 1$), the NIZK proof for L_{positive} becomes redundant. Because the combination of L_{balanced}, L_{zero} and L_{solvent} necessarily implies exactly one positive transfer amount, making explicit verification unnecessary.

Theorem 5.1. *If the* PKE *satisfies IND-CPA security in the multi-user setting, and* NIZK *component satisfies simulation sound extractability and adaptive zero-knowledge property, then the above generic PPABC construction satisfies authenticity, soundness and ledger-indistinguishability.*

Proof. We prove this theorem via following four lemmas.

Lemma 5.2. *Assuming the relation* R_{key} *specified by the algorithm* PKE.KeyGen *is a hard relation,* NIZK *is adaptive zero-knowledge and* $\text{NIZK}_{\text{solvent}}$ *has simulation sound extractability, then the above PPABC satisfies authenticity property.*

Proof. Let $\mathcal{A}$ be a PPT adversary attacking the authenticity property of the PPABC and S_i be the event that $\mathcal{A}$ wins in Game i. We prove this lemma via a sequence of games.

Game 0. The real experiment for authenticity. Specifically, the challenger $\mathcal{CH}$ interacts with $\mathcal{A}$ as below.

1. Setup: $\mathcal{CH}$ runs $pp_{\text{pke}} \leftarrow \text{PKE.}\mathsf{Setup}(1^\lambda)$, $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$ and $crs \leftarrow \text{NIZK.}\mathsf{CRSGen}(pp_{\text{nizk}})$, picks an integer $n \geq 2$, sends the public parameters $pp = (pp_{\text{pke}}, pp_{\text{nizk}}, crs, n)$ to $\mathcal{A}$.
2. Queries: $\mathcal{CH}$ answers queries to the oracles $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$ as described above.
3. Forge: $\mathcal{A}$ outputs $\mathsf{tx}^* = (\mathsf{memo}, \mathsf{aux})$ where $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$ and $\mathsf{aux} = \pi_{\text{legal}} = (\pi_{\text{balance}} \circ \pi_{\text{zero}} \circ \pi_{\text{positive}} \circ \pi_{\text{solvent}})$, and wins if $\mathsf{VerifyTx}(\mathsf{tx}^*) = 1$, $\mathsf{tx}^* \notin T_{\text{tx}}$, and there exists an $\ell_s \in [n]$ such that $C^*_{\ell_s} = \mathsf{Enc}(pk^*_{\ell_s}, v^*)$ where $v^* \in [-v_{\max}, 0)$ and $pk^*_{\ell_s} \notin T_{\text{corrupt}}$.

According to the definition, we have

$$\mathsf{Adv}_{\mathcal{A}} = \Pr[S_0].$$

Game 1. Same as Game 0, except that $\mathcal{CH}$ picks a target index $j \in [Q_{\text{honest}}]$ at the beginning, where Q_{honest} is the maximum number that $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{regH}}$, and $\mathcal{CH}$ aborts if any of the following two events happen: (i) in the query stage, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{extH}}$ with pk_j; (ii) in the forge stage, $\mathcal{A}$ outputs tx^* while $pk_j \notin \mathsf{AnonSet}$ or the transfer amounts corresponding to pk_j is non-negative. Conditioned on $\mathcal{CH}$ does not abort, $\mathcal{A}$'s view in Game 0 is identical to Game 1, we therefore have

$$\Pr[S_1] = \Pr[S_0] \cdot \frac{1}{Q_{\text{honest}}}.$$

Game 2. Same as Game 1, except that (i) in the setup stage, $\mathcal{CH}$ runs $(crs, \tau) \leftarrow \mathsf{Sim}_1(pp_{\text{nizk}})$; (ii) in the query stage, when $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{trans}}$, $\mathcal{CH}$ generates π_{legal} for $\mathsf{memo} \in L_{\text{legal}}$ via running $\mathsf{Sim}_2(crs, \tau, \mathsf{memo})$. Based on the adaptive zero-knowledge property of the NIZK, we have

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\lambda).$$

Below, we argue that no PPT adversary has non-negligible advantage in Game 2.

Claim 5.3. *Assuming the relation* R_{key} *specified by the algorithm* PKE.KeyGen *is a hard relation and* $\text{NIZK}_{\text{solvent}}$ *has simulation sound extractability, then* $\Pr[S_2] \leq \mathsf{negl}(\lambda)$ *for all PPT adversary* $\mathcal{A}$*.*

Proof. Suppose there exists an adversary $\mathcal{A}$ wins in Game 2 with non-negligible advantage, then we can construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and solves the hard relation R_{key}. Given the pp_{pke} and a target public key pk^*, $\mathcal{B}$ interacts with $\mathcal{A}$ as below:

1. Setup: $\mathcal{B}$ runs $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, $(crs, \tau) \leftarrow \mathsf{Sim}_1(pp_{\text{nizk}})$, picks an integer $n \geq 2$, sets $pp = (pp_{\text{pke}}, pp_{\text{nizk}}, crs, n)$ and sends pp to $\mathcal{A}$. $\mathcal{B}$ also picks a random index $j \in [Q_{\text{honest}}]$.
2. Queries: $\mathcal{B}$ answers $\mathcal{A}$'s queries by maintaining three lists T_{honest}, T_{corrupt} and T_{tx}, which are initially empty.
   - $\mathcal{O}_{\mathsf{regH}}$: on the i-th query, $\mathcal{B}$ picks a random balance $\tilde{v}$ and then proceeds as below: (i) if $i \neq j$, runs $(pk, sk, \tilde{C}) \leftarrow \mathsf{CreateAccount}(\tilde{v})$, appends $(pk, sk, \tilde{C}, \tilde{v})$ to the list T_{honest}; (ii) if $i = j$, sets $pk = pk^*$, computes $\tilde{C} \leftarrow \text{PKE.}\mathsf{Enc}(\tilde{v})$, appends $(pk, \perp, \tilde{C}, \tilde{v})$ to the list T_{honest}. Then $\mathcal{B}$ returns $(pk, \tilde{C})$ to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{regC}}$: upon receiving a query $(pk, \tilde{C})$ (which is not allowed to be the one in T_{honest}), $\mathcal{B}$ records $(pk, \perp, \tilde{C}, \perp)$ to the list T_{corrupt}.
   - $\mathcal{O}_{\mathsf{extH}}$: upon receiving a query $pk \in T_{\text{honest}}$, if $pk = pk_j$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ responds with the sk recorded in T_{honest}, and moves this entry to the T_{corrupt}.
   - $\mathcal{O}_{\mathsf{trans}}$: upon receiving a query $(pk_s, (pk_{r_i}, v_{r_i})_{i=1}^t)$, where $pk_s \in T_{\text{honest}}$, $\mathcal{B}$ responds as below: (i) honestly generates $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}})$; (ii) runs $(\pi_{\text{legal}}) \leftarrow \mathsf{Sim}_2(crs, \tau, \mathsf{memo})$; (iii) resets $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$ and returns $\mathsf{tx} = (\mathsf{memo}, \pi_{\text{legal}})$ to $\mathcal{A}$; (iv) records tx to the list T_{tx} and updates the states of associated accounts.
   - $\mathcal{O}_{\mathsf{inject}}$: upon receiving a transaction $\mathsf{tx} = (\mathsf{memo}, \pi_{\text{legal}})$, if it holds $\text{NIZK.}\mathsf{Verify}(\mathsf{memo}, \pi_{\text{legal}}) = 1$, $\mathcal{B}$ inserts tx to the list T_{tx}, otherwise, $\mathcal{B}$ ignores.
3. Forge: Finally, $\mathcal{A}$ outputs $\mathsf{tx} = (\mathsf{memo}, \mathsf{aux})$ where $\mathsf{memo} = (k, \mathsf{AnonSet}, \mathbf{C})$ and $\mathsf{aux} = \pi_{\text{legal}} = (\pi_{\text{balance}} \circ \pi_{\text{zero}} \circ \pi_{\text{positive}} \circ \pi_{\text{solvent}})$, and it satisfies conditions as defined in the authenticity property.

$\mathcal{B}$ checks whether pk^* is the actual sender. Specifically, $\mathcal{B}$ checks whether $pk^* \in \mathsf{AnonSet}$ and the corresponding transfer amount v^* is negative (via computing v^* as below: for each $pk_j \neq pk^*$, runs $v_j \leftarrow \mathrm{PKE.Dec}(sk_j, C_j)$, computes $v^* = -\sum_{j:pk_j \neq pk^*} v_j$). If not, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ invokes the extractor $\mathsf{Ext}_{\mathrm{solvent}}$ of the sub-protocol $\mathrm{NIZK}_{\mathrm{solvent}}$ (which has the ability of rewinding $\mathcal{A}$) on input the statement $(pp, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}}, \pi_{\mathrm{solvent}})$ and the transcripts of $\mathcal{A}$'s random oracle queries, obtains sk^* and others. Then $\mathcal{B}$ forwards sk^* to its own challenger. It is obvious that $\mathcal{B}$ simulates $\mathcal{A}$'s view in Game 2 perfectly. Thus, $\mathcal{B}$ wins with the same advantage as $\mathcal{A}$ wins. This proves Claim 5.3. □

This proves Lemma 5.2. □

Lemma 5.4. *Assuming the soundness of* NIZK*, then the above PPABC satisfies soundness.*

Proof. According to the definition of soundness property, we have $\mathsf{VerifyTx}(\mathsf{tx}^*) = 1 \ \wedge \ f_{\mathsf{legal}}(\mathsf{memo}^*) \neq 1$ directly implies that $\mathrm{NIZK.Verify}(crs, \mathsf{memo}^*, \pi^*_{\mathrm{legal}}) = 1 \wedge \mathsf{memo}^* \notin L_{\mathrm{legal}}$. The reduction is straightforward, we thus omit the details here. □

Lemma 5.5. *Assuming the* PKE *is IND-CPA secure in the multi-user setting and the* NIZK *is adaptive zero-knowledge, then the above PPABC satisfies ledger-indistinguishability.*

Proof. Let $\mathcal{A}$ be a PPT adversary attacking the ledger-indistinguishability property of the PPABC and S_i be the event that $\mathcal{A}$ wins in Game i. We prove this lemma via a sequence of games.

Game 0. The real experiment for ledger-indistinguishability. Specifically, the challenger $\mathcal{CH}$ interacts with $\mathcal{A}$ as below.

1. Setup: $\mathcal{CH}$ runs $pp_{\mathrm{pke}} \leftarrow \mathrm{PKE.Setup}(1^\lambda)$, $pp_{\mathrm{nizk}} \leftarrow \mathrm{NIZK.Setup}(1^\lambda)$ and $crs \leftarrow \mathrm{NIZK.CRSGen}(pp_{\mathrm{nizk}})$, picks an integer $n > 2$, sends public parameters $pp = (pp_{\mathrm{pke}}, pp_{\mathrm{nizk}}, crs, n)$ to $\mathcal{A}$.
2. Pre-challenge queries: $\mathcal{CH}$ answers queries to the oracles $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$ as described above.
3. Challenge: $\mathcal{A}$ outputs two sets $\mathbf{PK}^*_{\mathrm{H}} \subset T_{\mathrm{honest}}$, $\mathbf{PK}^*_{\mathrm{C}} \subset T_{\mathrm{corrupt}}$, and three vectors of transfer amounts $\mathbf{v}^*_{\mathrm{H},0}$, $\mathbf{v}^*_{\mathrm{H},1}$, $\mathbf{v}^*_{\mathrm{C}}$, subject to that (i) both $\mathbf{v}^*_{\mathrm{H},0}$ and $\mathbf{v}^*_{\mathrm{H},1}$ contain exactly one negative element; (ii) the number of positive elements is identical in both $\mathbf{v}^*_{\mathrm{H},0}$ and $\mathbf{v}^*_{\mathrm{H},1}$; (iii) both $(\mathbf{PK}^*, \mathbf{v}^*_0)$, $(\mathbf{PK}^*, \mathbf{v}^*_1)$ constitute valid transactions, where $\mathbf{PK}^* = \{\mathbf{PK}^*_{\mathrm{H}}, \mathbf{PK}^*_{\mathrm{C}}\}$, $\mathbf{v}^*_0 = (\mathbf{v}^*_{\mathrm{H},0}, \mathbf{v}^*_{\mathrm{C}})$ and $\mathbf{v}^*_1 = (\mathbf{v}^*_{\mathrm{H},1}, \mathbf{v}^*_{\mathrm{C}})$. $\mathcal{CH}$ first picks a random bit β, then computes $\mathsf{tx}^* \leftarrow \mathsf{CreateTx}(sk^*_s, \mathbf{PK}^*, \mathbf{v}^*_\beta)$, inserts tx^* to T_{tx}, updates the states of corresponding accounts, then send tx^* to $\mathcal{A}$ as the challenge.
4. Post-challenge queries: $\mathcal{CH}$ answers queries to the oracles $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$ as in the pre-challenge stage, but rejects in the following two cases: (i) $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{extH}}$ with accounts in $\mathbf{PK}^*$; (ii) $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{trans}}$, such that there exists an account $pk_i \in \mathbf{PK}^*_{\mathrm{H}}$ for which either $\tilde{v}_i + v^*_{\mathrm{H},0,i} + v_{\mathrm{sum},i}$ or $\tilde{v}_i + v^*_{\mathrm{H},1,i} + v_{\mathrm{sum},i}$ falls outside the interval $[0, v_{\max}]$, where $\tilde{v}_i$ is the balance of pk_i before tx^* and $v_{\mathrm{sum},i}$ is the accumulation of the transfer amounts in $\mathcal{A}$'s queries to $\mathcal{O}_{\mathsf{trans}}$.
5. Guess: $\mathcal{A}$ outputs a guess β' for β and wins if $\beta' = \beta$.

According to the definition, we have

$$\mathsf{Adv}_{\mathcal{A}} = |\Pr[S_0] - 1/2|.$$

Game 1. Same as Game 0, except that $\mathcal{CH}$ picks a target subset $S \subset [Q_{\mathrm{honest}}]$, $|S| \leq n$ at the beginning, Q_{honest} is the maximum number that $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{regH}}$, and $\mathcal{CH}$ aborts if any of the following two events happen: (i) in the pre-challenge query stage, $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{extH}}$ with pk_j, $j \in S$; (ii) in the challenge stage, $\mathcal{A}$ outputs $\mathbf{PK}^*_{\mathrm{H}}$ while $\mathbf{PK}^*_{\mathrm{H}} \neq \{pk_j\}_{j \in S}$. Let W be the event that $\mathcal{CH}$ does not abort, then it holds $\Pr[W] = \frac{1}{n} \cdot \frac{(|S|)!(Q-|S|)!}{Q!} \geq \frac{1}{n \cdot (Q-|S|+1)^{|S|}}$, where $Q = Q_{\mathrm{honest}}$. Conditioned on $\mathcal{CH}$ does not abort, $\mathcal{A}$'s view in Game 0 is identical to Game 1, we therefore have

$$\Pr[S_1] = \Pr[S_0] \cdot \Pr[W].$$

Game 2. Same as Game 1, except that (i) in the setup stage, $\mathcal{CH}$ runs the simulator $(crs, \tau) \leftarrow \mathsf{Sim}_1(pp_{\text{nizk}})$; (ii) in the query stage when $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{trans}}$ and in the challenge stage, $\mathcal{CH}$ generates π_{legal} for $\mathsf{memo} \in L_{\text{legal}}$ via running $\mathsf{Sim}_2(crs, \tau, \mathsf{memo})$. Based on the adaptive zero-knowledge property of the NIZK, we have

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\lambda).$$

Below, we argue that no PPT adversary has non-negligible advantage in Game 2.

Claim 5.6. *Assuming the* PKE *is IND-CPA secure in the multi-user setting, then* $\Pr[S_2] \leq \mathsf{negl}(\lambda)$ *for all PPT adversary* $\mathcal{A}$.

Proof. Suppose there exists an adversary $\mathcal{A}$ wins in Game 2 with non-negligible advantage, then we can construct an adversary $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and breaks the IND-CPA security of the PKE. Given the pp_{pke} and a vector of target public keys $\mathbf{PK}' = (pk'_1, \ldots, pk'_m)$, $\mathcal{B}$ interacts with $\mathcal{A}$ as below:

1. Setup: $\mathcal{B}$ runs $pp_{\text{nizk}} \leftarrow \text{NIZK.}\mathsf{Setup}(1^\lambda)$, $(crs, \tau) \leftarrow \mathsf{Sim}_1(pp_{\text{nizk}})$, picks an integer $n > 2$, sets $pp = (pp_{\text{pke}}, pp_{\text{nizk}}, crs, n)$ and sends pp to $\mathcal{A}$. $\mathcal{B}$ also picks a set of random indices $S = \{i_1, \ldots, i_m\} \subset [Q_{\text{honest}}]$.

2. Pre-challenge queries: $\mathcal{B}$ answers $\mathcal{A}$'s queries by maintaining three lists T_{honest}, T_{corrupt} and T_{tx}, which are initially empty.

   - $\mathcal{O}_{\mathsf{regH}}$: on the i-th query, $\mathcal{B}$ picks a random balance $\tilde{v}$ and then proceeds as below: (i) if $i \notin S$, runs $(pk, sk, \tilde{C}) \leftarrow \mathsf{CreateAccount}(\tilde{v})$, appends $(pk, sk, \tilde{C}, \tilde{v})$ to the list T_{honest}; (ii) if $i = i_j$, sets $pk = pk'_j$, computes $\tilde{C} \leftarrow \text{PKE.}\mathsf{Enc}(\tilde{v})$, appends $(pk, \perp, \tilde{C}, \tilde{v})$ to the list T_{honest}. Then $\mathcal{B}$ returns $(pk, \tilde{C})$ to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{regC}}$: upon receiving a query $(pk, \tilde{C})$ (which is not allowed to be the one in T_{honest}), $\mathcal{B}$ records $(pk, \perp, \tilde{C}, \perp)$ to the list T_{corrupt}.
   - $\mathcal{O}_{\mathsf{extH}}$: upon receiving a query $pk \in T_{\text{honest}}$, if $pk \in \{pk_{i_1}, \ldots, pk_{i_m}\}$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ responds with the sk recorded in T_{honest}, and moves this entry to the T_{corrupt}.
   - $\mathcal{O}_{\mathsf{trans}}$: upon receiving a query $(pk_s, (pk_{r_i}, v_{r_i})_{j=1}^t)$, where $pk_s \in T_{\text{honest}}$, $\mathcal{B}$ responds as below: (i) honestly generates $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}})$; (ii) runs $\pi_{\text{legal}} \leftarrow \mathsf{Sim}_2(crs, \tau, \mathsf{memo})$. Next, $\mathcal{B}$ resets $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$ and returns $\mathsf{tx} = (\mathsf{memo}, \pi_{\text{legal}})$ to $\mathcal{A}$, then records tx to the list T_{tx} and updates the states of associated accounts.
   - $\mathcal{O}_{\mathsf{inject}}$: upon receiving $\mathsf{tx} = (\mathsf{memo}, \pi_{\text{legal}})$, if $\text{NIZK.}\mathsf{Verify}(\mathsf{memo}, \pi_{\text{legal}}) = 1$, $\mathcal{B}$ inserts tx to T_{tx}; otherwise, $\mathcal{B}$ ignores.

3. Challenge: $\mathcal{A}$ outputs two sets $\mathbf{PK}^*_{\text{H}} \subset T_{\text{honest}}$, $\mathbf{PK}^*_{\text{C}} \subset T_{\text{corrupt}}$, and three vectors of transfer amounts $\mathbf{v}^*_{\text{H},0}$, $\mathbf{v}^*_{\text{H},1}$, $\mathbf{v}^*_{\text{C}}$, which are subject to conditions as Game 2. If $\mathbf{PK}^*_{\text{H}} \neq \mathbf{PK}'$, then $\mathcal{B}$ aborts. Else, $\mathcal{B}$ submits $\mathbf{v}^*_{\text{H},0}$ and $\mathbf{v}^*_{\text{H},1}$ to its own challenger and receives back $\mathbf{C}^*_{\text{H}}$, then it generates the transaction via the following steps: (i) honestly computes the corresponding transfer amounts' ciphertexts $C^*_{\text{C},i} = \mathsf{Enc}(PK^*_{\text{C},i}, v^*_{\text{C},i})$, sets $\mathsf{memo} = (t^*, \mathsf{AnonSet}, \mathbf{C}, \tilde{\mathbf{C}})$ where $\mathsf{AnonSet} = \{\mathbf{PK}^*_{\text{H}}, \mathbf{PK}^*_{\text{C}}\}$ and $\mathbf{C} = \{\mathbf{C}^*_{\text{H}}, \mathbf{C}^*_{\text{C}}\}$. (ii) runs $\pi_{\text{legal}} \leftarrow \mathsf{Sim}_2(crs, \tau, \mathsf{memo})$. Next, $\mathcal{B}$ resets $\mathsf{memo} = (t, \mathsf{AnonSet}, \mathbf{C})$ and returns $\mathsf{tx}^* = (\mathsf{memo}, \pi_{\text{legal}})$ to $\mathcal{A}$, then records tx^* to the list T_{tx} and updates the states of associated accounts.

4. Post-challenge queries: $\mathcal{B}$ answers queries to the oracles $\mathcal{O}_{\mathsf{regH}}$, $\mathcal{O}_{\mathsf{regC}}$, $\mathcal{O}_{\mathsf{extH}}$, $\mathcal{O}_{\mathsf{trans}}$ and $\mathcal{O}_{\mathsf{inject}}$ as in the pre-challenge stage, but with the following exception: (i) $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{extH}}$ with accounts in $\mathsf{AnonSet}$; (ii) $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{trans}}$, such that there exists an account $pk_i \in \mathbf{PK}^*_{\text{H}}$ for which either $\tilde{v}_i + v^*_{\text{H},0,i} + v_{\text{sum},i}$ or $\tilde{v}_i + v^*_{\text{H},1,i} + v_{\text{sum},i}$ falls outside the interval $[0, v_{\max}]$, where $\tilde{v}_i$ is the balance of pk_i before tx^* and $v_{\text{sum},i}$ is the accumulation of the transfer amounts in $\mathcal{A}$'s queries to $\mathcal{O}_{\mathsf{trans}}$.

5. Guess: Finally, $\mathcal{A}$ outputs β'.

Then, $\mathcal{B}$ forwards β' to its own challenger. It is obvious that $\mathcal{B}$ simulates $\mathcal{A}$'s view in Game 2 perfectly. Thus, $\mathcal{B}$ wins with the same advantage as $\mathcal{A}$ wins. This proves Claim 5.6. □

This proves Lemma 5.5. □

Putting all of the above together, Theorem 5.1 immediately follows. □

6 An Efficient Instantiation: Anonymous PGC

Following the generic construction of multi-receiver PPABC in Section 5.2, we present an efficient instantiation, *Anonymous PGC*. For the homomorphic PKE component, we adopt twisted ElGamal [CMTA20], which is compatible with ZKPs whose statements are expressed as Pedersen commitments. For the NIZK component, we employ our novel k-out-of-n proofs to construct the proof for legality in a modular way.

6.1 ZKP-Friendly Homomorphic PKE

Definition 6.1 (Twisted ElGamal PKE)**.** Twisted ElGamal PKE [CMTA20] consists of four algorithms as below:

- $\mathsf{Setup}(1^\lambda)$: runs $(\mathbb{G}, g, p) \leftarrow \mathsf{GroupGen}(1^\lambda)$, picks $h \xleftarrow{\mathrm{R}} \mathbb{G}$, outputs $pp = (\mathbb{G}, g, h, p)$.
- $\mathsf{KeyGen}(pp)$: picks $sk \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, outputs $(pk, sk) = (g^{sk}, sk)$.
- $\mathsf{Enc}(pk, m; r)$: computes $C_L = pk^r$, $C_R = g^r h^m$, outputs $C = (C_L, C_R)$.
- $\mathsf{Dec}(sk, C)$: parses $C = (C_L, C_R)$, recovers m from $h^m = C_R / C_L^{sk^{-1}}$.

Same as standard exponential ElGamal PKE, Twisted ElGamal is IND-CPA secure based on the DDH assumption [CMTA20].

ZKP-Friendly PKE. Twisted ElGamal PKE is superior to the standard exponential ElGamal PKE due to its ZKP-friendly nature. Specifically, the right part of the ciphertext is exactly a Pedersen commitment under the commitment key (g, h), whose DL relation is unknown to all users, and even the ciphertexts under different public keys have the same commitment key. Such structure makes twisted ElGamal PKE compatible with all ZKPs whose statement is of the form of Pedersen commitment. This property enables further decomposing and simplifying legality languages, and subsequently designing dedicated ZKPs for each sub-language.

By plugging the twisted ElGamal PKE into the generic construction of PPABC in Section 5.2, we have $\mathsf{memo} = (t, (pk_j)_{j=1}^n, (C_{j,L}, C_{j,R})_{j=1}^n)$. Through querying the balance ciphertext $\tilde{C}_j = (\tilde{C}_{j,L}, \tilde{C}_{j,R})$ for each j-th account pk_j, it can be extended to $\mathsf{memo} = (t, (pk_j)_{j=1}^n, (C_{j,L}, C_{j,R})_{j=1}^n, (\tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n)$. Based on the decomposition of the language L_{legal} in Section 5.2 and the ZKP-friendly nature of the twisted ElGamal PKE, we can further decompose and simplify languages L_{balance}, L_{zero}, L_{positive} and L_{solvent} as below:

(i) Balance: the ciphertexts are well-formed and sum of v_j equals zero.

$$\begin{aligned} L_{\text{balance}} = \{&(pp, (pk_j, C_{j,L}, C_{j,R})_{j=1}^n) \mid \exists\ (\mathbf{v}, \mathbf{r}) \text{ s.t.} \\ &\forall j \in [n], C_{j,L} = pk_j^{r_j} \wedge C_{j,R} = g^{r_j} h^{v_j} \wedge \sum v_j = 0\} \end{aligned}$$

(ii) Decoy integrity: exactly $n - t - 1$ of the v_j equal zero.

$$\begin{aligned} L_{\text{zero}} = \{&(pp, (C_{j,R})_{j=1}^n) \mid \exists\ (J, (r_j)_{j \in J}) \text{ s.t.} \\ &J \subset [n] \wedge |J| = n - t - 1 \wedge \forall j \in J, C_{j,R} = g^{r_j}\} \end{aligned}$$

(iii) Receiver validity: exactly t of the v_j are positive.

$$L_{\text{positive}} = \{(pp, (C_{j,R})_{j=1}^n) \mid \exists\ (S, (v_j)_{j\in S}, (r_j)_{j\in S}) \text{ s.t.} \\ S \subset [n] \wedge |S| = t \wedge \forall j \in S, C_{j,R} = g^{r_j} h^{v_j} \wedge v_j \in (0, v_{\max}]\}$$

(iv) Sender solvency: exactly one of v_j is negative and the corresponding balance v^* is non-negative.

$$L_{\text{solvent}} = \{(pp, (pk_j, C_{j,L}, C_{j,R}, \tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n) \mid \exists\ (sk, \ell_s, v^*, v_{\ell_s}, r_{\ell_s}) \text{ s.t.} \\ \ell_s \in [n] \wedge C_{\ell_s,R} = g^{r_{\ell_s}} h^{v_{\ell_s}} \wedge -v_{\ell_s} \in (0, v_{\max}] \wedge pk_{\ell_s} = g^{sk} \wedge \\ h^{v^*} = (\tilde{C}_{\ell_s,R} \cdot C_{\ell_s,R})/(\tilde{C}_{\ell_s,L} \cdot C_{\ell_s,L})^{sk^{-1}} \wedge v^* \in [0, v_{\max}]\}$$

6.2 NIZK for Transaction Legality

Below, we present the construction of NIZK proofs for the aforementioned sub-statements, leveraging the homogeneous and the heterogeneous k-out-of-n proofs introduced in earlier sections. NIZK for language L_{legal} can be obtained by carefully putting together these NIZKs.

NIZK for L_{balance}. Since the statement that sum of n committed values equals to 0 (i.e., $\sum v_j = 0$) can be efficiently proved via a standard Σ-protocol, we thus focus on proving the well-formedness of the ciphertexts, or rather, proving knowledge of n plaintexts and n randomnesses. We denote the simplified language as $L_{\text{well-formed}}$. While one could employ standard Σ-protocols in parallel n times to give proofs for $L_{\text{well-formed}}$, this approach would result in a proof size that scales linearly with the size of the anonymity set n, thereby limiting the overall scalability of the system. Obviously, the language $L_{\text{well-formed}}$ is a specific case of heterogeneous n-out-of-n statement, of which $F_1, \ldots, F_n \in \text{Hom}(\mathbb{Z}_p^2, \mathbb{G}^2)$ and $F_j : (v_j, r_j) \mapsto (pk_j^{r_j}, g^{r_j} h^{v_j})$, $j \in [n]$. Therefore, we can construct the protocol Π_{wellform} by directly instantiating the optimized version of $\Pi_{\text{hetero-koon}}$ in the $k = n$ case.

NIZK for L_{zero}. The language L_{zero} is a standard homogeneous k-out-of-n statement where $k = n - t - 1$. Thus, any homogeneous k-out-of-n proofs can be used to prove it, including the protocol $\Pi_{\text{hetero-koon}}$ in Section 4 and $\Pi_{\text{homo-koon}}$ in Section 3, and other practical protocols, such as that in [ACF21, ZGSX23]. For our implementation, we adopt the protocol from [ZGSX23] due to its superior efficiency. It's worth noting that, based on our modular design, any advancement in the k-out-of-n proofs can be leveraged to improve the efficiency of the Anonymous PGC.

NIZK for L_{positive}. NIZK for L_{positive} can be constructed by adapting the protocol homogeneous k-out-of-n range proof $\Pi_{\text{homo-koon}}^{\text{range}}$ (Section 3.3), where $k = t$, to support half-open range proof, following the method outlined in Section 2.4. Note that, in the case where there is only one receiver, there is no need to generate π_{positive}, as detailed in Section 5.2.

NIZK for L_{solvent}. Notably, the language L_{solvent} involves both homogeneous and heterogeneous 1-out-of-n range proofs.

Proving "exactly one v_j is negative" needs a homogeneous 1-out-of-n range proof. Specifically, given a list of Pedersen commitments $(C_{j,R})_{j=1}^n$, the sender proves knowledge of ℓ_s, v_{ℓ_s}, r_{ℓ_s} subject to $C_{\ell_s,R} = g^{r_{\ell_s}} h^{v_{\ell_s}}$ and $-v_{\ell_s} \in (0, v_{\max}]$. In this scenario, the instance is a list of Pedersen commitments with the same commitment key g, h, and thus it can be proved via the homogeneous 1-out-of-n range proof, i.e., the GK 1-out-of-n range proof (Section 3.1). As noted in Section 3.3, it is essentially an optimized special case of our general homogeneous k-out-of-n range proof when $k = 1$.

Proving "balance v^* is non-negative" needs a heterogeneous 1-out-of-n range proof. Specifically, given a list of group elements $(pk_j, C_{j,L}, C_{j,R}, \tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n$, the sender proves knowledge of ℓ_s, v^*, sk^{-1} subject to $pk_{\ell_s}^{sk^{-1}} = g$, $(\tilde{C}_{\ell_s,L} \cdot C_{\ell_s,L})^{sk^{-1}} h^{v^*} = \tilde{C}_{\ell_s,R} \cdot C_{\ell_s,R}$, and $v^* \in [0, v_{\max}]$. In this scenario, the instance is a list of group elements $(g, \tilde{C}_{j,R} \cdot C_{,R})_{j=1}^n$ mapped by distinct homomorphisms (i.e., $F_j : (x_1, x_2) \mapsto$

$(pk_j^{x_1}, (\tilde{C}_{j,L} \cdot C_{j,L})^{x_1} h^{x_2}) \in \mathrm{Hom}(\mathbb{Z}_p^2, \mathbb{G}^2)$, $j \in [n]$), and thus it can be proved via the heterogeneous 1-out-of-n range proof described in Section 4.3.

In summary, the language L_{solvent} can be decomposed into the following two sub-languages $L_{\text{sender-trans}}$ and $L_{\text{non-negative}}$. By reusing the commitment S_{ℓ_s}, we ensure both sub-proofs share the same witness ℓ_s.

$$
\begin{aligned}
L_{\text{negative}} = \{&(pp, (C_{j,R})_{j=1}^n, S_{\ell_s}) \mid \exists\ (\ell_s, v_{\ell_s}, r_{\ell_s}, \mathbf{s}, r_S) \text{ s.t.} \\
&\mathbf{s} \in \{0,1\}^n \wedge \langle \mathbf{s}, \mathbf{1}^n \rangle = 1 \wedge \langle \mathbf{s}, \mathbf{n} \rangle = \ell_s \wedge S_{\ell_s} = \mathbf{g}^{\mathbf{s}} \mathbf{h}^{\mathbf{s}-\mathbf{1}^n} u^{r_S} \wedge \\
&C_{\ell_s,R} = g^{r_{\ell_s}} h^{v_{\ell_s}} \wedge -v_{\ell_s} \in (0, v_{\max}]\} \\
L_{\text{non-negative}} = \{&(pp, (pk_j, C_{j,L}, C_{j,R}, \tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n, S_{\ell_s}) \mid \exists\ (sk, \ell_s, v^*, \mathbf{s}, r_S) \text{ s.t.} \\
&\mathbf{s} \in \{0,1\}^n \wedge \langle \mathbf{s}, \mathbf{1}^n \rangle = 1 \wedge \langle \mathbf{s}, \mathbf{n} \rangle = \ell_s \wedge S_{\ell_s} = \mathbf{g}^{\mathbf{s}} \mathbf{h}^{\mathbf{s}-\mathbf{1}^n} u^{r_S} \wedge \\
&pk_{\ell_s}^{sk^{-1}} = g \wedge (\tilde{C}_{\ell_s,L} \cdot C_{\ell_s,L})^{sk^{-1}} h^{v^*} = \tilde{C}_{\ell_s,R} \cdot C_{\ell_s,R} \wedge v^* \in [0, v_{\max}]\}
\end{aligned}
$$

where $\mathbf{n} = (1, \ldots, n) \in \mathbb{Z}_p^n$ and $\mathbf{s} \in \{0,1\}^n$ is the indication bit vector of the index ℓ_s (i.e., $s_j = 1$ iff $j = \ell_s$).

Instead of concatenating proofs π_{negative} and $\pi_{\text{non-negative}}$ directly, we present an optimized proof for L_{solvent} by reusing the overlaps of the two proofs. The full protocol Π_{solvent} for the language L_{solvent} is depicted in Figure 8.

6.3 Efficiency Analysis of Anonymous PGC

A transaction in Anonymous PGC comprises the following four parts: the number of receivers t, an anonymity set $\mathsf{Anonset}$, the ciphertexts $\{C_i\}_{i\in[n]}$, and the legality proof π_{legal}. The latter is further composed of four sub-proofs: π_{balance}, π_{zero}, π_{positive}, and π_{solvent}.

Among these components, the legality proof π_{legal} is the primary determinant of the system's efficiency. Its computational complexity dominates both transaction generation and verification. Transaction generation requires $O(n \log n + t\nu)$ operations, where the primary cost comes from the homogeneous k-out-of-n range proofs used in π_{solvent} (with $k = 1$) and π_{positive} (with $k = t$). Transaction verification has a complexity of $O(n + t\nu)$, which is similarly dominated by these two sub-proofs. Here, $\nu = \log v_{\max}$ denotes the bit-length of the maximum transfer amount, t is the number of receivers, and n is the size of the anonymity set. In terms of transaction size, it is determined by all components: $\mathsf{Anonset}$, $\{C_i\}_{i\in[n]}$ and π_{legal}. Concretely, it consists of $3n + 20\log n + 10\log\nu + 4\log t + 64$ group elements and $(t+1)\log n + 53$ field elements, yielding an asymptotic size of $O_\lambda(n + t\log n + \log(t\nu))$, where λ is the security parameter.

7 Implementation and Evaluation

We implement and evaluate our core k-out-of-n protocols, including the homogeneous k-out-of-n proof $\Pi_{\text{homo-koon}}$ (Figure 4), the homogeneous k-out-of-n range proof $\Pi_{\text{homo-koon}}^{\text{range}}$ (Figure 5), and the heterogeneous k-out-of-n proof $\Pi_{\text{hetero-koon}}$ (Figure 6). Building on these protocols, we then construct the Anonymous PGC in a modular fashion and evaluate its performance.

7.1 Experimental Setup

All our experiments are conducted on a laptop equipped with an Intel(R) Core(TM) i7-11800 CPU operating at 2.3GHz and 16GB of RAM, running Ubuntu 22.04.05 LTS. We report results from a single-threaded configuration. The protocols are built atop elliptic curve NIST P-256 [Ope] with 128 bit security (where $|\mathbb{G}| = 33$ bytes, $|\mathbb{Z}_p| = 32$ bytes).

7.2 Performance of k-out-of-n Proofs

With the experimental setup established, we now evaluate the efficiency of our core k-out-of-n proofs.

$\mathsf{st} = (pp, (pk_j, C_{j,L}, C_{j,R}, \tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n)$

$\mathcal{P}(\mathsf{st}; sk, \ell_s, v^*, v_{\ell_s}, r_{\ell_s})$ | $\mathcal{V}(\mathsf{st})$

Run $\Pi^{\text{range}}_{\text{hetero-1oon}}$ (Section 4.3) on input $((pk_j, C_{j,L}, C_{j,R}, \tilde{C}_{j,L}, \tilde{C}_{j,R})_{j=1}^n; \ell_s, sk^{-1}, v^*)$, set $F_j : (x_1, x_2) \mapsto (pk_j^{x_1}, (\tilde{C}_{j,L} \cdot C_{j,L})^{x_1} h^{x_2})$, $j \in [n]$
// prove $L_{\text{non-negative}}$

$\boldsymbol{\ell}_s \in \{0,1\}^m$ s.t. $\langle \boldsymbol{\ell}_s, \mathbf{2}^m \rangle = \ell_s$
$r_B \xleftarrow{R} \mathbb{Z}_p$, $B := \mathbf{g}^{\boldsymbol{\ell}_s} u^{r_B}$
$(state_{\text{bit-linear}}, A, C, D) \leftarrow \mathcal{P}_{\text{bit-linear}}(pp_{\text{bit-linear}}, B; \boldsymbol{\ell}_s, r_B)$
For $d = 0, \ldots, m-1$
$\alpha_d \xleftarrow{R} \mathbb{Z}_p$, $C_{(\text{c})\deg_d} := \prod_{j=1}^n C_{j,R}^{p_{j,d}} g^{\alpha_d}$
$\beta_d \xleftarrow{R} \mathbb{Z}_p$, $C_{(\text{s})\deg_d} := \prod_{j=1}^n (g_j h_j)^{p_{j,d}} u^{\beta_d}$
$(state^1_{\text{range}}, V_1, S_1) \leftarrow \mathcal{P}_{\text{range}}(pp_{\text{range}}; -v_{\ell_s})$
$(state^2_{\text{range}}, V_2, S_2) \leftarrow \mathcal{P}_{\text{range}}(pp_{\text{range}}; -v_{\ell_s} - 1)$

A, B, C, D, $(C_{(\text{c})\deg_d})_{d=0}^{m-1}$, $(C_{(\text{s})\deg_d})_{d=0}^{m-1}$, V_1, S_1, V_2, S_2 →

$y_1, z_1, y_2, z_2 \xleftarrow{R} \mathbb{Z}_p$

← y_1, z_1, y_2, z_2

$(state^1_{\text{range}}, T_{11}, T_{12}) \leftarrow \mathcal{P}_{\text{range}}(y_1, z_1; state^1_{\text{range}})$
$(state^2_{\text{range}}, T_{21}, T_{22}) \leftarrow \mathcal{P}_{\text{range}}(y_2, z_2; state^2_{\text{range}})$

$T_{11}, T_{12}, T_{21}, T_{22}$ →

$x_0, x_1, x_2 \xleftarrow{R} \mathbb{Z}_p$

← x_0, x_1, x_2

$(\mathbf{f}, z_A, z_C) \leftarrow \mathcal{P}_{\text{bit-linear}}(x_0; state_{\text{bit-linear}})$
$z_G := r_{\ell_s} \cdot x_0^m - \sum_{d=0}^{m-1} \alpha_d x_0^d$
$z_S := r_S \cdot x_0^m + \sum_{d=0}^{m-1} \beta_d x_0^d$
$(state^1_{\text{range}}, \mu_1, \hat{t}_1, \mathbf{l}_1, \mathbf{r}_1) \leftarrow \mathcal{P}_{\text{range}}(x_1; state^1_{\text{range}})$
$(state^2_{\text{range}}, \mu_2, \hat{t}_2, \mathbf{l}_2, \mathbf{r}_2) \leftarrow \mathcal{P}_{\text{range}}(x_2; state^2_{\text{range}})$
$\tau_{r1} = \tau_{12} \cdot x_1^2 + \tau_{11} \cdot x_1 + z_1^2 \cdot \gamma$
$\tau_{r2} = \tau_{22} \cdot x_2^2 + \tau_{21} \cdot x_2 + z_2^2 \cdot \gamma$
where $\gamma = -z_G / x_0^m$,
and (τ_{11}, τ_{12}), (τ_{21}, τ_{22}) can be got from $state^1_{\text{range}}$, $state^2_{\text{range}}$ respectively

$\mathbf{f}, z_A, z_C, z_S$, $(\mu_1, \hat{t}_1, \mathbf{l}_1, \mathbf{r}_1, \tau_{r1})$, $(\mu_2, \hat{t}_2, \mathbf{l}_2, \mathbf{r}_2, \tau_{r2})$ →

Compute:
$G := \prod_{j=1}^n (C_{j,R})^{p_j} \cdot \prod_{d=0}^{m-1} C_{(\text{c})\deg_d}^{-x_0^d}$
$S' := \left(\prod_{j=1}^n h_j\right)^{-x_0^m} \prod_{j=1}^n (g_j h_j)^{p_j} \cdot \prod_{d=0}^{m-1} C_{(\text{s})\deg_d}^{-x_0^d}$

Accept iff all the checks pass:
$\mathcal{V}_{\text{bit-linear}}(pp_{\text{bit-linear}}, B, A, C, D, x, \mathbf{f}, z_A, z_C) \stackrel{?}{=} 1$
$S^{x_0^m} / S' \stackrel{?}{=} u^{z_S}$, where S is recorded in $\pi^{\text{range}}_{\text{hetero-1oon}}$
Π_{range} as in Section 2.4 except for altering
$g^{\tau_{r1}} h^{\hat{t}_1} \stackrel{?}{=} (G^{-x_0^{-m}})^{z_1^2} \cdot h^{\delta(y_1, z_1)} \cdot T_{11}^{x_1} \cdot T_{12}^{x_1^2}$
$g^{\tau_{r2}} h^{\hat{t}_2} \stackrel{?}{=} (G^{-x_0^{-m}} / h)^{z_2^2} \cdot h^{\delta(y_2, z_2)} \cdot T_{21}^{x_2} \cdot T_{22}^{x_2^2}$

Figure 8: Π_{solvent} for L_{solvent}: proof for sender solvency

Performance of the homogeneous k-out-of-n (range) proof. We benchmark our homogeneous k-out-of-n proof $\Pi_{\text{homo-koon}}$ and its integrated version with Bulletproofs range proofs, $\Pi^{\text{range}}_{\text{homo-koon}}$, across various parameters (Table 3). To evaluate the integration overhead, we also implement the standard Bulletproof range proof Π_{range} and define the following *rate* metric:

$$\textit{rate} = \frac{\text{cost of } \Pi^{\text{range}}_{\text{homo-koon}}}{\text{cost of } \Pi_{\text{homo-koon}} + \text{cost of } \Pi_{\text{range}}}.$$

Here, $\textit{rate} > 1$ indicates that there are bridging costs when combining a k-out-of-n proof with Bulletproofs; $\textit{rate} = 1$ indicates that there are no bridging costs; $\textit{rate} < 1$ indicates that there are overlaps between the two protocols and these parts are reused in the combination. A smaller *rate* implies lower overhead.

The results in Table 3 show that the values of rate for prover time, verifier time, and proof size are all below 1. This indicates that our homogeneous k-out-of-n proof, $\Pi_{\text{homo-koon}}$, integrates seamlessly with the range proof by reusing the overlapping components between the two protocols.

Table 3: The performance of the homogeneous k-out-of-n range proof

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

Performance of the heterogeneous k-out-of-n proof. To demonstrate the efficiency of our heterogeneous k-out-of-n proof $\Pi_{\text{hetero-koon}}$, we compare it with the ACF k-out-of-n proof [ACF21, Protocol 3] under different parameter settings (see Table 4). To ensure a fair comparison, we implement our $\Pi_{\text{hetero-koon}}$ protocol for a homogeneous statement by setting the homomorphism $F : x \mapsto g^x$, thereby aligning it with the ACF protocol. Despite this alignment, we emphasize that our protocol provides richer functionality, as it also supports proving heterogeneous statements. The results in Table 4 show that, although the proof size is doubled, $\Pi_{\text{hetero-koon}}$ achieves a 1.4–5.6× improvement in prover time and a 2.4–10.9× improvement in verifier time, compared to the ACF protocol. Values that demonstrate the superior performance of our protocol are highlighted.

7.3 Performance of Anonymous PGC

In this work, we focus on the transaction layer and aim to advance the cryptographic foundations of account-based cryptocurrencies. Thus, we give a prototype implementation of Anonymous PGC as a standalone cryptocurrency in C++ to showcase its performance. To ensure a fair comparison, we re-implement

Table 4: Performance comparison of heterogeneous k-out-of-n proof and ACF proof

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

Anonymous Zether in C++ and exclude the epoch-handling overhead, as countermeasures against replay attacks are not considered in this work (see Section 5 for details). Since Anonymous Zether does not natively support multi-receiver transactions, we use t independent single-receiver transactions as a t-receiver transfer, consistent with the approach in [GKPH24]. We emphasize again that Anonymous PGC offers stronger security guarantee than Anonymous Zether (as stated in Section 5).

In Table 2, we have already compared Anonymous PGC with closely related account-based cryptocurrencies in terms of asymptotic complexity and security. In Table 5 shown as below, we give a more fine-grained measurements of the transaction sizes of Anonymous PGC and Anonymous Zether, which is an important metric in the realm of cryptocurrencies. It turns out that Anonymous PGC has a significant advantage when conducting multi-receiver transfers. In the multi-receiver scenarios, the transaction size of Anonymous PGC grows only linearly in t with a logarithmic factor in n (i.e., $O(t \log n)$), whereas that of Anonymous Zether grows linearly in t with a multiplicative factor of n (i.e., $O(tn)$). In the single-receiver case ($t = 1$) of Anonymous PGC, the proof size is even smaller than its generalized case degrading to 1, cause the combination of proofs for balance, decoy integrity and sender solvency already imply the existence of exactly one positive transfer amount, making the proof for receiver validity redundant.

Table 5: Transaction sizes of Anonymous PGC and Anonymous Zether

Systems	Parameters	Transaction Size	
		$\#\,\mathbb{G}$	$\#\,\mathbb{Z}_p$
Anon. Zether	$t = 1$	$2n + 8\log n + 2\log \nu + 15$	$2\log n + 9$
	$t > 1$	$t(2n + 8\log n + 2\log \nu + 15)$	$t(2\log n + 9)$
Anon. PGC	$t = 1$	$3n + 16\log n + 6\log \nu + 48$	$\log n + 35$
	$t > 1$	$3n + 20\log n + 10\log \nu + 4\log t + 64$	$(t+1)\log n + 53$

Here, $\nu = \log v_{\max}$, indicating the length of the maximum amounts (i.e., 32 in practice). n is the size of the anonymity set and t is the number of receivers in a single transaction.

Table 6 reports the performances of Anonymous PGC and Anonymous Zether. The results show that, in addition to asymptotic improvement and security enhancement, Anonymous PGC also exhibits superior concrete performance. For an anonymity set size of 64 and 8 receivers, Anonymous PGC achieves a 2.4× and 5.7× improvement in transaction generation and verification times, respectively, along with a 2.2× reduction in transaction size. Furthermore, the advantages of Anonymous PGC in terms of computational costs (transaction generation and verification) become more pronounced as the anonymity set size and the number of receivers increase.

To demonstrate the scalability advantages of Anonymous PGC, we compare it with Anonymous Zether

Table 6: Comparisons between Anonymous PGC and 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

Here, n is the size of the anonymity set and t is the number of receivers. We set the maximum transfer amount as $v_{\max} = 2^{32} - 1$.

by fixing the anonymity set size to 2^8 and varying the number of receivers from 1 to 2^7. In this comparison, we focus on efficiency metrics and do not consider deployment constraints such as block size limits, which in practice bound the maximum transaction size [MS23]. As shown in Figure 9, the costs of Anonymous Zether grow linearly with the number of receivers across all three metrics, whereas Anonymous PGC exhibits substantially slower growth, particularly in transaction size. These results demonstrate that Anonymous PGC achieves significantly better scalability in large-scale, multi-receiver settings.

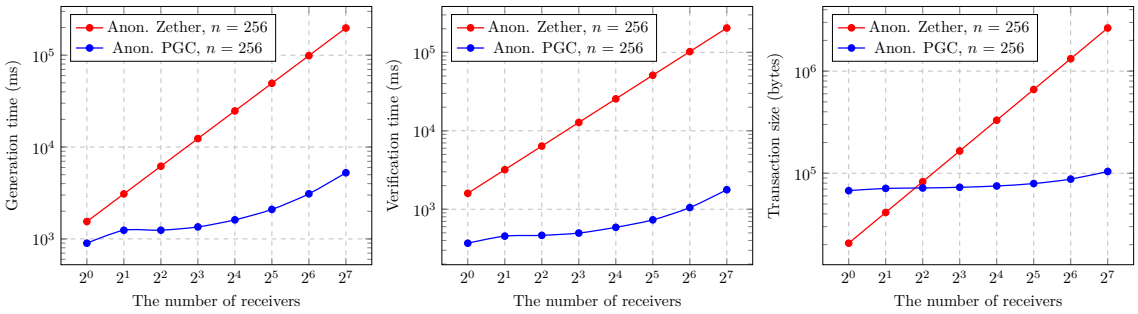

Figure 9: Comparisons between Anonymous PGC and Anonymous Zether under varying parameters.

Acknowledgments

We are grateful to Xiaoyun Wang for encouraging us to initiate this work. We thank Zhiying Cui and Hongxu Yi for valuable discussions during the early stage of this work.

References

[AAB+21] Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen. Acyclicity programming for sigma-protocols. In *TCC*, 2021.

[ABFV22] Gennaro Avitabile, Vincenzo Botta, Daniele Friolo, and Ivan Visconti. Efficient proofs of knowledge for threshold relations. In *ESORICS*, 2022.

[AC20] Thomas Attema and Ronald Cramer. Compressed sigma-protocol theory and practical application to plug & play secure algorithmics. In *CRYPTO*, 2020.

[ACF21] Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In *CRYPTO*, 2021.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *FC*, 2020.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE SP*, 2018.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT*, 2000.

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *ESORICS*, 2015.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, 2016.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE SP*, 2014.

[BCG+20] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: enabling decentralized private computation. In *IEEE SP*, 2020.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60. Springer, 2016.

[BCT21] Aritra Banerjee, Michael Clear, and Hitesh Tewari. zkhawk: Practical private smart contracts from mpc-based hawk. *CoRR*, abs/2104.09180, 2021.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *FC*, 2014.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *ASIACRYPT*, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, 1993.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, 1994.

[CMTA20] Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. PGC: decentralized confidential payment system with auditability. In *ESORICS*, 2020.

[Das] Dash. `https://www.dash.org`.

[Dia21] Benjamin E. Diamond. Many-out-of-many proofs and applications to anonymous zether. In *IEEE SP*, 2021.

[EKRN24] Liam Eagen, Sanket Kanjalkar, Tim Ruffing, and Jonas Nick. Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments. In *EUROCRYPT*, 2024.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, STOC 1990*, pages 308–317, 1990.

[FMMO19] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *ASIACRYPT*, 2019.

[FP01] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*, 2001.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

[GGHK22] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking sigmas: A framework to compose sigma-protocols for disjunctions. In *EUROCRYPT*, 2022.

[GHKS23] Aarushi Goel, Mathias Hall-Andersen, Gabriel Kaptchuk, and Nicholas Spooner. Speed-stacking: Fast sublinear zero-knowledge proofs for disjunctions. In *EUROCRYPT*, 2023.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT*, 2008.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT*, 2015.

[GKPH24] Yue Guo, Harish Karthikeyan, Antigoni Polychroniadou, and Chaddy Huussin. Pride CT: towards public consensus, private transactions, and forward secrecy in decentralized payments. In *IEEE SP*, 2024.

[GN18] Brandon Goodell and Sarang Noether. Thring signatures and their applications to spender-ambiguous digital currencies. *IACR Cryptol. ePrint Arch.*, page 774, 2018.

[Gri] Grin. `https://grin-tech.org/`.

[Gro04] Jens Groth. Evaluating security of voting schemes in the universal composability framework. In *ACNS*, 2004.

[HAB+17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *NDSS*, 2017.

[JM20] Aram Jivanyan and Tigran Mamikonyan. Hierarchical one-out-of-many proofs with applications to blockchain privacy and ring signatures. *AsiaJCIS*, 2020.

[KMS+16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE SP*, 2016.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology*, 2003.

[Max13] Greg Maxwell. Coinjoin: Bitcoin privacy for the real world, 2013. `https://people.xiph.org/~greg/confidential_values.txt`.

[Max15] Greg Maxwell. Confidential transactions, 2015. `https://people.xiph.org/~greg/confidential_values.txt`.

[MS23] Varun Madathil and Alessandra Scafuro. Prifhete: Achieving full-privacy in account-based cryptocurrencies is possible, 2023. `https://eprint.iacr.org/2023/710`.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[NMT16] Shen Noether, Adam Mackenzie, and Monero Core Team. Ring confidential transactions. 2016. https://lab.getmonero.org/pubs/MRL-0005.pdf.

[NVV18] Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *USENIX NSDI*, 2018.

[Ope] https://github.com/openssl/openssl/blob/4e6647506331fc3b3ef5b23e5dbe188279ddd575/include/openssl/obj_mac.h.

[Poe16] Andrew Poelstra. Mimblewimble, 2016. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf.

[RMK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *ESORICS*, 2014.

[WGL+24] Chenke Wang, Zhonghui Ge, Yu Long, Xian Xu, Shi-Feng Sun, and Dawu Gu. Blindshuffler: Universal and trustless mixing for confidential transactions. In *ASIA CCS*, 2024.

[Woo14] Gavin Wood. Ethereum: A secure decentralized transaction ledger. http://gavwood.com/paper.pdf, 2014.

[ZGSX23] Tianyu Zheng, Shang Gao, Yubo Song, and Bin Xiao. Leaking arbitrarily many secrets: Any-out-of-many proofs and applications to ringct protocols. In *IEEE SP*, 2023.

[ZLH+22] Gongxian Zeng, Junzuo Lai, Zhengan Huang, Yu Wang, and Zhiming Zheng. Dag-Σ: A dag-based sigma protocol for relations in CNF. In *ASIACRYPT*, 2022.

[ZSLW24] Jianning Zhang, Ming Su, Xiaoguang Liu, and Gang Wang. Springproofs: Efficient inner product arguments for vectors of arbitrary length. In *IEEE SP*, 2024.