PriDe CT: Towards Public Consensus, Private Transactions, and Forward Secrecy in Decentralized Payments

Yue Guo*, Harish Karthikeyan†, Antigoni Polychroniadou‡
J.P. Morgan AI Research, J.P. Morgan AlgoCRYPT CoE
New York, NY, USA
*Email: *yue.guo@jpmchase.com, †harish.karthikeyan@jpmchase.com, ‡antigonipoly@gmail.com*

***Abstract*—Anonymous Zether, proposed by Bünz *et al.* (FC, 2020) and subsequently improved by Diamond (IEEE S&P, 2021) is an account-based confidential payment mechanism that works by using a smart contract to achieve privacy (i.e. identity of receivers to transactions and payloads are hidden). In this work, we look at simplifying the existing protocol while also achieving batching of transactions for multiple receivers, while ensuring consensus and forward secrecy. To the best of our knowledge, this work is the first to formally study the notion of forward secrecy in the setting of blockchain, borrowing a very popular and useful idea from the world of secure messaging. Specifically, we introduce:**

- **FUL-Zether, a forward-secure version of Zether (Bünz *et al.* , FC, 2020).**
- **PRIvate DEcentralized Confidential Transactions (PriDe CT), a much-simplified version of Anonymous Zether that achieves competitive performance and enables batching of transactions for multiple receivers.**
- **PRIvate DEcentralized Forward-secure Until Last update Confidential Transactions (PriDeFUL CT), a forward-secure version of PriDe CT.**

We also present an open-source, Ethereum-based implementation of our system. PriDe CT uses linear homomorphic encryption as Anonymous Zether but with simpler zero-knowledge proofs. PriDeFUL CT uses an updatable public key encryption scheme to achieve forward secrecy by introducing a new DDH-based construction in the standard model.

In terms of transaction sizes, Quisquis (Asiacrypt, 2019), which is the only cryptocurrency that supports batchability (albeit in the UTXO model), has 15 times more group elements than PriDe CT. Meanwhile, for a ring of N receivers, Anonymous Zether requires $6 \log N$ more terms even without accounting for the ability to batch in PriDe CT. Further, our implementation indicates that, for $N = 32$, even if there were 7 intended receivers, PriDe CT outperforms Anonymous Zether in proving time and gas consumption.

1. Introduction

Blockchain systems are stateful primitives used to achieve consensus among mutually distrusted parties. For cryptocurrencies, this state can consist of information about each user's balance and the transactions affecting these balances. They store the state "in the clear," allowing any user to verify the consensus. There are two approaches to encoding the state of a blockchain: the Unspent Transaction Output (UTXO) model and the account-based model. Both models have been subjects of research to provide privacy-preserving alternatives.

In the UTXO model, several privacy-preserving alternatives have been proposed, including ZCash [1], Monero [2], SwapCT [3], Confidential Assets [4], and Quisquis [5].On the other hand, in the account-based model, which supports smart contracts, the approach is to use the smart contract itself as a "cryptocurrency" [6], [7]. This results in an efficient wallet, as users only need to know their secret key and account to transact. In contrast, cryptocurrencies in the UTXO model require scanning the entire payment history or relying on a third-party "filter" to determine available coins for spending. UTXO-based privacy solutions often face challenges in maintaining a monotonically increasing state, as spent tokens cannot be easily purged without identification. Furthermore, it has been shown by empirical analysis that privacy-preserving cryptocurrencies in the UTXO model often fail to provide privacy to participants [8], [9]. This inspired another approach to privacy - "mixing" the inputs and the outputs, thereby severing any links between the input and the output coins but some require active participation of users.

In secure messaging [10], [11], forward secrecy [12], is a crucial security goal, that ensures past communications remain secure even if a user's security is compromised. This attack involves an adversary storing messages for future decryption. However, in the context of blockchain, the challenge is more significant due to its public transaction history. It is even more pertinent, especially with the possibility of

"harvest now, decrypt later" attacks [13], leveraging that the blockchains already have the data harvested. Most papers aiming to tackle this issue of forward secrecy advocate for the naive approach of constantly generating new key pairs, redirecting funds to the new account and then deleting the old keys. This leads to the proliferation of accounts that are needlessly receiving funds or participating in decoy transactions, which can lead to loss of privacy as one can easily identify such dormant accounts. To ensure provable security guarantees on old transactions, it is crucial to explore key evolving primitives in the blockchain context, whereby confidentiality of past transactions is guaranteed even in the event of a current compromise. Moreover, interest is growing in the vulnerability of historical blockchain data to future attacks. Some blockchain technologies, like Algorand, are pursuing an orthogonal line of research to protect their state from potential quantum threats by integrating post-quantum secure state-proofs into their infrastructure. This motivates further research to establish forward secrecy in the blockchain.

1.1. Problem Statement

Our motivation for this line of research is two-fold. On the one hand, the academic research initiated by Zether [6] and improved upon by Anonymous Zether [7] which desired to transfer assets, while preserving consensus and ensuring that the transaction is confidential (i.e., the payload is hidden), private (i.e., the identity of the parties to the transaction remain private), and at the same time leverage the blockchain as an immutable ledger to achieve a decentralized transaction platform. Meanwhile, it is also driven by business needs of the blockchain launched by J.P.Morgan [14]. At its core, Onyx blockchain is an Ethereum-based blockchain, operating in the account based model. The current architecture is built on top of Quorum [15] which has attractive features of public consensus. However, this public consensus is sacrificed at the altar of privacy as the current proposal for privacy involves using Tessera [16] which is a "privacy transaction manager". The idea behind this solution is for two parties who wish to transact confidentially to insert a transaction record in the public blockchain, while the actual transaction corresponding to that record is solely existing on the encrypted storage of the clients. Therefore, while everyone can observe a transaction was inserted but can never agree to a public state as the transaction details are hidden from everyone but the participating clients. Meanwhile, this solution also requires some off-chain communication to ensure that the details of the transaction are finalized between the two parties. Our work is primarily motivated by this practical business needs on achieving public consensus while offering privacy-preserving transactions. More specifically, we aim to build a provably secure protocol that achieves the following properties:

- It is set in the account-based model, where we employ the smart contract as a cryptocurrency in an effort to have an efficient wallet.
- It is interoperable with other smart contracts, a key facet of Ethereum smart contracts, and building atop the Ethereum blockchain, thereby enhancing functionality.
- It allows for the passive participation of users, i.e., a party need not be present online to ensure a transaction has to succeed.
- It offers a payment mechanism where transactions are confidential with receiver privacy.
- It supports batching of multiple transactions in one big message, i.e., allowing multiple receivers to receive funds in one posted message.
- It can support the concurrency of transactions without an expensive locking mechanism, i.e., where a receiver's account has to be locked by a sender, thereby ensuring that only that sender can transact with the receiver.
- It offers support for forward secrecy, without requiring active receiver participation, a mechanism that can be implemented with little to no friction.

1.2. Our Contribution

1.2.1. PriDe CT In this work, we present PriDe CT which stands for **Pri**vate, **De**centralized, **C**onfidential **T**ransactions. This is a paradigm that builds on the work of Zether [6] and Anonymous Zether [7] which focuses on the property where a client - with access to the blockchain - can, in constant time, determine its own state (and independent of the state of the blockchain). As remarked earlier, this is a feature that is a departure from existing cryptocurrencies that offer privacy-preserving alternatives mentioned before. We improve on the earlier works of Zether and Anonymous Zether. by also offering the perk of batchability, i.e., where a single transaction message can be used to send payloads to multiple receivers, all while simplifying the code base and the zero-knowledge proofs required. The resulting solution offers competitive performance, as we will see in Section 7. Finally, our scheme supports a subset of concurrent transactions without requiring an expensive locking mechanism outlined in earlier works. Specifically, multiple senders can send to a particular receiver and all honest transactions will succeed without needing any locking mechanism. Note that both Zether and Anonymous Zether heavily relied upon a locking mechanism, whereby a sender locks the account of its receiver(s) and then unlocks the account after transacting. This is an expensive process that can lead to delays.

Let us pause to review the approach undertaken. As done in both Zether and Anonymous Zether, the state of the system is a global table of "accounts" that employs ElGamal encryption [17]. Specifically, the table is a mapping from a public key pk to encryption of the user's balance $balance$ under that public key, i.e., $\mathsf{pk} \mapsto (g^r, \mathsf{pk}^r \cdot g^{balance})$ for some randomness r and g which is a generator of a cyclic group $\mathbb{G}$ of prime order p where DDH problem is hard. To send funds, the sender chooses a ring of receivers. However, unlike in Anonymous Zether where (a) the sender and the receiver were hidden among the ring of decoys and (b) only one of the ring members can actually receive money, PriDe CT

fixes the index of the sender and allows for any of the remaining ring members to be a recipient. Then, to transact, the sender encrypts the payload for each ring member, under their respective public key and also encrypts the debited balance with its own public key. The environment, say a smart contract, updates the balances homomorphically. Since the entire transaction happens under the hood of encryption, the sender attaches proof to demonstrate that it has behaved correctly. While sender anonymity is not offered, as argued later both Zether and Anonymous Zether do not offer the same because the identity of who invokes a smart contract is always revealed in Ethereum. Meanwhile, heaving dummy senders periodically post messages, even when not intending to transact, can support weak sender anonymity. Finally, PriDe CT can interoperate with existing smart contracts, in a manner similar to the use-cases defined in Zether [6]. More specifically, our protocol prioritizes interoperation with Ethereum, much like Zether and Anonymous Zether. It can interoperate with some ERC-20-compliant token contracts. Much like Zether, we can exhibit a functionality to tie accounts in PriDe CT to arbitrary smart contracts. Meanwhile, any ZK-proofs can only be constructed by the owner of the account as it requires a secret state that is unavailable to the arbitrary (and possibly malicious) smart contract. This functionality would help unlock applications such as sealed-bid auctions where the confidentiality of the payloads during the bidding process is paramount. We refer the reader to Zether [6, §8] for a detailed discussion on how to similarly use PriDe CT to interoperate with other smart contracts to unlock more functionalities.

1.2.2. FUL-security and PriDeFUL CT Note that in Forward-Secure Public Key Encryption (FS-PKE) [12], time is divided into epochs with the sender evolving the public key and the receiver evolving the secret key consistent with the epoch. Implicit in the security definition is that the sender and the receiver's epochs are the same, and thus when the compromise occurs it is of the most recent key. However, a more natural setting is one which allows the sender's epoch to run ahead of the receiver's epoch and the compromised key could belong to an epoch older than the sender's epoch. In such a setting, a receiver can remain offline and only periodically come online at which time it evolves the key and gets caught up with all the updates locally. Therefore, we model "**F**oward-secure **U**ntil **L**ast update" which leaks the receiver's epoch's key, where this epoch could be older than the sender's current epoch. Our security definition models a weakening of FS-PKE whereby the sender(s) aids the evolution of the receiver's key. We show how to effectively bootstrap Zether [6] to achieve a FUL-secure version. Finally, we present PriDeFUL CT which is the FUL-secure version of PRIvate DEcentralized Confidential Transactions.

Our technical tool in this setting is Updatable Public Key Encryption [10], [11], [18], [19], [20] where the sender aids the evolution of the receiver's key, a relaxation of FS-PKE where both the sender and the receiver evolves key independently but consistently. We begin by showing that ElGamal Encryption [17] is a secure UPKE at the basic security level. We then present efficient zero-knowledge proof to satisfy stronger security definition and finally prove that using this UPKE we can bootstrap Zether to build FUL-secure Zether. Along the way, we also show that ElGamal Encryption also satisfies the property of fast-forwarding [19] whereby a receiver who is in epoch i can jump to current epoch i by decrypting locally no more than $O(\log(j-i))$ ciphertexts. It is to be stated that the resulting construction is the *most efficient* construction of UPKE with the Fast-Forwarding feature outperforming both of the constructions proposed by Dodis *et al.* [19].

In summary, our results are as follows:

- We show that ElGamal Encryption is a secure UPKE under the DDH Assumption. We also show that it inherits the property of fast-forwarding [19].
- We formalize the security definition of sender-aided forward secrecy until the last update (FUL), which was never considered before.
- We show that using the ElGamal-based UPKE and appropriate Σ-protocol, we achieve FUL security of Zether under the DDH Assumption.
- We introduce PriDe CT which simplifies the Anonymous Zether protocol while allowing for batched transactions and receiver anonymity. We prove the security of our interactive proof protocol under the DDH Assumption.
- We show how to use ElGamal Encryption-based UPKE to bootstrap PriDe CT to build PriDeFUL CT that is FUL Secure, by using a similar approach as FUL-Zether.
- We present an open-sourced implementation of our protocol implemented with Javascript, Node.js, and Solidity.
- Finally, our experiments reveal that PriDe CT offers competitive performance when compared with the running time of Anonymous Zether by leveraging batchability.

Technical Challenges. Zether [6] used one-out-of-many proofs [21] to achieve anonymity. It was later refined by Anonymous Zether [7] using many-out-of-many proofs. Despite elegance, Anonymous Zether faced sender anonymity issues, an artifact of Ethereum where invocation to smart contract trivially reveals the identity of the invoking party. In our work, we simplify, prioritizing receiver anonymity with fixed sender location, supporting multiple receivers and concurrent transactions efficiently [7].

Forward Security. Traditional constructions of FS-PKE, built from HIBE [22], [23], are inefficient and lack interoperability with existing homomorphic encryption schemes. Therefore, we have to rely on achieving a weakening of forward security where the sender(s) help the receiver evolve the secret key. UPKE allows the sender to help evolve the secret key of the receiver by encrypting an update offset δ with the receiver's public key, enabling the receiver to update their secret key with the recovered δ. The sender

can also update the public key with the knowledge of δ, allowing the receiver to stay offline for epochs. Unfortunately, while the hashed ElGamal construction was a secure UPKE [10], [11], it lacks efficient zero-knowledge proofs for the sender to prove it has behaved honestly because of the hash function used in the encryption process. Therefore, we require a standard model construction. Existing standard model constructions of UPKE [18], [20], [24] unfortunately have inefficient parameter sizes or are un-interoperable with the current ElGamal-scheme-based approach. Our approach focuses on achieving forward secrecy in the account-based model, which can be easily implemented on Solidity. Therefore, we refocus our efforts on ElGamal encryption and prove various properties of it.

2. Preliminaries

Let $\mathbb{G}$ denote a cyclic group of prime order q, with a generator g. Additional generators are denoted by $h, v, u \in \mathbb{G}$. Further, we denote by $\mathbb{Z}_q$ or $\mathbb{F}_q$ the ring of integers modulo the prime q and $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. Now, let $\mathbb{G}^n, \mathbb{F}_q^n$ be vector spaces of dimension n for $\mathbb{G}$ and $\mathbb{F}_q$ respectively.

We use $\leftarrow\!\!\$$ to denote the uniform sampling of an element. For example, $y \leftarrow\!\!\$ \mathbb{F}_q$ denotes the uniform sampling of y from $\mathbb{F}_q$. We will use the bold font, in lower case, to denote vectors and bold font, in upper case, to denote matrices. For example $\mathbf{a} \in \mathbb{F}_q^n$ denotes a vector with elements $a_1, \ldots, a_n \in \mathbb{F}_q$ and $\mathbf{A} \in \mathbb{F}_q^{n \times m}$ denotes a matrix with n rows and m columns such that each elements is in $\mathbb{F}_q$.

With these notations in place, we can define the following set of operations involving vectors and scalars:

- For a scalar $c \in \mathbb{F}_q$ and a vector $\mathbf{a} \in \mathbb{F}_q^n$, we denote by $\mathbf{b} = c \cdot \mathbf{a} \in \mathbb{F}_q^n$ the scalar multiplication where $b_i = c \cdot a_i$.
- For vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$, we denote its inner product by $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i \cdot b_i \in \mathbb{F}_q$.
- For vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$, we define the concatenation of two vectors by $\mathbf{a} \parallel \mathbf{b}$.
- For $0 \leq \ell \leq n$, we define the slice of a vector $\mathbf{a}$ as follows: $\mathbf{a}_{[:\ell]} = (a_1, \ldots, a_\ell)$, $\mathbf{a}_{[\ell:]} = (a_{\ell+1}, \ldots, a_n)$ Therefore, $\mathbf{a}_{[:\ell]} \parallel \mathbf{a}_{[\ell:]} = \mathbf{a}$.
- For a scalar $k \in \mathbb{F}_q^*$, we use $\mathbf{k}^n$ to denote the vector containing the first n powers of k, i.e., $\mathbf{k}^n = (1, k, \ldots, k^{n-1}) \in \mathbb{F}_q^{*n}$ Specifically, $\mathbf{0}^n$ (resp. $\mathbf{1}^n$) are vectors of length n where each element is 0 (resp. 1) and $\mathbf{2}^n = (1, 2, 4, \ldots, 2^{n-1})$.

We can define vector polynomials as $p(X) = \sum_{i=0}^d \mathbf{p}_i \cdot X^i \in \mathbb{F}_q^n[X]$ where the coefficient $\mathbf{p}_i$ is a vector in $\mathbb{F}_q^n$. Then, the inner product between two such vector polynomials $l(X), r(X)$ is defined as:

$$\langle l(X), r(X) \rangle = \sum_{i=0}^{d} \sum_{j=0}^{i} \langle \mathbf{l}_i, \mathbf{r}_j \rangle \cdot X^{i+j} \in \mathbb{F}_q[X]$$

An important feature of the inner product of vector polynomials is that evaluating a polynomial at a point $x \in \mathbb{F}_q$ and then taking the inner product is the same as evaluating the inner product polynomial $t(X) = \langle l(X), r(x) \rangle$ on this point x.

3. Syntax and Security

3.1. Syntax

The payment system is supposed to offer support for the following operations: (a) account creation, (b) account funding, (c) transfer of assets, and (d) burning of the account. However, for simplicity, we only focus on the version of a payment that only offers the transfer of assets. This is similar to the syntax of the work by Diamond [7] and is simpler to analyze. Further, as remarked earlier, sender anonymity trivially breaks as the identity of the invocation of the smart contract is always revealed. Therefore we only focus on receiver anonymity. Further, Anonymous Zether allows for a transaction to have exactly one sender and one receiver. In PriDe CT, we streamline the cryptography involved by allowing multiple non-zero payloads, thereby utilizing Bulletproofs to provide range proofs on multiple payloads, which may include zero values.

Definition 1 (Decentralized Payment System)**.** *A payment system* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}\ \mathsf{Read}, \mathsf{Transact}, \mathsf{Verify}, \mathsf{Insert})$ *consists of a tuple of PPT algorithms with the following syntax:*

- $\mathsf{pp}, \mathsf{state} \leftarrow\!\!\$ \ \mathsf{Setup}(1^\kappa)$*: On input of security parameter* κ *in unary,* Setup *produces the public parameters* pp*. It also initializes the state* state *of the payment system.*
- $\mathsf{pk}, \mathsf{sk} \leftarrow\!\!\$ \ \mathsf{KeyGen}(\mathsf{pp})$*: On input of public parameters* pp*,* KeyGen *outputs a public key/secret key pair.*
- $\mathsf{acc} \leftarrow\!\!\$ \ \mathsf{Encrypt}(\mathsf{pk}, \mathsf{b})$*: On input of balance* b *and public key of a party* pk*,* Encrypt *outputs a ciphertext* acc *which encrypts* b *under* pk*.*
- $\mathsf{b} \leftarrow \mathsf{Read}(\mathsf{acc}, \mathsf{sk})$*: On input of the encrypted balance* acc *and the secret key* sk*,* Read *returns the balance* b*.*
- $\mathsf{tx} \leftarrow\!\!\$ \ \mathsf{Transact}(\mathsf{state}, \mathsf{sk}_i, \mathbf{B})$*: On input of valid sender secret key* sk_i *corresponding to some* pk_i*, a set of valid payload information* $\mathbf{B} = \{(\mathsf{pk}_j, \mathsf{pl}_j)\}_{j=1}^{t_R}$*, and state* state*,* Transact *produces a transaction message* tx*.*
- $\{0, 1\} \leftarrow \mathsf{Verify}(\mathsf{tx}, \mathsf{state})$*: On input of a transaction message* tx *and state* state*,* Verify *checks the consistency of the message and outputs* 1 *if consistent, and 0 otherwise.*

3.2. Security Definitions

Our security definitions will follow the game-based paradigm. We first look at the various oracles that the adversary has access to:

- $\mathcal{O}_{\mathsf{Corrupt}}(\cdot)$: On input of index i, it returns sk_i. Further, it adds i to a list $\mathcal{C}$ of corrupted users.
- $\mathcal{O}_{\mathsf{Transact}}(\mathsf{state}, \cdot, \cdot)$: It requires as input an index i, and a set of balances and public keys as indicated by $\mathbf{B}$. In response, the oracle computes $\mathsf{tx} \leftarrow\!\!\$ \ \mathsf{Transact}(\mathsf{state}, \mathsf{sk}_i, \mathbf{B})$, and invokes

$\mathcal{O}_{\mathsf{Insert}}(\mathsf{state}, \mathsf{tx})$. This models an honest creation and insertion of a transaction. Note that the receivers in $\mathbf{B}$ may involve parties that were generated by the adversary, and not by the challenger.

- $\mathcal{O}_{\mathsf{Insert}}(\mathsf{state}, \cdot)$: On input of a transaction tx, the system runs $\mathsf{Verify}(\mathsf{tx}, state)$ and upon successful verification, updates the state to get the updated state new-state. This models an adversarial transaction created and which is successfully inserted into the blockchain.

3.2.1. Overdraft-Safety Experiment Note that the $\mathsf{Transact}$ algorithm takes as input some secret key sk of the sender. Therefore, it requires that no non-corrupted user's (i.e., secret key unknown to the adversary) balance is reduced at the end of the experiment. It also follows that there is no non-corrupted party P_j in any transaction such that $\mathsf{pl}_j < 0$. Further, it also requires that the corrupted user's combined balance does not increase at the end of the experiment. We omit the formal definition due to space constraints and refer the readers to the work of Diamond [7]

3.2.2. Ledger-Indistingusishability Experiment As remarked earlier, we only define receiver anonymity in this experiment. The ledger-indistinguishability experiment $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}(\kappa)$ is modeled by a game where the adversary has to distinguish between two adversarially chosen transaction payloads, even with access to $\mathcal{O}_{\mathsf{Transact}}, \mathcal{O}_{\mathsf{Insert}}$. We omit the formal definition due to space constraints and refer the reader to the work of Diamond [7]

3.2.3. FUL Security Formally, we model in this game the semantic security of old transactions even if the receiver's current secret key is revealed. Note that again we allow the sender's epoch to be ahead of the receiver's epoch when the compromise occurs. In the FUL model, the adversary is allowed to update the key pairs for any receiver and provide two sets of payloads for adversarially chosen honest receivers. One of these payloads is randomly chosen, and an honest transaction is generated and inserted to update the keys of the receivers and their balances. The adversary receives encryptions corresponding to epoch i and also the keys at epoch $i+1$. The security experiment $\mathsf{FUL}_{\mathcal{A},\Pi}(\kappa)$ strengthens the $\mathcal{O}_{\mathsf{Insert}}$ oracle to update the keys of the receiver when the transaction is inserted into the blockchain. The state state is a dictionary that maps each user i to their current public key and encryption of their current balance under their current public key. This game implicitly models the stronger setting where every receiver's key is updated for every transaction.

1) Run $\mathsf{pp}, \mathsf{state}, \mathsf{txpool}, \mathsf{b}_{\mathsf{total}}, \mathsf{MAX} \leftarrow_{\$} \mathsf{Setup}(1^{\kappa})$.
2) Now, the game does the following:

   for $i = 1$ **to** n **do**

   Sample a random balance b_i

   $(\mathsf{sk}_i^{(0)}, \mathsf{pk}_i^{(0)}) \leftarrow_{\$} \mathsf{KeyGen}(\mathsf{pp})$

   $\mathsf{acc}_i \leftarrow_{\$} \mathsf{Encrypt}(\mathsf{pk}_i^{(0)}, \mathsf{b}_i)$

   Set $\mathsf{state}[i] = \mathsf{pk}_i^{(0)}, \mathsf{acc}_i$
3) $(i, \mathbf{B}_0^*, \mathbf{B}_1^*, \mathbf{R}) \leftarrow_{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Transact}}, \mathcal{O}_{\mathsf{Insert}}}(\mathsf{state}, \mathsf{pk}_1^{(0)}, \ldots, \mathsf{pk}_n^{(0)})$. Here $\mathbf{B}_0^*, \mathbf{B}_1^*$ are two arrays which correspond to the payload for each receiver in adversarially chosen $\mathbf{R}$ and $\mathsf{pk}_i \notin \mathbf{R}$. As before, we will require that the payload corresponding to corrupted users $j \in \mathbf{R}$ are the same in both sets of payloads.
4) Sample $\beta \leftarrow_{\$} \{0,1\}$. Run $\mathsf{tx}^* \leftarrow_{\$} \mathsf{Transact}(\mathsf{state}, \mathsf{sk}_i, \mathbf{B}_\beta)$ and $\mathcal{O}_{\mathsf{Insert}}(\mathsf{state}, \mathsf{tx}^*)$.
5) $\beta' \leftarrow_{\$} \mathcal{A}(\mathsf{tx}^*, \mathbf{SK})$ where $\mathbf{SK}$ contains the updated secret key (after tx^* was inserted) of every user in $\mathbf{R}$.
6) The game outputs 1 iff $\beta = \beta'$.

Definition 2. *A payment system* Π *is* Forward-secure Until Last update *if, for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr[\mathsf{FUL}_{\mathcal{A},\Pi}(\kappa) = 1] \leq \frac{1}{2} + \mathrm{negl}(\kappa)$$

4. Construction of FUL Zether

4.1. Zether, a Recap

As remarked before, Zether uses ElGamal encryption [17] as the underlying encryption scheme. Due to space constraints, we omit a detailed exposition on the syntax, security, and homomorphic properties of the encryption scheme. Let Alice (with key pair $(\mathsf{sk}_A, \mathsf{pk}_A)$) and Bob (with key pair $(\mathsf{sk}_B, \mathsf{pk}_B)$) be two users in the system with respective balances $\mathsf{b}_A, \mathsf{b}_B$. In other words, the state of the table looked similar to the following:

$$\mathsf{pk}_A \mapsto \left(\mathsf{oC}_A = g^x, \mathsf{oD}_A = \mathsf{pk}_A^x \cdot g^{\mathsf{b}_A}\right)$$
$$\mathsf{pk}_B \mapsto \left(\mathsf{oC}_B = g^{x'}, \mathsf{oD}_B = \mathsf{pk}_B^{x'} \cdot g^{\mathsf{b}_B}\right)$$

Say, Alice wishes to send Bob a payload of pl. Alice begins by encrypting $g^{-\mathsf{pl}}$ under its public key pk_A and then encrypts g^{pl} under Bob's public key pk_B. Formally, we have:

$$\left(C = g^r, D_0 = (\mathsf{pk}_A, \mathsf{pk}_A^r \cdot g^{-\mathsf{pl}}), D_1 = (\mathsf{pk}_B, \mathsf{pk}_B^r \cdot g^{\mathsf{pl}})\right).$$

Note that multiplying C with oC_A along with multiplying D_0 with oD_A yields a ciphertext pair: $(\mathsf{nC}_A = g^{r+x}, \mathsf{nD}_A = \mathsf{pk}_A^{r+x} \cdot g^{\mathsf{b}_A - \mathsf{pl}})$. Therefore, $(\mathsf{nC}_A, \mathsf{nD}_A)$ is a valid ElGamal ciphertext encrypting the balance of Alice after the transaction. Let $bal' := \mathsf{b}_A - \mathsf{pl}$ be this residual balance. However, a cheating user can try to do the following: spend tokens of others, create inconsistent ciphertext pairs, spend tokens it does not have, or try to steal tokens from others. This requires a zero-knowledge proof of honesty whereby Alice needs to prove:

- the knowledge of secret key sk_A for which $g^{\mathsf{sk}_A} = \mathsf{pk}_A$.
- the knowledge of randomness r such that $g^r = C$ (knowledge of randomness)
- the net-zero conservation of balance $(\mathsf{pk}_A \cdot \mathsf{pk}_B)^r = (D_0 \cdot D_1)$
- the well-formedness of sender-related ciphertext: $D_0 = g^{-\mathsf{pl}} \cdot C^{\mathsf{sk}_A}$ and $\mathsf{nD}_A = g^{bal'} \cdot \mathsf{nC}_A^{\mathsf{sk}_A}$

- the protection against overflow and overdraft: $\mathsf{pl}, bal' \in [0, \mathsf{MAX}]$

$$\mathcal{R}_Z : \left\{ \begin{array}{cc|c} (\mathsf{pk}_A, D_0, \mathsf{pk}_B, D_1, C) & : \mathsf{pl} & g^{\mathsf{sk}_0} = \mathsf{pk}_0 \wedge C^{\mathsf{sk}_A} \cdot g^{-\mathsf{pl}} = D_0 \\ (\mathsf{nC}_A, \mathsf{nD}_A) & bal' & \mathsf{nC}^{\mathsf{sk}_0} \cdot g^{bal'} = \mathsf{nD}_0 \\ & \mathsf{sk}_A & C = g^r \wedge, (\mathsf{pk}_A \cdot \mathsf{pk}_B)^r = (D_0 \cdot D_1) \\ & r & \mathsf{pl}, bal' \in \{0, \ldots, \mathsf{MAX}\} \end{array} \right\}$$

Their proof of zero knowledge involved the usage of Σ-Bullets to prove the statement - a combination of Σ protocol and bulletproofs. We refer the readers to the original work for a detailed construction of the zero-knowledge protocol.

Note that we do not focus on gases and fees that are to be paid by the sender to process the transaction sent to the smart contract. Prior works have shown that it is simple to augment the code to also support these functionalities.

4.2. Our Technical Tool: Updatable Public Key Encryption

As covered in the earlier sections, forward security is a desired feature. However, the existing constructions of FS-PKE are not compatible with the framework of using ElGamal Encryption. Therefore, we focus on sender-aided forward secrecy which is dubbed UPKE.

Definition 3. *An updatable public key encryption (UPKE) scheme is a set of six polynomial-time algorithms* $\mathsf{UPKE} = (\mathsf{U\text{-}PKEG}, \mathsf{U\text{-}Enc}, \mathsf{U\text{-}Dec}, \mathsf{Upd\text{-}Pk}, \mathsf{Upd\text{-}Sk})$ *with the following syntax:*

- ***Key generation****:* $\mathsf{U\text{-}PKEG}$ *takes as parameter* 1^κ *where* κ *is the security parameter and outputs a fresh secret key* sk_0 *and a fresh initial public key* pk_0.
- ***Encryption****:* $\mathsf{U\text{-}Enc}$ *receives a public key* pk *and a message* m *to produce a ciphertext* c.
- ***Decryption****:* $\mathsf{U\text{-}Dec}$ *receives a secret key* sk *and a ciphertext* c *to produce message* m.
- ***Update Public Key****:* $\mathsf{Upd\text{-}Pk}$ *receives a public key* pk *to produce an update ciphertext* up *and a new public key* pk'.
- ***Update Secret Key****:* $\mathsf{Upd\text{-}Sk}$ *receives an update ciphertext* up *and secret key* sk *to produce a new secret key* sk'.

4.2.1. UPKE IND-CR-CPA Security Game We present a simplified version of INDistinguishable under Chosen Randomness, Chosen Plaintext Attack (IND-CR-CPA) security game, adapted from the earlier definition of Dodis *et al.* [18].

Definition 4. *An updatable public key encryption scheme* $\mathsf{UPKE} = (\mathsf{U\text{-}PKEG}, \mathsf{U\text{-}Enc}, \mathsf{U\text{-}Dec}, \mathsf{Upd\text{-}Pk}, \mathsf{Upd\text{-}Sk})$ *is said to be IND-CR-CPA Secure if:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow_\$ \mathsf{U\text{-}PKEG}(1^\lambda); b \leftarrow_\$ \{0,1\} \\ (m_0, m_1, state, r_0, r_1, \ldots, r_{q-1}) \leftarrow_\$ \mathcal{A}(\mathsf{pk}_0) \\ (\mathsf{pk}_q, \mathsf{sk}_q) \leftarrow \mathsf{UpdateAll}(\mathsf{pk}_0, \mathsf{sk}_0, r_0, \ldots, r_{q-1}) \\ C \leftarrow_\$ \mathsf{U\text{-}Enc}(\mathsf{pk}_q, m_b); r^* \leftarrow_\$ \mathcal{R} \\ (\mathsf{up}^*, \mathsf{pk}^*) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_q; r^*) \\ \mathsf{sk}^* \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_q, \mathsf{up}^*) \; . \\ b' \leftarrow_\$ \mathcal{A}(C, \mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*, state) \end{array} \right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

where $\mathsf{UpdateAll}$ *is as defined below:*

$\mathsf{UpdateAll}(\mathsf{pk}_0, \mathsf{sk}_0, r_0, \ldots, r_{q-1})$
  for $i = 0$ **to** $q-1$ **do**
    $(\mathsf{up}_{i+1}, \mathsf{pk}_{i+1}) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_i; r_i)$
    $\mathsf{sk}_{i+1} \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_i, \mathsf{up}_{i+1})$

Construction 1 (UPKE Scheme UPKE)**.** For a group $\mathbb{G}$ of prime order q, with generator g, we define the following scheme:

- U-PKEG: Outputs, $\mathsf{sk} \leftarrow_\$ \mathbb{F}_q, \mathsf{pk} = g^{\mathsf{sk}}$
- U-Enc($\mathsf{pk}, m \in \mathbb{F}_q; r \in \mathbb{F}_q$): Outputs $(C = g^r, D = \mathsf{pk}^r \cdot g^m)$
- U-Dec($\mathsf{sk}, c = (C, D)$): Outputs m such that $g^m = D \cdot C^{-\mathsf{sk}}$
- Upd-Pk($\mathsf{pk}; r, \delta \in \mathbb{F}_q$): Outputs $\mathsf{pk}' = \mathsf{pk} \cdot g^\delta, \mathsf{up} = \mathsf{U\text{-}Enc}(\mathsf{pk}, g^\delta; r)$
- Upd-Sk(sk, up): Outputs $\mathsf{sk}' = \mathsf{sk} + \delta$ where $\delta = \mathsf{U\text{-}Dec}(\mathsf{sk}, \mathsf{up})$

4.2.2. Circular Secure + Leakage Resilience (CS+LR) Security Game

Definition 5. *A public key encryption scheme* $\mathcal{E}$ *with message space* $\mathcal{M}$*, ciphertext space* $\mathcal{C}$ *is said to be CS+LR secure) if:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{U\text{-}PKEG}(1^\lambda); \\ b \leftarrow_\$ \{0,1\} \\ (m_0, m_1) \leftarrow_\$ \mathcal{A}(\mathsf{pk}) \\ C \leftarrow_\$ \mathsf{U\text{-}Enc}(\mathsf{pk}, m_b) \\ C' \leftarrow_\$ \mathsf{U\text{-}Enc}(\mathsf{pk}, \mathsf{sk}) \\ \delta^* \leftarrow_\$ \mathcal{M}; \ell \leftarrow \mathsf{sk} + \delta^* \\ b' \leftarrow_\$ \mathcal{A}(C, C', \ell) \end{array} \right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

Theorem 1. *Construction 1 is CS+LR secure, under the DDH assumption.*

Proof. Let us look at the distribution D_0 that $\mathcal{A}$ receives when playing the original CS+LR Security game.

$$\mathsf{pk}, C = (g^r, \mathsf{pk}^r \cdot g^{m_b}), C' = (g^{r'}, \mathsf{pk}^{r'} \cdot g^{\mathsf{sk}}),$$
$$\ell = \mathsf{sk} + \delta^*; r, \delta^*, r' \leftarrow_\$ \mathbb{F}_q$$

The proof proceeds through a sequence of hybrids which is summarized below.

Hybrid	Hybrid Definition	Security
D_0	The Original CS+LR Security Game	Identical
D_1	D_0 except $C' = (g^{r'}, \mathsf{pk}^{r'+1})$	Identical
D_2	D_1 except except $C = g^r, g^{r \cdot \mathsf{sk}} \cdot g^{m_b}$	Identical
D_3	D_2 except each $\ell = \delta^* + \mathsf{sk}$ is replaced by $\delta \leftarrow_\$ \mathbb{F}_q$	DDH
D_4	D_3 except $g^{r \cdot \mathsf{sk}} \cdot g^{m_b}$ replaced by $U \leftarrow_\$ \mathbb{G}$	

□

Theorem 2. *Under the DDH Assumption,* UPKE *(presented as Construction 1) is IND-CR-CPA Secure.*

Proof. We will prove that if UPKE was a secure CS+LR construction, then it is also IND-CR-CPA secure. In other words, let there be an adversary $\mathcal{A}$ that breaks the IND-CR-CPA security of UPKE, then we will use it to construct

an adversary $\mathcal{B}$ that breaks CS+LR security. Due to space constraints, we briefly summarize how the reduction proceeds. The challenger of the CS+LR game first samples a key pair $\mathsf{pk}_0, \mathsf{sk}_0$, and pk_0 is given to $\mathcal{B}$. $\mathcal{B}$ runs $\mathcal{A}$ on pk_0. $\mathcal{A}$ responds with challenge messages $m_0, m_1, \{\delta_i\}_{i=0}^{q-1}$. $\mathcal{B}$ simply buffers and records every choice of δ_i and uses $\mathcal{A}$'s choice of m_0, m_1 as its own challenge message. In return, $\mathcal{B}$ receives first an encryption of m_b under pk_0. Call this (c_1, c_2). Meanwhile $\mathcal{A}$ expects an encryption of m_b under pk_q. However, ElGamal has the key-homomorphism feature whereby $\mathcal{B}$ computes $\Delta = \sum_{i=0}^{q-1} \delta_i$ and can generate the correct encryption under pk_q by simply outputting $(c_1, c_2 \cdot c_1^{\Delta})$ which is an encryption of m_b under pk_q.

It now sets $\mathsf{sk}^* = \ell + \Delta$ where $\ell = \mathsf{sk}_0 + \delta^*$ that $\mathcal{B}$ receives from CS+LR challenger. It can now compute $\mathsf{pk}^* = g^{\mathsf{sk}^*}$. The only thing remaining to be computed is for $\mathcal{B}$ to generate an encryption of δ^* under pk_q, without knowledge of δ^*. For this, $\mathcal{B}$ use the encryption of sk_0 under pk_0 that the challenger sends (call it c_1', c_2') and knowledge of $\ell = \mathsf{sk}_0 + \delta^*$, to can generate an encryption of δ^* under pk_0 by doing the following:

- First $(c_1'^{-1}, c_2'^{-1})$ is a valid encryption of $-\mathsf{sk}_0$ under pk_0. Call this $c_1'^*, c_2'^*$)
- Then multiplying $c_2'^*$ by g^{ℓ} we get a valid encryption of $\ell - \mathsf{sk}_0 = \delta^*$ under pk_0

Then, it can use the key homomorphism property defined before to generate an encryption under pk_q. It is easy to verify that $\mathcal{B}$ produces the correct distribution for $\mathcal{A}$ and therefore $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s. □

4.2.3. UPKE IND-CU-CPA Security Game We present a simplified version of INDistinguishable under Chosen Update, Chosen Plaintext Attack (IND-CU-CPA) security game, adapted from the earlier definition of Dodis *et al.* [18].

Definition 6. *An updatable public key encryption scheme* $\mathsf{UPKE} = (\mathsf{U\text{-}PKEG}, \mathsf{U\text{-}Enc}, \mathsf{U\text{-}Dec}, \mathsf{Verify}, \mathsf{Upd\text{-}Pk}, \mathsf{Upd\text{-}Sk})$ *is said to be IND-CU-CPA Secure if for all PPT-adversaries* $\mathcal{A}$*:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow_{\$} \mathsf{U\text{-}PKEG}(1^{\lambda}); b \leftarrow_{\$} \{0,1\} \\ (m_0, m_1, state, \{(\mathsf{pk}_i, \mathsf{up}_i)\}_{i=1}^{q}) \leftarrow_{\$} \mathcal{A}(\mathsf{pk}_0) \\ (\mathsf{pk}_q, \mathsf{sk}_q) \leftarrow \mathsf{UpdateAll}(\mathsf{pk}_0, \mathsf{sk}_0, \{(\mathsf{pk}_i, \mathsf{up}_i)\}_{i=1}^{q}) \\ C \leftarrow_{\$} \mathsf{U\text{-}Enc}(\mathsf{pk}_q, m_b); r^* \leftarrow_{\$} \mathcal{R} \\ (\mathsf{up}^*, \mathsf{pk}^*) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_q; r^*) \\ \mathsf{sk}^* \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_q, \mathsf{up}^*)\ . \\ b' \leftarrow_{\$} \mathcal{A}(C, \mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*, state) \end{array} \right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

where UpdateAll *is as defined below:*

$\mathsf{UpdateAll}(\mathsf{pk}_0, \mathsf{sk}_0, \{(\mathsf{pk}_i, \mathsf{up}_i)\}_{i=1}^{q})$
  for $i = 1$ **to** q **do**
    if $\mathsf{Verify\text{-}Upd}(\mathsf{pk}_{i-1}, \mathsf{up}_i, \mathsf{pk}_i) == 1$ **then**
      $(\mathsf{up}_{i+1}, \mathsf{pk}_{i+1}) \leftarrow \mathsf{Upd\text{-}Pk}(\mathsf{pk}_i; r_i)$
      $\mathsf{sk}_i \leftarrow \mathsf{Upd\text{-}Sk}(\mathsf{sk}_{i-1}, \mathsf{up}_i)$
    else ***ABORT***

Remark 1. For succinct notating, our definitions of IND-CU-CPA and IND-CR-CPA is written out as non-adaptive adversary. It is easy to extend the same to allow for adaptive queries. The work by Dodis *et al.* [18] presents the more general adaptive security. Further, we will use the adaptive definition in our proofs.

We first begin by presenting a NIZK which will be employed in our UPKE scheme. This can be done with the help of a simple Σ-protocol.

Construction 2 (NIZK for $\mathcal{R}_{ElG}$)**.** The following Non-Interactive Proof of Knowledge $\mathsf{nizk}_{Elg} = \mathsf{Setup}_{Elg}, \mathsf{Prove}_{Elg}, \mathsf{Verify}_{ElG}$ is defined for the following relation:

$$\mathcal{R}_{ElG} = \left\{ \mathsf{pk}, \mathsf{pk}', \mathsf{up}; \delta, r' \;\middle|\; \mathsf{pk}' = \mathsf{pk} \cdot g^{\delta} \wedge \mathsf{up} = (g^{r'}, \mathsf{pk}^{r'} \cdot g^{\delta}) \right\}$$

- $\mathsf{Setup}_{ElG}(1^{\kappa})$: Chooses an appropriate cyclic group $\mathbb{G}$ of prime order q with generators g, h and sets $\mathsf{pp} = (\mathbb{G}, q, g)$. It also sets hash function $H : \{0,1\}^* \to \mathbb{Z}_q$
- $\mathsf{Prove}_{ElG}(\mathsf{pp}, (\mathsf{pk}, \mathsf{pk}', \mathsf{up}), (\delta, r'))$: Outputs the following:
  - Sample k_{δ}, k_r
  - Compute: $A_r = \mathsf{pk}^{k_r}, A_{\delta} = g^{k_{\delta}}$
  - Compute: $c = H(A_r, A_{\delta})$
  - Compute $s_{\delta} = k_{\delta} + c \cdot \delta, s_r = k_r + c \cdot r'$
  - Output $\pi = (A_r, A_{\delta}, s_{\delta}, s_r)$
- $\mathsf{Verify}_{ElG}(\mathsf{pp}, \pi = (A_r, A_{\delta}, s_{\delta}, s_r), (\mathsf{pk}, \mathsf{pk}', \mathsf{up}))$
  - Parse $\mathsf{up} = (c_1, c_2)$
  - Compute: $c = H(A_r, A_{\delta})$
  - Check if: $g^{s_r} \stackrel{?}{=} A_r \cdot c_1^{c}$
  - Check if: $A_{\delta} \cdot A_r \cdot g^{-s_{\delta}} \cdot \mathsf{pk}^{-s_r} \stackrel{?}{=} (c_2)^{-c}$
  - Check if: $A_{\delta} \cdot g^{-s_{\delta}} \stackrel{?}{=} (\mathsf{pk}'/\mathsf{pk})^{-c}$

Theorem 3. *The above satisfies perfect completeness, perfect honest verifier zero-knowledge, and statistical witness-extended emulation for extracting a valid witness x. This extractor $\mathcal{E}_{ElG}$ runs the prover with four different values of the hash function to generate four different challenges c where H is modeled as a random oracle.*

We omit the proof due to space constraints.

Construction 3 (UPKE Scheme UPKE')**.** For a group $\mathbb{G}$ of prime order q, with generator g, we define the following scheme:

- U-PKEG: Outputs, $\mathsf{sk} \leftarrow_{\$} \mathbb{F}_q, \mathsf{pk} = g^{\mathsf{sk}}$
- $\mathsf{U\text{-}Enc}(\mathsf{pk}, m \in \mathbb{F}_q; r \in \mathbb{F}_q)$: Outputs $(C = g^r, D = \mathsf{pk}^r \cdot g^m)$
- $\mathsf{U\text{-}Dec}(\mathsf{sk}, c = (C, D))$: Outputs m such that $g^m = D \cdot C^{-\mathsf{sk}}$
- $\mathsf{Upd\text{-}Pk}(\mathsf{pk}; r, \delta \in \mathbb{F}_q)$: Outputs
- $\mathsf{Upd\text{-}Sk}(\mathsf{sk}, \mathsf{up}')$:
  - Parse $\mathsf{up}' = (\mathsf{up}, \pi)$
  - **if** $\mathsf{Verify}_{ElG}(\pi, (\mathsf{pk}, \mathsf{pk}', \mathsf{up})) = 1$, then outputs $\mathsf{sk}' = \mathsf{sk} + \delta$ where $\delta = \mathsf{U\text{-}Dec}(\mathsf{sk}, \mathsf{up})$

Theorem 4. *If* UPKE *is IND-CR-CPA secure, then* UPKE' *is IND-CU-CPA secure.*

4.3. Achieving FUL Zether

We require that Alice (a sender) does the following additional steps:

- Sample a random $\delta \in \mathbb{F}_q$.
- Then compute an update ciphertext that would encrypt this δ. Looking ahead, this δ would be updating the secret key of Bob (receiver) from sk_B to $\mathsf{sk}_B + \delta$. $\mathsf{up} = (g^{r'}, \mathsf{pk}_B^{r'} \cdot g^{\delta})$
- Update the public key of Bob to $\mathsf{pk}'_B = \mathsf{pk}_B \cdot g^{\delta}$. We will use Σ-proof to prove that this update is correct, corresponding to secret δ.
- Also provide the offset of $E = \mathsf{nC}_B^{\delta}$ where $\mathsf{nC}_B = C \cdot \mathsf{oC}_B$, with knowledge of the secret δ, relying again on Σ-proof to prove that this offset is correct. The purpose of this offset is to evolve the updated balance of Bob to the new epoch, corresponding to the public key pk'_B.

The posted message looks as follows:

$$\left(C = g^r, D_0, D_1, \mathsf{up}, \mathsf{pk}'_B, E\right).$$

The additional zero-knowledge proof involves:

- knowledge of r' and δ such that $\mathsf{up} = (g^{r'}, \mathsf{pk}_B^{r'} \cdot g^{\delta})$
- knowledge of δ such that $E = (\mathsf{nC}_B)^{\delta}$
- knowledge of δ such that $\mathsf{pk}'_B = \mathsf{pk}_B \cdot g^{\delta}$

$$\mathcal{R}_{\mathsf{FUL}\text{-}Z} : \left\{ \begin{array}{ccc|c} (\mathsf{pk}_A, D_0, \mathsf{pk}_B, D_1, C) & : \mathsf{pl} & & g^{\mathsf{sk}_0} = \mathsf{pk}_0 \wedge C^{\mathsf{sk}_A} \cdot g^{-\mathsf{pl}} = D_0 \\ (\mathsf{nC}_A, \mathsf{nD}_A) & bal' & & \mathsf{nC}^{\mathsf{sk}_0} \cdot g^{bal'} = \mathsf{nD}_0 \\ (\mathsf{nC}_B, \mathsf{up}, \mathsf{pk}'_B, E) & \mathsf{sk}_A & & C = g^r \wedge, (\mathsf{pk}_A \cdot \mathsf{pk}_B)^r = (D_0 \cdot D_1) \\ & r & & \mathsf{pl}, bal' \in \{0, \ldots, \mathsf{MAX}\} \\ & r' & & \mathsf{up} = (g^{r'}, \mathsf{pk}_B^{r'} \cdot g^{\delta}) \\ & \delta & & \mathsf{pk}'_B = \mathsf{pk}_B \cdot g^{\delta} \wedge E = \mathsf{nC}_B^{\delta} \end{array} \right\}$$

We present the prover-verifier interaction to prove the relation $\mathcal{R}_{\mathsf{FUL}\text{-}Z}$ in Figure 1. Due to space constraints, we only discuss the additional proving and verification steps needed. We refer the readers to the Σ-protocol for the original Zether in their paper [6, §G]. Specifically, Figure 1 only contains the Σ-protocol for the new terms of the update ciphertext up, the updated public key pk'_B, and the offset E. This is to be interleaved with the steps from the original proof protocol from [6] which proves the well-formedness of C, D_0, D_1 with the knowledge of sk_A and the necessary range proofs.

Theorem 5 (Forward-secure Until Last update)**.** *Under DDH Assumption, the construction of FUL-Zether (as described in Section 4.3) is FUL-secure.*

We refer the readers to the full version of this paper [25].

Remark 2. The above construction fundamentally employs the UPKE Construction based on ElGamal Encryption. This scheme can thus leverage the graph-based fast-forwarding paradigm of Dodis *et al.* [19] that helps a receiver who is stuck in epoch i to evolve the key to epoch j in time $\log(j - i)$. This is faster than the naive approach of evolving key sequentially. They build this primitive from any homomorphic UPKE and it can be shown that indeed the ElGamal construction is homomorphic UPKE. Furthermore, the ElGamal encryption scheme is the most efficient construction of UPKE with a Fast-Forwarding feature, besting

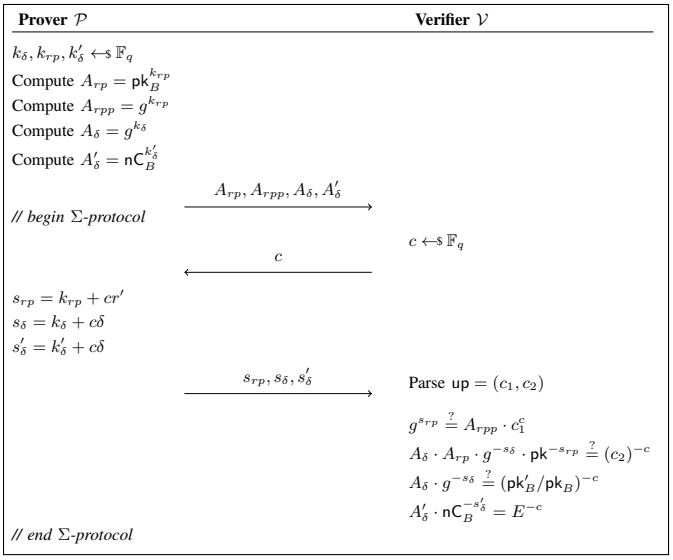

Figure 1. Additional Steps in Interactive Proof Protocol for $\mathcal{R}_{\mathsf{FUL}\text{-}Z}$.

even the constructions laid out in [19]. Despite the benefits, it is important to note that a recipient must take a discrete log to retrieve the δ; however, if they only wish to check their balance infrequently, this step is omitted and only undertaken when necessary.

To avoid the discrete log, one can potentially use the hashed ElGamal encryption which is also proven to be IND-CR-CPA secure by Jost *et al.* [10]. However, there is a critical issue for our purposes where all posted transactions need to have proof of well-formedness. Unfortunately, encrypting the offset δ as $(g^r, H(\mathsf{pk}^r) \oplus \delta)$ does not have an efficient proof of well-formedness. The closest one can achieve is to use ZKBoo [26] to prove the well-formedness of the ciphertext, which is computationally expensive.

5. PRIvate DEcentralized Confidential Transactions

5.1. Anonymous Zether

We begin by first taking a brief look at Anonymous Zether [7]. This will serve as a useful launchpad for our construction of PriDe CT. At its core, Anonymous Zether uses the idea of anonymity sets whereby the sender chooses a set of public keys pk_i from $i = 0$ to $t - 1$, which included the sender at index sen and the receiver at index rec. These indices were secret. Every receiver at index $i \neq \mathsf{rec}, \mathsf{sen}$ would receive a zero payload, i.e., their encryptions would merely serve as decoys. If the underlying encryption was secure, then an adversary, without decrypting, would be unable to determine whether the payload was 0, the positive payload pl (corresponding to rec), or the negative payload $-\mathsf{pl}$ (corresponding to sen) by observing the ciphertexts. The sender then had to prove:

- the correctness of the indices: $\mathsf{sen}, \mathsf{rec} \in \{0, \ldots, t-1\}$

- the knowledge of the secret key sk such that: $g^{\mathsf{sk}} = \mathsf{pk}_{\mathsf{sen}}$
- the knowledge of randomness r such that: $g^r = C$
- the conservation of balances: $(\mathsf{pk}_{\mathsf{sen}} \cdot \mathsf{pk}_{\mathsf{rec}})^r = D_{\mathsf{sen}} \cdot D_{\mathsf{rec}}$
- the well-formedness of the remaining ciphertexts: $\forall i \in \{0, \ldots, t-1\} \setminus \{\mathsf{sen}, \mathsf{rec}\}\ \mathsf{pk}_i^r = D_i$
- the well-formedness of the sender-related ciphertexts: $D_{\mathsf{sen}} = C^{\mathsf{sk}} \cdot g^{-\mathsf{pl}},\ \mathsf{nD}_{\mathsf{sen}} = \mathsf{nC}_{\mathsf{sen}}^{\mathsf{sk}} \cdot g^{bal'}$
- the protection against overflow and overdraft: $bal' \in [0, \mathsf{MAX}] \wedge \mathsf{pl} \in [0, \mathsf{MAX}]$

In addition, there were additional requirements of the values of sen and rec being of opposite parity along with the notion of epoch key. For simplicity, we do not discuss these aspects in our brief survey of this work. We refer the reader to the work of Diamond [7] for a detailed discussion.

Asymptotic Performance of Anonymous Zether. Despite the asymptotic behavior of many-out-of-many proofs, careful observations and optimizations (such as the observation that only 2 out of the N ciphertexts encrypt non-zero while the remaining encrypted zero) allowed Anonymous Zether to achieve $O(N \log N)$ proving time and $O(N \log N)$ verification time with the proof size being $O(\log N)$ and transaction size being $O(N)$ where N is the size of the receiver set. The proving time $O(N \log N)$ is achieved through a careful optimization technique discussed in detail in the original work. This corresponds to one sender sending a single transaction to a receiver. To achieve t such transactions, we incur a multiplicative factor of t using Anonymous Zether [7] where we run the protocol t different times as Anonymous Zether does not support batching natively.

5.2. Construction of PriDe CT

Sender S_i with secret key $(\mathsf{sk}_0, \mathsf{pk}_0)$ is sending $\mathsf{pl}_1, \ldots, \mathsf{pl}_t$ to receivers $R_1, \ldots, R_t$ with public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_t$. It also means that $\mathsf{pl}_0 = \sum_{j=1}^{t} \mathsf{pl}_j$ is debited from the account of the sender S. We need to ensure that $\mathsf{pl}_0, \mathsf{pl}_1, \ldots, \mathsf{pl}_t$ are all positive. Further, let bal' be the balance left in the sender's account, after this transaction. We will also need $bal' \geq 0$. The transaction is posted as a set of ciphertexts defined as: $C = g^r$, $D_j = \mathsf{pk}_j^r \cdot g^{\mathsf{pl}_j}$ for $j = 1, \ldots, t$, and $D_0 = \mathsf{pk}_0^r \cdot g^{-\mathsf{pl}_0}$. Further, let nC_0 and nD_0 be the result of the homomorphic addition of the account balance of sender S_i, as a result of this transaction. The statement that the sender needs to prove is the following:

- Validity of Sender: The sender needs to establish its validity by proving knowledge of some secret, i.e., it proves the knowledge of its secret key sk_0 that corresponds with the public key pk_0 for sender S_i.
- Validity of Encryptions: The sender needs to prove that (a) it is the party that has generated the ciphertext by proving the knowledge of the randomness which is being used in the ciphertext, and (b) that there is a conservation of payloads, i.e., the sum total of all payload is 0.In other words, it proves the knowledge of r such that: 1) $C = g^r$, and 2) balance of Transactions Holds, i.e., money debited is also money credited: $\prod_{j=0}^{t} D_j = \left(\mathsf{pk}_i \cdot \prod_{j=1}^{t} \mathsf{pk}_j\right)^r$.
- Validity of Payload: The sender needs to prove that it has not spent more than it has in its account and also specifically prove that it has not maliciously tried to receive money. These are the proof statements: $bal', \mathsf{pl}_0, \mathsf{pl}_1, \ldots, \mathsf{pl}_t \in \{0, \ldots, \mathsf{MAX}\}$.
- Well-formedness of ciphertext: 1) $D_0 = C^{\mathsf{sk}_0} \cdot g^{-\mathsf{pl}_0}$, 2) for $\mathsf{nC}_0, \mathsf{nD}_0$ being the result of homomorphic addition of S_i's balance we have: $\mathsf{nD}_0 = \mathsf{nC}_0^{\mathsf{sk}_0} \cdot g^{bal'}$,[1] and 3) the ciphertext is correctly formed, i.e., that $D_j = (\mathsf{pk}_j)^r \cdot g^{\mathsf{pl}_j}$ for $j = 1, \ldots, t$. This is similar to the validity of the encryptions step.

TABLE 1. GLOSSARY OF VARIABLES. THE VARIABLES THAT REMAIN SECRET ARE INDICATED BY THE SYMBOL ♣.

sk	Secret key of the sender ♣
pk_0	Public Key of the sender
$\mathsf{pk}_1, \ldots, \mathsf{pk}_t$	Public Keys of the receivers
pl_0	Total money sent by the sender ♣
bal'	Residual balance of sender. We set $\mathsf{pl}_{t+1} = bal'$ ♣
$\mathsf{pl}_1, \ldots, \mathsf{pl}_t$	Amount sent to receivers with public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_t$ ♣
r	randomness used for ElGamal encryption ♣
C	The g^r term of the ElGamal Ciphertext
D_0	Encryption of $-\mathsf{pl}_0$ under public key pk_0
$D_1, \ldots, D_t$	Encryption of pl_j under public key pk_j for $j = 1, \ldots, t$
$(\mathsf{nC}_0, \mathsf{nD}_0)$	Valid ElGamal encryption of pl_{t+1} Obtained by homomorphic multiplication.

Formally, we can combine the conditions, for each sender S_i, to the following relation:

$$\mathcal{R}_{\mathsf{PriDe\ CT}} : \left\{ \begin{array}{cc|c} (\mathsf{pk}_j, D_j)_{j=0}^t & (\mathsf{pl}_j)_{j=0}^t & g^{\mathsf{sk}_0} = \mathsf{pk}_0 \wedge C^{\mathsf{sk}_i} \cdot g^{-\mathsf{pl}_0} = D_0 \wedge \\ C, (\mathsf{nC}_0, \mathsf{nD}_0) & bal', \mathsf{sk}_0, r & \mathsf{nC}_0^{\mathsf{sk}_0} \cdot g^{bal'} = \mathsf{nD}_0 \\ & & \left(D_j = (\mathsf{pk}_j)^r \cdot g^{\mathsf{pl}_j}\right)_{j=1}^t \wedge C = g^r \wedge \\ & & \prod_{j=0}^t D_j = \left(\prod_{j=0}^t \mathsf{pk}_j\right)^r \\ & & \mathsf{pl}_0, \ldots \mathsf{pl}_t, bal' \in \{0, \ldots, \mathsf{MAX}\} \end{array} \right\}$$

5.2.1. Interactive Proof Protocol We present a zero-knowledge proof protocol for the relation $\mathcal{R}_{\mathsf{PriDe\ CT}}$, which is a modification of the protocol proposed by Diamond [7]. This protocol is based on the Σ-bullets proof protocol, which combines Bulletproofs and Σ-protocols. Our protocol is simpler and more concise compared to Diamond's protocol, thanks to the clear distinction between sender and receiver indices and our support for batching of transactions.

The protocol relies on the proof techniques introduced by Bünz *et al.* [27] to adapt bulletproofs for range proofs. Before diving into the proof protocol, we establish the public parameters, which are summarized in Table1 for reference. The proof protocol begins as presented in Figure 2.

[1]This check helps in ensuring that there is no overdraft as bal' needs to necessarily be greater than or equal to 0 and it also needs to know what bal' is.

Prover Input: $\left(C, \mathsf{nC}_0, D'_0, \{\mathsf{pk}_j, D_j\}_{j=0}^{t}; \{\mathsf{pl}_j\}_{j=0}^{t}, \mathsf{pl}_{t+1} = bal', \mathsf{sk}, r\right)$
Verifier Input: $\left(C, \mathsf{nC}_0, D'_0\{\mathsf{pk}_j, D_j\}_{j=0}^{t}\right)$

$\mathcal{P}$:
1: $\alpha, \rho \leftarrow\$ \mathbb{F}_q$
2: Compute $\mathbf{a}_L \in \{0,1\}^{t'n}$ such that:
3: **for** $j = 1, \ldots, t'$
4: $\left\langle \mathbf{2}^n, \mathbf{a}_{L[(j-1)\cdot n : jn-1]} \right\rangle = \mathsf{pl}_{j-1}$
5: $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{t'n} \in \mathbb{F}_q^{t'n}$
6: $A = h^{\alpha} \cdot \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$
7: $\mathbf{s}_L, \mathbf{s}_R \leftarrow\$ \mathbb{F}_q^{t'n}$
8: $S = h^{\rho} \cdot \mathbf{g}^{\mathbf{s}_L} \cdot \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$
$\mathcal{P} \rightarrow \mathcal{V}$: A, S
$\mathcal{V}$:
9: $y, z \leftarrow\$ \mathbb{F}_q$
$\mathcal{V} \rightarrow \mathcal{P}$: y, z
$\mathcal{P}$:
10: $l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^{t'n}) + \mathbf{s}_L \cdot X \in \mathbb{F}_q^{t'n}[X]$
11: $r(X) = (\mathbf{y}^{t'\cdot n} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{t'n} + \mathbf{s}_R \cdot X)$
$+ \sum_{j=1}^{t'} z^{1+j} \cdot (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(t'-j)\cdot n}) \in \mathbb{F}_q^{t'n}[X]$
12: $t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 \cdot X^2 \in \mathbb{F}_q[X]$
13: $\tau_1, \tau_2 \leftarrow\$ \mathbb{F}_q$
14: $T_i = g^{t_i} \cdot h^{\tau_i} \in \mathbb{G}$ for $i = 1, 2$
$\mathcal{P} \rightarrow \mathcal{V}$: T_1, T_2
$\mathcal{V}$:
15: $x \leftarrow\$ \mathbb{F}_q$
$\mathcal{V} \rightarrow \mathcal{P}$: x
$\mathcal{P}$:
16: $\mathbf{l} = l(x) = (\mathbf{a}_L - z \cdot \mathbf{1}^{t'n}) + \mathbf{s}_L \cdot x \in \mathbb{F}_q^{t'n}$
17: $\mathbf{r} = r(x) = (\mathbf{y}^{t'\cdot n} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{t'n} + \mathbf{s}_R \cdot x)$
$+ \sum_{j=1}^{t'} z^{1+j} \cdot (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(t'-j)\cdot n}) \in \mathbb{F}_q^{t'n}$
18: $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{F}_q$
19: $\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2$
20: $\mu = \alpha + \rho \cdot x \in \mathbb{F}_q$
21: $k_{\mathsf{sk}}, k_r, k_\tau, k_b \leftarrow\$ \mathbb{F}_q$
22: $A_C = g^{k_r}$
23: $A_y = g^{k_{\mathsf{sk}}}$
24: $A_b = g^{k_b} \cdot (C^{-z^2} \cdot \mathsf{nC}_0^{z^{t'+1}})^{k_{\mathsf{sk}}} \cdot \prod_{j=1}^{t} (\mathsf{pk}_j)^{k_r \cdot z^{2+j}}$
25: $A_X = \left(\prod_{j=0}^{t} \mathsf{pk}_j\right)^{k_r}$
26: $A_\tau = g^{-k_b} \cdot h^{k_\tau}$
$\mathcal{P} \rightarrow \mathcal{V}$: $\hat{t}, \mu, A_y, A_c, A_b, A_X, A_t$
// begin Σ-protocol
$\mathcal{V}$:
27: $c \leftarrow\$ \mathbb{F}_q$
$\mathcal{V} \rightarrow \mathcal{P}$: c
$\mathcal{P}$:
28: $s_{\mathsf{sk}} = k_{\mathsf{sk}} + c\mathsf{sk}$
29: $s_r = k_r + cr$
30: $s_b = k_b + c \sum_{j=1}^{t'} z^{1+j} \mathsf{pl}_{j-1}$
31: $s_\tau = k_\tau + c\tau_x$
$\mathcal{P} \rightarrow \mathcal{V}$: $s_{\mathsf{sk}}, s_r, s_b, s_\tau$
// end Σ-protocol

Figure 2. Interactive Proof Protocol for the relation $\mathcal{R}_{\mathsf{PriDe\ CT}}$

Then, the verifier verifies the Σ-proofs of knowledge:

$$32: A_y \stackrel{?}{=} g^{s_{\mathsf{sk}}} \cdot \mathsf{pk}^{-c}$$

$$33: A_C \stackrel{?}{=} g^{s_r} \cdot C^{-c}$$

$$34: \frac{A_b}{g^{s_b}} \stackrel{?}{=} \frac{\left(C^{-z^2} \cdot \mathsf{nC}_0^{z^{t'+1}}\right)^{s_{\mathsf{sk}}}}{\left(D_0^{-z^2} \cdot D_0'^{z^{t'+1}}\right)^{c}} \cdot \prod_{j=1}^{t} \left(\frac{\mathsf{pk}_j^{s_r}}{D_j^c}\right)^{z^{j+2}}$$

$$35: A_X \stackrel{?}{=} \left(\prod_{j=0}^{t} \mathsf{pk}_j\right)^{s_r} \cdot \left(\prod_{j=0}^{t} D_j\right)^{-c}$$

$$36: \delta(y,z) = (z - z^2)\left\langle \mathbf{1}^{t'\cdot n}, \mathbf{y}^{t'\cdot n} \right\rangle - \sum_{j=1}^{t'} z^{j+2} \left\langle \mathbf{1}^n, \mathbf{2}^n \right\rangle$$

$$37: g^{c\cdot\hat{t}} \cdot h^{s_\tau} \stackrel{?}{=} g^{c\cdot\delta(y,z)} \cdot g^{s_b} \cdot A_\tau \cdot (T_1^x \cdot T_2^{x^2})^c$$

Then, $\mathcal{P}$ and $\mathcal{V}$ engage in protocol 1 of [27] on inputs $(\mathbf{g}, \mathbf{h}', Ph^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$ in which $\mathbf{h}' = \mathbf{h}^{\mathbf{y}^{-t'n}} = \left(h_0, h_1^{y^{-1}}, h_2^{y^{-2}}, \ldots, h_{t'\cdot n-1}^{y^{-t'n+1}}\right)$ and $P = A \cdot S^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h}'^{z\cdot\mathbf{y}^{t'n}} \cdot \prod_{j=1}^{t'} \mathbf{h}'^{z^{j+1}\cdot\mathbf{2}^n}_{[(j-1)\cdot n : j\cdot n-1]}$.

Theorem 6. *The argument of knowledge, as defined in Section 5.2.1, is a secure argument of zero-knowledge which has perfect completeness, perfect honest verifier zero-knowledge, and computational witness extended emulation. The extractor $\mathcal{E}_{\mathsf{ANON}}$ runs the prover with $t' \cdot n$ different values of y, $t'+2$ different values of z, 3 different values of x, and 2 different values of the challenge c (in addition to the extractor $\mathcal{E}_{\mathsf{IP}}$) to extract valid witnesses $\mathsf{sk}, \mathbf{b} = (\mathsf{pl}_0, \ldots, \mathsf{pl}_t, bal'), r$ or a non-trivial discrete log relation between independently chosen generators.*

We omit the proof due to space constraints.

Theorem 7. *If the discrete logarithm problem is hard with respect to group $\mathbb{G}$, then PriDe CT is overdraft-safe.*

Theorem 8. *Under the DDH Assumption, PriDe CT is ledger-indistinguishable.*

We refer the readers to the full version of this paper [25].

6. Towards FUL-secure PriDe CT

In this section, we will present our construction of PriDeFUL CT. The approach we take is similar to the approach of "bootstrapping" Zether with an updatable public key encryption scheme to achieve Forward-secure Until the Last update construction. As discussed in the introduction, there exist some inherent concurrency-related issues about PriDeFUL CT that are not necessarily pertinent to PriDe CT. Specifically, how one handles the setting where multiple senders are transacting with a receiver in that epoch. While they are capable of proving consistency with respect to a particular state of the receiver and the contract can multiply all of them together to create the new state, combining the evolution of keys with transacting concurrently is challenging. Our solution for PriDeFUL CT is to

simply apply the solution for FUL-Zether to PriDe CT. In FUL-Zether, the sender chose the update for the receiver and provided the necessary information to (a) help the receiver update the secret key, and (b) the contract to evolve the ciphertext to be consistent with the updated public key. For PriDeFUL CT we provide two options: (1) either concurrent transactions are performed with the update ignored, or (2) a transaction and a corresponding update to that receiver is performed, ignoring other concurrent transactions to that receiver.

Our construction of PriDeFUL CT achieves FUL-security, allowing the sender to post a single message (either for one or all of the receivers) by including some additional messages (similar to the one for FUL-Zether from Section 4) in their standard communication. The result is mathematically correct and enjoys the same benefits of forward security as obtained from FUL-Zether. Unfortunately, this comes with an inherent limitation of concurrent transactions not being supported any more.

7. Experimental Results for PriDe CT

We implement PriDe CT as an Ethereum smart contract based on the open-source code of Anonymous Zether [7] at https://github.com/benediamond/anonymous-zether. Same as Anonymous Zether, the client-end wallet is implemented with JavaScript and Node.js module and the verifier smart contract is implemented with Solidity. Our implementation interoperates with ERC-20-compliant token contracts, can be deployed on any blockchain platform supporting Ethereum-style smart contracts, and can interact with other smart contracts to provide anonymity to a variety of services. With the "register" method, a user can open a new escrow account in the contract. Then with "deposit" and "withdraw" methods, PriDe CT allows users to transfer ERC-20 tokens into and out of the escrow account. With the account funded, a user can perform transactions with multiple registered recipients with the "transfer" method which generates the statement and the proof, sends them to the contract, and updates the local status on receiving confirmation of the transaction succeeding.

We use the Ganache/Truffle suite which is widely used in smart contract development and testing to measure the performance of our scheme and compare it with Anonymous Zether. The smart contracts are compiled and deployed with Truffle on a local Ethereum chain simulated with Ganache. We measure the performance of PriDe CT and Anonymous Zether in the same environment, for the same security level, and report the performance of both protocols for $N = 4$ to $N = 64$ as reported in [7]. We compare the proving time (i.e., the total computation time of the prover, including time taken to generate the statement) on the client side and the gas consumption of verification on the smart contract side in a transfer transaction in Table 2. Our experiments, much like those reported in Anonymous Zether [7], assumes that a posted transaction is updated instantaneously, in an effort to abstract away the consensus layer. The proving time is measured on a standard MacBook Pro with a 10-core Apple M1 Pro chip and 16GB memory. The gas consumption reveals the computation cost of the verification process in smart contracts and also how expensive a transaction is. We omit the real-world time measurement of the verification time as Solidity language does not provide a functionality to measure the real-world execution time due to the on-chain nature of the computation and gas consumption is a better measurement of the computation cost.

In Table 2, we compare the running time of PriDe CT and Anonymous Zether. The performance is parametrized by N and we let N be a power of two for ease of implementing the inner product verifier of Bulletproofs. Here, N is the ring of public keys posted as a part of the transaction message. In the case of Anonymous Zether, this consists of $N-1$ public keys distinct from the sender's public key. Meanwhile, in PriDe CT, there are $N-2$ public keys distinct from the sender's public key. Further, let t be the number of intended recipients. In Anonymous Zether, $t = 1$ always and in PriDe CT $t \leq N-2$. Further, in Table 2, we compute a ratio of proving times as follows:

$$\frac{\text{Proving Time for PriDe CT}}{\text{Proving Time for Anonymous Zether}}.$$

Similarly, we compute the ratio for the gas consumption. The table shows us that for a ring size of $N = 32$ if $t \geq 7$ then PriDe CT performs much better than naively running Anonymous Zether t-times.

While Anonymous Zether is the only construction in the account-based model, we also present comparisons of the transaction size in the UTXO-based payment system known as Quisquis [5]. Much like Anonymous Zether, we are unable to offer an implementation-based comparison with Quisquis due to the vast differences in programming languages and the underlying infrastructure. We instead offer comparisons with Quisquis, based on the sizes of the transaction, expressed as a function of N, in terms of the number of Group Field Elements. We also compare with Anonymous Zether on the same parameters and note that we are smaller than even a single Anonymous Zether transaction (even ignoring our benefits from batching) for the same N. This is presented in Table 4.

Performance of PriDeFUL CT. For completeness, we study the performance of PriDeFUL CT by identifying the overhead of a sender in order to evolve the key per receiver to achieve FUL security. Our experiments show that a sender needs to spend an additional 166ms to generate the additional terms and prove that the key evolution is correct (i.e., that the update ciphertext and the new public key is evolved consistently with the same offset - Prover steps in Figure 1) and the verifier consumes an additional 82,147 units of gas (Verifier steps in Figure 1), for a sender to evolve the key of a single receiver. These numbers are per receiver and thus *independent* of the size of the anonymity set. For the case where a sender updates X keys for X receivers at the same time then the cost and the gas incurs a multiplicative factor of X.

In Table 5, we compare the various existing solutions, with our work, along an assortment of properties.

TABLE 2. PERFORMANCE COMPARISON BETWEEN PRIDE CT AND PRIDEFUL CT. GAS COSTS OF VERIFICATION ARE COMPUTED USING THE LATEST STANDARD PRICES OF GAS AT THE TIME OF WRITING, WHICH WAS 41 GWEI, SOURCED FROM [28] AT UTC 04:25 PM 6 NOVEMBER 2023. NOTE THAT THE PRICES ARE SUBJECT TO MARKET FLUCTUATIONS, BUT THE GAS COSTS TEND TO BE STABLE AND MORE MEANINGFUL.

	Proving Time			Gas Consumption				
	Anonymous Zether	PriDe CT	Ratio	Anonymous Zether		PriDe CT		Ratio
				Gas Cost	Ether	Gas Cost	Ether	
Transfer(4)	1,897	3,543	1.87	3,453,438	0.142	3,812,298	0.156	1.10
Transfer(8)	2,066	6,757	3.27	4,332,444	0.178	8,106,123	0.332	1.87
Transfer(16)	2,699	12,910	4.78	6,325,889	0.259	16,877,598	0.692	2.67
Transfer(32)	3,672	25,263	6.88	10,919,626	0.448	35,758,365	1.466	3.27
Transfer(64)	6,266	51,445	8.21	22,022,114	0.903	77,024,171	3.158	3.50
Transfer(N)	$O(N \log N)$	$O(N \log N)$		$O(N \log N)$		$O(N \log N)$		

TABLE 3. PERFORMANCE COMPARISON BETWEEN PRIDE CT AND PRIDEFUL CT. GAS COSTS OF VERIFICATION ARE COMPUTED USING THE LATEST STANDARD PRICES OF GAS AT THE TIME OF WRITING, WHICH WAS 41 GWEI, SOURCED FROM [28] AT UTC 04:25 PM 6 NOVEMBER 2023. NOTE THAT THE PRICES ARE SUBJECT TO MARKET FLUCTUATIONS, BUT THE GAS COSTS TEND TO BE STABLE AND MORE MEANINGFUL.

	Proving Time			Gas Consumption				
	PriDeCT	PriDeFULCT	Ratio	PriDeCT		PriDeFULCT		Ratio
				Gas Cost	Ether	Gas Cost	Ether	
N=16	12,910	13,076	1.012	16,877,598	0.692	16,959,745	0.695	1.005
N=32	25,263	25,429	1.007	35,758,365	1.466	35,840,512	1.469	1.002

TABLE 4. TRANSACTION SIZES AS A FUNCTION OF N AND t.

	# Group Elements	# Field Elements	Model
Quisquis	$30N + 22\sqrt{N} + 52 + 2\log(t)$	$6N + 10\sqrt{N} + 39$	UTXO
Anon. Zether	$2N + 8\log(N) + 20$	$2\log(N) + 10$	A/C
PriDe CT	$2N + 2\log(N) + 16$	9	A/C

TABLE 5. COMPARISON BETWEEN DIFFERENT ANONYMOUS CRYPTOCURRENCIES

System	DSE	RA	DEN	NIT	CS	IOH	EW	RPT	CONC
SwapCT	✓	✓	✓	✓		✓		1	
Zcash		✓		✓				1	
Monero	✓		✓	✓				1	
Lelantus	✓							1	
Quisquis	✓	✓	✓	✓	?	✓		n	
Basic Zether	✓			✓	✓	✓	✓	1	
Anon. Zether	✓	✓	✓	✓	✓	✓	✓	1	
This Work	✓	✓	✓	✓	✓	✓	✓	n	✓

DSE	Decentralized Setup, i.e., does not require a trusted setup
RA	Receiver Anonymity
DEN	Deniability, i.e., one cannot determine if a user is to be involved in a given transaction
NIT	Non-interactive transaction, i.e., does not require out-of-band communication
CS	Constant Storage, per user, on full nodes.
COIT	The computation overhead is independent of total funds spent
EW	Efficient Wallet, i.e., constant amount of state from a node
RPT	Number of Receivers per Transaction
CONC	Concurrency without any additional locking mechanism

8. Conclusion

In this work, we present PriDe CT that can function as a "cryptocurrency" using a smart contract and which is interoperable with other smart contracts, akin to the examples proposed in Zether [6]. It offers, in addition to the features of Anonymous Zether, that of batchability (where multiple receivers can be supported without additional overhead), concurrency (multiple senders can send payload to a particular receiver, without additional mechanism), and support for forward secrecy. The resulting protocol offers competitive performance with Anonymous Zether. We identify the following lines of future research that mitigate some of the inherent limitations:

- Identifying a mechanism to incentivize decoy senders to post transactions where each payload is zero to achieve sender anonymity.
- Currently, recovering the update offset from the ciphertext requires one to perform a discrete logarithm. Even though these are infrequent, one can leverage class groups [24] to avoid discrete logarithm computation. Furthermore, whether one can leverage HIBE-based FS-PKE constructions to achieve full forward secrecy (avoiding sender-aided forward secrecy) remains an interesting area of research.

References

[1] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 18–21, 2014, pp. 459–474.

[2] S. Noether, A. Mackenzie, and t. M. Research Lab, "Ring confidential transactions," *Ledger*, vol. 1, p. 1–18, Dec. 2016. [Online]. Available: https://www.ledgerjournal.org/ojs/ledger/article/view/34

[3] F. Engelmann, L. Müller, A. Peter, F. Kargl, and C. Bösch, "SwapCT: Swap Confidential Transactions for Privacy-Preserving Multi-Token Exchanges," *Proceedings on Privacy Enhancing Technologies, PoPETs*, vol. 2021, no. 4, pp. 270–290, 2021. [Online]. Available: https://doi.org/10.2478/popets-2021-0070

[4] A. Poelstra, A. Back, M. Friedenbach, G. Maxwell, and P. Wuille, "Confidential assets," in *FC 2018 Workshops*, ser. Lecture Notes in Computer Science, A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, and M. Sala, Eds., vol. 10958. Nieuwpoort, Curaçao: Springer, Heidelberg, Germany, Mar. 2, 2019, pp. 43–63.

[5] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "Quisquis: A new design for anonymous cryptocurrencies," in *Advances in Cryptology – ASIACRYPT 2019, Part I*, ser. Lecture Notes in Computer Science, S. D. Galbraith and S. Moriai, Eds., vol. 11921. Kobe, Japan: Springer, Heidelberg, Germany, Dec. 8–12, 2019, pp. 649–678.

[6] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *FC 2020: 24th International Conference on Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, J. Bonneau and N. Heninger, Eds., vol. 12059. Kota Kinabalu, Malaysia: Springer, Heidelberg, Germany, Feb. 10–14, 2020, pp. 423–443.

[7] B. E. Diamond, "Many-out-of-many proofs and applications to anonymous zether," in *2021 IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society Press, May 24–27, 2021, pp. 1800–1817.

[8] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in zcash," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18. USA: USENIX Association, 2018, p. 463–477.

[9] A. Biryukov and D. Feher, "Privacy and linkability of mining in zcash," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 118–123.

[10] D. Jost, U. Maurer, and M. Mularczyk, "Efficient ratcheting: Almost-optimal guarantees for secure messaging," in *Advances in Cryptology – EUROCRYPT 2019, Part I*, ser. Lecture Notes in Computer Science, Y. Ishai and V. Rijmen, Eds., vol. 11476. Darmstadt, Germany: Springer, Heidelberg, Germany, May 19–23, 2019, pp. 159–188.

[11] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, "Security analysis and improvements for the IETF MLS standard for group messaging," in *Advances in Cryptology – CRYPTO 2020, Part I*, ser. Lecture Notes in Computer Science, D. Micciancio and T. Ristenpart, Eds., vol. 12170. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 17–21, 2020, pp. 248–277.

[12] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *Advances in Cryptology – EUROCRYPT 2003*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 2656. Warsaw, Poland: Springer, Heidelberg, Germany, May 4–8, 2003, pp. 255–271.

[13] W. contributors, "Harvest now, decrypt later," *Wikipedia*, 6 2023. [Online]. Available: https://en.wikipedia.org/wiki/Harvest_now,_decrypt_later#:~:text=Harvest%20now%2C%20decrypt%20later%2C%20also,it%20readable%20in%20the%20future.

[14] "Onyx by j.p.morgan." [Online]. Available: https://www.jpmorgan.com/onyx/index

[15] "Consensys quorum." [Online]. Available: https://consensys.net/quorum/

[16] "Consensys tessera." [Online]. Available: https://docs.tessera.consensys.net

[17] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[18] Y. Dodis, H. Karthikeyan, and D. Wichs, "Updatable public key encryption in the standard model," in *TCC 2021: 19th Theory of Cryptography Conference, Part III*, ser. Lecture Notes in Computer Science, K. Nissim and B. Waters, Eds., vol. 13044. Raleigh, NC, USA: Springer, Heidelberg, Germany, Nov. 8–11, 2021, pp. 254–285.

[19] Y. Dodis, D. Jost, and H. Karthikeyan, "Forward-secure encryption with fast forwarding," in *TCC 2022: 20th Theory of Cryptography Conference, Part II*, ser. Lecture Notes in Computer Science, E. Kiltz and V. Vaikuntanathan, Eds., vol. 13748. Chicago, IL, USA: Springer, Heidelberg, Germany, Nov. 7–10, 2022, pp. 3–32.

[20] C. A. Haidar, B. Libert, and A. Passelègue, "Updatable public key encryption from DCR: Efficient constructions with stronger security," in *ACM CCS 2022: 29th Conference on Computer and Communications Security*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. Los Angeles, CA, USA: ACM Press, Nov. 7–11, 2022, pp. 11–22.

[21] J. Groth and M. Kohlweiss, "One-out-of-many proofs: Or how to leak a secret and spend a coin," in *Advances in Cryptology – EUROCRYPT 2015, Part II*, ser. Lecture Notes in Computer Science, E. Oswald and M. Fischlin, Eds., vol. 9057. Sofia, Bulgaria: Springer, Heidelberg, Germany, Apr. 26–30, 2015, pp. 253–280.

[22] J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in *Advances in Cryptology – EUROCRYPT 2002*, ser. Lecture Notes in Computer Science, L. R. Knudsen, Ed., vol. 2332. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 28 – May 2, 2002, pp. 466–481.

[23] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," in *Advances in Cryptology – ASIACRYPT 2002*, ser. Lecture Notes in Computer Science, Y. Zheng, Ed., vol. 2501. Queenstown, New Zealand: Springer, Heidelberg, Germany, Dec. 1–5, 2002, pp. 548–566.

[24] H. Karthikeyan and A. Polychroniadou, "Updatable public-key encryption based on class groups: Strong security with better efficiency," Manuscript, 2023.

[25] Y. Guo, H. Karthikeyan, and A. Polychorniadou, "PriDe CT: Towards Public Consensus, Private Transactions, and Forward Secrecy in Decentralized Paments," in *45th IEEE Symposium on Security and Privacy*, 2024.

[26] I. Giacomelli, J. Madsen, and C. Orlandi, "Zkboo: Faster zero-knowledge for boolean circuits," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 1069–1083. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli

[27] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society Press, May 21–23, 2018, pp. 315–334.

[28] "Ethereum gas tracker." [Online]. Available: https://etherscan.io/gasTracker

[29] S. Meiklejohn and R. Mercer, "Möbius: Trustless tumbling for transaction privacy," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 2, pp. 105–121, 2018. [Online]. Available: https://doi.org/10.1515/popets-2018-0015

[30] A. Rondelet and M. Zajac, "ZETH: on integrating zerocash on ethereum," *CoRR*, vol. abs/1904.00905, 2019. [Online]. Available: http://arxiv.org/abs/1904.00905

[31] D. V. Le and A. Gervais, "AMR: autonomous coin mixer with privacy preserving reward distribution," *CoRR*, vol. abs/2010.01056, 2020. [Online]. Available: https://arxiv.org/abs/2010.01056

[32] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity, or all languages in np have zero-knowledge proof systems," *Journal of the ACM*, vol. 38, 1 1991.

[33] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology – EUROCRYPT 2016, Part II*, ser. Lecture Notes in Computer Science, M. Fischlin and J.-S. Coron, Eds., vol. 9666. Vienna, Austria: Springer, Heidelberg, Germany, May 8–12, 2016, pp. 327–357.

[34] J. Groth and Y. Ishai, "Sub-linear zero-knowledge argument for correctness of a shuffle," in *Advances in Cryptology – EUROCRYPT 2008*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., vol. 4965. Istanbul, Turkey: Springer, Heidelberg, Germany, Apr. 13–17, 2008, pp. 379–396.

[35] Y. Lindell, "Parallel coin-tossing and constant-round secure two-party computation," *Journal of Cryptology*, vol. 16, no. 3, pp. 143–184, Jun. 2003.

[36] J. Groth, "Honest verifier zero-knowledge arguments applied," in *Dissertation Series DS-04-3*. BRICS, 2004, phD thesis.

[37] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology – CRYPTO'89*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 20–24, 1990, pp. 239–252.

[38] M. Bellare, A. Boldyreva, and J. Staddon, "Randomness re-use in multi-recipient encryption schemeas," in *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, ser. Lecture Notes in Computer Science, Y. Desmedt, Ed., vol. 2567. Miami, FL, USA: Springer, Heidelberg, Germany, Jan. 6–8, 2003, pp. 85–99.

[39] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology – CRYPTO'91*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed., vol. 576. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 11–15, 1992, pp. 129–140.

Appendix A. Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

A.1. Summary

This paper proposes a private, anonymous, and decentralized payment mechanism while offering forward secrecy. It introduces many new constructions FUL-Zether, PriDe CT, and PriDeFUL CT.

A.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

A.3. Reasons for Acceptance

1) The paper advances the prior studies in the literature and introduces new constructions like forward secrecy until the last update (FUL) and shows that ElGamal Encryption is a secure UPKE.
2) Experimental construction of PriDe-CT shows feasibility as a cryptocurrency on top of Ethereum.
3) Clear advantage in extension in the removal of the reliance on locking, and the optimization of transaction size when compared to Zether
4) Comprehensive coverage of construction of Zether, and extensions into PriDe.

Appendix B. Smart Contract as a Mixer.

Mobiüs [29] facilitated fund transfers between Alice and Bob with provable security guarantees. It achieves low off-chain communication compared to existing centralized and decentralized mixers. To ensure privacy, Alice and Bob need to share Bob's master public key, a shared secret, and a nonce to derive new "decoy public keys" that sever linkability between addresses. A smart contract is used to gather sufficient user deposits before Bob can withdraw. Alice can notify Bob off-chain about the contract address (or Bob can find it independently if one were to assume all contracts are registered). Bob then withdraws by creating a linkable ring signature using the available decoy public keys, by proving that it has a secret corresponding to one of these public keys and also ensuring that this proof can only be used once. Unfortunately, Mobiüs has fixed denominations for transactions, requires waiting for participants, mandates withdrawal after each mixing, and requires active user participation though it supports auditability.

ZETH [30] is a smart contract-based implementation of the Zerocash [1] protocol on Ethereum blockchain. It allows

users to exchange funds for zethNotes which can be reused repeatedly within the contract. Rather than actually transferring these notes from one user to another, the contract works by having the sender consume a set of input notes and produce a set of output notes. Those input notes cannot be used again. ZETH uses zero-knowledge proofs to ensure the correctness of all required operations and to prove that every input note was destroyed while producing output notes. However, ZETH suffers from information leakage, which compromises sender privacy. It also inherits some drawbacks of Zerocash such as trusted setup and monotonically increasing size of coins/commitments. ZETH avoids off-chain communication by using the smart contract to notify, and supports arbitrary denominations for transactions with minimal-to-no latency. Nevertheless, it requires the active participation of users as the receiver needs to monitor events emitted by the sender's invocation of contracts, retrieve all output notes, and verify the sender was honest in providing output commitments.

AMR [31] is a contract-based coin mixer that employs a zk-SNARK-based approach similar to ZETH. It allows deposited funds to accrue interest and enables privacy-preserving reward payment. Clients deposit coins and obtain a note, which is used to withdraw the funds later. The zk-SNARK-based Merkle Tree is used to generate proof that the note corresponds to a transaction and has not been withdrawn already. The note and proof are used to withdraw funds to a new address, severing the linkability between the source and destination addresses. AMR achieves a more efficient zk-SNARK than ZETH but suffers from several drawbacks, such as natively not supporting the transfer of coins between two users, requiring fixed denomination of coins, and the depositor having to withdraw into a new account address, which is an overhead for the client. Furthermore, this does not admit efficient wallet as the state corresponding to the user might require scanning the entire blockchain which can cause issues for auditing.

Appendix C. Cryptographic Primitives

In this section, we describe and formally introduce some of our cryptographic primitives that we used in this paper.

C.1. Zero-Knowledge Proofs

A zero-knowledge proof, informally, is a proof that reveals no information beyond the validity of the statement, especially any and all secret information used in the generation of the proof. It is known that a ZK proof can be generated for any NP statement [32].

Definition 7 (Argument of Knowledge)**.** *Let $\mathcal{R}$ be any polynomial-time decidable relation. Then, an argument of knowledge consists of a triple of algorithms:* $(\mathsf{Setup}_{\mathcal{R}}, \mathcal{P}, \mathcal{V})$ *where:*

- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda)$ *which generates the common reference string* crs *with* λ *as the security parameter. This defines the following NP Language:*

$$\mathcal{L}_{\mathsf{crs}} = \{\mathsf{stmt} | \exists \mathsf{wit} : (\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}\}$$

- $\mathcal{P}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$ *is the randomized prover prover algorithm which is run on inputs* $\mathsf{crs}, \mathsf{stmt}$ *and the secret* wit *which helps prove that* $\mathsf{stmt} \in \mathcal{R}$*. This input* wit *is not provided to the verifier. Instead,* $\mathcal{V}(\mathsf{crs}, \mathsf{stmt})$ *represents the randomized verifier algorithm which is executed on inputs* crs *and* stmt*.*
- *Then,* $\mathrm{tr} \leftarrow\!\!\$\ \langle \mathcal{P}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle$ *represents the transcript generated by one such interaction between $\mathcal{P}$ and $\mathcal{V}$. Often time, we simplify the notation to simply write as* $b \leftarrow\!\!\$\ \langle \mathcal{P}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle$ *where b is a single bit representing the validity of the transcript.*

Definition 8 (Security of Arguments of Knowledge)**.** *The triple* $(\mathsf{Setup}_{\mathcal{R}}, \mathcal{P}, \mathcal{V})$ *is called a secure argument of knowledge for relation $\mathcal{R}$ if it satisfies the following two definitions:*

- *Perfect Completeness: for all non-uniform polynomial time adversaries $\mathcal{A}$, we have:*

$$\Pr\left[\begin{array}{c} (\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}) \notin \mathcal{R} \vee \\ \langle \mathcal{P}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle = 1 \end{array} \middle| \begin{array}{c} (\mathsf{crs}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda) \\ (\mathsf{stmt}, \mathsf{wit}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs}) \end{array} \right] = 1$$

- *Computational Witness-Extended Emulation: for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for all pairs of interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$, there exists a negligible function* $\mathrm{negl}(\lambda)$ *such that:*

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}_1(\mathrm{tr}) = 1 \;\middle|\; \begin{array}{c} (\mathsf{crs}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda) \\ (\mathsf{stmt}, \mathsf{wit}) \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{crs}) \\ \mathrm{tr} \leftarrow\!\!\$\ \langle \mathcal{P}^*(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle \end{array} \right] - \\ \Pr\left[\begin{array}{l} \mathcal{A}_1(\mathrm{tr}) = 1 \wedge \\ (\mathsf{accept}(\mathrm{tr}) \implies (\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}') \in \mathcal{R}) \end{array} \;\middle|\; \begin{array}{c} (\mathsf{crs}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda) \\ (\mathsf{stmt}, \mathsf{wit}) \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{crs}) \\ (\mathrm{tr}, \mathsf{wit}') \leftarrow\!\!\$\ \mathcal{E}^{\mathcal{O}}(\mathsf{crs}, \mathsf{stmt}) \end{array} \right] \end{array} \right| \leq \mathrm{negl}(\lambda)$$

  where $\mathcal{O} = \langle \mathcal{P}^*(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle$*, and permits rewinding to a point and then proceeding with fresh randomness for the verifier from this point onwards and* $\mathsf{accept}(\mathrm{tr})$ *is a predicate that returns true iff the transcript* tr *is accepting.*

In the definition of witness-extended emulation, one can interpret wit to be the state of $\mathcal{P}^*$, including any randomness. Therefore, whenever $\mathcal{P}^*$ convinces the verifier when in state wit, $\mathcal{E}$ is able to extract a witness. In particular, if the adversary is capable of producing a satisfying argument with some probability, then the emulator $\mathcal{E}$ can produce an identically distributed argument and a witness with the same probability. Here, witness-extended emulation is used to define knowledge soundness which was used by prior work including [27], [33] and defined in [34], [35], [36]. However, for our purposes, we will also define a special soundness security property. Recall that a *move* is defined as a message sent from $\mathcal{P}$ to $\mathcal{V}$ or vice versa.

Definition 9 (Special Soundness)**.** *A $2\mu + 1$-move, public-coin argument of knowledge is $(n_1, \ldots, n_\mu)$-special sound*

if, there exists a PPT extractor $\mathcal{E}$ for which, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\lambda)$ *such that:*

$$\Pr\left[(\mathsf{crs},\mathsf{stmt},\mathsf{wit}) \notin \mathcal{R} \;\middle|\; \begin{array}{c}(\mathsf{crs}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda)\\ (\mathsf{stmt},\mathsf{wit},\mathsf{tree}) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{crs})\\ \mathsf{wit} \leftarrow\!\!\$\ \mathcal{E}(\mathsf{crs},\mathsf{stmt},\mathsf{tree})\end{array}\right] \leq \mathrm{negl}(\lambda)$$

where tree *is a* $(n_1,\ldots,n_\mu)$*-tree of accepting transcripts whose challenges are distinct.*

Definition 10 (Public Coin)**.** *An argument of knowledge is called public coin if all messages sent from the verifier to the prover are chosen uniformly at random and independently of any messages from the prover.*

Definition 11 (Computationally Special Honest-Verifier Zero Knowledge)**.** *A public coin argument of knowledge is a special honest verifier zero-knowledge argument for $\mathcal{R}$ if there exists a PPT simulator* Sim *such that for all non-uniform PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{c} b = b' \end{array} \;\middle|\; \begin{array}{c}(\mathsf{crs}) \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda); b \leftarrow\!\!\$\ \{0,1\}\\ (\mathsf{stmt},\mathsf{wit},\rho) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{Setup}_{\mathcal{R}})\\ \mathrm{tr}_0 = \langle \mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit}), \mathcal{V}(\mathsf{crs},\mathsf{stmt};\rho)\rangle\\ \mathrm{tr}_1 = \mathsf{Sim}(\mathsf{crs},\mathsf{stmt},\rho)\\ b' \leftarrow\!\!\$\ \mathcal{A}(\mathrm{tr}_b)\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

where ρ is the public coin randomness used by the verifier. When $\mathrm{negl}(\lambda) = 0$*, we get perfect special honest-verifier zero-knowledge.*

Definition 12 (Σ-protocols)**.** *Σ-protocols are honest-verifier public-coin zero-knowledge interactive proofs which consist of three moves:*

- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}_{\mathcal{R}}(1^\lambda)$
- $a \leftarrow\!\!\$\ \mathcal{P}(\mathsf{crs},\mathsf{stmt},\mathsf{wit})$*: Given* $(\mathsf{crs},\mathsf{stmt},\mathsf{wit}) \in \mathcal{R}$*, it generates an initial message a*
- $c \leftarrow\!\!\$\ \{0,1\}^\lambda$*: A public coin challenge c is chosen uniformly at random by the verifier.*
- $z \leftarrow \mathcal{P}(c)$*: z is the response from $\mathcal{P}$ on the challenge c*
- $b \leftarrow\!\!\$\ \mathcal{V}(\mathsf{crs},\mathsf{stmt},a,c,z)$*: Outputs 1 if accepts, or else 0.*

A commonly used proof of knowledge which satisfies the three move definition of Σ protocol is due to Schnorr [37]. We define that next and this proof system is used in our proof protocol. This protocol is used to prove knowledge of x such that $y = g^x$ where $\mathsf{stmt} = y, \mathsf{wit} = x$

Construction 4 (Schnorr Protocol)**.** The protocol is defined for a cyclic group $\mathbb{G}$ of prime order q with generator g.

- $\mathsf{Setup}_{\mathcal{R}}(1^\lambda)$ chooses a cyclic group $\mathbb{G}$ of prime order q with generator g and sets $\mathsf{crs} = (\mathbb{G}, q, g)$
- $\mathcal{P}(\mathsf{crs}, y, x)$: Given crs, y, x such that $y = g^x$, $\mathcal{P}$ samples r at random and outputs $t = g^r$ as the first message.
- $\mathcal{V}$ replies with a challenge c chosen a random.
- Then, $\mathcal{P}$ responds with $s = r + c \cdot x$
- $\mathcal{V}(\mathsf{crs}, y)$ checks if: $g^s = t \cdot y^c$

Remark 3. It is easy to see that this is an argument of knowledge because it has witness-extended emulation where an extractor can use two different challenges c_1, c_2, and use $s_1 = r + c_1 x$ and $s_2 = r + c_2 x$ to output $(s_1 - s_2)(c_1 - c_2)^{-1}$ as witness x.

It is also trivial to see that the Schnorr protocol satisfies Definition 11 where Sim, with knowledge of the verifier randomness c, can simulate an identically generated transcript where $s = r$ and $t = g^r \cdot y^{-c}$ and this correctly verifies.

Theorem 9. *The Schnorr Protocol satisfies perfect completeness, perfect honest verifier zero-knowledge, and statistical witness-extended emulation for extracting a valid witness x. This extractor $\mathcal{E}_\Sigma$ runs the prover with two different values of the challenge c.*

C.2. Bulletproofs

The work of Bünz *et al.* [27] successfully introduced the powerful abstraction known as Bulletproofs that helped achieve improved efficiency for zero-knowledge proof of inner-product argument. This work improved on prior work of Bootle *et al.* [33] and achieved a communication complexity of $2\log_2 n$ where n is the dimension of the vectors. More formally, they showed that:

Theorem 10 ([27, Theorem 1])**.** *Let $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$ be independent generators of the group $\mathbb{G}$, $c \in \mathbb{F}_q$, and $P \in \mathbb{G}$. Then, there exists an efficient proof system* $(\mathsf{IPProve}, \mathsf{IPVerify})$ *for the following relation:*

$$\mathcal{R}_{\mathsf{IP}} = \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{F}_q; \mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n | P = \mathbf{g}^{\mathbf{a}} \cdot \mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b}\rangle\}$$

This proof system has perfect completeness and statistical witness-extended-emulation for either extracting a non-trivial discrete logarithm relation between $\mathbf{g}, \mathbf{h}$ or extracting a valid witness $\mathbf{a}, \mathbf{b}$. This extractor $\mathcal{E}_{\mathsf{IP}}$ requires $O(n^2)$ transcripts.

Theorem 11 ([27, Theorem 3])**.** *Let $g, h \in \mathbb{G}$ be independent generators of the group $\mathbb{G}$. Then, there exists a proof system* $(\mathsf{RPProve}, \mathsf{RPVerify})$ *for the following relation:*

$$\mathcal{R}_{\mathsf{RP}} = \{(g, h \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^{t'}; \mathbf{b}, \boldsymbol{\gamma} \in \mathbb{F}_q^{t'} | V_j = h^{\gamma_j} \cdot g^{b_j} \wedge b_j \in [0, 2^n - 1]\}$$

which has perfect completeness, perfect honest verifier zero-knowledge, and computational witness-extended emulation. Further, this aggregated range proof uses $2 \cdot \lceil \log_2(t' \cdot n) \rceil + 4$ group elements and 5 elements in $\mathbb{F}_q$.

Corollary 12. *There exists an efficient extractor $\mathcal{E}_{\mathsf{RP}}$ which satisfies the computational witness-extended emulation property, producing either a valid witness $\mathbf{b}, \boldsymbol{\gamma}$ or a non-trivial discrete logarithm relation between independently chosen generators. $\mathcal{E}_{\mathsf{RP}}$ runs the prover with $t' \cdot n$ different values of y [2], $(t' + 2)$ different values of z, and 3 different values of the challenge x (in addition to the extractor $\mathcal{E}_{\mathsf{IP}}$), i.e., rewinding the prover $3 \cdot (t'+2) \cdot t' \cdot n \cdot O(n^2) = O(n^3 \cdot m^2)$.*

[2] y, z, x are elements in $\mathbb{F}_q$ which are randomness provided by the verifier in this interactive protocol.

Corollary 13. *There exists a simulator* $\mathsf{Sim}_{\mathsf{RP}}$ *running in time* $O(\mathsf{RPVerify} + \mathsf{IPProve})$ *which satisfies the honest verifier zero-knowledge property.*

We discuss a more expansive discussion of this result, in the context of our proof requirements.

C.3. Public Key Encryption

Let us recall the definition of a public-key encryption scheme.

Definition 13. *An encryption scheme is a set of three polynomial-time algorithms* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- ***Key generation:*** Gen *receives* 1^λ *where* λ *is the security parameter and outputs a fresh secret* sk *and outputs a fresh public key* pk.
- ***Encryption:*** Enc *receives a public key* pk *and a message* m *to produce a ciphertext* c.
- ***Decryption:*** Dec *receives a secret key* sk *and a ciphertext* c *to produce message* m.

Definition 14 (Correctness of PKE)**.** *For all messages* $m \in \mathcal{M}$*, we require the following to hold:*

$$\Pr\left[\begin{array}{c|c} m' = m & \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, m) \\ m' = \mathsf{Dec}(\mathsf{sk}, c)\end{array}\end{array}\right] = 1$$

Definition 15 (IND-CPA Security)**.** *For all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathrm{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{c|c} b' = b & \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda); b \leftarrow\!\!\$\ \{0,1\} \\ (m_0, m_1) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk}) \\ c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, m_b) \\ b' \leftarrow\!\!\$\ \mathcal{A}(c)\end{array}\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

Construction 5 (ElGamal Encryption [17])**.** For a group $\mathbb{G}$ of prime order q, with generator g, we define the following scheme:

- Gen: Outputs, $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{F}_q, \mathsf{pk} = g^{\mathsf{sk}}$
- $\mathsf{Enc}(\mathsf{pk}, m \in \mathbb{F}_q; r \in \mathbb{F}_q)$: Outputs $(C = g^r, D = \mathsf{pk}^r \cdot g^m)$
- $\mathsf{Dec}(\mathsf{sk}, c = (C, D))$: Outputs m such that $g^m = D \cdot C^{-\mathsf{sk}}$

The above construction is IND-CPA secure under the DDH Assumption. The important point here is that ElGamal Encryption is actually a homomorphic encryption scheme. Specifically, given $c_1 = (C_1, D_1)$ as an encryption of m under pk and an encryption of $c_2 = (C_2, D_2)$ as an encryption of m', also under pk we have that $c' = (\mathsf{nC} = C_1 \cdot C_2, \mathsf{nD} = D_1 \cdot D_2)$ is an encryption of $m + m'$ under pk.

Definition 16 (MR-RR-INDCPA Security of ElGamal [38])**.** *For all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathrm{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{c|c} b' = b & \begin{array}{c} N \leftarrow\!\!\$\ \mathcal{A}(\mathbb{G}, q, g) \\ \textbf{for}\ \ i = 1\ \textbf{to}\ N:\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{Gen}(1^\kappa) \\ \{m_{b,i}\}_{b=0,i=1}^{b=1,i=N} \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N) \\ b \leftarrow\!\!\$\ \{0,1\}; r \leftarrow\!\!\$\ \mathbb{F}_q; C = g^r \\ \textbf{for}\ \ i = 1\ \textbf{to}\ N: D_i = \mathsf{pk}_i^r \cdot m_{b,i} \\ b' \leftarrow\!\!\$\ \mathcal{A}(C, D_1, \ldots, D_N)\end{array}\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

Remark 4. A useful technique that we will use in this paper is to alternatively view (pk, D) as an encryption under the "public key" $C = g^r$. Consequently, given a set of encryptions, under the same randomness, but different public keys, we can view them as encryptions under the same public key, but different randomness. In other words, let $m_0, \ldots, m_t$ be $t+1$ messages to be encrypted under public keys $\mathsf{pk}_0, \ldots, \mathsf{pk}_t$ with the same randomness r. Let $(C, D_i) \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i; r)$ for $i = 0, \ldots, t$. Then, $(\mathsf{pk}_0, D_0), \ldots, (\mathsf{pk}_t, D_t)$ are valid encryptions of $m_0, \ldots, m_t$ under "public key" C with randomness $\mathsf{sk}_0, \ldots, \mathsf{sk}_t$.

C.4. Commitments

Definition 17 (Non-Interactive Commitments)**.** *A non-interactive commitment scheme* com *consists of a pair of probabilistic algorithms* $(\mathsf{Setup}, \mathsf{Commit}_{\mathsf{pp}})$ *where:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *generates public parameters* pp *for the scheme, where* λ *is the security parameter.*
- $com \leftarrow \mathsf{Commit}_{\mathsf{pp}}(x; \alpha)$ *where* $com \in \mathcal{C}$ *is the commitment to message* $x \in \mathcal{M}$*, with randomness* $\alpha \leftarrow\!\!\$\ \mathcal{R}$ *where* $\mathcal{M}$ *is the message space,* $\mathcal{C}$ *is the commitment space, and* $\mathcal{R}$ *is the randomness space.*

Such a commitment scheme is defined to be additively homomorphic if for abelian groups $\mathcal{M}, \mathcal{R}, \mathcal{C}$ *and for all* $x_1, x_2 \in \mathcal{M}, r_1, r_2 \in \mathcal{R}$ *we have:*

$$\mathsf{Commit}_{\mathsf{pp}}(x_1; \alpha_1) + \mathsf{Commit}_{\mathsf{pp}}(x_2; \alpha_2) = \mathsf{Commit}_{\mathsf{pp}}(x_1 + x_2; \alpha_1 + \alpha_2)$$

Definition 18 (Security of Commitment Scheme)**.** *A non-interactive commitment scheme* com *is secure if:*

- *Hiding Property: for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c|c} b = b' & \begin{array}{c}(\mathsf{pp}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ (x_0, x_1, st) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp}) \\ b \leftarrow\!\!\$\ \{0,1\}, \alpha \leftarrow\!\!\$\ \mathcal{R} \\ com = \mathsf{Commit}_{\mathsf{pp}}(x_b; \alpha) \\ b' \leftarrow\!\!\$\ \mathcal{A}(com, st)\end{array}\end{array}\right] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

  where the probability is over $b, \alpha, \mathsf{Setup}, \mathcal{A}$*.*
- *Binding Property: for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c|c} \mathsf{Commit}_{\mathsf{pp}}(x_0; \alpha_0) = \mathsf{Commit}_{\mathsf{pp}}(x_1; \alpha_1) \wedge x_0 \neq x_1 & \begin{array}{c}(\mathsf{pp}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ (x_0, x_1, \alpha_0, \alpha_1, st) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp})\end{array}\end{array}\right] \leq \mathrm{negl}(\lambda)$$

  where the probability is over $\mathsf{Setup}, \mathcal{A}$*. If the probability is 0, then we say that the scheme is perfectly binding.*

We have the following commitment scheme from Pedersen [39].

Construction 6 (Pedersen Commitment Schemes)**.** Here, $\mathcal{M}, \mathcal{R} = \mathbb{F}_q, \mathcal{C} = \mathbb{G}$ of order q.

- Setup: Outputs $g, h \leftarrow_\$ \mathbb{G}$
- $\mathsf{Commit}_{\mathsf{pp}}(x; \alpha)$: Outputs $g^x \cdot h^\alpha$

This can be extended to the commit to a vector, with the same randomness α, as follows:

- Setup: Outputs $\mathbf{g} = (g_1, \ldots, g_n), h \leftarrow_\$ \mathbb{G}$
- $\mathsf{Commit}_{\mathsf{pp}}(\mathbf{x} = (x_1, \ldots, x_n); \alpha)$: Outputs $\mathbf{g}^{\mathbf{x}} \cdot h^\alpha = h^\alpha \cdot \prod_{i=1}^n g_i^{x_i}$

The above construction is *perfectly hiding* and *computationally binding* under the discrete logarithm assumption.

Remark 5. When $\alpha = 0$, i.e., $C = \mathbf{g}^{\mathbf{x}}$ is binding (but not hiding) commitment to the vector $\mathbf{a}$. Given C, and another vector $\mathbf{y} \in \mathbb{F}_q^*$, we can treat C as a new commitment to $\mathbf{x} \circ \mathbf{y}$. This follows by defining $\mathbf{g}'$ such that $g'_i = g_i^{y_i^{-1}}$ with $C = \prod_{i=1}^n (g'_i)^{x_i \cdot y_i}$.

Remark 6. The ElGamal ciphertext of $(g^r, g^{\mathsf{sk} \cdot r} \cdot g^m)$ can be seen as a commitment to m where $h = g^r$ and sk is the blinding randomness. This is perfectly hiding and computationally binding.

Appendix D. Deferred Proofs

D.1. Proof for Theorem 6

Proof. We present a brief proof for each component of the statement.

Proof of Perfect Completeness. The perfect completeness, while straightforward and a consequence of the modular components of the aggregate range proof and the deployed Schnorr Protocol will be verified in this proof.

- **Check for A_y.** On the right-hand side, we have:

$$g^{s_{\mathsf{sk}}} \cdot \mathsf{pk}^{-c} = g^{k_{\mathsf{sk}} + c \cdot \mathsf{sk}} \mathsf{pk}^{-c} = g^{k_{\mathsf{sk}}} \mathsf{pk}^c \mathsf{pk}^{-c} = g^{k_{\mathsf{sk}}} = A_y$$

- **Check for A_C.** On the right-hand side, we have:

$$g^{s_r} \cdot C^{-c} = g^{k_r + c \cdot r} \cdot C^{-c} = g^{k_r} \cdot C^c \cdot C^{-c} = g^{k_r} = A_C$$

- **Check for A_X.** First, observe that $\mathsf{pl}_0 + \mathsf{pl}_1 + \ldots + \mathsf{pl}_t = 0$. Therefore, $\left(\prod_{j=0}^t D_j\right) = \left(\prod_{j=0}^t \mathsf{pk}_j\right)^r$. On the right-hand side, we have:

$$\left(\prod_{j=0}^t \mathsf{pk}_j\right)^{s_r} \left(\prod_{j=0}^t D_j\right)^{-c} = \left(\prod_{j=0}^t \mathsf{pk}_j\right)^{k_r + r} \left(\prod_{j=0}^t \mathsf{pk}_j\right)^{-cr}$$

$$= \left(\prod_{j=0}^t \mathsf{pk}_j\right)^{k_r} = A_X$$

- **Check for A_b.** We will prove this by comparing the coefficients of each exponent of z on the LHS and RHS.
  - Coefficient of z^2. On the LHS we have: $C^{-k_{\mathsf{sk}}} \cdot g^{-c \cdot \mathsf{pl}_0}$. Observe that $C^{\mathsf{sk}} = D_0 \cdot g^{\mathsf{pl}_0}$. On the RHS we have:

$$\begin{aligned} C^{-s_{\mathsf{sk}}} \cdot D_0^c &= C^{-k_{\mathsf{sk}} - c \cdot \mathsf{sk}} \cdot D_0^c \\ &= C^{-k_{\mathsf{sk}}} \cdot D_0^{-c} \cdot g^{-\mathsf{pl}_0 \cdot c} \cdot D_0^c \\ &= C^{-k_{\mathsf{sk}}} \cdot g^{-c \cdot \mathsf{pl}_0} \end{aligned}$$

  - Coefficient of $z^{t'+1}$. On the LHS we have: $\mathsf{nC}_0^{k_{\mathsf{sk}}} \cdot g^{-c \cdot \mathsf{pl}_{t'-1}}$. Observe that $\mathsf{nC}_0^{\mathsf{sk}} = D'_0 \cdot g^{-\mathsf{pl}_{t'-1}}$. On the RHS we have:

$$\begin{aligned} \mathsf{nC}_0^{s_{\mathsf{sk}}} \cdot D_0'^{-c} &= \mathsf{nC}_0^{k_{\mathsf{sk}} + c \cdot \mathsf{sk}} \cdot D_0'^{-c} \\ &= \mathsf{nC}_0^{k_{\mathsf{sk}}} \cdot D_0'^{c} \cdot g^{-\mathsf{pl}_{t'-1} \cdot c} \cdot D_0'^{-c} \\ &= \mathsf{nC}_0^{k_{\mathsf{sk}}} \cdot g^{-c \cdot \mathsf{pl}_{t'-1}} \end{aligned}$$

  - Coefficient of z^{2+j} for $j = 1, \ldots, t$. On the LHS we have: $\mathsf{pk}_j^{k_r} \cdot g^{-c \cdot \mathsf{pl}_j}$. Observe that $D_j^{-1} \cdot \mathsf{pk}_j^r = g^{-\mathsf{pl}_j}$. On the RHS, we have:

$$\begin{aligned} \left(D_j^{-c} \cdot \mathsf{pk}_j^{s_r}\right) &= \left(D_j^{-c} \cdot \mathsf{pk}_j^{k_r + c \cdot r}\right) \\ &= \mathsf{pk}_j^{k_r} \cdot (D_j^{-1} \cdot \mathsf{pk}_j^r)^c = \mathsf{pk}_j^{k_r} \cdot g^{-c \cdot \mathsf{pl}_j} \end{aligned}$$

- **Check for A_τ.** Now, observe that $t_0 = \delta(y,z) + \sum_{j=1}^{t'} z^{1+j} \cdot \mathsf{pl}_{j-1} = \delta(y,z) + z^2 \cdot \left\langle \mathbf{z}^{t'}, \mathbf{b} \right\rangle$. Now, $g^{s_b} = g^{k_b} \cdot g^{c \cdot z^2 \cdot \left\langle \mathbf{z}^{t'}, \mathbf{b} \right\rangle}$. Therefore, on the RHS we have:

$$\begin{aligned} g^{c \cdot \delta(y,z)} \cdot g^{s_b} \cdot A_\tau &= g^{k_b} \cdot g^{c \cdot \left(\delta(y,z) + z^2 \cdot \left\langle \mathbf{z}^{t'}, \mathbf{b} \right\rangle\right)} \cdot g^{-k_b} \cdot h^{k_\tau} \\ &= g^{c \cdot t_0} \cdot h^{k_\tau} \end{aligned}$$

$$\begin{aligned} T_1^x \cdot T_2^{x^2} := \left(g^{t_1} \cdot h^{\tau_1}\right)^x \cdot \left(g^{t_2} \cdot h^{\tau_2}\right)^{x^2} &= h^{\tau_x} \cdot g^{t_1 \cdot x + t_2 \cdot x^2} \\ &= h^{\tau_x} \cdot g^{\hat{t} - t_0} \end{aligned}$$

Therefore, the RHS simplifies to:

$$\begin{aligned} g^{c \cdot \delta(y,z)} \cdot g^{s_b} \cdot A_\tau \cdot (T_1^x \cdot T_2^{x^2})^c &= g^{c \cdot t_0} \cdot h^{k_\tau} \cdot h^{c \cdot \tau_x} \cdot g^{c \cdot \hat{t} - c \cdot t_0} \\ &= g^{c \cdot \hat{t}} \cdot h^{k_\tau + c \cdot \tau_x} = g^{c \cdot \hat{t}} \cdot h^{s_\tau} \end{aligned}$$

Proof of SHVZK The protocol is honest verifier zero-knowledge because one can create a simulator that generates verifying transcripts without access to the witnesses. The simulator merely samples a random challenge c, along with the $s_{\mathsf{sk}}, s_r, s_b, s_\tau$. Then, it reconstructs the values of $A_y, A_C, A_b, A_X, A_\tau$ based on the verification equations. Then, if $g, h, \prod_{j=0}^t \mathsf{pk}_j, C$ are group generators, each of the values generated is a random group element in the honest protocol and in the simulated transcript. Further, A_X and A_C form a DDH tuple with basis $g, \prod_{j=0}^t \mathsf{pk}_j$ making A_X, A_C computationally indistinguishable in the honest and simulated transcripts.

Proof of Computational Witness Extended Emulation We have already seen how to extract witnesses from a Σ protocol. Using two transcripts involving two different values of c as c_1 and c_2 (and induced values of $s_{\mathsf{sk},1}, s_{\mathsf{sk},2}, s_{r_1}, s_{r_2}, s_{b_1}, s_{b_2}$ one can compute:

$$\mathsf{sk} = \frac{s_{\mathsf{sk},1} - s_{\mathsf{sk},2}}{c_1 - c_2}, r = \frac{s_{r,1} - s_{r,2}}{c_1 - c_2}, b = \frac{s_{b,1} - s_{b,2}}{c_1 - c_2}, \tau = \frac{s_{\tau,1} - s_{\tau,2}}{c_1 - c_2}$$

Here, note that b corresponds to $\sum_{j=1}^{t'} z^{1+j} \mathsf{pl}_{j-1}$. Now, we need to rely on the $\mathcal{E}_{\mathsf{RP}}$ (relying on Corollary 12) to extract the witness **b**. □

D.2. Deferred Proofs of PriDe CT

Proof. In this proof, recall that n is such that $\mathsf{MAX} = 2^n - 1$, i.e., n is the maximum number of bits that is supported as payload. Then, consider the following sequence of hybrids:

$\mathcal{H}_0$ Corresponds to $\mathsf{Overdraft}_{\mathcal{A},\Pi}(\kappa)$;

$\mathcal{H}_1$ Same as $\mathcal{H}_0$, except that the interactive proof protocol from Section 5.2 is used for all oracle queries (both to $\mathcal{O}_{\mathsf{Transact}}$ and $\mathcal{O}_{\mathsf{Insert}}$.)

Claim 14. *For some polynomial q, there exists a negligible function* $\mathrm{negl}(\kappa)$ *such that:*

$$\mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-0}} \leq \mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-1}} \cdot q(\kappa)^{4+\log(t' \cdot n)} + \mathrm{negl}(\kappa)$$

The proof for this claim is quite straightforward. It boils down to inconsistencies being generated in the honest transcript corresponding to $\mathcal{H}_0$ which uses the random oracle to query, vs the actual value sent by the verifier in the interactive protocol corresponding to $\mathcal{H}_1$. In other words, there are 4 + $\log(N \cdot n)$ places where a challenge from the verifier is provided and these are the spots where one can identify an inconsistency. The latter term comes from the inner-product verification protocol from Bulletproofs [27].

$\mathcal{H}_2$ Same as $\mathcal{H}_1$, except that the experimenter reruns $\mathcal{A}$ with the same randomness but different challenges y, z, x, c and obtains a $(t' \cdot n, t' + 2, 3, 2)$-tree of accepting transcripts after obtaining the final transcript (tx^*), and returns 0 if any of the challenges feature collisions. In addition, we also run the extractor of $\mathcal{E}_{\mathsf{IP}}$, generating a 4-ary tree of depth $\log(t' \cdot n)$. Finally, the experimenter imposes the winning condition of $\mathcal{H}_1$ on all leaves.

Claim 15. *For some polynomial q, there exists a negligible function* $\mathrm{negl}'(\kappa)$ *such that:*

$$\left(\mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-1}}\right)^{(t' \cdot n) \cdot (t'+2) \cdot 3 \cdot 2 \cdot 4^{\log(t' \cdot n)}} \leq \mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-2}} + \mathrm{negl}'(\kappa)$$

The proof of this claim follows from applying Jensen's inequality to each level in the tree of transcripts.

$\mathcal{H}_3$ Same as $\mathcal{H}_2$ except that the experimenter runs the extractor $\mathcal{X}$ to extract witnesses corresponding to the statement.

Claim 16. *There exists a negligible function* $\mathrm{negl}(\kappa)$ *such that:*

$$\left(\mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-2}}\right) \leq \mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-3}} + \mathrm{negl}(\kappa)$$

The proof of this claim follows from the Computational Witness Extended Emulation property. Furthermore, it is clear from these witnesses that the conditions 6.a and 6.b from $\mathsf{Overdraft}_{\mathcal{A},\Pi}$ are met.

$\mathcal{H}_4$ Same as $\mathcal{H}_3$ except that the experimenter chooses a random y^* for the sender of the challenge transaction tx^*.

Claim 17. $\left(\mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-3}}\right) \leq N \cdot \mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-4}} + \mathrm{negl}(\kappa)$ where N is the total number of honestly generated users in the experiment.

Claim 18. *Under the Discrete Logarithm Assumption, there exists a negligible function* $\mathrm{negl}(\kappa)$ *such that:*

$$\left(\mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-4}}\right) = \mathrm{negl}(\kappa)$$

The proof of this claim follows similarly to the proof of Claim B.6 from the work of Anonymous Zether [7]. Let y^* be the Discrete Logarithm Experiment challenge. During the account creation process, the experimenter randomly chooses an index $i^* \in \{0, \ldots, N-1\}$ and sets $\mathsf{pk}_i = y^*$. For every Transact query with i^* as the sender, the experimenter uses SHVZK property to simulate the transcript. For any query to corrupt with $i = i^*$, abort or else answer honestly. Then, after it receives the challenge transaction tx^*, it rewinds to generate a tree of accepting transcripts as defined in $\mathcal{H}_2$. If the sender in the transaction tx^* is not i^*, we abort as well. Otherwise, it uses the extractor to return an element sk such that $g^{\mathsf{sk}} = y^*$ and returns this value of sk. □

Theorem 8. *Under the DDH Assumption, PriDe CT is ledger-indistinguishable.*

Proof. The proof of this theorem follows the proof of Theorem 6.5 from Anonymous Zether [7]. However, there are some notable differences, owing to the simplification of our scheme and the security game. To begin with, we will reduce the security to the multi-recipient, randomness-reusing IND-CPA security game proposed by Bellare *et al.* [38]

$\mathcal{H}_0$ Corresponds to $\mathsf{L\text{-}IND}_{\mathcal{A},\Pi}(\kappa)$

$\mathcal{H}_1$ Same as $\mathcal{H}_0$, except that all Σ-proofs are simulated. *We have already argued that there is no difference advantage between $\mathcal{H}_0$ and $\mathcal{H}_1$ because of the SHVZK property of the scheme. Therefore, under DDH, these two Games are computationally indistinguishable.*

$\mathcal{H}_2$ Same as $\mathcal{H}_1$, except that, in the challenge transaction tx^*, the receivers' payload is replaced with a random value.

Claim 19. *Under the MR-RR-INDCPA security of ElGamal Encryption, there exists a negligible function* negl *such that:*

$$\left(\mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-2}}\right) \leq \mathrm{Adv}^{\textit{PriDe CT}}_{\text{Game-1}} + \mathrm{negl}(\kappa)$$

Proof: We will use an adversary capable of distinguishing $\mathcal{A}$ between $\mathcal{H}_1$ and $\mathcal{H}_2$ to build an adversary $\mathcal{B}$ capable of winning the MR-RR-INDCPA security

game (see Definition 16. Upon receiving the group description from its challenger $\mathcal{B}$ does the following:

- It generates the necessary pp consistent with these inputs from its challenger.
- It runs $\mathcal{A}(\mathsf{pp})$ and receives the balances $\mathsf{b}_{,1}, \ldots, \mathsf{b}_{,N}$ as inputs.
- $\mathcal{B}$ then returns N to its challenger receiving $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ in response.
- $\mathcal{B}$ generates the initial state by computing $\mathsf{U\text{-}Enc}(\mathsf{pk}_i, \mathsf{b}_{,i})$,
- $\mathcal{B}$ responds to the queries from $\mathcal{A}$ as follows:
  - Any input to $\mathcal{O}_{\mathsf{Insert}}$ queries are merely handled by verification followed by homomorphic multiplication.
  - Any input to $\mathcal{O}_{\mathsf{Transact}}$ requires $\mathcal{B}$ to generate a valid multi-recipient, randomness-reusing ciphertext along with a simulated proof, as specified earlier.
- $\mathcal{B}$ now receives the challenge payload as $i, \mathbf{B}_0^*, \mathbf{B}_1^*, \mathbf{R}$ which is provided as input to its challenger, receiving the encrypted challenge ciphertexts. Again, proofs are simulated even though $\mathcal{B}$ does not know which of the payloads were used

□

D.3. FUL Security of FUL-Zether

Theorem 5 (Forward-secure Until Last update)**.** *Under DDH Assumption, the construction of FUL-Zether (as described in Section 4.3) is FUL-secure.*

Proof. Note that interactive proof protocol is a standard Σ protocol which is honest verifier zero knowledge, as one can simulate verifying transcripts without having access to the witness. Specifically, the simulator samples its random challenge c, and the randomness values $s_{rp}, s_\delta, s'_\delta$ (in addition to the overall protocol's randomness of $s_{\mathsf{sk}}, s_r, s_b, s_\tau$). Then, it can follow the verification steps to compute

$$A_{rp} = g^{s_{rp}} \cdot c_1^{-c}, A_\delta = g^{s_\delta} \cdot (\mathsf{pk}'/\mathsf{pk})^{-c}, A'_\delta = E^{-c} \cdot \mathsf{nC}_B^{s'_\delta}$$

If g, nC_B are not equal to 1, i.e., they are group generators, then under the DDH assumption, the honest values of A cannot be distinguished from the simulated values. Therefore, the simulated transcript is identically distributed.

We will prove the security of our construction by reducing it to the IND-CU-CPA security (as defined in Definition 6) of the UPKE scheme (defined in Definition 3) Specifically, we will show that if there exists an adversary $\mathcal{A}_{\mathsf{FUL}}$ that compromises its security in $\mathsf{FUL}_{\mathcal{A},\Pi}$, then we can build an adversary that breaks the $\mathcal{A}$ In $\mathsf{FUL}_{\mathcal{A},\Pi}$, the challenger simulates a set of random balances, generates keys, and then encrypts the balance accordingly. To reduce it to the IND-CR-CPA security, the challenger of the IND-CR-CPA game maintains a secret key that is unknown to the adversary.

For ease of notation let us assume that there exist the following functions:

- $\mathsf{Current\text{-}PK}(A)$ returns the current public key of user A
- $\mathsf{Current\text{-}SK}(A)$ returns the current secret key of user A

Further, throughout this proof, we will refer to the sender of the challenge transaction as Alice and the receiver as Bob.

$\mathcal{H}_0$ Corresponds to $\mathsf{FUL}_{\mathcal{A},\Pi}(\kappa)$

$\mathcal{H}_1$ Same as $\mathcal{H}_0$, except that all Σ-proofs are simulated.
We have already argued that there is no difference advantage between $\mathcal{H}_0$ and $\mathcal{H}_1$ because of the HVZK property of the scheme. Therefore, under DDH, these two Games are computationally indistinguishable.

$\mathcal{H}_2$ Same as $\mathcal{H}_1$, except that the challenge transaction is generated as follows:

$$D_1 = \mathsf{Current\text{-}PK}(Bob)^r \cdot g^{\mathsf{pl}}$$
$$D_0 = \left(\frac{\mathsf{Current\text{-}PK}(Alice)}{\mathsf{Current\text{-}PK}(Bob)}\right)^r \cdot g^{-\mathsf{pl}}$$
$$A_y = \mathsf{Current\text{-}PK}(Alice)^{s_r} \cdot (D_0 \cdot D_1)^c$$

where pl is either B_0^* or B_1^*
It is easy to verify that this modification still yields an accepting transcript. Further, under DDH, these two distributions are indistinguishable.

$\mathcal{H}_3$ Same as $\mathcal{H}_1$, except that, in the challenge transaction tx^*, the receiver's payload is replaced with a random value. Let us call this receiver as Bob.

Then, we can reduce the distinguishing between $\mathcal{H}_2$ and $\mathcal{H}_3$ by to the security of the UPKE Scheme. Let $\mathcal{B}$ be the adversary capable of distinguishing the two games. Then, we will build $\mathcal{A}$ that can break the UPKE Scheme's IND-CU-CPA. $\mathcal{A}$ does the following: (a) use the challenger of the IND-CU-CPA scheme to maintain Bob's account and (b) the rest is maintained locally. Therefore, it receives Bob's public key for time period 0, as $\mathsf{pk}_0^{(0)}$ from the challenger. Then, it samples the balance and encrypts it using $\mathsf{pk}_0^{(0)}$.

Query Phase. We now look at how $\mathcal{A}$ responds on receiving queries from $\mathcal{B}$. Recall that $\mathcal{B}$ can make oracle calls to either:

- $\mathcal{O}_{\mathsf{Transact}}$ with inputs sender's index i, and $\mathbf{B}$ containing the receiver's public key and the payload pl.
- $\mathcal{O}_{\mathsf{Insert}}$ with input as a transaction tx.
- Any calls to $\mathcal{O}_{\mathsf{Transact}}$ with Bob as the sender will require an honest generation of the transaction but Bob's secret key is unknown to $\mathcal{A}$. However, as defined in $\mathcal{H}_2$, this will anyway be simulated, as illustrated earlier in the proof.
- Any calls to $\mathcal{O}_{\mathsf{Transact}}$ with Bob as the receiver will simply require $\mathcal{A}$ to randomly sample the δ corresponding to an update, update the balance, and provide the correct input to its Challenger.
- Any calls to $\mathcal{O}_{\mathsf{Insert}}$ with Bob as the receiver will require the evolution of Bob's keys which is maintained by its challenger. Recall that the IND-CU-CPA requires as inputs the update ciphertext up and pk', both of which are provided by $\mathcal{B}$. Further, it uses the input E from $\mathcal{B}$ as a part of this transaction message to evolve the balance of Bob correctly.

- Any calls to $\mathcal{O}_{\mathsf{Insert}}$ with Bob as the sender will be locally maintained, and no interaction with the IND-CU-CPA challenger is required.
- Clearly, any transactions received from $\mathcal{B}$ either as input to $\mathcal{O}_{\mathsf{Insert}}$ or $\mathcal{O}_{\mathsf{Transact}}$ involving any party that is not the receiver is handled easily by $\mathcal{A}$. It can honestly generate transaction messages and also issue them.

Challenge Phase. $\mathcal{A}$ receives from $\mathcal{B}$ the following inputs: i, the index of the honest sender, payload choices $\mathsf{pl}_0^*, \mathsf{pl}_1^*$, and the receiver's identity/public key. $\mathcal{A}$ forwards to its challenger its challenge messages: $g^{\mathsf{pl}_0^*}$ and $g^{\mathsf{pl}_1^*}$. In response, it receives:

- Challenge Ciphertext: $C = g^r, D_1 = (\mathsf{Current\text{-}PK}(Bob))^r \cdot g^{\mathsf{pl}_b}$ for a randomly tossed bit b.
- Update Ciphertext corresponding to some choice of δ^* as follows: $g^{r'}, \mathsf{Current\text{-}PK}(Bob)^{r'} \cdot g^{\delta}$ and proof π about this values well-formedness.
- Updated Public Key of Bob as $\mathsf{Current\text{-}PK}(Bob) \cdot g^{\delta}$ and secret key sk_B^*.

$\mathcal{A}$ ignores the proof π and instead simulates it (as defined earlier). To compute D_1, it does the following:

$$D_0 := C^{\mathsf{Current\text{-}SK}(Alice)} \cdot D_1^{-1}$$

$\mathcal{A}$ provides the correctly simulated transcript corresponding to these generated statements as the challenge transaction. In addition, $\mathcal{A}$ also provides the secret key of the receiver received as sk_B^* from its challenger. $\mathcal{B}$'s guess is forwarded as $\mathcal{A}$'s own guess. It is easy to see that if the tossed bit was 0, then $\mathcal{A}$ correctly simulates the distribution that corresponds to the payload choice of pl_0^* and similarly if the tossed bit was 1, the distribution will be the case when pl_1^*. The secret key is correctly distributed. Therefore, $\mathcal{B}$'s advantage in distinguishing Games $\mathcal{H}_2$ and $\mathcal{H}_3$ is the same as $\mathcal{A}$'s advantage in winning the IND-CU-CPA security game. $\square$