Confidential Transactions
Theory Justification

Iftach Haitner*

August 7, 2025

Abstract

[**Iftach's Note: TODO**]

Contents

*Stellar Development Foundation. E-mail: `iftach.haitner@stellar.org`..

1 Introduction

[**Iftach's Note: TODO**]

2 Preliminaries

2.1 Notation

We use calligraphic letters to denote sets, uppercase for random variables, and lowercase for integers and functions. Let $\mathbb{N}$ denote the set of natural numbers. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$ and $(n) := \{0, \ldots, n\}$. For a relation $\mathcal{R}$, let $\mathcal{L}(\mathcal{R})$ denote its underlying language, i.e., $\mathcal{L}(\mathcal{R}) := \{x \colon \exists w \colon (x, w) \in \mathcal{R}\}$.

2.2 Homomorphic Encryption

An homomorphic encryption over $\mathbb{Z}_q$ is a triplet $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of efficient algorithms, with the standard correctness and semantic security properties. In addition, there exist an efficient addition operation denote $+$ such that for uniformly generated public key pk, and any two messages $x_0, x_1 \in \mathbb{Z}_q$, it holds that $\mathsf{Enc}_{\mathsf{pk}}(x_0) + \mathsf{Enc}_{\mathsf{pk}}(x_1)$ are computationally indistinguishable from $\mathsf{Enc}_{\mathsf{pk}}(x_0 + x_1 \bmod q)$.

3 The Confidential Transaction Protocol

3.1 The Ideal Functionality

Functionality 3.1 ($\mathcal{F}_{\mathsf{ConfTrans}}$: Confidential transactions)**.**

Parties: Issuer I, chain holder C and users $\mathsf{U}_1, \ldots, \mathsf{U}_n$.

Parameters: $p_{\mathsf{pcount}}, p_{\mathsf{size}}, q \in \mathbb{N}$.

Init. Upon receiving init from all parties:

1. For each $i \in [n]$: $\mathsf{avlBlance}_i, \mathsf{pndBalance}_i \leftarrow 0$, $\mathsf{tcount}_i \leftarrow 0$, and $\mathsf{log}_i \leftarrow \emptyset$.
2. $\mathsf{log} \leftarrow \emptyset$.

Issue. Upon receiving $(\mathsf{sid}, \mathsf{issue}, x, d)$ from C and I:

1. Assert($x \in (p_{\mathsf{size}})$, $\mathsf{tcount}_d \leq p_{\mathsf{pcount}}$ and $d \in [n]$).
2. tcount_d^{++}.
3. $\mathsf{pndBalance}_d \mathrel{+}= x$.
4. Set $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{issue}, d, x, \mathsf{tcount}_d)$.

Transfer. Upon receiving $(\mathsf{sid}, \mathsf{transfer}, d)$ from C and U_s, with U_s using private input x.

1. Assert($x \in (p_{\mathsf{pcount}})$, $\mathsf{tcount} \leq p_{\mathsf{size}}$, $\mathsf{avlBlance}_s \geq x$ and $d \in [n]$).

2. tcount^{++}.
3. $\mathsf{avlBlance}_s \mathrel{-}= x$.
4. $\mathsf{pndBalance}_d \cup= x$.
5. Set $\mathsf{log}_d \cup= (\mathsf{sid}, \mathsf{transfer}, s, x)$
6. Set $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{transfer}, s, d)$

Rollover. Upon receiving $(\mathsf{sid}, \mathsf{rollover})$ from party U_i and C, party C

1. $\mathsf{tcount} \leftarrow 0$.
2. Set $\mathsf{avlBlance}_i \mathrel{+}= \mathsf{pndBalance}_i$.
3. Set $\mathsf{pndBalance}_i \leftarrow 0$.
4. Set $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{rollover}, i)$

Withraw. Upon receiving $(\mathsf{sid}, \mathsf{withraw}, x)$ from party U_i and C, party C

1. Assert($x \in \mathbb{N}$, $\mathsf{avlBlance}_i \geq x$ and $i \in [n]$).
2. $\mathsf{avlBlance}_i \mathrel{-}= x$.
3. Set $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{withraw}, i, x)$

History. Upon receiving $(\mathsf{sid}, \mathsf{history})$ from party U_i and C:

Send $(\mathsf{log}, \mathsf{log}_i)$ to U_i.

Audit. [**Iftach's Note: Later**]

3.2 The Protocol

Throughout, we fix a security parameter κ and omit is from the notation. We also fix an homomorphic encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over $\mathbb{Z}_q$ with randomness domain $\mathcal{D}$. We require that $\mathsf{Dec}_{sk}(\overline{A})$ outputs $(a; r)$ such that $\overline{A} = \mathsf{Enc}(a; r)$.

Main protocol. We split the protocol into several sub-protocols defined below, and use the following environment to define the common part the different sub-protocols share, e.g., global parameter.

Protocol 3.2 ($\Pi_{\mathsf{ConfTrans}}$: Confidential transactions)**.**

Parties: Issuer I, chain-holder C and users $\mathsf{U}_1, \ldots, \mathsf{U}_n$.

Parameters: $p_{\mathsf{pcount}}, p_{\mathsf{size}}, q \in \mathbb{N}$.

Subprotocols: See below.

Init. This sub-protocol is were the encryption key are sampled and shared, and the chain manager C set the initial values of the chain. The protocol uses ZKPOK proof for the relation:

Key generation: $\mathcal{R}_{\mathsf{KeyGen}} = \{(\mathsf{pk}, w)) \colon \mathsf{KeyGen}(w) = (\cdot, \mathsf{pk})\}$.

Protocol 3.3 ($\Pi_{\mathsf{ConfTrans}}$.Init)**.**

Participating parties. All parties.

Proofs: $\Pi^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{KeyGen}}}$.

Algorithms: KeyGen.

Operation:

1. U_i, for all $i \in [n]$:
   (a) Set $(pk_i, sk_i) \xleftarrow{\text{R}} \mathsf{KeyGen}(r_i)$ for $r_i \xleftarrow{\text{R}} \mathcal{D}$.
   (b) Store sk_i.
   (c) Let $\pi_i \xleftarrow{\text{R}} \mathsf{P}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{KeyGen}}}(pk_i, r_i)$.
   (d) Send (pk_i, π_i) to C.
2. C:
   (a) Call $\{\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{KeyGen}}}(pk_i, \pi_i)\}_{i\in[n]}$. Abort and publish i, if the i^{th} proof is not verified.
   (b) Store $\{pk_i\}_{i\in[n]}$.
3. C:
   (a) Broadcast $\{\overline{P}_i \xleftarrow{\text{R}} \mathsf{Enc}_{pk_i}(0), \overline{B}_i \leftarrow \mathsf{Enc}_{pk_i}(0), \mathsf{tcount}_i \leftarrow 0\}_{i\in[n]}$.
   (b) Broadcast $\mathsf{log} \leftarrow \emptyset$.

Issue.

Protocol 3.4 ($\Pi_{\mathsf{ConfTrans}}$.Issue)**.**

Participating parties. I and C.

Common input. sid, $x \in \mathbb{N}$ and $d \in [n]$.

Operation: C

C:

1. Assert($x \in [p_{\mathsf{size}}]$ and $\mathsf{tcount}_d \leq p_{\mathsf{pcount}}$)
2. Set $\overline{P}_i \mathrel{+}= \mathsf{Enc}_{pk_i}(x)$).
3. Broadcast $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{issue}, d, x, \overline{P}_i, \mathsf{tcount}_d)$.

Transfer. The protocol uses ZK and ZKPOK proofs for the following relations:

In range. $\mathcal{R}_{\mathsf{Rp}} = \big\{((\mathsf{pk}, \overline{A}, b), (a, r)) \colon \mathsf{Enc}_{\mathsf{pk}}(a; r) = \overline{A} \wedge a \in (b)\big\}$, i.e., encryption of values in $[p_{\mathsf{size}}]$.

Equality. $\mathcal{R}_{\mathsf{Eq}} = \{((\mathsf{pk}_0, \mathsf{pk}_1, \overline{A}_0, \overline{A}_1), (a, r_0, r_1)) \colon \forall i \in \{0,1\}\ \mathsf{Enc}_{\mathsf{pk}_i}(a; r_i) = \mathrm{A}_i\}$, i.e., encryptions of the same pair under different public keys.

Larger than. $\mathcal{R}_{\mathsf{LrgerEq}} = \{((\mathsf{pk}, \overline{A}_0, \overline{A}_1), (a_0, r_0, a_1, r_1)) \colon \forall i \in \{0,1\}\ \mathsf{Enc}_{\mathsf{pk}}(a_i; r_i) = \overline{A}_i \wedge a_1 - a_0 \in (q)\}$, i.e., encryptions of the pair of values (a_0, a_1), under the same public key, with $a_1 \geq a_0$.

Protocol 3.5 ($\Pi_{\mathsf{ConfTrans}}$.Transfer)**.**

Participating parties: U_s and C.

Proofs: $\Pi^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Rp}}}$,$\mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Eq}}}$,$\mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{LrgerEq}}}$

Algorithms: Dec.

Common input: $d \in [n]$.

U_s's private input. $x \in \mathbb{N}$.

Operation:

1. U_s:
   (a) $X_d \xleftarrow{\mathrm{R}} \mathsf{Enc}_{\mathsf{pk}_d}(x; r)$ for $r^d \xleftarrow{\mathrm{R}} \mathcal{D}$.
   (b) $\pi^{\mathsf{Rp}} \xleftarrow{\mathrm{R}} \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Rp}}}((\mathsf{pk}_d, X_s, p_{\mathsf{size}}), (x, r))$.
   (c) $X_s \xleftarrow{\mathrm{R}} \mathsf{Enc}_{\mathsf{pk}_s}(x; r)$ for $r^s \xleftarrow{\mathrm{R}} \mathcal{D}$.
   (d) $\pi^{\mathsf{Eq}} \xleftarrow{\mathrm{R}} \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Eq}}}((\mathsf{pk}_s, \mathsf{pk}_d, X_s, X_s), (x, r_s, r_d))$.
   (e) $(b, r^b) \leftarrow \mathsf{Dec}_{sk_s}(\overline{B}_s)$.
   (f) $\pi^{\mathsf{LrgerEq}} \xleftarrow{\mathrm{R}} \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{LrgerEq}}}((\mathsf{pk}_s, X_s, \overline{B}_s), (x, r_s, r_b))$.
   (g) Send $(X_s, X_d, \pi^{\mathsf{Rp}}, \pi^{\mathsf{Eq}}, \pi^{\mathsf{LrgerEq}})$ to C.

2. C:
   (a) Call $\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{Rp}}}((\mathsf{pk}_d, X_s, p_{\mathsf{size}}), \pi^{\mathsf{Rp}})$,
   $\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathsf{Eq}}((\mathsf{pk}_s, \mathsf{pk}_d, X_s, X_s), \pi^{\mathsf{Rp}})$ and $\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{LrgerEq}}}((\mathsf{pk}_s, X_s, \overline{B}_s), \pi^{\mathsf{Rp}})$.
   (b) Verify $\mathsf{tcount}_d \leq p_{\mathsf{pcount}}$.
   (c) Set $U_s \mathrel{-}= X_s$.
   (d) Set $\overline{P}_d \mathrel{+}= X_d$.
   (e) tcount_d^{++}.
   (f) Publish $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{transfer}, s, d, U_s, \overline{P}_d)$.

Rollover.

Protocol 3.6 ($\Pi_{\mathsf{ConfTrans}}$.Rollover)**.**

Participating parties. U_i and C.

Operation: C:

1. $\overline{B}_i \mathrel{+}= \overline{P}_i$.
2. $\overline{P}_i \mathrel{-}= \overline{P}_i$.
3. Set $\mathsf{tcount}_i \leftarrow 0$.
4. $\mathsf{log} \mathrel{+}= (\mathsf{sid}, \mathsf{rollover}, i, \overline{B}_i, \overline{P}_i)$

History.

Protocol 3.7 ($\Pi_{\mathsf{ConfTrans}}$.History)**.**

Participating parties. U_i and C.

Operation: C: send log to U_i.

Audit. [**Iftach's Note: TODO**]

3.2.1 Security of Protocol 3.2

Theorem 3.8 (Security of Protocol 3.2)**.** [**Iftach's Note: TODO**]

4 The Chunk ElGamal Encryption Scheme

In this section we define the (efficient) close to being additive homomorphic encryption scheme based on ElGamal multiplicative homomorphic encryption scheme. [**Iftach's Note: give citations**] The idea is to bootstrap the so-called *ElGamal in-the exponent* additive homomorphic encryption scheme,[1] which in turn is based on the ElGamal multiplicative homomorphic encryption scheme, that lacks efficient decryption algorithm, by splitting the plain text into small "chunks". That is, we present message $m \in Z_t$ as $\sum_{i\in(t/c)} 2^{ic} \cdot a_i$, where c, the chunk size, is some integer that divides $[t]$, and encrypt using additive homomorphic EG each of the a_i. To decry $\overline{A} = (A_0, \ldots, A_{t/c})$, one

1. Decrypt each A_i to get $a_i \cdot G$.
2. Use brute force to find a.[2]
3. Reconstruct a.

In Section 4.1 we formally define the ElGamal in-the-exponent scheme, and a few ZK proofs for the NP-relations the scheme induces. The chunk ElGamal scheme is defined in Section 4.2 and the related ZK proofs, are defined in Section 4.3. Finally, in Section 4.4 we explain how to adjust Protocol 3.2 tp work with the new scheme.

[1] It is called ElGamal "in-the-exponent" due to typical multiplicative group notation. Here use additive group notation, but keep the name for historical reason.

[2] One can use standard processing to speed-up this part from c group operations to $\sqrt{c}$ operations, or even [**Iftach's Note: cite**] to $\sqrt[3]{c}$.

4.1 ElGamal In-the-Exponent Scheme

Throughout we fix a cyclic additive q-size group $\mathcal{G}$ with generator G. The ElGamal in-the-exponent encryption scheme $(\mathsf{EgGen}, \mathsf{EgEnc}, \mathsf{EgDec})$ is define as follows:

Algorithm 4.1 ($(\mathsf{EgGen}, \mathsf{EgEnc}, \mathsf{EgDec})$: ElGamal in-the-exponent encryption)**.**

Key generation: $\mathsf{EgGen}(1^b)$ *samples* $e \overset{R}{\leftarrow} \mathbb{Z}_q$*, and outputs* $((1^b, e), E \leftarrow e \cdot G)$*.*

Encryption: $\mathsf{EgEnc}_E(a)$ *samples* $e \overset{R}{\leftarrow} \mathbb{Z}_q$*, and outputs* $\widetilde{A} \leftarrow (r \cdot G, r \cdot E + a \cdot G)$

Decryption: $\mathsf{EgDec}_{1^b,e}(\widetilde{A})$*,*

1. *Let* $M \leftarrow \widetilde{A}_2 - e \cdot \widetilde{A}_2$*.*
2. *Find (using brute force)* $m \in (-b, b) \in \mathbb{Z}_q$ *so that* $m \cdot G = M$*. Abort if no such* m *exists.*
3. *Output* m*.*

Addition: Addition over $\mathcal{G}^2$*.*[a]

Minus: The inverse in $\mathcal{G}^2$*.*

[a] For $\widetilde{A}, \widetilde{B} \in \mathcal{G}^2$: $\widetilde{A} + \widetilde{B} := (\widetilde{A}_0 + \widetilde{B}_0, \widetilde{A}_1 + \widetilde{B}_1)$.

When clear from the context, will omit the parameter 1^b from the secret key of the scheme.

Theorem 4.2 (Security of Algorithm 4.1)**.** *Assuming DDH is hard over* $\mathcal{G}$*, then Algorithm 4.1 is a perfectly binding, semantically secure additively homomorphic scheme over* $\mathbb{Z}_q$*, with the following caveat: the description only guaranteed to work on encryptions of explain in* (b)*, for* 1^b *being the input of the key generation algorithm:*

4.1.1 Zero-Knowledge Proofs

In Section 4.2 we make use of ZK proofs for the following relations regrading the above scheme.

Knowledge of secret key.

> **Task:** ZKPOK for $\mathcal{R}_{\mathsf{EGKeyGen}} = \{(E, e)) \colon e \cdot G = E\}$.
>
> **Proof:** [**Iftach's Note: Schnorr proof. Give ref**]

Knowledge of plain text.

> **Task:** ZKPOK for $\mathcal{R}_{\mathsf{EgEnc}} = \left\{((E, \widetilde{A}), (a, r)) \colon \mathsf{EgEnc}_E(a; r) = \widetilde{A}\right\}$.
>
> **Proof:** [**Iftach's Note: Schnorr proof. Give ref**]

Equality.

> **Task:** ZKP for $\mathcal{R}_{\mathsf{Eq}} = \left\{((E_0, E_1, \widetilde{A}_0, \widetilde{A}_1), (a, r_0, r_1)) \colon \forall i \in \{0, 1\}\ \mathsf{EgEnc}_{E_i}(a; r_i) = \widetilde{A}_i\right\}$.
>
> **Proof:** [**Iftach's Note: Schnorr proof. Give ref**]

In range.

> **Task:** ZKP for $\mathcal{R}_{\mathsf{EgRp}} = \left\{((E, \widetilde{A}, b), (a, r)) \colon \mathsf{Enc}_E(a; r) = \widetilde{A} \wedge a \in (b)\right\}$.
>
> **Proof:** [**Iftach's Note: Bullet proof. Give ref**]

4.2 The Chunk ElGamal Scheme

In the following we fix $t, c \in \mathbb{N}$ with $t \leq q$ and $\ell \leftarrow t/c \in \mathbb{N}$. The chunk ElGamal encryption scheme is defined as follows:

Definition 4.3 (Base factorization)**.** *For $a \in \mathbb{Z}_q$ let $a_0, \ldots, a_{\ell-1}$ so that $a = \sum_{i\in(\ell)} 2^{ic} \cdot a_i$.*

Algorithm 4.4 (($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$): Chunk ElGamal adaptively homomorphic encryption)**.**

Key generation: $\mathsf{KeyGen}(1^b)$*: act as* $\mathsf{EgGen}(1^b)$.

Encryiption: $\mathsf{Enc}_{\mathsf{pk}}(a)$

1. *Compute* $(a_0, \ldots, a_{\ell-1}) \leftarrow \mathsf{baseF}(a)$.
2. *For each* $i \in (\ell)$*: let* $\widetilde{A}_i \overset{R}{\leftarrow} \mathsf{EgEnc}_{\mathsf{pk}}(a_i)$.
3. *Output* $\overline{A} \leftarrow (\widetilde{A}_0, \ldots, \widetilde{A}_{\ell-1})$.

Decription: $\mathsf{Dec}_{\mathsf{sk}}(\overline{M}, b)$

1. *For each* $i \in (\ell)$*: let* $m_i \overset{R}{\leftarrow} \mathsf{EgDec}_{\mathsf{sk}}(\overline{M}_i, b)$.
2. *Let* $m \leftarrow \sum_{i\in(\ell)} 2^{ic} \cdot m_i$.
3. *Output* m.

Addition: Vector addition.[a]

Minus: Vector negation.

[a] For $\overline{A}, \overline{B} \in (\mathcal{G}^2)^\ell$, $\overline{A} + \overline{B} := (\widetilde{A}_0 + \widetilde{B}_0, \ldots, \widetilde{A}_{\ell-1} + \widetilde{B}_{\ell-1})$.

Theorem 4.5 (Security of Algorithm 4.4)**.** *Assuming* DDH *is hard over* $\mathcal{G}$*, then Algorithm 4.4 is a perfectly binding, semantically secure additively homophobic scheme over* $\mathbb{Z}_q$*, with the following caveat work on encryptions of plaintext* a *so that* $\mathsf{baseF}(a) \in (-b, b)^\ell$ *(*1^b *being the input of the key generation algorithm).*

4.3 Zero-Knowledge Proofs for the Scheme

In this section, we define the ZK and POK proofs used in Section 3. In the following, we omit the parameter b from the input list of Dec. We will address its value in Section 4.4.

Knowledge of secret key.

Task: ZKPOK for $\mathcal{R}_{\mathsf{KeyGen}} = \{(\mathsf{pk}, w)) \colon \mathsf{KeyGen}(w) = (\cdot, \mathsf{pk})\}$.

Proof: Same as $\Pi^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{EGKeyGen}}}$.

Knowledge of plain text.

Task: ZKPOK for $\mathcal{R}_{\mathsf{Enc}} = \{((\mathsf{pk}, A), (a, r)) \colon \mathsf{Enc}_{\mathsf{pk}}(a; r) = A\}$.

P**:** On input $((\mathsf{pk}, \overline{A}), (\overline{a}, \overline{r})$.

1. For each $i \in (\ell)$: let $\pi_i \leftarrow \Pi^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{EgEnc}}}((\mathsf{pk}, \overline{A}_i), (\overline{a}_i, \overline{r}_i))$.
2. Output $\pi \leftarrow (\pi_0, \ldots, \pi_{\ell-1})$.

V**:** On input $((\mathsf{pk}, \overline{A}), \pi = (\pi_0, \ldots, \pi_{\ell-1}))$:

Accept iff $\mathsf{V}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgEnc}}}((\mathsf{pk}, \widetilde{A}_i), \pi_i)$ for all $i \in (\ell)$.

Equality.

Task: ZKP for $\mathcal{R}_{\mathsf{Eq}} = \big\{((\mathsf{pk}_0, \mathsf{pk}_1, \overline{A}_0, \overline{A}_1), (a, r_0, r_1)) \colon \forall i \in \{0, 1\}\ \mathsf{Enc}_{\mathsf{pk}_i}(a; r_i) = \mathrm{A}_i\big\}$.

P**:** On input $((\mathsf{pk}_0, \mathsf{pk}_1, \overline{A}_0, \overline{A}_1), (\overline{a}, \overline{r}_0, \overline{r}_1))$:

1. Let $a \leftarrow \sum_{i \in (\ell)} 2^c \cdot \overline{a}_i$.
2. For both $j \in \{0, 1\}$:
   (a) $\widetilde{A}_j \leftarrow \sum_i 2^c \cdot (\overline{A}_j)_i$.
   (b) $r_j \leftarrow \sum_{i \in (\ell)} 2^c \cdot (\overline{r}_j)_i$.
3. Output $\pi \leftarrow \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Eq}}}((\mathsf{pk}, \widetilde{A}_0, \widetilde{A}_1), (a, r_0, r_1))$.

V**:** On input $((\mathsf{pk}_0, \mathsf{pk}_1, \overline{A}_0, \overline{A}_1), \pi)$:

1. Generate $\widetilde{A}_0$ and $\widetilde{A}_1$ as done by P.
2. Apply $\mathsf{V}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Eq}}}((\mathsf{pk}, \widetilde{A}_0, \widetilde{A}_1), \pi)$.

In range.

Task: ZK for $\mathcal{R}_{\mathsf{Rp}} = \big\{((\mathsf{pk}, \overline{A}, b), (a, r)) \colon \mathsf{Enc}_{\mathsf{pk}}(a; r) = \overline{A} \wedge a \in (b)\big\}$.

P**:** On input $((\mathsf{pk}, \overline{A}, b), (\overline{a}, \overline{r}))$:

1. $a \leftarrow \sum_{i \in (\ell)} 2^c \cdot \overline{a}_i$.
2. $\widetilde{A} \leftarrow \sum_{i \in (\ell)} 2^c \cdot \overline{A}_i$.
3. $r \leftarrow \sum_{i \in (\ell)} 2^c \cdot \overline{r}_i$.
4. Output $\pi \leftarrow \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgRp}}}((\mathsf{pk}, \widetilde{A}, b), (a, r))$.

V**:** On input $((\mathsf{pk}, \overline{A}, b), \pi)$:

1. Generate $\widetilde{A}$ as by P.
2. Output $\mathsf{V}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgRp}}}((\mathsf{pk}, \widetilde{A}, b), \pi)$.

Larger than.

Task: POK for $\mathcal{R}_{\mathsf{LrgerEq}} = \big\{((\mathsf{pk}, \overline{A}_0, \overline{A}_1), (a_0, r_0, a_1, r_1)) \colon \forall i \in \{0, 1\}\ \mathsf{Enc}_{\mathsf{pk}}(a_i; r_i) = \overline{A}_i \wedge a_1 - a_0 \in (q)\big\}$.

P**:** On input $((\mathsf{pk}_0, \mathsf{pk}_1, \overline{A}_0, \overline{A}_1), (\overline{a}, \overline{r}_0, \overline{r}_1))$:

1. For both $j \in \{0,1\}$:
   (a) $a_j \leftarrow \sum_{i\in(\ell)} 2^c \cdot (\overline{a}_j)_i$.
   (b) $\widetilde{A}_j \leftarrow \sum_i 2^c \cdot (\overline{A}_j)_i$.
   (c) $r_j \leftarrow \sum_{i\in(\ell)} 2^c \cdot (\overline{r}_j)_i$.
2. Let $\widetilde{A} \leftarrow \widetilde{A}_1 - \widetilde{A}_0$, $a \leftarrow a_1 - a_0$ and $r \leftarrow r_1 - r_0$.
3. Output $\pi \leftarrow \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgRp}}}((\mathsf{pk}, \widetilde{A}, q), (a, r))$.

V**:** On input $((\mathsf{pk}_0, \mathsf{pk}_1, \overline{A}_0, \overline{A}_1), \pi$:

1. Generate $\widetilde{A}$ as by P.
2. Output $\mathsf{V}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgRp}}}((\mathsf{pk}, \widetilde{A}, q), \pi)$.

Decryptability.

Task: ZKP for $\mathcal{R}_{\mathsf{Dec}} = \big\{((\mathsf{pk}, \overline{A}, b), (\overline{a}, \overline{r}))\colon \forall i \in (\ell)\colon \mathsf{EgEnc}_{\mathsf{pk}}(\overline{a}_i; \overline{r}_i) = \overline{A}_i \wedge \overline{a}_i \in (b)\big\}$.

P**:** On input $((\mathsf{pk}, \overline{A}, b), (\overline{a}, \overline{r})$:

1. For each $i \in (\ell)$: $\pi_i \leftarrow \mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgRp}}}((\mathsf{pk}, \widetilde{A}_i, b), (\overline{a}_i, \overline{r}_i))$.
2. Output $\pi \leftarrow (\pi_0, \ldots, \pi_{\ell-1})$.

V**:** On input $((\mathsf{pk}, \overline{A}, b), \pi = (\pi_0, \ldots, \pi_{\ell-1}))$:
Accept iff $\mathsf{V}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{EgRp}}}((\mathsf{pk}, \widetilde{A}_i, b), \pi_i)$ for all $i \in (\ell)$.

4.4 Adjusting Protocol 3.2

Since the chunk ElGamal scheme has some shortcoming to be considered as truly additive homomorphic shame, see **??**, instantiating Protocol 3.2 with the new scheme requires some adjustments.

Init: Set the parameter of the encryption key generation algorithm to $b \leftarrow 2^c \cdot p_{\mathsf{pcount}}$.

Transfer. The sender also provide proofs that X_d is decryptable (i.e., using $\mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Dec}}}$ with parameter $b \leftarrow 2^c$).

Rollover: The rollover over operation should be updated to allow the account holder to "normalize" it active balance: to make it decryptable. Specifically

(a) U_i:
   i. Decrypt $\overline{P}_i$ and $\overline{B}_i$ to get value (p_i, r_i) and (b_i, w_i) respectively.
   ii. Generate a fresh encryption $\overline{B}'_i$ of $(p_i + b_i)$ and $\overline{P}'_i$ of 0.
   iii. Generate a proof π (i.e., using $\mathsf{P}^{\mathsf{ZK}}_{\mathcal{R}_{\mathsf{Eq}}}$) that $\overline{P}_i + \overline{B}_i = \overline{P}'_i + \overline{B}'_i$.
   iv. Send $(\overline{P}'_i, \overline{B}'_i, \pi)$ to C.

(b) C:
   i. Verify π.
   ii. Set $\overline{P}_i \leftarrow \overline{P}'_i$ and $\overline{B}_i \leftarrow \overline{B}'_i$.
   iii. Continue as in the original protocol.