

Confidential Transactions

Theory Justification

Iftach Haitner*

August 5, 2025

Abstract

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

Contents

1	Introduction	2
2	Preliminaries	2
2.1	Notation	2
2.2	Homomorphic Encryption	2
3	The Confidential Transaction Protocols	2
3.1	The Ideal Functionality	2
3.2	The Protocol	3
3.2.1	Security of Protocol 3.2	6
4	The ElGammal-Based Additive Homomorphic Encryption	6
4.1	ElGammal In-the-Exponent Scheme	6
4.2	The Scheme	7
4.3	Proofs	7
4.4	Adjusting Section 3.2	7

*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 $[n] := \{1, \dots, n\}$ and $(n) := \{0, \dots, n\}$. For a relation \mathcal{R} , let $\mathcal{L}(\mathcal{R})$ denote its underlying language, i.e., $\mathcal{L}(\mathcal{R}) := \{x : \exists w : (x, w) \in \mathcal{R}\}$.

2.2 Homomorphic Encryption

An homomorphic encryption is a triplet $(\text{KeyGen}, \text{Enc}, \text{Dec})$ of efficient algorithms, with the standard correctness and semantic security properties. In addition, there is addition operation denote $+$ over any two (valid) ciphertexts such that for any validly generated public key pk and valid ciphertexts x_0, x_1 , it holds that $\text{Enc}_{sk}(x_0) + \text{Enc}_{pk}(x_1) \in \text{Supp}(\text{Enc}_{sk}(x_0 + x_1 \bmod q))$, where $q \in \mathbb{N}$ is efficiently determined by pk .

[Iftach's Note: Do we really need the homomorphic properties or only for the proofs?]

3 The Confidential Transaction Protocols

3.1 The Ideal Functionality

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

Parties: Issuer I , chain holder C and users U_1, \dots, U_n .

Parameters: $p_{\text{num}}, p_{\text{size}} \in \mathbb{N}$.

Init. Upon receiving `init` from all parties:

1. For each $i \in [n]$: $\text{avlBalance}_i, \text{pndBalance}_i \leftarrow 0$, $\text{transnum}_i \leftarrow 0$, and $\text{log}_i \leftarrow \emptyset$.
2. $\text{log} \leftarrow \emptyset$.

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

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

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

1. Assert($x \in (p_{\text{num}})$, $\text{transnum} \leq p_{\text{size}}$, $\text{avlBalance}_s \geq x$ and $d \in [n]$).
2. transnum^{++} .
3. $\text{avlBalance}_s -= x$.
4. $\text{pndBalance}_d \cup = x$.
5. Set $\log_d \cup = (\text{sid}, \text{transfer}, s, x)$
6. Set $\log \cup = (\text{sid}, \text{transfer}, s, d)$

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

1. $\text{transnum} \leftarrow 0$.
2. Set $\text{avlBalance}_i += \text{pndBalance}_i$.
3. Set $\text{pndBalance}_i \leftarrow 0$.
4. Set $\log \cup = (\text{sid}, \text{rollover}, i)$

Withdraw. Upon receiving $(\text{sid}, \text{withdraw}, x)$ from party U_i and C, party C

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

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

Send (\log, \log_i) to P_i .

Audit. [Iftach's Note: Later]

3.2 The Protocol

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

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

Parties: Issuer I, chain-holder C and users U_1, \dots, U_n .

Subprotocols: See below.

Init. We use the of key-generation relation

$$\mathcal{R}_{\text{KeyGen}} = \{(pk, w) : \text{KeyGen}(w) = (\cdot, pk)\}.$$

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

Participating parties. All parties.

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

Algorithms: **KeyGen**.

Operation:

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

Issue.

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

Participating parties. I and C .

C 's input. sid , $x \in \mathbb{N}$ and $i \in [n]$.

Operation:

1. I : Send (x, i) to C .
2. C :
 - (a) Assert($x \in [p_{\text{size}}]$ and $\bar{P}_i \leq p_{\text{size}}$)
 - (b) Set $\bar{P}_i += \text{Enc}_{pk_i}(x)$.
 - (c) Publish $\log \cup = (\text{sid}, \text{issue}, x, i, \bar{P}_i)$.

Transfer. We use proofs for the following relations

$$\mathcal{R}_{\text{rp}} = \{((pk, A), (a, r)) : \text{Enc}_{pk}(a; r) = A \wedge a \in [p_{\text{size}}]\}$$

I.e., encryption of values in $[p_{\text{size}}]$.

$$\mathcal{R}_{\text{eq}} = \{((pk_0, pk_1, A_0, A_1), (a, r_0, r_1)) : \forall i \in \{0, 1\} \text{Enc}_{pk_i}(a; r_i) = A_i\}$$

I.e., encryptions of the same pair under different public keys.

$$\mathcal{R}_{\text{lrger}} = \{((pk, A_0, A_1), (a_0, r_0, a_1, r_1)) : \forall i \in \{0, 1\} \text{Enc}_{pk}(a_i; r_i) = A_i \wedge a_1 - a_0 \in [q]\}$$

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 ($\Pi_{\text{ConfTrans}}.\text{Transfer}$).

Participating parties: P_s and C .

Proofs: $\Pi_{\mathcal{R}_{rp}}^{\text{ZK-POK}}, \mathbf{p}_{\mathcal{R}_{eq}}^{\text{ZK-POK}}, \mathbf{p}_{\mathcal{R}_{lrger}}^{\text{ZK-POK}}$

Algorithms: Dec.

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

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

Operation:

1. P_s :

- (a) $X_d \xleftarrow{R} \text{Enc}_{pk_d}(x; r)$ for $r^d \xleftarrow{R} \mathcal{D}$.
- (b) $\pi^{rp} \xleftarrow{R} \mathbf{p}_{\mathcal{R}_{rp}}^{\text{ZK-POK}}((pk_d, X_s, p_{\text{size}}), (x, r))$.
- (c) $X_s \xleftarrow{R} \text{Enc}_{pk_s}(x; r)$ for $r^s \xleftarrow{R} \mathcal{D}$.
- (d) $\pi^{eq} \xleftarrow{R} \mathbf{p}_{\mathcal{R}_{eq}}^{\text{ZK-POK}}((pk_s, pk_d, X_s, X_s), (x, r_s, r_d))$.
- (e) $(b, r^b) \leftarrow \text{Dec}_{sk_s}(\overline{B}_s)$.
- (f) $\pi^{\text{lrger}} \xleftarrow{R} \mathbf{p}_{\mathcal{R}_{lrger}}^{\text{ZK-POK}}((pk_s, X_s, \overline{B}_s), (x, r_s, r_b))$.
- (g) Send $(X_s, X_d, \pi^{rp}, \pi^{eq}, \pi^{\text{lrger}})$ to C .

2. C :

- (a) Call $\mathbf{V}_{\mathcal{R}_{rp}}^{\text{ZK-POK}}((pk_d, X_s, p_{\text{size}}), \pi^{rp})$,
 $\mathbf{V}_{\text{eq}}^{\text{ZK-POK}}((pk_s, pk_d, X_s, X_s), \pi^{eq})$ and $\mathbf{V}_{\mathcal{R}_{lrger}}^{\text{ZK-POK}}((pk_s, X_s, \overline{B}_s), \pi^{\text{lrger}})$.
- (b) Set $U_s \leftarrow X_s$.
- (c) Set $\overline{P}_d \leftarrow X_d$.
- (d) Publish $\log \cup = (\text{sid}, \text{transfer}, s, d, U_s, \overline{P}_d)$.

Rollover.

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

Participating parties. P_i and C .

Operation:

C :

1. $\bar{B}_i += oP_i.$
2. $\bar{P}_i -= \bar{P}_i.$
3. $\log += (\text{sid}, \text{rollover}, i, \bar{B}_i, \bar{P}_i)$

History.

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

Participating parties. P_i and C .

Operation: C : send \log to P_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 ElGammal-Based Additive Homomorphic Encryption

In this section we define the (efficient) additive homomorphic encryption scheme based on ElGammal multiplicative homomorphic encryption scheme. [Iftach's Note: give citations]

The idea is to bootstrap the so-called *ElGammal in-the exponent* additive homomorphic encryption scheme,¹ which in turn is based on the ElGammal 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

$\sum_{i \in (t/c)} m_i \cdot 2^{ic}$, where c , the chunk size, is some integer that divides $[t]$, and encrypt using additive homomorphic EG each of the m_i . To decrypt $\bar{M} = (M_0, \dots, M_{t/c})$, one

1. Decrypt each M_i to get $m_i \cdot G$.
2. Use brute force to find m .²
3. Reconstruct m .

4.1 ElGammal In-the-Exponent Scheme

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

Key generation: EgGen() samples $e \xleftarrow{R} \mathbb{Z}_q$, and outputs $(e, e \cdot G)$.

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

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

Encryption: $\text{EgEnc}_E(m)$ samples $e \xleftarrow{\mathbb{R}} \mathbb{Z}_q$, and outputs $\widetilde{M} \leftarrow (r \cdot G, r \cdot E + m \cdot G)$

Decryption: $\text{EgDec}_e(\widetilde{M})$,

1. Let $M \leftarrow \widetilde{M}_2 - e \cdot \widetilde{M}_1$.
2. Find (using brute force) m so that $m \cdot G = M$.
3. Output m .

4.2 The Scheme

In the following we fix $t, c \in \mathbb{N}$ with $t \leq q$ and $\ell \leftarrow t/c \in \mathbb{N}$. The encryption scheme ($\text{KeyGen}, \text{Enc}, \text{Dec}$) is defined as follows:

Key generation: $\text{KeyGen}()$

1. For each $i \in (\ell)$: sample $(e_i, E_i) \xleftarrow{\mathbb{R}} \text{EgGen}()$.
2. Output $(\bar{e} \leftarrow (e_0, \dots, e_{\ell-1}), \bar{E} \leftarrow (E_0, \dots, E_{\ell-1}))$

Encryption: $\text{Enc}_{\bar{E}}(m)$

1. Compute $m_0, \dots, m_{\ell-1}$ do that $m = \sum_{i \in (\ell)} m_i \cdot 2^{ic}$.
2. For each $i \in (\ell)$: let $\widetilde{M}_i \xleftarrow{\mathbb{R}} \text{EgEnc}_{E_i}(m_i)$.
3. Output $\bar{M} \leftarrow (\widetilde{M}_0, \dots, \widetilde{M}_{\ell-1})$.

Decryption: $\text{EgDec}_{\bar{e}}(\bar{M})$

1. For each $i \in (\ell)$: let $m_i \xleftarrow{\mathbb{R}} \text{EgDec}_{E_i}(\widetilde{M}_i)$.
2. Let $m \leftarrow \sum_{i \in (\ell)} m_i \cdot 2^{ic}$.
3. Output m .

4.3 Proofs

Knowledge of secret key. Proof $\Pi_{\mathcal{R}_{\text{KeyGen}}}^{\text{ZK-POK}}$.

For each of the ℓ public keys, apply the (standard) ElGamal POK for the secret key relation.

Knowledge of plain text. Protocol $\Pi_{\mathcal{R}_{\text{KeyGen}}}^{\text{ZK-POK}}$.

Proof of knowledge for the ciphertexts relation

$$\mathcal{R}_{\text{enc}} = \{((pk, A), (a, r)) : \text{Enc}_{pk}(a; r) = A\}.$$

On plaintext \bar{M} , for each \bar{M}_i :

1. Apply the standard ElGamal POK for the ciphertexts relation.
2. Apply range proof, i.e., $\Pi_{\mathcal{R}_{\text{rp}}}^{\text{ZK-POK}}$, to show that the encrypted plaintext is in $(2^c \cdot p_{\text{size}})$

4.4 Adjusting Section 3.2