Confidential Transactions
Theory Justification

Iftach Haitner*

August 5, 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 is a triplet $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ 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 $\mathsf{Enc}_{sk}(x_0) + \mathsf{Enc}_{pk}(x_1) \in \mathrm{Supp}(\mathsf{Enc}_{sk}(x_0 + x_1 \bmod q)$, 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 ($\mathcal{F}_{\mathsf{ConfTrans}}$: Confidential transactions)**.**

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

Init. Upon receiving init from all parties:

1. For each $i \in [n]$: $\mathsf{avlBlance}_i, \mathsf{pndBalance}_i \leftarrow 0$ 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($x \in \mathbb{N}$ and $d \in [n]$).
2. $\mathsf{pndBalance}_d \mathrel{+}= x$.
3. Set $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{issue}, x, d)$.

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

1. Assert($x \in \mathbb{N}$, $\mathsf{avlBlance}_s \geq x$ and $s, d \in [n]$).
2. $\mathsf{avlBlance}_s \mathrel{-}= x$.

3. $\mathsf{pndBalance}_d \cup= x$.
4. Set $\mathsf{log}_d \cup= (\mathsf{sid}, \mathsf{transfer}, s, x)$
5. 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. Set $\mathsf{avlBlance}_i \mathrel{+}= \mathsf{pndBalance}_i$.
2. Set $\mathsf{pndBalance}_i \leftarrow 0$.
3. 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 P_i and C:

Send $(\mathsf{log}, \mathsf{log}_i)$ to P_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 $\overline{A} = \mathsf{Enc}(a; r)$.

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

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

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

Subprotocols: See below.

We use the of key-generation relation

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

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. P_i, for all $i \in [n]$:

   (a) Set $(pk_i, sk_i) \stackrel{\mathrm{R}}{\leftarrow} \mathsf{KeyGen}(r_i)$ for $r_i \stackrel{\mathrm{R}}{\leftarrow} \mathcal{D}$.
   (b) Store sk_i.
   (c) Let $\pi_i \stackrel{\mathrm{R}}{\leftarrow} \mathsf{P}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{KeyGen}}}(pk_i, r_i)$.
   (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 $\{\overline{P}_i \leftarrow 0, \overline{P}_i \stackrel{\mathrm{R}}{\leftarrow} \mathsf{Enc}_{pk_i}(0), \overline{B}_i \leftarrow \emptyset\}_{i\in[n]}$.
   (b) Broadcast $\mathsf{log} \leftarrow \emptyset$.

Protocol 3.4 ($\Pi_{\mathsf{ConfTrans}}$.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($x \in [p_{\mathsf{size}}]$ and $\overline{P}_i \le p_{\mathsf{num}}$)
   (b) Set $\overline{P}_i \mathrel{+}= \mathsf{Enc}_{pk_i}(x)$).
   (c) Publish $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{issue}, x, i, \overline{P}_i)$.

We use proofs for the following relations

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

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

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

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

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

I.e., encryptions of the pair of values (a_0, a_1), under the same public key, with $a_1 \ge a_0$.

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

Participating parties: P_s and C.

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

Algorithms: Dec.

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

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

Operation:

1. P_s:
   (a) $X_d \xleftarrow{\text{R}} \mathsf{Enc}_{pk_d}(x; r)$ for $r^d \xleftarrow{\text{R}} \mathcal{D}$.
   (b) $\pi^{\mathsf{rp}} \xleftarrow{\text{R}} \mathsf{P}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{rp}}}((pk_d, X_s, p_{\mathsf{size}}), (x, r))$.
   (c) $X_s \xleftarrow{\text{R}} \mathsf{Enc}_{pk_s}(x; r)$ for $r^s \xleftarrow{\text{R}} \mathcal{D}$.
   (d) $\pi^{\mathsf{eq}} \xleftarrow{\text{R}} \mathsf{P}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{eq}}}((pk_s, pk_d, X_s, X_s), (x, r_s, r_d))$.
   (e) $(b, r^b) \leftarrow \mathsf{Dec}_{sk_s}(\overline{B}_s)$.
   (f) $\pi^{\mathsf{lrger}} \xleftarrow{\text{R}} \mathsf{P}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{lrger}}}((pk_s, X_s, \overline{B}_s), (x, r_s, r_b))$.
   (g) Send $(X_s, X_d, \pi^{\mathsf{rp}}, \pi^{\mathsf{eq}}, \pi^{\mathsf{lrger}})$ to C.
2. C:
   (a) Call $\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{rp}}}((pk_d, X_s, p_{\mathsf{size}}), \pi^{\mathsf{rp}})$,
   $\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathsf{eq}}((pk_s, pk_d, X_s, X_s), \pi^{\mathsf{rp}})$ and $\mathsf{V}^{\mathsf{ZK\text{-}POK}}_{\mathcal{R}_{\mathsf{lrger}}}((pk_s, X_s, \overline{B}_s), \pi^{\mathsf{rp}})$.
   (b) Set $U_s \mathrel{-}= X_s$.
   (c) Set $\overline{P}_d \mathrel{+}= X_d$.
   (d) Publish $\mathsf{log} \cup= (\mathsf{sid}, \mathsf{transfer}, s, d, U_s, \overline{P}_d)$.

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

Participating parties. P_i and C.

Operation:

C:

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

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

Participating parties. P_i and C.

Operation: C: send log to P_i.

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 decry $\overline{M} = (M_0, \ldots, M_{t/c})$, 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 $(\mathsf{EgGen}, \mathsf{EgEnc}, \mathsf{EgDec})$ is define as follows:

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

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

Decription: $\mathsf{EgDec}_e(\widetilde{M})$,

1. Let $M \leftarrow \widetilde{M}_2 - e \cdot \widetilde{M}_2$.
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 $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

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

[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 $\sqrt[3]{c}$.

1. For each $i \in (\ell)$: sample $(e_i, E_i) \xleftarrow{\text{R}} \mathsf{EgGen}()$.
2. Output $(\overline{e} \leftarrow (e_0, \ldots, e_{\ell-1}), \overline{E} \leftarrow (E_0, \ldots, R_{\ell-1})$

Encryiption: $\mathsf{Enc}_{\overline{E}}(m)$

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

Decription: $\mathsf{EgDec}_{\overline{e}}(\overline{M})$

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

4.3 Normalization

4.4 Proofs

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