

Confidential Transactions

Theory Justification

Iftach Haitner*

July 23, 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

*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

[Iftach's Note:

1. Rollover
 2. Pending and available balances
 3. Destroy
-]

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

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

Init. Upon receiving `init` from all parties:

1. For each $i \in [n]$: $\text{ActBalance}_i, \text{PndBalance}_i \leftarrow 0$ and $\log_i \leftarrow \emptyset$.
2. $\log \leftarrow \emptyset$.

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

1. $\text{Assert}(x \in \mathbb{N} \text{ and } d \in [n])$.

2. $\text{PndBalance}_d += x$.
3. Set $\log \cup = (\text{sid}, \text{issue}, x, d)$.

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

1. Assert($x \in \mathbb{N}$, $\text{ActBalance}_s \geq x$ and $s, d \in [n]$).
2. $\text{ActBalance}_s -= x$.
3. $\text{PndBalance}_d \cup = x$.
4. Set $\log_d \cup = (\text{sid}, \text{transfer}, s, x)$
5. Set $\log \cup = (\text{sid}, \text{transfer}, s, d)$

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

1. Set $\text{ActBalance}_i += \text{PndBalance}_i$.
2. Set $\text{PndBalance}_i \leftarrow 0$.
3. Set $\log \cup = (\text{sid}, \text{rollover}, i)$

Destroy. Upon receiving $(\text{sid}, \text{destroy}, x)$ from party \mathcal{U}_i and \mathcal{C} , party \mathcal{C}

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

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

Send (\log, \log_i) to \mathcal{P}_i .

[Iftach's Note: TODO

1. Should the receiver be part of the call in which it gets money.

2. Auditor?

]

3.2 The Protocol

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

Parties: Issuer \mathcal{I} , Chain \mathcal{C} and users $\mathcal{U}_1, \dots, \mathcal{U}_n$.

Parameters: $1^{\kappa_{\mathcal{C}}}$.

Subprotocols: See below.

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

Participating parties. All parties.

Operation:

1. P_i , for all $i \in [n]$,
 - (a) Set $(pk_i, sk_i) \xleftarrow{R} \text{KeyGen}(1^{\kappa_c})$.
 - (b) Store sk_i .
 - (c) Send pk_i to all parties.
2. All parties store $\{pk_i\}_{i \in [n]}$.
3. C:
 - (a) For all $i \in [n]$: Set $B_i \xleftarrow{R} \text{Enc}_{pk_i}(0)$ and $H_i \leftarrow \emptyset$.
 - (b) Set $\log \leftarrow \emptyset$.

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

Participating parties. I and C.

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

Operation:

1. I: Send (x, i) to C.
2. C: Set $H_i \cup = \{\text{sid}, \text{issue}, (\text{Enc}_{pk_i}(x))\}$.
3. C: Set $\log \cup = (\text{sid}, \text{issue}, x, i)$.

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

Participating parties. P_s and C.

Proof's systems: $\Pi^{\text{pos}}, \Pi^{\text{lrg}}$

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

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

Operation:

1. P_s :
 - (a) $X \xleftarrow{R} \text{Enc}_{pk_d}(x; r)$ for $r \xleftarrow{R} \{0, 1\}^{\kappa_c}$.
 - (b) $\pi^{\text{pos}} \xleftarrow{R} \text{Plrg}((pk_d, X), (x, r))$.
 - (c) $\pi^{\text{lrg}} \xleftarrow{R} \text{Plrg}((pk_s, pk_d, B_i, X), (sk_s, x, r))$.
 - (d) Send $(X, \pi^{\text{pos}}, \pi^{\text{lrg}})$ to C.
2. C:

- (a) $V^{\text{pos}}(pk_d, X)$.
- (b) $V^{\text{lrg}}(pk_s, pk_d, B_i, X)$.
- (c) Set $H_d \cup = (\text{sid}, s, X)$.
- (d) Set $\log \cup = (\text{sid}, \text{transfer}, s, d)$.

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

Participating parties. P_i and C .

Operation: C

1. $B_i += \sum_{(\cdot, X) \in H_i} X$.
2. $H_i \leftarrow \emptyset$.
3. $\log += (\text{sid}, \text{rollover}, i)$

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

Participating parties. P_i and C .

Operation: C sends (\log, H_i) to P_i .