Astraea
Anonymous and Secure Auditing Based on Private Smart Contracts for Donation Systems

Li, Meng; Chen, Yifei; Zhu, Liehaung; Zhang, Zijian; Ni, Jianbing; Lal, Chhagan; Conti, Mauro

DOI
10.1109/TDSC.2022.3204287

Publication date
2023

Document Version
Final published version

Published in
IEEE Transactions on Dependable and Secure Computing

Astraea: Anonymous and Secure Auditing Based on Private Smart Contracts for Donation Systems

Meng Li, *Senior Member, IEEE*, Yifei Chen, Liehuang Zhu, *Senior Member, IEEE*, Zijian Zhang, Jianbing Ni, *Senior Member, IEEE*, Chhagan Lal, *Member, IEEE*, and Mauro Conti, *Fellow, IEEE*

Abstract—Many regions are in urgent need of facial masks for slowing down the spread of COVID-19. To fight the pandemic, people are contributing masks through donation systems. Most existing systems are built on a centralized architecture which is prone to the single point of failure and lack of transparency. Blockchain-based solutions neglect fundamental privacy concerns (*donation privacy*) and security attacks (*collusion attack*, *stealing attack*). Moreover, current auditing solutions are not designed to achieve donation privacy, thus not appropriate in our context. In this work, we design a decentralized, anonymous, and secure auditing framework *Astraea* based on private smart contracts for donation systems. Specifically, we integrate a Distribute Smart Contract (DiSC) with an SGX Enclave to distribute donations, prove the integrity of donation number (intention) and donation sum while preserving donation privacy. With DiSC, we design a Donation Smart Contract to refund deposits and defend against the stealing attack the collusion attack from malicious collector and transponder. We formally define and prove the privacy and security of Astraea by using security reduction. We build a prototype of Astraea to conduct extensive performance analysis. Experimental results demonstrate that Astraea is practically efficient in terms of both computation and communication.

Index Terms—Donation systems, auditing, donation privacy, security, private smart contract, commitment

1 Introduction

The COVID-19 pandemic has caused more than 5.70 million deaths so far according to WHO statistics [1]. It strikes the health care systems of many countries overwhelmingly. One serious consequence is the short supply of facial masks [2], which is crucial to slow the spread of COVID-19 [3].

To fight the pandemic, people are voluntarily donating facial masks to the health care systems, hoping their contributions will be distributed to points of highest need. In this work, we focus on the underlying donation systems. Each donation system has a donation chain as depicted in Fig. 1. For instance, multiple institutions typically act as collectors or intermediaries [6]; when a donator provides an asset or donation, a collector will receive the asset and distribute it to a donee. Besides, some logistics companies are responsible for delivering assets between the three parties.

Existing donation systems include Eleo [7], DonorSnap [8], and easyTithe [9]. They are centralized donor management platforms based on cloud computing, which makes them inherently suffer from a single point of failure and lack of transparency [10]. To mitigate these effects, some blockchain-based solutions are proposed [11], [12], [13]. They are proof-of-delivery solution of shipped physical items based on Ethereum [11], blockchain-based proof-of-delivery of physical assets with multiple transporters [12], and donation tracking system using blockchain [13]. However, these solutions neglect some fundamental privacy concerns and security attacks.

- Privacy concerns. *Donation privacy* [14], [15], [16], [17], which is an important feature of donation

- *Meng Li is with the Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education; School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui 230601, China, and with the Anhui Province Key Laboratory of Industry Safety and Emergency Technology, Hefei, Anhui 230601, China, and also with the Intelligent Interconnected Systems Laboratory of Anhui Province (Hefei University of Technology), Hefei, Anhui 230601, China. E-mail: mengli@hfut.edu.cn.*
- *Yifei Chen is with Solution and Architecture Research Department, NSFOCUS, Milpitas, CA 95035 USA. E-mail: chenyifei@nsfocus.com.*
- *Liehuang Zhu is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: liehuangz@bit.edu.cn.*
- *Zijian Zhang is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Southeast Institute of Information Technology, Beijing Institute of Technology, Putian, Fujian 351100, China. E-mail: zhangzijian@bit.edu.cn.*
- *Jianbing Ni is with the Department of Electrical and Computing Engineering, Queen's University, Kingston, ON K7L 3N6, Canada. E-mail: jianbing.ni@queensu.ca.*
- *Chhagan Lal is with the Department of Intelligent Systems, CyberSecurity Group, TU Delft, 2628 Delft, CD, Netherlands. E-mail: c.lal@tudelft.nl.*
- *Mauro Conti is with the Department of Mathematics and HIT Center, University of Padua, 35131 Padua, Italy, and also with the Department of Intelligent Systems, CyberSecurity Group, TU Delft, 2628 Delft, CD, Netherlands. E-mail: conti@math.unipd.it.*

Manuscript received 12 July 2022; accepted 1 September 2022. Date of publication 5 September 2022; date of current version 11 July 2023.
This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62002094, in part by the Anhui Provincial Natural Science Foundation under Grant 2008085MF196, in part by the National Key Research and Development Program of China under Grant 2021YFB2701200, in part by the National Natural Science Foundation of China (NSFC) under Grants 62172040, U1836212, and 61872041, and in part by EU LOCARD Project under Grant H2020-SU-SEC-2018-832735.
(Corresponding authors: Liehuang Zhu and Zijian Zhang.)
Digital Object Identifier no. 10.1109/TDSC.2022.3204287

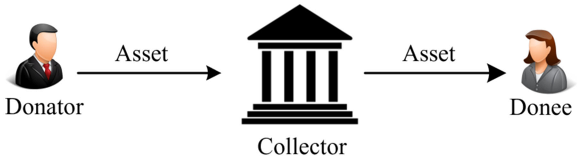


Fig. 1. Traditional system model of donation systems.

systems. It involves the protection of 1) Number: number of masks in a donation, 2) Sum: total number of masks processed by a collector, and 3) Donator/donee: identities of donators and donees. (Detailed explanations are given in Section 8.2.)

- Security attacks. (1) Collusion attack, i.e., intermediate entities on a donation chain colludes to acquire the number, sum, and identity of donator/donee. (2) Stealing attack, i.e., intermediate entities steal assets.

Such privacy concerns [4] and security attacks mainly arise from the unreliability of collectors, such as Red Cross and commonwealth organizations. It is reported that Red Cross moved thousands of masks to a "more secure location" which should be distributed to fight the COVID-19 [18]. In the US, scammers have held millions of masks without ever delivering them to individuals [19]. Therefore, there is a strong need for anonymous and secure auditing carefully designed for the donation systems. Although existing work [6], [20] provide privacy-preserving auditing, they are not designed to cover donation privacy, so it is natural that they are not entirely applicable for donation systems with unique features.

Our motivations come from a closer look into blockchain, smart contracts (SCs), and Intel Software Guard eXtensions (SGX) from the perspective of system architecture. Blockchains [21], [22], [23], [24] are gaining significant popularity as verifiable and reliable records of transactions. SC [25], [26], [27] allows automatic execution of predefined codes on top of a blockchain. Blockchain along with SC can solve the single point of failure and achieve autonomous execution of functions. However, they ensure verifiability and reliability by making all transactions and data public, resulting in privacy concerns that hinder a wide adoption of potential uses. SGX [28], [29], [30] opens a new way of supporting data confidentiality and integrity by performing codes in an isolated space. This introduces the fundamental *problem*: how to eliminate the privacy concerns and defend against the collusion attacks? To fully realize this framework, there are four technical challenges. *Technical challenge 1*: from the privacy and auditing perspective, how to protect the donation privacy (identity, number, and sum) while achieving auditing in an open donation system based on a blockchain? For donators and donees who have different positions and requirements in a donation system, we need to consider different anonymization strategies from existing ones. Especially, in non-blockchain-based donation systems, privacy can be guaranteed if we secure the communication channels. However, we adopt the blockchain to channel donation systems with unforgeability and verifiability, which makes the transmitted messages transparent [31]. Intuitively, encryption will protect the number and sum of donations. However, we need to audit them after an untrustworthy collector distributes the donations, it is challenging to generate auditing proofs while not undermining the confidentiality of number and sum. *Technical challenge 2*: from the security and auditing perspective, how to defend against collusion attacks and stealing attacks in an anonymous donation system? Under a non-colluding and honest-but-curious model, it is not difficult to achieve auditing. However, in our target anonymous donation systems, there will be inside adversaries who steal the donations and incur financial and social consequences. This requires the auditing to consider the gap between the digital world and the physical world. *Technical challenge 3*: from the formalization perspective, how to formally define and prove the privacy and security of an anonymous and secure donation system? Such a system is also a protocol including entities and functions. Different from cryptographic protocols, it needs adjustments when we are treating privacy and security. *Technical challenge 4*: from the performance and auditing perspective, how to build an anonymous and secure donation system while maintaining acceptable costs? We aim to provide several privacy and security objectives by using cryptographic techniques and trusted hardware. Instead of arbitrarily stuffing the system with such techniques, we should integrate them reasonably to keep the system up and running. Furthermore, we should consider cost control by adjusting parameters and optimizing data structures without sacrificing privacy or security.

To address these challenges, we present Astraea, a decentralized, anonymous, and secure auditing scheme for donation systems. First, a donator holding several accounts sends a donation to a community of donees. Each collector publishes a Distribute Smart Contract (DiSC) that explicitly clarifies the distribution logic. We propose a new building block for decentralized, anonymous, and secure donation auditing, i.e., Private Smart Contract (PSC) by integrating the DiSC into an SGX Enclave. We design an efficient Proof-of-Distribution (PoD) to protect the numbers while providing designated verifiability. We design an efficient Proof-of-Asset (PoA) based on Pedersen commitments to protect the sum while achieving public verifiability. Second, the collector publishes a normal Donation Smart Contract (DoSC). Two transponders and the collector in each donation put down a deposit [34], [35], [36] on the DoSC to track a donation. Each donation has a complete chain for the DoSC and DiSC to refund deposits and defend against the stealing attack. Consequently, the DiSC protects the link between donator and donee in both donation distribution and deposit refunding, thwarting collusion attack. Third, we carefully define the privacy and security of Astraea by using cryptographic games. Our contributions are as follows.

- Our work on Astraea represents the first formal treatment of privacy and security on distributed donation systems - a new architecture that aligns with the process of current donation systems. Astraea provides a formal privacy and security model that captures the requirements of collectors when migrating donations onto public ledgers.
- To protect donation privacy and prevent security attacks, we design an auditing framework Astraea based on PSCs to automatically, privately, and securely distribute donations. We design PoA and

PoD to enforce donation privacy while achieving verifiability.

- We give formal analysis of privacy and security. We report on our prototype implementation of Astraea based on Ethereum and SGX to present experimental results, demonstrating its feasibility and efficiency. We also provide a performance comparison with existing work.

The remainder of this paper is organized as below. We review related work in Section 2. We introduce our system architecture, threat model, and design objectives in Section 3. We review preliminaries in Section 4. We present the proposed Astraea in Section 5. In Section 6, we formally analyze the privacy and security of Astraea. In Section 7, we show the implementation of Astraea, evaluate its performance, and compare with existing work. Finally, we give discussions in Section 8 and conclude our work in Section 9.

2 Related Work

In this section, we review proof-of-delivery, two blockchain-based auditing frameworks, and private SC.

2.1 Proof-of-Delivery

Hasan et al. [11] proposed a blockchain based proof-of-delivery scheme of physical items that leverages Ethereum SCs to track items in a decentralized manner with integrity, reliability, and immutability. The physical items are delivered from a seller to a buyer by a transporter. A trusted arbitrator intervenes when a dispute happens. Each of the participants has a public Ethereum address and agrees on all the conditions of a contract. An SC attestation authority (SCAA) attests the SC to guarantee that the SC follows the signed conditions. However, the existence of a SCAA contradicts with the decentralization of blockchain.

Hasan et al. [12] proposed another proof-of-delivery scheme to track the delivery of a shipped items regardless of the number of intermediate transporters. When an item is delivered between two participants, a chain of contracts is created according to the number of transporters. They require all participants to act honestly by using a double deposit collateral and achieve automated payment to ensure that each participant gets a fair reward upon successful delivery.

Singh et al. [13] claimed that charities are not highly transparent. They proposed a blockchain-based solution to solve this problem by eliminating the presence of third parties between donors and charities. The proposed solution is implemented upon Ethereum and tested to track donation process. All donations are made using SCs such that donors know when and how their donations will be received.

2.2 Blockchain-Based Auditing

Solidus [6] is a distributed ledger system that hides transaction values and the transaction graph amount between clients (banks) while keeping verifiability. Specifically, identifying an edge in the transaction graph is equal to identifying which account's balance changes with a transaction. Hence, putting all balances in an Oblivious RAM (ORAM) offers transaction graph confidentiality. In Solidus, clients have to prove an application-defined notion of correctness for each update. To do so, authors propose a Publicly-Verifiable Oblivious RAM Machine (PVORM) that defines legal application-specific operations and they use Generalized Schnorr Proofs (GSPs) to guarantee that account balances and PVORMs are properly updated and remain valid. Although Solidus provides private transactions, it only achieves auditing by disclosing all the keys to an auditor and opening transactions.

zkLedger [20] is a distributed auditing system among banks to support strong transaction privacy, public verifiability, and practical, useful auditing. Specifically, it provides fast and rich auditing by using Schnorr-type non-interactive zero-knowledge proofs. In addition, it uses a columnar ledger to prevent banks from hiding transactions from the auditor, and banks can use rolling caches to produce and verify answers. However, a transaction in zkLedger contains an entry for every Bank so that each non-participating bank has to commit to 0 and create three proofs, leading to extra computational costs and communication overhead.

The promotion over existing works in this paper is that Astraea provides strong donation privacy in a decentralized framework while defending against two security attacks.

2.3 Private Smart Contract

Existing SC-based systems lack privacy and blockchain consensus causes poor performance in SCs. Blockchains and Trusted Execution Environments (TEEs) have complementary properties: the former guarantees availability and persistence of its state, and the latter offers minimal overhead and verifiable computation with confidential state [37]. To take advantage of the two techniques, Cheng et al. proposed Ekiden, a system for highly performant and confidentiality-preserving SCs [37]. Ekiden is enabled by a delicate and secure integration of blockchains and trusted hardware that separates computation from consensus. *Compute nodes* are used to execute SC computation over private data off chain in TEEs, then attest to their correct execution on chain. The blockchain is maintained by *consensus nodes* which are not equipped with trusted hardware. Hawk [25] is a blockchain model of cryptography and privacy-preserving smart contracts. It preserves transactional privacy from the public's view and allows a programmer to write a PSC in an intuitive manner. Arbitrum [38] is a platform for SCs with scalability and privacy. It allows SCs to execute privately while publishing only (saltable) hashes of SCs. Our proposed scheme differs from existing schemes in designing a PSC for donation privacy while maintaining scalability.

3 Problem Statement

Before delving into technical details, we introduce the system architecture, threat model, and design objectives of Astraea. As we shall show, Astraea supports the full donation lifecycle while achieving privacy, security, and audit.

3.1 System Architecture

The system model consists of donator, transponder, collector, donee, auditor, and consortium blockchain (Fig. 2).

Donator. A donator is a person who donates some asset (e.g., a package of facial masks) to the community of a donee

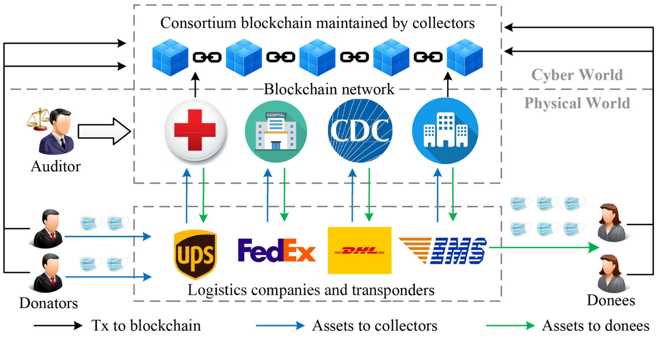


Fig. 2. System architecture of astraea.

via a collector. A donator hands the asset to a transponder with the collector being the receiver. The donator encrypts his donation response res and sends to the collector a response message rm. Finally, the donator uploads a donation transaction $\mathrm{Tx}^{\mathrm{don}}$ to the consortium blockchain. In auditing, each donator also collects the transactions sent from DiSC to check whether his donation is distributed to his intended donee.

Transponder. A transponder is hired by a logistics company to transport the assets from donator to collector, and from collector to donee's community. The logistics company has warehouses for asset storage and logistics stations for final pick-up. Transportation includes truck, airplane, freight train, and ship. Upon receiving an asset from a donator, the transporter takes a photo of the asset, obtains a digital signature of the donator via an electronic hand-held device and uploads a deliver transaction $\mathrm{Tx}^{\mathrm{del}_1}$. Upon delivering the asset to the donee, the transporter uploads a deliver transaction $\mathrm{Tx}^{\mathrm{del}_2}$. If the donee is not available for delivery, the transporter obtains a receipt after storing the item in a logistics station.

Collector. A collector is in charge of collecting and distributing assets. A collector can be Red Cross, Centers for Disease Control, companies, and commonwealth organizations. A predetermined set of collectors maintain a consortium blockchain. Each collector publishes a DiSC to distribute donations. The collector publishes and updates an official website to record donation requests from donees. For the requests, the collector publishes a DoSC to track donations. After receiving transponder's package, the collector stores the package and uploads a store transaction $\mathrm{Tx}^{\mathrm{sto}}$. The collector ships out the asset to a donee's community and uploads a ship transaction $\mathrm{Tx}^{\mathrm{shi}}$. At the end of a donation period (e.g., a day), the Enclave processes all responses to upload a batch of distribute transactions $\{\mathrm{Tx}^{\mathrm{dis}}\}$ and send a checklist message to the donee's community. After donees send a feedback message to the collector, the Enclave sends a refund transaction $\mathrm{Tx}^{\mathrm{ref}}$ to the DoSC to refund deposits.

Donee. A donee is the receiver of the asset. To complete the handover, a donee has to provide a real-world address. But for privacy concerns, individual donees assign the address to a local community and go to the community for donations. In the beginning, each donee publishes a donation request on the collector's website stating the number of donations in need and a community public key. We use the community key with as a cloaked identity for donees. When donees receive assets in their community, the community goes through the checklist to see whether all donations have arrived. The donee who successfully receives an asset encrypts a donation feedback fb and sends to the collector a feedback message fm. She uploads a receive transaction $\mathrm{Tx}^{\mathrm{rec}}$.

Auditor. An auditor is responsible for monitoring the donation distributions by collectors. In specific, the auditor collects the transactions sent to DoSC and the transactions sent from DiSC. It checks whether the aggregate of donation numbers is tampered with by the collectors. We require that each donation transaction and distribution transaction be marked with a date so that auditing proceeds in terms of individual days. This time period is flexible according to different audit requirements.

Consortium Blockchain. We choose to use a consortium blockchain maintained by all collectors to record donation responses and track each donation. The underlying consensus protocol can be a (crash-tolerant) consensus protocol [39], PBFT [40], or PoW [21]. We model each donation as a Finite State Machine (FSM) whose state transitions are executed by the DoSC. Transactions will invoke the functions in the DoSC to trigger state transitions. Each donation has six states: submitted, delivery, stored, shipped, delivered, and received.

Interactions Between Two Worlds. There are two worlds involved in our system: application (cyber world) and distribution of physical goods (physical world). The interactions between the two worlds happen when an entity sends or receives an asset. The first case includes a donator's handing the asset to a transponder and sending a response message (donation transaction) to a collector (DoSC); a transponder's delivering the asset to the collector and sending a delivery transaction to DoSC; a collector's receiving the asset and sending a store transaction; a collector's handing the asset to a transponder and sending a ship (distribution) transaction to DoSC; a transponder's delivering the asset to the collector and sending a delivery transaction to DoSC; a donee's receiving the asset and sending a feedback message (receive transaction) to DiSC (DoSC).

3.2 Threat Model

Donator and Donee. Donators/donees are honest-but-curious. Each donator/donee will follow the defined protocol but is curious about the donation number of other donators. We assume that the donee's community also behaves according to the protocol and does not collude with other entities.

Collector and transponder.

(1) Collusion attack. A collector and a transponder attempt to collude with each other to acquire the number, sum, and donator/donee identity.

(2) Stealing attack. A malicious collector/transponder intercepts the donations and steals asset to sell them at a higher price.

Besides the two attacks, we assume that collectors, as blockchain maintainers, will not try to undermine the blockchain. Specifically, the stealing attack is about accountability and we formally define its definition as follows.

Definition 1 (Stealing Attack). *Given a donation set $\mathcal{DO} = \{dn_1, \ldots, dn_{n_1}\}$, two transponders TP_1, TP_2, a collector C, a community, and a set of donees' public keys $\mathcal{PK} = \{pk_1, \ldots, pk_{n_2}\}$, we define a normal donation function $DF : \mathcal{DO} \times \mathcal{PK} \rightarrow sum$, where sum is the* donation sum in the physical world *after the TP_2 delivers the donations to the CM. When there are no stealing attacks, we have $sum \sum_{i=1}^{n_1} dn_i$. If a stealing attack happens, one of the three malicious parties (TP_1, C, and TP_2) intercepts some donations. For simplicity, we deem the three malicious parties as one adversary. We enhance the adversary's power by considering the ability to meddle with the donation chain and define a donation function under a stealing attack $DF' : \mathcal{DO} \times \mathcal{S} \times \mathcal{PK} \rightarrow sum'$, where $\mathcal{S}$ is the set of number of stolen donations $[1, n_1]$ and $sum' < \sum_{i=1}^{n_1} dn_i$.*

Auditor. We assume that auditors are honest-but-curious and behave faithfully when auditing the collectors. It means that auditors cannot determine who was involved in donations [20]. They do not collude with other entities to provide falsified audit reports.

Consortium Blockchain. We assume that the blockchain is available at all times and it is not susceptible to denial-of-service attacks [6].

3.3 Design Objectives

Our design objectives are three-fold: donation privacy, security, and auditing. We construct corresponding cryptographic games [41], [42] in which a Probabilistic Polynomial-Time (PPT) adversary interacts with the algorithm and we define the adversary's winning advantage. We say the algorithm is privacy-preserving and secure if the adversary's advantage in each game is negligible. For the collusion attack, we give the adversaries the ability to corrupt. Formally, the privacy and security objectives are defined below.

(1) Donation Privacy. The donation privacy is threefold. (1.1) Number: no adversaries except the donator/donee of the donation can acquire the exact number of a donation, (1.2) Sum: no adversaries can acquire the sum of donation numbers, and (1.3) Identity privacy: the identity of a donator/donee is protected from other entities.

(1.1) Number. We compute the advantage of an adversary $\mathcal{A}$ in observing the execution of the scheme $\Pi = (\mathsf{Setup}, \mathsf{Donate}, \mathsf{Distribute}, \mathsf{Receive}, \mathsf{Audit})$ and correctly guessing the donation number from two commitments. We call this advantage the number hiding advantage and denote it by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Num}}(\Pi)$. We define the *number hiding game* $\mathsf{Game}_{\mathcal{A},\Pi}^{\mathrm{Num}}$ as follows:

1. Setup: $\mathcal{A}$ generates dn_0, dn_1, and gives them to $\mathcal{C}$.
2. Execution: $\mathcal{C}$ executes Π to obtain (cm_0, tok) and (cm_1, tok) in two donation transactions.
3. Challenge: $\mathcal{C}$ chooses a uniform bit $b \in \{0, 1\}$ and sends cm_b to $\mathcal{A}$.
4. Guess: $\mathcal{A}$ outputs a bit b' sends it to $\mathcal{C}$. $\mathcal{A}$ wins the game if $b' = b$.

$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Num}}(\Pi)$ is the advantage of $\mathcal{A}$ in winning $\mathsf{Game}_{\mathcal{A},\Pi}^{\mathrm{Num}}$:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Num}}(\Pi) = |\Pr[\mathcal{A}\ wins] - \frac{1}{2}| = |\Pr[b' = b] - \frac{1}{2}|.$$

(1.2) Sum. Sum is different than number that is hidden in a commitment. We put the donation sum sum in the form of $\mathrm{Agg} = g^{sum}$ for PoA verification. We define the sum hiding game $\mathsf{Game}_{\mathcal{A},\Pi}^{\mathrm{Sum}}$ as follows.

1. Setup: $\mathcal{C}$ generates n donation numbers dn_i, $i \in [1, n]$ and computes $sum = \sum_i dn_i$.
2. Execution: $\mathcal{C}$ executes Π to obtain Agg.
3. Challenge: $\mathcal{C}$ sends Agg to $\mathcal{A}$.
4. Guess: $\mathcal{A}$ outputs a value sum' and sends it to $\mathcal{C}$. $\mathcal{A}$ wins the game if $sum' = sum$.

(1.3.1) Identity Privacy of Donator. We compute the advantage of an adversary $\mathcal{A}$ in correctly guessing the identity id_i of a donator DR_i. We call this advantage the identity indistinguishability advantage and denote it by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi)$. We define $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi)$ by using a challenge-response game. Let $\mathcal{DR} = \{DR_1, DR_2, \ldots, DR_n\}$ be the set of all donators, including the PPT adversary $\mathcal{A}$, the challenger $\mathcal{C}$, and the proposed Astraea scheme Π. We define the *identity indistinguishability game* $\mathsf{Game}_{\mathcal{A},\Pi}^{\mathrm{Ide}_1}$ as follows:

1. Setup: $\mathcal{C}$ collects identities id_i, $i \in \{1, 2, \ldots, n\}$.
2. Execution: $\mathcal{C}$ executes Π with all donators $\mathcal{DR}$ and records donation/distribution transactions.
3. Challenge: $\mathcal{C}$ chooses a $k \in [1, n]$ and sends id_i to $\mathcal{A}$.
4. Guess: $\mathcal{A}$ selects a value $k' \in [1, n]$ and sends it to $\mathcal{C}$. $\mathcal{A}$ wins the game if $k' = k$.

$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi)$ is the advantage of $\mathcal{A}$ in winning $\mathsf{Game}_{\mathcal{A},\Pi}^{\mathrm{Ide}_1}$:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi) = |\Pr[\mathcal{A}\ wins] - \frac{1}{n}| = |\Pr[k' = k] - \frac{1}{n}|,$$

where $\mathcal{A} \notin \mathcal{DR}$, i.e., $\mathcal{A}$ is the collector, a transponder, or a donee. $\Pr[k' = k]$ is the probability that $\mathcal{A}$ correctly guesses k chosen by $\mathcal{C}$. When $\mathcal{A} \in \mathcal{DR}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi) = \Pr[k' = k] - \frac{1}{n-1}$. We do not consider it a successful attack when DR_k is $\mathcal{A}$.

(1.3.2) Identity Privacy of Donee. We define a similar identity indistinguishability game $\mathsf{Game}_{\mathcal{A},\Pi}^{\mathrm{Ide}_2}$ as follows:

1. Setup: $\mathcal{C}$ collects identities id_i, $i \in [1, n]$.
2. Execution: $\mathcal{C}$ executes Π with all donees $\mathcal{DE}$ and records their receive transactions.
3. Challenge: $\mathcal{C}$ chooses a $k \in [1, n]$ and sends id_i to $\mathcal{A}$.
4. Guess: $\mathcal{A}$ selects a value $k' \in [1, n]$ and sends it to $\mathcal{C}$. $\mathcal{A}$ wins the game if $k' = k$.

$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_2}(\Pi) = |\Pr[\mathcal{A}\ wins] - \frac{1}{n}| = |\Pr[k' = k] - \frac{1}{n}|$, where $\mathcal{A} \notin \mathcal{DE}$. When $\mathcal{A} \in \mathcal{DE}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_2}(\Pi) = |\Pr[k' = k] - \frac{1}{n-1}|$.

(2) Security.

(2.1) Resistance to collusion attack. Astraea should prevent the collusion attack between a collector and transponder, i.e., the colluding parties cannot violate the donation privacy. We enhance the collector $\mathcal{A}$'s power by considering the ability to corrupt [43], [44]. Specifically, we model this ability as a corrupt query $\mathsf{Corrupt}()$. When $\mathcal{A}$ issues a query to $\mathsf{Corrupt}()$, $\mathcal{C}$ returns all the possessed information $\mathcal{I}_{TP}$ to $\mathcal{A}$. We compute the advantage of $\mathcal{A}$, i.e., a collector, colluding with a transponder TP in correctly guessing the number, sum, and identity of donator/donee. We call this advantage the collusion advantage and denote it by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Col}}(\Pi)$. If the collusion attack succeeds in guessing one of the targets, we consider the attack successful, i.e., $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Col}}(\Pi)$

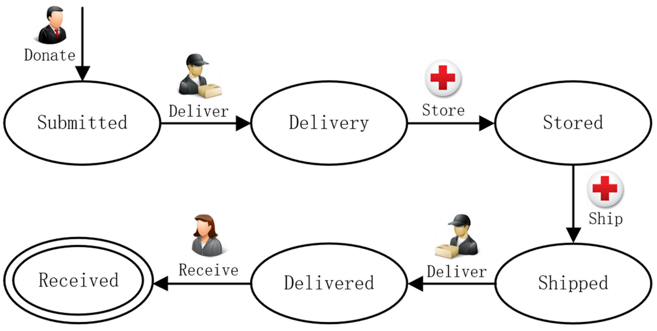


Fig. 3. The state machine model of astraea.

equals to the advantage in the each of the games defined above when attacking pertinent targets.

(2.2) Resistance to Stealing Attack. Astraea should prevent the stealing attack, i.e., intermediate entities are held accountable for their behaviors during each donation and malicious entities will be punished if they steal an asset. We construct a distinguisher D that interacts with a donation oracle with a goal to determine whether the output sum was generated by DF or DF'. We define the stealing game $\mathsf{Game}^{\text{Ste}}_{\mathcal{A},\Pi}$ as follows:

1. Setup: A donation system Π is initialized. The access to a donation oracle $\mathcal{O}$ is given to D.
2. Query: D chooses a donation set $\mathcal{DO} = \{dn_1, \ldots, dn_{n_1}\}$ and queries it to $\mathcal{O}$.
3. Respond: A random bit b is chosen. If $b = 0$, respond with $sum = DF(\mathcal{DO}, \mathcal{PK})$; otherwise, respond with $sum = DF'(\mathcal{DO}, s, \mathcal{PK})$ where s is randomly drawn from $\mathcal{S}$.
4. Distinguish: D outputs a bit b'.

For the stealing attack to be detected, it must be that $|\Pr[D^{DF}(sum)=1]$-$\Pr[D^{DF'}(sum')=1]|$ is non-negligible.

(3) Auditing. We have two requirements for auditing: equity and donation intention. These two requirements are consistent with the verification of PoA and PoD. If the two verification pass, we consider the auditing successful.

4 Preliminaries

In this section, we review the preliminaries, namely Pedersen commitments in Section 4.1, blockchain and SC in Section 4.2, and Intel SGX in Section 4.3.

4.1 Pedersen Commitments

Let $\mathbb{G}$ be a cyclic group with $|\mathbb{G}|$ elements. g and h are two random generators of $\mathbb{G}$. A Pedersen commitment to an integer $a \in \{1, 2, \ldots, m\}$ is computed as follows: choose a random number r and return the commitment $cm : g^a h^r$ [45].

Pedersen commitments have two properties: (1) perfectly hiding: cm reveals nothing about a, and (2) commitments are computationally binding: if an adversary can open a commitment cm from two different random numbers, then it can be invoked to break the discrete logarithm assumption in $|\mathbb{G}|$ by calculating $\log_h g$.

4.2 Blockchain and Smart Contract

Blockchain is a decentralized, verifiable, and tamper-proof ledger that was initially used as an underlying technology to solve the double-spending problem in cryptocurrencies [21]. It integrates economic modeling, peer-to-peer networking, decentralized consensus, and cryptography to achieve distributed database synchronization. SC is a piece of self-enforcing codes deployed on the blockchain with a unique address and state variables. There are predefined and self-executable functions in the SC triggered by specific transactions. Executing functions in SCs requires some execution fees to incentivize honest peers and mitigate denial of service attacks.

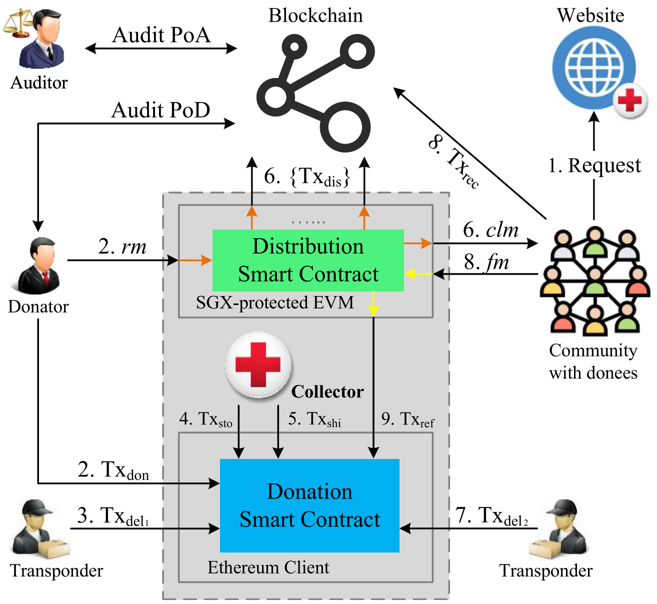

Auditor
Audit PoA
Blockchain
Website
Audit PoD
1. Request
8. $\mathrm{Tx}_{\mathrm{rec}}$
6. $\{\mathrm{Tx}_{\mathrm{dis}}\}$
......
2. rm
Distribution Smart Contract
6. clm
8. fm
SGX-protected EVM
Donator
Community with donees
Collector
4. $\mathrm{Tx}_{\mathrm{sto}}$
5. $\mathrm{Tx}_{\mathrm{shi}}$
9. $\mathrm{Tx}_{\mathrm{ref}}$
2. $\mathrm{Tx}_{\mathrm{don}}$
3. $\mathrm{Tx}_{\mathrm{del_1}}$
Donation Smart Contract
7. $\mathrm{Tx}_{\mathrm{del_2}}$
Transponder
Ethereum Client
Transponder

Fig. 4. Information flow of astraea.

4.3 SGX

Software Guard eXtensions (SGX) is a hardware extension of Intel Architecture that enables an application to establish a protected container, i.e., an enclave. It is protected by the processor through access control. Accesses to the enclave from outside the enclave are denied [28]. Only the processor and programs in the enclave can access the code and data in plaintext. Such an enclave provides confidentiality and integrity. To protect the enclave outside of the CPU, it is encrypted and integrity-protected by a specific key.

There are already many works on developing privacy-enhanced applications with SGX. For instance, VC3 [46] leveraged SGX processors to isolate memory regions on individual computers and secure distributed MapReduce computations. SDTE [32] established a secure execution environment atop SGX to protect the source data, and the data analysis results in data trading. SecGrid [33] designed a secure SGX-enabled smart grid system to support rich functionalities on customers' private data.

5 The Proposed Astraea

In this section, we first give an overview of Astraea and then present the details. We depict the state machine model of Astraea in Fig. 3 and the information flow of Astraea in Fig. 4. The pseudocodes for DiSC and DoSC are listed in Algorithms 1 and 2, respectively. Table 1 shows the notations frequently used in Astraea.

TABLE 1
Key Notations of Astraea

Notation	Definition
SGX	Software Guard eXtensions
SC, PSC	Smart Contract, Private Smart Contract
DoSC	Donation Smart Contract
DiSC	Distribution Smart Contract
PoA, PoD	Proof-of-Asset, Proof-of-Distribution
FSM	Finite State Machine
PPT	Probabilistic Polynomial-Time
TEE	Trusted Execution Environments
n_1/n_2	Number of donators/donees in games
$\mathcal{S}$	Number set of stolen donations
$\mathcal{PK}$	Set of donees' public keys
DR, dn, DE	Donator, donation number, donee
sum	Sum of donation numbers
C, CM	Collector, community
LC, TP	Logistics company, transponder
cl, clm	Checklist, checklist message
res, rm	Donation response, response message
Agg	g to the power of aggregate sum
$\mathcal{DA}$, R	Donation area, region
CB, Add	Consortium blockchain, address of SC
(pk, sk), dp	public/private key pair, deposit
$\mathbb{G}$; g, h	Cyclic group; generator
Σ	CCA2-secure public key cryptosystem
H, l	Secure hash function, length of H
fb, fm	Donation feedback, feedback message
cm, tok	Commitment, token
AggTok, AggCom	Aggregate of tokens, aggregate of commitments

5.1 Overview

To offer readers an overarching picture of the auditing procedures in the decentralized donation systems, we summarize five phases: Setup, Donate, Distribute, Receive, and Audit. In Setup, a donee publishes a donation request on a collector's website. The collector publishes to the blockchain a SGX-protected DiSC and a DoSC. In Donate, a donator hands the donation to a transponder, sends a donation message to DiSC and a donation transaction to DoSC. The transponder delivers the donation to the collector and uploads a delivery transaction to DoSC. In Distribute, the collector receives a batch of donations, uploads store transactions to the DoSC, uploads distribution transactions to the blockchain, and hands the donations to transponders. A transponder uploads a delivery transaction to DoSC. In Receive, a donee sends a feedback message to DiSC and uploads a receive transaction to the blockchain. In Audit, auditors verify PoA to check the sum of donations and donators verify PoD to check the distributions.

5.2 Setup

A cyclic group $\mathbb{G}$ is established by collectors with two random generators g and h. A CCA2-secure public key cryptosystem $\Sigma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is chosen with algorithms Setup, Encryption, and Decryption. A secure hash function $\mathsf{H}: \{0,1\}^* \rightarrow \{0,1\}^l$ is chosen.

The donation area is divided into a set of administrative region $\mathcal{DA} = \{R_1, R_2, \ldots, R_{|\mathcal{DA}|}\}$. The consortium blockchain is denoted as CB. The amount of deposit is dp. Each collector publishes its own DiSC with a unique address $\mathrm{Add}_{\mathrm{Di}}$. The identity of a SC is a hexadecimal address of 20 bytes. It is used when calling a function in the SC. The deployment of DiSC is different from normal Ethereum SCs because it is integrated into an SGX Enclave. A public/private key pair $(pk_{C,\mathrm{Di}}, sk_{C,\mathrm{Di}})$ is generated for each C's DiSC and stored in the Enclave by using the SGX Seal method. Note that the private key $sk_{C,\mathrm{Di}}$ is known to its Enclave but not C.

We assume that each collector C has a public/private key pair (pk_C, sk_C) where $pk_C = h^{sk_C}$ and an official website to call for donations. Each collector explicitly indicates the logistics companies that can be chosen by donators for asset delivery. Donators, donees, logistics companies, and transponders also have their own pair of keys.

A donee DE_i publishes on C's website her donation request $req_i = (R_i, C_i, pk_{CM_i}, dn_i)$ where CM_i is DE_i's community, dn_i is the donation number. DE_i also registers to the community with a public key. C publishes a SGX-protected DiSC with a unique address $\mathrm{Add}_{\mathrm{Di}}$ and a key pair $(pk_{C,\mathrm{Di}}, sk_{C,\mathrm{Di}})$. By checking a collector's website and looking into the SC, donators can quickly find the latest information.

5.3 Donate

A donator DR_j sees this request on collector C_i's website and donates to DE_i as follows. DR_j hands the asset, pk_{CM_i}, and $\mathsf{H}(cl_j)$ to a transponder TP_1 from LC, sets C_i as the receiver, and obtains an order ID id_{j1}. Here, cl_j is a checklist stating the detailed asset information with a random code used to reclaim deposits later. DR_j forms a donation response $res_j = (TP_1, LC, id_{j1}, dn_j, cl_j, pk_{CM_i}, \bar{r}_j, req_i, time)$ and encrypts it with $pk_{C_i,\mathrm{Di}}$ to obtain a response message $rm_j = \mathsf{Enc}(pk_{C_i,\mathrm{Di}}, res_j)$. $\bar{r}_j$ is a random number for PoD verification.

To enable anonymous and secure auditing, DR_j computes a commitment $cm_j := g^{dn_j} h^{r_j}$ where r_j is a random number. In addition, DR_j computes a token $tok_j = pk_{C_i}^{r_j}$ for PoA verification. Finally, DR_j sends rm_j to C_i and sends a donation transaction to DoSC:

$$\mathrm{Tx}^{\mathrm{don}}_{DR_j} = (''\mathrm{Donate}'', cm_j, tok_j, \mathsf{H}(res_j), time, \sigma), \tag{1}$$

where $\mathsf{H}(res_j)$ is a digital proof of donation, and σ is a signature.

Before delivering the asset to C_i, TP_1 sends a delivery transaction to C_i's DoSC including a deposit dp:

$$\mathrm{Tx}^{\mathrm{del}_1}_{TP_1} = (''\mathrm{Deliver}''_1, pk_{CM_i}, \mathsf{H}(cl_j), dp, time, \sigma), \tag{2}$$

where $\mathsf{H}(cl_j)$ is used for donation tracking and refunding deposits later.

5.4 Distribute

Upon receiving the donation from TP, C_i uploads a store transaction to DoSC:

$$\mathrm{Tx}^{\mathrm{sto}}_{C_i} = (''\mathrm{Store}'', pk_{CM_i}, \mathsf{H}(cl_j), time, \sigma). \tag{3}$$

After receiving the ciphertext rm_j from DR, C_i's SGX Enclave decrypts it with $sk_{C_i,\mathrm{Di}}$ to read res_j into the Enclave, which stores the response by calling the function `Respond` in Algorithm 1. At the end of a day, the Enclave performs donation distribution by calling the function `Distribute`. For each res, the Enclave performs as follows:

- Compute a new commitment $cm'_j \leftarrow g^{dn_j} h^{\bar{r}_j}$;
- Compute new token $tok'_j \leftarrow pk_{C_i}^{\bar{r}_j}$;
- Compute $sum = sum + dn_j$ for PoA verification;
- Prepare a distribute transaction for auditing: $\mathrm{Tx}_{C_i}^{\mathrm{dis}} = (''\mathrm{Distribute}'', pk_{CM_i}, \mathsf{H}(cl_j), cm'_j, tok'_j, time)$; (A signature will be added outside of the Enclave.)
- When all responses are processed, compute $\mathrm{Agg} = g^{sum}$ and a zero-knowledge proof π by using the Sigma-protocol, prepare a final distribute transaction $(''\mathrm{Distribute}'', \mathrm{Agg}, \pi, time, \sigma)$, and send $\{\mathrm{Tx}_{\mathrm{dis}}\}$ to CB;
- Reset the variables of DiSC for next batch of responses.

Next, C_i hands the asset to the second transponder TP_2, sets CM_i as the receiver, and obtains an order ID id_{j2}. C_i sends a ship transaction to DoSC:

$$\mathrm{Tx}_{C_i}^{\mathrm{shi}} = (''\mathrm{Ship}'', pk_{CM_i}, \mathsf{H}(cl_j), dp, time, \sigma). \tag{4}$$

More importantly, C_i uploads a distribution transaction to CB for each donation response:

$$\mathrm{Tx}_{C_i}^{\mathrm{dis}} = (''\mathrm{Distribute}'', pk_{CM_i}, \mathsf{H}(cl_j), cm'_j, tok'_j, time, \sigma). \tag{5}$$

Finally, the other transponder TP_2 sends a delivery transaction to DoSC:

$$\mathrm{Tx}_{TP_2}^{\mathrm{del}_2} = (''\mathrm{Deliver}''_2, pk_{CM_i}, \mathsf{H}(cl_j), dp, time, \sigma). \tag{6}$$

Algorithm 1. Pseudocode for DiSC

```
struct $res$; //Donation response Struct
struct $dis$; //Distribution Struct
struct $ref$; //Refund Struct
create $sum = 0$; //sum of donation numbers
create res[] Res; //Array of donation responses
create dis[] Dis; //Array of distribution
  transactions
create string[] List; //Array of donation
  checklists
create ref[] Ref; //Array of refund
  transactions
function Response($res_j$)
  Res{} ← ($res_j$);
function Distribute(Res{})
  while not at end of Res{}
    res ← Res{}; //$res = (TP, LC, id, dn_j, cl_j, pk_{CM_i}, \bar{r}_j,$
  $req_i, time)$
    $cm' \leftarrow g^{dn_j} h^{\bar{r}_j}$; //compute new commitment
    $tok' \leftarrow pk_{C_i}^{\bar{r}_j}$; //compute new token
    $sum = sum + dn_j$; //update sum
    Dis{} ← ($''\mathrm{Distribute}'', pk_{CM_i}, \mathsf{H}(cl_j), cm', tok', time)$);
    //to send distribute transactions
    List{} ← $cl_j$; //to send detailed asset
    information to donees
    Agg = $g^{sum}$; //for PoA verification
     $\pi = \mathrm{ZKP}\{(sk_{C_i}) | \mathrm{ACom}_{\mathrm{in}}^{'sk_{C_i}} = \mathrm{ATok}_{\mathrm{in}} \wedge \mathrm{ACom}_{\mathrm{out}}^{'sk_{C_i}} =$
     $\mathrm{ATok}_{\mathrm{out}}\}$;
  Dis{} ← ($''\mathrm{Distribute}'', \mathrm{Agg}, \pi, time, \sigma$); //for PoA
  verification
    sum = 0; //clear sum
    Return (Dis{}, List{});
    function Clear()
    Clear Dis{} and List{};
    function Refund(fb)
     Ref{} ← ($''\mathrm{Refund}'', pk_{CM}, \mathsf{H}(cl), time, \sigma$); //to send
refund transactions
    Return (Ref{});
```

Algorithm 2. Pseudocode for DoSC

```
struct $don$; //Donation Struct
mapping (bytes32 => don) Don; //Map of donation
pool
function Transfer(Tx)
  Switch (Tx, Don[$\mathsf{H}(cl)$].state){
    Case: ($\mathrm{Tx}_{\mathrm{don}}$, Empty)
      Don[$\mathsf{H}(cl)$].state = Submitted;
    Case: ($\mathrm{Tx}_{\mathrm{del}_1}$, Submitted)
      Don[$\mathsf{H}(cl)$].state ← Delivery;
      Don[$\mathsf{H}(cl)$].deposit += $dp$;
    Case: ($\mathrm{Tx}_{\mathrm{sto}}$, Delivery)
      Don[$\mathsf{H}(cl)$].state ← Stored;
      Don[$\mathsf{H}(cl)$].deposit += $dp$;
    Case: ($\mathrm{Tx}_{\mathrm{Shi}}$, Stored)
      Don[$\mathsf{H}(cl)$].state ← Shipped;
    Case: ($\mathrm{Tx}_{\mathrm{del}_2}$, Shipped)
      Don[$\mathsf{H}(cl)$].state ← Delivered;
      Don[$\mathsf{H}(cl)$].deposit += $dp$;
function Refund($\mathrm{Tx}_{\mathrm{ref}}$)
  if (Don[$\mathsf{H}(cl)$])
    if (Don[$\mathsf{H}(cl)$].state == Received)
      Refund $dp$ to $TP_1$, $TP_2$, and $C$;
      Don[$\mathsf{H}(cl)$].state ← Received;
      Don[$\mathsf{H}(cl)$].deposit = 0;
      Return "Donation Completed."
```

5.5 Receive

Upon receiving the donations from transponders including TP_2, CM_i distributes the donations to donees including DE in its community. After unpacking a package, DE_i obtains the checklist cl_j. She forms a donation feedback $fb_i = (TP_2, LC, id_i, dn_j, pk_{CM_i}, cl_j, time)$ and encrypts it with $pk_{C_i,\mathrm{Di}}$ to obtain a feedback message $fm_j = \mathsf{Enc}(pk_{C_i,\mathrm{Di}}, fb_i)$.

Finally, DE_i sends fm_i to C_i's DiSC and uploads a receive transaction to CB:

$$\mathrm{Tx}_{DE_i}^{\mathrm{rec}} = (''\mathrm{Receive}'', \mathsf{H}(fb_i), time, \sigma). \tag{7}$$

Note that the state transition of each donation is completed by calling the function `Transfer` in Algorithm 2.

After receiving the ciphertext fm_i from DE_i, C_i's SGX Enclave decrypts it with $sk_{C_i,\mathrm{Di}}$ to read fb_i into the Enclave.

The Enclave performs deposit refunding by calling the function `Refund` as follows:

- Compute $\mathsf{H}(cl_j)$;
- Prepare a refund transaction for deposit refunding: $\mathrm{Tx}_{C_i}^{\mathrm{ref}} = (''\mathrm{Refund}'', pk_{CM_i}, time, \mathsf{H}(cl_j), \sigma)$;
- Send $\mathrm{Tx}_{C_i}^{\mathrm{ref}}$ to DoSC.

5.6 Audit

The auditor and the donator audit the output of the enclave. For PoA verification, an auditor first records the transactions $\mathrm{Tx}_{\mathrm{don}}$ sent to a collector and the transactions $\mathrm{Tx}_{\mathrm{dis}}$ sent from the collector's Enclave in each donation period. Then the auditor verifies PoA as follows:

- Compute aggregate $\mathrm{ATok}_{\mathrm{in}} = \prod_j pk_{C_i}^{r_j}$;
- Compute aggregate $\mathrm{ATok}_{\mathrm{out}} = \prod_j pk_{C_i}^{\bar{r}_j}$;
- Compute aggregate $\mathrm{ACom}_{\mathrm{in}} = \prod_j g^{dn_j} \cdot h^{r_j}$;
- Compute aggregate $\mathrm{ACom}_{\mathrm{out}} = \prod_j g^{dn_j} \cdot h^{\bar{r}_j}$;
- Compute $\mathrm{ACom}'_{\mathrm{in}} = \mathrm{ACom}_{\mathrm{in}}/\mathrm{Agg}$;
- Compute $\mathrm{ACom}'_{\mathrm{out}} = \mathrm{ACom}_{\mathrm{out}}/\mathrm{Agg}$;
- Verify $\log_{\mathrm{ACom}'_{\mathrm{in}}} \mathrm{ATok}_{\mathrm{in}} \overset{?}{=} \log_{\mathrm{ACom}'_{\mathrm{out}}} \mathrm{ATok}_{\mathrm{out}}$.

We give a proof of correctness here:

$$\mathrm{AggTok}_{\mathrm{in}} = \prod_j pk_{C_i}^{r_j} = pk_{C_i}^{\sum r_j},$$
$$\mathrm{AggTok}_{\mathrm{out}} = \prod_j pk_{C_i}^{\bar{r}_j} = pk_{C_i}^{\sum \bar{r}_j},$$
$$\mathrm{AggCom}_{\mathrm{in}} = \prod_j g^{dn_j} \cdot h^{r_j} = g^{\sum dn_j} \cdot h^{\sum r_j},$$
$$\mathrm{AggCom}_{\mathrm{out}} = \prod_j g^{dn_j} \cdot h^{\bar{r}_j} = g^{\sum dn_j} \cdot h^{\sum \bar{r}_j},$$
$$\mathrm{Agg} = g^{sum} = g^{\sum dn_j},$$
$$\mathrm{AggCom}'_{\mathrm{in}} = \mathrm{AggCom}_{\mathrm{in}}/\mathrm{Agg} = h^{\sum r_j},$$
$$\mathrm{AggCom}'_{\mathrm{out}} = \mathrm{AggCom}_{\mathrm{out}}/\mathrm{Agg} = h^{\sum \bar{r}_j},$$
$$\log_{\mathrm{AggCom}'_{\mathrm{in}}} \mathrm{AggTok}_{\mathrm{in}} = \log_{h^{\sum r_j}} pk_{C_i}^{\sum r_j} = sk_{C_i},$$
$$\log_{\mathrm{AggCom}'_{\mathrm{out}}} \mathrm{AggTok}_{\mathrm{out}} = \log_{h^{\sum \bar{r}_j}} pk_{C_i}^{\sum \bar{r}_j} = sk_{C_i}.$$

For PoD verification, each donator records the transaction sent from the collector's Enclave in each donation period, i.e., the $\mathrm{Tx}_{C_i}^{\mathrm{dis}}$ with the $\mathsf{H}(cl_j)$ being his own. Then the donator dn_j verifies PoD by computing $cm''_j := g^{dn_j} h^{\hat{r_j}}$ and checking $cm''_j \overset{?}{=} com'$. Since the random number is generated by DR_j and passed to the Enclave, the donator can easily and secretly check whether the donation number in the new commitment is consistent with the one in the previous commitment.

6 Privacy and Security Analysis

In this section, we formally prove the privacy, security, and auditing w.r.t. adversary model and design objectives.

6.1 Privacy

Theorem 1. *Astraea is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Num}}(\Pi)$-number hiding against PPT adversaries under the Decisional Diffie-Hellman (DDH) assumption, where $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Num}}(\Pi) \leq \mathsf{negl}(n)$.*

Proof: In the DDH assumption, no PPT adversary can distinguish between (h, h^x, h^y, h^{xy}) and (h, h^x, h^y, h^z). We assume that $\mathcal{A}$ which is given (g, h, pk_C) can generate dn_0 and dn_1 such that it can distinguish the commitment and tokens to dn_0 from the commitment and token to dn_1, i.e., $\mathcal{A}$ can distinguish the $(g^{dn_0}h^r, h^{sk_C r}, h^{sk_C})$ and $(g^{dn_1}h^r, h^{sk_C r}, h^{sk_C})$. We show how to use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ to break the DDH assumption.

1. $\mathcal{A}'$ is given a challenge (h, X, Y, Z) where Z is either h^{xy} or h^z.
2. $\mathcal{A}'$ samples a random generator g and calls $\mathcal{A}$ on input (g, h, X) where X acts as pk_C.
3. $\mathcal{A}$ generates two values dn_0, dn_1, and sends them to $\mathcal{A}'$.
4. $\mathcal{A}'$ chooses a random bit $b \in \{0, 1\}$, computes $cm_k = g^{dn_k}Y$, and sets $tok = Z$. $\mathcal{A}'$ sends (cm_k, tok) to $\mathcal{A}$.
5. $\mathcal{A}$ outputs a guess b' and returns b' to $\mathcal{A}'$.
6. $\mathcal{A}'$ outputs b'.

$\mathcal{A}'$ runs in polynomial time since $\mathcal{A}$ does. There are two cases to consider:

Case 1: $\mathcal{A}'$ is given a challenge (h, h^x, h^y, h^z). In this case, the inputs to $\mathcal{A}$ have information-theoretically no information about dn_b: h^z is unrelated to $g^{dn_b}h^y$. Thus,

$$\begin{aligned}&\Pr[\mathcal{A}'(\mathbb{G}, h, h^a, h^b, h^c) = 1]\\&= \Pr[\mathcal{A}(\mathbb{G}, g, h, h^x, g^{dn_b}h^y, h^z) = 1] = \frac{1}{2}.\end{aligned}$$

Case 2: $\mathcal{A}'$ is given a challenge (h, h^x, h^y, h^{xy}). In this case, $\mathcal{A}$ has what it expects in the number hiding game with $sk_C = x$ and $r = y$. Thus,

$$\begin{aligned}&\Pr[\mathcal{A}'(\mathbb{G}, h, h^x, h^y, h^{xy}) = 1]\\&= \Pr[\mathcal{A}(\mathbb{G}, g, h, h^x, g^{dn_b}h^y, h^{xy}) = 1].\end{aligned}$$

Given that the DDH problem is hard relative to $\mathbb{G}$, there is a negligible function negl such that

$$\begin{aligned}&\mathsf{negl}(n)\\&\geq |\Pr[\mathcal{A}'(\mathbb{G}, h, h^a, h^b, h^c) = 1] - \Pr[\mathcal{A}'(\mathbb{G}, h, h^x, h^y, h^{xy}) = 1]|\\&= |\Pr[\mathcal{A}(\mathbb{G}, g, h, h^x, g^{dn_b}h^y, h^z) = 1]\\&\quad - \Pr[\mathcal{A}(\mathbb{G}, g, h, h^x, g^{dn_b}h^y, h^{xy}) = 1]|\\&= |\frac{1}{2} - \Pr[\mathcal{A}(\mathbb{G}, g, h, h^x, g^{dn_b}h^y, h^{xy}) = 1]|\end{aligned}$$

This implies $\Pr[\mathcal{A}(\mathbb{G}, g, h, h^x, g^{dn_b}h^y, h^{xy}) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$, i.e., the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Num}}(\Pi)$ is negligible, completing the proof. ■

Theorem 2. *Astraea is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sum}}(\Pi)$-sum hiding against PPT adversaries under the Discrete Logarithm (DL) assumption.*

Proof: In one of the distribution transactions released by a collector, we hide sum in $\mathrm{Agg} = g^{sum}$ for PoA verification. The underlying security is straightly drawn from the DL assumption. ■

Theorem 3. *Astraea is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi)$-identity indistinguishable against PPT adversaries under the Elliptic Curve Discrete Logarithm Problem (ECDLP), where $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi) \leq \frac{1}{2^{|h|}}$.*

Proof: We prove according to the four types of adversaries: donator, collector, transponder, and donee.

(1) Donator as $\mathcal{A}$. $\mathcal{A}$, who is not the chosen DR_k, can correctly guess k by either correctly guessing the value of sk or (2) randomly guessing k. As $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ide}_1}(\Pi) = \Pr[k' = k] - \frac{1}{n-1}$, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Ide}_1}_{\mathcal{A}_1}(\Pi) &= \frac{n-1}{n} * \frac{1}{2^{|h|}} + \frac{1}{n-1} - \frac{1}{n-1}\\ &= \frac{n-1}{n} * \frac{1}{2^{|h|}} < \frac{1}{2^{|h|}},\end{aligned}$$

which is negligible under the ECDLP.

(2) Collector as $\mathcal{A}$. A malicious collector is outside of the donator set and its target is among the n donators. Although it handles the matching for the n users, the plaintexts are in the Enclave and hidden from the collector. We have

$$\mathsf{Adv}^{\mathsf{Ide}_1}_{\mathcal{A}}(\Pi) = \Pr[k' = k] - \frac{1}{n} = \frac{1}{2^{|h|}} + \frac{1}{n} - \frac{1}{n} = \frac{1}{2^{|h|}},$$

which is negligible.

(3) Transponder and Donee as $\mathcal{A}$. The transponder and donee are is also outside of the donator set. Similar to the collector scenario, we have $\mathsf{Adv}^{\mathsf{Ide}_1}_{\mathcal{A}}(\Pi) = \frac{1}{2^{|h|}}$.

To summarize, the proposed Astraea scheme is $\mathsf{Adv}^{\mathsf{Ide}_1}_{\mathcal{A}}$ (Π)-identity indistinguishable against PPT adversaries. ■

Theorem 4. *Astraea is* $\mathsf{Adv}^{\mathsf{Ide}_2}_{\mathcal{A}}(\Pi)$*-identity indistinguishable against PPT adversaries, where* $\mathsf{Adv}^{\mathsf{Ide}_2}_{\mathcal{A}}(\Pi) \leq \frac{1}{2^{|h|}}$.

Proof: Similar to the proof of Theorem 1, we also have four types of adversaries here: donator, collector, transponder, and donee. The donees only interact with the system (blockchain) when uploading a receive transaction, so in the first three cases, the adversaries are observers and the advantage $\mathsf{Adv}^{\mathsf{Ide}_2}_{\mathcal{A}}(\Pi) = \Pr[k' = k] = \frac{1}{2^{|h|}}$. In the last case, the malicious donee has to guess the identity of other $n-1$ donees which leads to $\mathsf{Adv}^{\mathsf{Ide}_2}_{\mathcal{A}}(\Pi) = \Pr[k' = k] < \frac{1}{2^{|h|}}$. Therefore, the proposed Astraea scheme is $\mathsf{Adv}^{\mathsf{Ide}_2}_{\mathcal{A}}(\Pi)$-identity indistinguishable against PPT adversaries. ■

6.2 Security

Theorem 5. *Astraea is* $\mathsf{Adv}^{\mathsf{Col}}_{\mathcal{A}}(\Pi)$*-collusion resistant against PPT adversaries under the DDH assumption, the DL assumption, and the ECDLP.*

Proof: A malicious collector $\mathcal{A}$ colludes with a transponder (s) to share information, i.e., $\mathcal{A}$ has the ability to corrupt. To disclose the number or sum of a donator, $\mathcal{A}$ and transponders are given challenges as in the first two games in Section 3.3. The SGX-protected secure distribution environment has cut off the operability from the collector, which leaves no possibility for the colluding parties to learn more than what has already been known. To disclose the identity of a donator/donee, $\mathcal{A}$ can correctly guess k by either correctly guessing the value of sk, or (2) randomly guessing k. Even though $\mathcal{A}$ obtains all information of the transponder (s), they are only observers regarding identity. Therefore, $\mathsf{Adv}^{\mathsf{Col}}_{\mathcal{A}}(\Pi)$ is negligible.

Formally, we prove Theorem 5 by using a sequences of games [47], [48]. Let S_i be the event that $b' = b$ in Game_i for $0 \leq i \leq 3$. It is sufficient to prove that the view of $\mathcal{A}$ in the real world and that in the simulated world is computationally indistinguishable [49]. To this end, we define the following three games, between whom the indistinguishabilities can be reduced to underlying assumption/problem:

- Game_0: This is the real world where the system is running between honest donators/donees, and some transponders controlled by a malicious collector $\mathcal{A}$.
- Game_1: This is identical to Game_0 except that every commitment cm generated by honest donators is replaced with a simulated one $cm' = g^{r_1}h^{r_2}$, where r_1 and r_2 are two random numbers. Indistinguishability between Game_0 and Game_1 comes from the DDH assumption, i.e., $|\Pr[S_0] - \Pr[S_1]| = \epsilon_{ddh}$, where ϵ_{ddh} is the DDH-advantage of some efficient algorithm.
- Game_2: This is identical to Game_1 except that every aggregate sum Agg is generated by computing a random $Agg' = g^r$. Indistinguishability between Game_1 and Game_2 comes from the DL assumption, i.e., $|\Pr[S_1] - \Pr[S_2]| = \epsilon_{dl}$.
- Game_3: This is identical to Game_2 except that every pair of private key and public key of an honest donator/donee is replaced by a different one. Indistinguishability between Game_2 and Game_3 comes from the ECDLP, i.e., $|\Pr[S_2] - \Pr[S_3]| = \epsilon_{ecdlp}$.

It is evident that $\Pr[S_3] = 1/2$. Combining the three equations above, we have $|\Pr[S_0] - 1/2| \leq \epsilon_{ddh} + \epsilon_{dl} + \epsilon_{ecdlp}$, which is negligible. We complete the proof. ■

Theorem 6. *Astraea resists to the stealing attack under the ECDLP.*

Proof: Recall that we require that donees registers to their community (Section 5.2. Setup) with a public key. In each donation, entities except the donator are required to put down a security deposit [34], [35], [36]. In event of stealing, the malicious parties will not be able to send a receive transaction to the blockchain on behalf of a donee. The stealing will be caught and the deposits will not be returned to the thief.

Let Π be the proposed Astreae and Π' be the ECDLP. Let $\mathcal{A}$ (TP_1, C, TP_2) be a PPT adversary attacking Π and $\mathcal{A}'$ be a PPT adversary attacking Π'. Let Att be the event that $\mathcal{A}$ successfully steals a donation without being caught and Att' be the event that $\mathcal{A}'$ breaks the Π'.

Now we assume that $\mathcal{A}$ has non-negligible advantage in launching a stealing attack against Π and show how to construct $\mathcal{A}'$ breaking Π' with a non-negligible advantage:

1. $\mathcal{A}'$ invokes $\mathcal{A}$ as a subroutine.
2. $\mathcal{A}$ inputs s to DF'.
3. $\mathcal{A}'$ output what $\mathcal{A}$ outputs.

$\mathcal{A}'$ runs in polynomial time since $\mathcal{A}$ does. If Att happens, then Att' happens, i.e., $\Pr[\mathsf{Att}'] = \Pr[\mathsf{Att}]$. This is because when a stealing attack succeeds, it must be that $\mathcal{A}$ has computed or acquired at least one private key of a donee and then submits a receive transaction to cover the malicious act, i.e., $\mathsf{Adv}^{\mathsf{Ste}}_{\mathcal{A}}(\Pi)$ is negligible, thus allowing $\mathcal{A}'$ to solve the ECDLP. Since the ECDLP is difficult to be broken, we have $|\Pr[D^{DF}(sum) = 1] - \Pr[D^{DF'}(sum) = 1]|$ is non-negligible and D can distinguish DF' from DF, completing the proof. ■

6.3 Auditing

For equity, we allow an auditor to check the donation sum of processed donation transactions in one donation period by efficiently leveraging the aggregate of commitments and the aggregate of tokens. When the PoA verification pass, the auditor will be convinced that the collector does not tamper with the sum, thus auditing the equity.

For donation intention, we allow each donator to check the committed number in the associated distribution transaction. When the PoD verification pass, the donator believes that his contribution is counted, thus auditing donation intention.

6.4 Possible Attacks and Security Assumptions

Lying Donators. A malicious donator can overstate his donated assets. This is an interesting attack and we give an idea of how to address it. First, we build a TEE-enabled monitoring system with a micro-controller to periodically sense the inside of a donation package, generate unforgeable data, and upload an "encrypted" donation number to the blockchain [50]. However, this brings the donators extra costs that we have to consider cost control. Second, we allow the auditor to securely verify the consistency of claimed donation number and the real donation number via zero-knowledge proof of knowledge.

Lying Donees. A malicious donee can pretend not to receive the assets that are indeed delivered. Since we ask the donees to go to their community for donations, thereby protecting their home address, this attack can be prevented by asking the community (where an automatic device distributes assets or staff manually operates) to confirm the final delivery by interacting with the blockchain.

Multiple Auditors. Putting the entire trust on a single auditor naturally makes one associate with single-point-of-failure. Operating multiple auditors is a better solution that not only reduces the workload for one auditor but also increases the credibility of the auditing and the cost of dishonest auditing.

Collusion Between Donator/Donee and With Transponder. Such an attack does not convey many financial incentives compared with our proposed collusion attack, still, it may cause some consequences. For example, a donator and a transponder fabricate a donation; a transponder does not deliver a donator's asset to a collector but secretly hands it to a colluding donator/donee; and a transponder does not deliver collector's distributions to a community but secretly hands it to a colluding donator/donee. The first attack can be easily defended because the collector does not receive the actual asset; the second attack is a stealing attack that will be detected by the honest donator; the last attack can be eliminated by asking the community to send a receive transaction to the blockchain and enhance the consistency of the donation chain.

All the four cases above are beneficial to our future work and we believe there are more to be explored.

6.5 51% Attack and Transaction Reordering Attack

We have used the Proof-Of-Authority (POA) consensus in the Ethereum-based experiments. Different from the 51% attack against the PoW, the 51% attack against POA requires an adversary to gain control over 51% of network nodes, which is more difficult than obtaining 51% computational power.

The Ethereum classic uses the PoW consensus algorithm similar to Bitcoin, which is vulnerable to the 51% attack. However, a leading organization behind the Ethereum Classic network, ETC Labs, proposed defense mechanisms to stabilize the network's plummeting hashrate and resist future 51% attacks [51]. Moreover, the Ethereum co-founder Vitalik Buterin is leading the charge in quickly organizing potential solutions to 51% attacks by moving to the Proof-of-Stake (PoS) mining from PoW mining [52].

Since distributed networks possess some degree of latency, they are prone to a transaction reordering attack. To mitigate this attack, we can leverage *a transaction counter* in the smart contract. When there is a state-modifying transaction triggering a function in the smart contract, we add one to the counter. We can also send a *transactionCount value* to when we initiate our transaction. If the counter's value does not equal to the one we have given, the transaction reverts.

7 Performance Analysis

In this section, we build a prototype of Astraea based on a Ethereum test network and Intel SGX to evaluate its computational costs, communication overhead, and monetary cost. We also compare Astraea with existing work.

7.1 Experimental Settings

We initiate a collector on a laptop (Lenovo Thinkpad X1 Carbon) with Ubuntu18.04, 8GB RAM, and a SGX-supported CPU (i5-10210u, 4-core, 4.20GHz). We install geth (Ethereum client) on our Collector node to communicate with other three computers (blockchain nodes) and deploy the DoSC contract. We use `puppeth` to create the genesis block with a Proof-of-Authority consensus mechanisms (Clique). Our smart contract is written in Solidity language, and other parts of the program are written in Go language. We choose $\mathbb{G}$ to be the group of points on the elliptic curve secp256k1 with $|g| = 512$. The hash function is SHA256. We have uploaded the codes to https://github.com/UbiPLab/Astraea.

7.2 SGX-protected EVM Configuration

To prepare the SGX runtime and development environment, we clone the SGX source codes published by Intel and install the required tools. We install the SGX-driver, SGX-SDK, and SGX-PSW. To run a SC inside SGX, we extract Ethereum Virtual Machine (EVM) source code from Go-ethereum repository,[1] and adopt the EGo framework.[2] EVM is the runtime environment for SCs (transaction execution) while EGo runs EVM in SGX.

7.3 Smart Contract Deployment

We use the browser-side SC development tool Remix to write codes. We use truffle, a javascript based Ethereum development framework to deploy DoSC. The DiSC is compiled into bytecode and export Application Binary Interface (ABI) information by using `Solidity Compiler` (SolC). EVM can read the bytecode and ABI file to run DiSC.

[1] https://github.com/ethereum/go-ethereum

[2] https://www.ego.dev/

TABLE 2
Comparison of Computational Costs (Time Unit: Ms)

Scheme		Donator	Transponder 1	Collector	Transponder 2	Donee	Audit
Delivery1 [11]	Theory	$T_{hash} + T_{sig}$	T_{sig}	n/a	n/a	T_{sig}	n/a
	Practice	0.025	0.025	n/a	n/a	0.025	n/a
Delivery2 [12]	Theory	T_{sig}	T_{sig}	n/a	T_{sig}	T_{sig}	n/a
	Practice	0.025	0.025	n/a	0.025	0.025	n/a
Tracking [13]	Theory	$3T_{sig}$	n/a	nT_{sig}	n/a	n/a	n/a
	Practice	0.075	n/a	0.025*n	n/a	n/a	n/a
Solidus [6]	Theory	$2T_{enc} + T_{sig}$	n/a	$n(T_{dec} + 2T_{pro} + 2T_{enc} + 2T_{sig})$	n/a	n/a	$n(T_{ver} + T'_{ver} + T_{dec})$
	Practice	0.097	n/a	(2.384t + 1.373)n	n/a	n/a	(1.344t + 0.913)n
zkLedger [20]	Theory	$n(2T_{mul} + T_{add} + T^A_{pro} + T^B_{pro} + T^C_{pro})$	n/a	n/a	n/a	n/a	$T^A_{ver} + T^B_{ver} + T^C_{ver}$
	Practice	5.212n	n/a	n/a	n/a	n/a	9.96
Astraea	Theory	$T_{hash} + T_{enc} + 3T_{mul} + T_{add} + T_{sig}$	T_{sig}	$2T_{hash} + (3n+1)T_{sig} + 2nT_{dec} + (3n+3)T_{mul} + (5n-2)T_{add}$	T_{sig}	$T_{hash} + T_{enc} + T_{sig}$	$2((n+2)T_{mul} + (n+1)T_{add} + T_{hash})$
	Practice	0.442	0.025	162.7n + 161.2	0.025	0.097	0.24n+0.4656

T_{hash}: hash; T_{enc}: Encryption; T_{dec}: decryption; T_{sig}: signing; T_{mul}: scala multiplication in $\mathbb{G}$; T_{add}: point addition in $\mathbb{G}$; T_{log}: logarithm operation. $T_{pro} = (5 + 10t)T_{mul} + tT_{enc}$: generate a proof (value$\in [0, 2^t)$) [6]; $T_{ver} = (7 + 12t)T_{mul}$: verify a proof; T'_{ver}: verify a signature; T^A_{pro}: generate a proof of asset in zkLedger; S^B_{pro}: generate a proof of balance; S^C_{pro}: generate a proof-of-consistency. T^A_{ver}: verify a proof of asset; S^B_{ver}: verify a proof of balance; S^C_{ver}: verify a proof-of-consistency.

TABLE 3
Comparison of Communication Overhead (Size Unit: KBytes)

Scheme		Donator	Transponder 1	Collector	Transponder 2	Donee
Delivery1 [11]	Theory	$3S_{str} + S_{hash} + S_{sig}$	$5S_{str} + S_{sig}$	n/a	n/a	$5S_{str} + S_{sig}$
	Practice	0.193	0.225	n/a	n/a	0.225
Delivery2 [12]	Theory	$3S_{str} + S_{hash} + S_{sig}$	$5S_{str} + S_{sig}$	n/a	$5S_{str} + S_{sig}$	$5S_{str} + S_{sig}$
	Practice	0.193	0.225	n/a	0.225	0.225
Tracking [13]	Theory	$3(2S_{str} + S_{sig})$	n/a	$n(S_{str} + S_{sig})$	n/a	n/a
	Practice	0.387	n/a	0.097n	n/a	n/a
Solidus [6]	Theory	$S_{str} + 2S_{enc} + S_{sig}$	n/a	$n(S_{str} + 2S_{enc} + 2S_{pro} + 2S_{sig})$	n/a	n/a
	Practice	0.211	n/a	0.82n	n/a	n/a
zkLedger [20]	Theory	$n(3S_g + S^A_{pro} + S^B_{pro} + S^C_{pro})$	n/a	n/a	n/a	n/a
	Practice	0.7n	n/a	n/a	n/a	n/a
Astraea	Theory	$S_{enc} + 3S_{str} + 2S_g + S_{hash} + S_{sig}$	$3S_{str} + S_g + S_{hash} + S_{sig}$	$(2n+6)S_{str} + (3n+5)S_g + (n+3)S_{hash} + (n+3)S_{sig}$	$3S_{str} + S_g + S_{hash} + S_{sig}$	$S_{enc} + 2S_{str} + S_{hash} + S_{sig}$
	Practice	0.362	0.257	0.353n+0.803	0.257	0.202

(String) $S_{str} = 256$ bits; (Point in $\mathbb{G}$) $S_g = 512$ bits; (Hash) $S_{hash} = 265$ bits; (Signature) $S_{sig} = 520$ bits; S_{pro}: range proof; S^A_{pro}: proof-of-asset; S^B_{pro}: proof-of-balance; S^C_{pro}: proof-of-consistency.

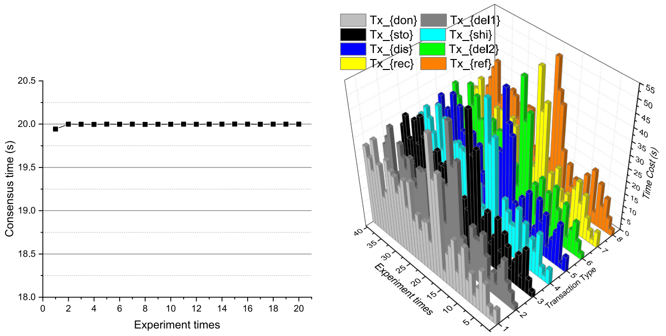
7.4 Computational Costs

We now analyze the computational costs for five entities (donators, transponder 1, collector processing n donations, transponder 2, and donee) and audit through counting the cryptographic operations in four phases. The theoretical and practical results are recorded in Table 2.

In Donate, a donator computes a donation message, which is an encryption of donation response T_{enc}. The donator also computes a donation transaction, which consists of a hash of checklist T_{hash}, a commitment on donation number $2T_{mul} + T_{add}$ (one scale multiplication and two and on point addition in $\mathbb{G}$), a token T_{mul}, and a signature T_{sig}. The transponder 1 computes a delivery transaction and the computational cost is from generating a signature T_{sig}. In Distribute, the collector computes one store transaction, decrypts n response messages, computes n commitments, n tokens, n zero-knowledge proofs, Agg, $n+1$ distribution transactions, and one ship transaction. Each proof generation requires $(4n-2)T_{add} + 2T_{mul} + 2T_{hash}$. The total cost is $2T_{hash} + (2n+1)T_{sig} + nT_{dec} + (3n+3)T_{mul} + (5n-2)T_{add}$. The transponder 2 computes one delivery transaction. In Receive, the donee computes a hash of donation feedback, an encryption of donation feedback, and a receive transaction, which is $T_{hash} + T_{enc} + T_{sig}$. The collector computes n decryption of feedback messages, and n refund transactions, which is $n(T_{dec} + T_{sig})$. In Audit, the auditor computes $2(n-1)$ multiplications of tokens, $2(n-1)$ multiplications of commitments, 2 divisions, and verifies n zero-knowledge proofs. The proof verification requires only $2(2T_{mul} + T_{add} + T_{hash})$. The total cost is $(4n-1)T_{add} + T_{hash}$. The donator computes one commitment, i.e., $2T_{mul} + T_{add}$.

7.5 Communication Overhead

We analyze the computational costs for five entities by counting the length of transmitted messages in three phases. The theoretical and practical results are recorded in Table 3.



(a) Consensus Time (b) Transaction Confirmation Time

Fig. 5. Network delay.

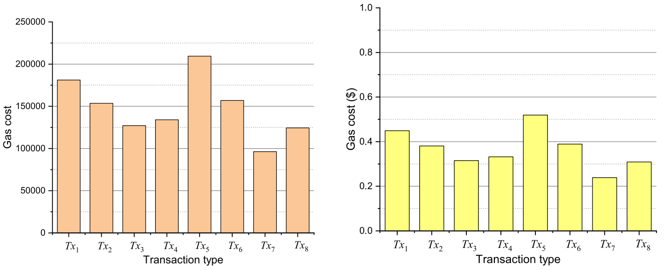
In Donate, a donator sends a response message to a collector and the communication overhead is $|rm| = S_{enc}$. The donator also uploads a donate transaction $\mathrm{Tx}^{\mathrm{don}}$ consisting of a string "Donate", two points cm and tok, a hash value $\mathsf{H}(cl)$, a string dp, a timestamp $time$, and a signature σ. The communication overhead is $S_{str} + 2S_g + S_{hash} + S_{str} + S_{str} + S_{sig} = 3S_{str} + S_{hash} + S_{sig}$. The transponder 1 uploads a deliver transaction $\mathrm{Tx}^{\mathrm{del}_1}$ with a length of $3S_{str} + S_g + S_{hash} + S_{sig}$. In Distribute, the collector uploads a store transaction $\mathrm{Tx}^{\mathrm{sto}}$, $(n+1)$ distribute transactions $\{\mathrm{Tx}^{\mathrm{dis}}\}$, and a ship transaction $\mathrm{Tx}^{\mathrm{shi}}$. The total communication overhead is $(2n+6)S_{str} + (3n+5)S_g + (n+3)S_{hash} + (n+3)S_{sig}$. The transponder 2 uploads the same amount of messages as the transponder 1. In Receive, the donee sends a feedback message of length S_{enc} and a receive transaction $\mathrm{Tx}^{\mathrm{rec}}$ of length $2S_{str} + S_{hash} + S_{sig}$.

7.6 Network Delay

We test the consensus time of the underlying blockchain network and the transaction confirmation time. We set the block creation time to be 20 seconds. Experimental results in Fig. 5a show that the real consensus time fluctuates around 20 seconds due to the hardware interference. We test the transaction confirmation time by using *waitForTransactionReceipt()* and record elapsed time by using *time.time()*. We set 5 seconds as the interval time between two donation transactions. Experimental results in Fig. 5b show that the confirmation time for the donators varies due to network delay and consensus mechanism.

7.7 Gas Costs and Monetary Costs

The monetary costs originate from the gas costs of invoking Ethereum SCs. In Astraea, there are eight types of transactions: $\mathrm{Tx}^{\mathrm{don}}$, $\mathrm{Tx}^{\mathrm{del}_1}$, $\mathrm{Tx}^{\mathrm{sto}}$, $\mathrm{Tx}^{\mathrm{shi}}$, $\mathrm{Tx}^{\mathrm{dis}}$, $\mathrm{Tx}^{\mathrm{del}_2}$, $\mathrm{Tx}^{\mathrm{rec}}$, and $\mathrm{Tx}^{\mathrm{ref}}$. The test script was written in Python 3.7 with web3py library, and we collect the GasUsed value through transaction receipts. We repeated the experiment for 1 to 10 ride requests and obtained the gas costs for each function in the SC. The ether price is \$2532.83 on January 23, 2022[3] and the gas price is 1 Gwei (1 Gwei $= 10^{-9}$ wei). The gas cost and monetary cost for each type of transactions are shown in Figs. 6a and 6b where it only costs \$0.44 to send a donation transaction.



(a) Gas Cost (b) Monetary Cost

Fig. 6. Gas costs and monetary costs.

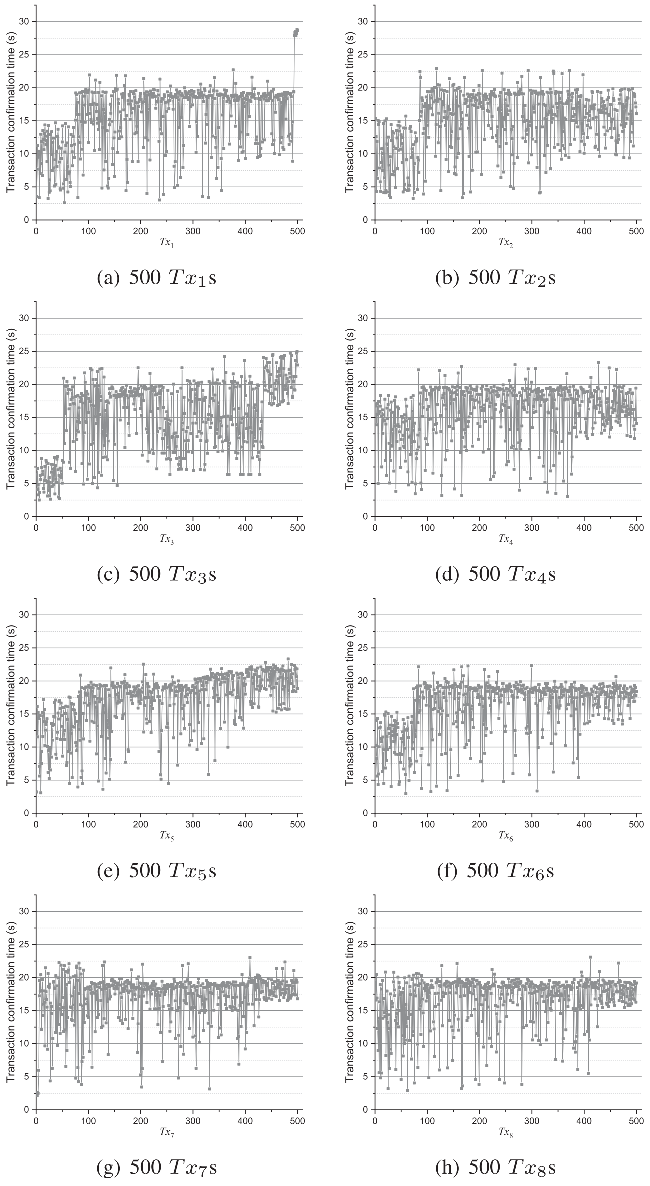
7.8 Scalability

Scalability refers to how the blockchain scales with the increase in the network size [53]. We create 500 donators using geth client. Each account receives 1000 ether from one pre-funded miner account. Since web3py is a none thread-safe library, we write transaction scripts in a loop. We measure the transaction confirmation time for the eight types of transactions. Experimental results in Fig. 7 indicate that the total time needed for reaching consensus of a new block is stable regardless of the network size. Therefore, we can support more donators to participate in the system. Furthermore, authorities can decrease block interval and increase block size according to the increase in number of donations.

7.9 Comparison With Existing Work

We also compare Astraea with existing work: Delivery1 [11], Delivery2 [12], Tracking [13], Solidus [6], and zkLedger [20] in terms of communication and computation costs. The comparison results are recorded in Tables 2 and 3. In [11], we consider seller as donator and buyer as donee. The scheme is rather simple for only hashing keys, signing transactions, and sending string keys and signatures. The operations in [12] is similar to [11] while supporting multiple transponders. In [13], an NGO, as a collector, submits a donation request for each donation to an SC. Donators interacts with the SC via buying tokens and donating tokens. In Solidus [6], a sending user (donator) signs a transaction (transfer a value v to a receiving user at bank 2) and gives it to bank 1, which decrypts two ciphertexts, generates a range proof, encrypts the value and a public key, and updates its publicly-verifiable oblivious RAM machine. The bank 2 generates a range proof and produces a signature. We see the two banks as a whole, i.e., a collector. The range proof of $v \in [0, 2^t)$ requires $5 + 10t$ multiplications on elliptic curve and t encryptions. The proof verification requires $7 + 12t$ multiplications. In zkLedger, a transaction happens between a sending bank and a receiving bank. Each bank has to send a commitment, an audit token, a proof-of-asset, a proof-of-balance, and a proof-of-consistency. The first three comparison schemes have lower costs and overhead for using simplified framework and lacking security mechanisms. ZkLedger's proof generation time is almost twelve times than ours and it increases with n. The comparison results show that Delivery1 [11] and Delivery2 [12] have the lowest cost/overhead for donators and donees for using a simplified system; Astraea performs the best and the same with Delivery1 [11] and Delivery2 [12] regarding two transponders' computational costs (not the communication

[3] https://coinmarketcap.com



(a) 500 Tx_1s

(b) 500 Tx_2s

(c) 500 Tx_3s

(d) 500 Tx_4s

(e) 500 Tx_5s

(f) 500 Tx_6s

(g) 500 Tx_7s

(h) 500 Tx_8s

Fig. 7. Scalability of astraea.

overhead for sending extra information); Astraea does not stand out for collector's computational cost due to the generation of PoA and PoD (but is communication overhead is less than Solidus [6]). Although Astraea does not perform the best, the other schemes cannot achieve all objectives in a donation, while Astraea provides strong donation privacy.

8 Discussions

8.1 Other Forms of Donation Chain

It is possible that other forms of donation chains exist. For example, a donator directly drops an asset at a collector's front gate and the donation chain is donator-collector-transponder-donee. For other chains, we can also design corresponding smart contract and adapt to their settings.

8.2 Donation Privacy

Public donation releases the identity of donators and the donation information. Astraea is essentially a private donation system that protects the number of a donation, the sum of donation numbers, and identity privacy of donators.

The number of a donation is important for three reasons. (1) Sensitivity of the assets. Some assets are sensitive and they convey the privacy of donators. For example, donating a large number of facial masks during the pandemic may reveal that the donator has a stable source of facial masks and plenty of money. This will attract attention from malicious parties who will list the donator as a robbery target. When donating antiques, famous paintings, and manuscripts to a museum, one item may not stand out, but three or more items probably link the donator to a specific historical event, which the donator tries to lie deep in the heart. (2) Importance of the assets. Some assets are vital to donees' basic living and they should be protected during transit. For example, one hospital/state donates organs/food to another hospital/state that falls short of supply. Such assets are moving targets for malicious merchants who attempt to loot the assets to trade for money. (3) Moral abduction. Some celebrities prefer to hide their donation number in case of moral abduction. If a famous movie star does not donate the expected number of money to a red cross at a place after an earthquake, she/he may be facing criticism from unfriendly cyber users or fans who dislike her/him.

Sum of Donation Numbers. The explanation above induces the importance of the sum of donation numbers. The collector in Astraea may become a target if the sum is revealed after receiving and distributing all donations.

Identity privacy of donator and donee exists in two worlds: cyber world and physical world. Although there are few works addressing the identity privacy in physical world, it is not difficult to see that a transponder actually can see a donator's face (some kind of identity) during asset collection, thus violating identity privacy. To mitigate this attack, we can ask transponder to retrieve the asset from a logistics station to avoid the direct contact between donator and transponder. In the cyber world, if a donator only makes a one-time donation, then the identity privacy is well protected. For multiple donations, we can ask the donators to register multiple Ethereum accounts and use a new account for each donation.

8.3 Security Concerns of Trusted Hardware

Although there exist practical attacks on the trusted hardware, there have been many defense methods. For example, program obfuscation for branch prediction attack [54], modifying kernel's memory management service for memory attack [55], partitioning the set associative memory structures for contention-based and access-based cache attacks [56], and padding executing time to a constant by dummy loop for timing-channel attack [57]. Another attack is rollback attack, where the adversary violates the integrity of a protected application state by replaying old persistently stored data or by starting multiple application instances [58]. Matetic et al. [58] designed a rollback protection system named ROTE to protect integrity as a distributed system between multiple enclaves running on separate processors. The defense idea is based on the fact that the owner of processors can assign multiple processors to assist each other.

8.4 Adoption of TEE and Hyperledger Fabric

Hyperledger Fabric is a prominent permissioned blockchain framework. The adoption of TEE and Hyperledger Fabric is also possible. Brandenburger et al. [59] proposed a solution for secure smart-contract execution on a blockchain using TEE (Intel SGX) and Hyperledger Fabric. It is a modular platform for consortium blockchains and it offers a notion of consensus whose outputs are always final to avoid the protocol-inherent rollback attack. We did not compare with the Hyperledger Fabric-based framework because our research group has been conducting experiments with the widely adopted Ethereum platform (we refer the interested readers to https://github.com/UbiPLab for our codes). However, we believe that it is interesting to realize TEE upon Hyperledger Fabric and verify computation/communication costs. We believe there will be slight differences regarding efficiency given their differences on objective, confidentiality, consensus, and language.

8.5 Extensions

We put the donation at the heart of the paper for three reasons. First, the social and economic disruption caused by the COVID-19 is devastating, and people around the world have been fighting against the pandemic for more than two years. Second, people of goodwill are donating facial masks to slow down the spread of the virus but we are also witnessing the loss and theft of valuable donations. Third, the decentralized donation systems have some unique features, especially on privacy and security, and there are corresponding technical challenges to be addressed, which are worthy to be investigated.

The system is a case study and it can be extended to other use cases (with some small modifications accordingly) where goods (materials) in transit are important to senders or receivers and auditing is essential to all stakeholders. First, the system can be a private delivery system that transports precious antiques and personal documents. Second, Astraea applies to important medical donation systems that transport organs (e.g., heart and kidney) for a patient far away. In such systems, donation privacy is quite important due to the nature of the assets. Third, Astraea is suitable for a supply chain where food, e.g., water, fruits, and plants, is delivered to places running short of supply. Such a case calls for auditing.

9 Conclusion

We have proposed Astraea to realize decentralized, anonymous, and secure auditing for donation systems. Based on private smart contracts, Astraea provides donation privacy for donators and security against collusion attacks and stealing attacks. Donators can make donations without privacy concerns on their identity and donation, the collectors can efficiently process the donations in a batch manner, the donators and an auditor can verify the distribution. We formally define and prove the privacy and security of Astraea. We implement a prototype based on Ethereum and SGX to evaluate its performance. The experimental results indicate that Astraea exhibits practical performance.

References

[1] "WHO coronavirus (COVID-19) dashboard," Accessed: Feb. 6, 2022. [Online]. Available: https://covid19.who.int

[2] A. Hufford, "Face masks are again in short supply as Covid-19 cases surge," Accessed: Feb. 6, 2022. [Online]. Available: https://www.wsj.com/articles/face-masks-are-again-in-short-supply-as-covid-19-cases-surge-11604499588

[3] Y. Cheng et al., "Face masks effectively limit the probability of SARS-CoV-2 transmission," *Science*, vol. 372, no. 6549, pp. 1439–1443, 2021.

[4] "Protecting donor privacy - Philanthropic freedom, anonymity and the first amendment," 2021. Accessed: Feb. 6, 2022. [Online]. Available: https://www.philanthropyroundtable.org/docs/default-source/default-document-library/protecting-philanthropic-privacy_white_paper.pdf

[5] M. Cerullo, "Supplies of N95 masks running low as COVID-19 surges," 2020. Accessed: Feb. 6, 2022. [Online]. Available: https://www.cbsnews.com/news/ppe-n95-mask-shortage-covid-19

[6] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi, "Solidus: Confidential distributed ledger transactions via PVORM," in *Proc. 24th ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 701–717.

[7] Eleo, 2022. Accessed: Feb. 6, 2022. [Online]. Available: https://eleoonline.com/capterra-link/?utm_source=GetApp

[8] DonorSnap, 2022. Accessed: Feb. 6, 2022. [Online]. Available: https://info.gartnerdigitalmarkets.com/donorsnap-gdm-lp?category=donation-management&utm_source=GetApp

[9] easyTithe, 2022. Accessed: Feb. 6, 2022. [Online]. Available: https://www.easytithe.com/ppc/join

[10] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Dependale Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.

[11] H. R. Hasan and K. Salah, "Blockchain-based solution for proof of delivery of physical assets," in *Proc. 3rd Int. Conf. Blockchain*, 2018, pp. 139–152.

[12] H. R. Hasan and K. Salah, "Blockchain-based proof of delivery of physical assets with single and multiple transporters," *IEEE Access*, vol. 6, pp. 46 781–46 793, 2018.

[13] A. Singh, R. Rajak, H. Mistry, and P. Raut, "Aid, charity and donation tracking system using blockchain," in *Proc. IEEE 4th Int. Conf. Trends Electron. Inform.*, 2020, pp. 457–462.

[14] "Donor privacy policy," 2015, Accessed: Feb. 6, 2022. [Online]. Available: https://www.feedingamerica.org/privacy-policy/donor-privacy-policy

[15] S. Parnell, "The legal and political landscape of donor privacy," 2017. Accessed: Feb. 6, 2022. [Online]. Available: https://www.philanthropyroundtable.org/philanthropy-magazine/article/spring-2017-the-legal-and-political-landscape-of-donor-privacy

[16] M. Breuer, U. Meyer, S. Wetzel, and A. Mühlfeld, "A privacy-preserving protocol for the kidney exchange problem," in *Proc. 19th Workshop Privacy Electron. Soc.*, 2020, pp. 151–162.

[17] J. Jeong, D. Kim, Y. Lee, J. -W. Jung, and Y. Son, "A study of private donation system based on blockchain for transparency and privacy," in *Proc. IEEE Int. Conf. Electron. Informat. Commun.*, 2020, pp. 1–4.

[18] K. Rivera, "Red cross moves thousands of masks stored at Port of Stockton to 'more secure location'," 2020. Accessed: Feb. 6, 2022. [Online]. Available: https://www.abc10.com/article/news/health/coronavirus/red-cross-moves-thousands-of-masks-stored-at-port-of-stockton/103-d15c4299-bf72–4289-8b9a-af71da784482

[19] A. Sternlicht, "Almost $800 Million in mask scams alleged in the U.S. alone," 2020. Accessed: Feb. 6, 2022. Avalaible: https://www.forbes.com/sites/alexandrasternlicht/2020/05/27/almost-800-million-in-mask-scams-alleged-in-the-us-alone/?sh=3b3dbbf712be

[20] N. Narula, W. Vasquez, and M. Virza, "zkLedger: Privacy-preserving auditing for distributed ledgers," in *Proc. 15th USENIX Symp. Networked Syst. Des. Implementation*, 2018, pp. 65–80.

[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009, Accessed: Feb. 6, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[22] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, Apr.–Jun. 2019.

[23] "Ethereum," 2022. Accessed: Feb. 6, 2022. [Online]. Available: https://ethereum.org/en

[24] M. Li, Y. Chen, C. Lal, and M. Conti, M. Alazab, and D. Hu, "Eunomia: Anonymous and secure vehicular digital forensics based on blockchain," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2021.3130583.

[25] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE 37th Symp. Secur. Privacy*, 2016, pp. 839–858.

[26] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An authenticated data feed for smart contracts," in *Proc. 23rd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 270–282.

[27] J. Guarnizo and P. Szalachowski, "PDFS: Practical data feed service for smart contracts," in *Proc. 24th Eur. Symp. Res. Comput. Secur.*, 2019, pp. 767–789.

[28] F. McKeen et al., "Innovative instructions and software model for isolated execution," in *Proc. 2nd Int. Workshop Hardware Architectural Support Secur. Privacy*, 2013, pp. 1–8.

[29] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *Proc. 2nd Int. Workshop Hardware Architectural Support Secur. Privacy*, 2013, pp. 1–7.

[30] Intel, "Intel® software guard extensions (Intel® SGX) developer guide," 2022. Accessed: Feb. 6, 2022. [Online]. Available: https://download.01.org/intel-sgx/latest/linux-latest/docs/Intel_SGX_Developer_Guide.pdf

[31] C. Cai, Y. Zheng, Y. Du, Z. Qin, and C. Wang, "Towards private, robust, and verifiable crowdsensing systems via public blockchains," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1893–1907, Jul./Aug. 2021.

[32] W. Dai, C. Dai, K.-K. R. Choo, C. Cui, D. Zou, and H. Jin, "SDTE: A secure blockchain-based data trading ecosystem," *IEEE Trans. Informat. Forensics Secur.*, vol. 15, pp. 725–737, 2020.

[33] S. Li, K. Xue, D. S. L. Wei, H. Yue, N. Yu, and P. Hong, "SecGrid: A secure and efficient SGX-enabled smart grid system with rich functionalities," *IEEE Trans. Informat. Forensics Secur.*, vol. 15, pp. 1318–1330, 2020.

[34] S. Dziembowski, L. Eckey, and S. Faust, "FairSwap: How to fairly exchange digital goods," in *Proc. 25th ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 967–984.

[35] G. Malavolta, P. Moreno-Sanchez, C. Schneidewindy, A. Katez, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

[36] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in Industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6564–6574, Oct. 2020, doi: 10.1109/TII.2020.2974537.

[37] R. Cheng et al., "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2019, pp. 185–200.

[38] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1353–1370.

[39] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for Internet-scale systems," in *Proc. 21st USENIX Annu. Tech. Conf.*, 2010, pp. 1–11.

[40] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Oper. Syst. Des. Implementation*, 1999, pp. 173–186.

[41] I. Bilogrevic, M. Jadliwala, V. Joneja, K. Kalkan, J. Hubaux, and I. Aad, "Privacy-preserving optimal meeting location determination on mobile devices," *IEEE Trans. Informat. Forensics Secur.*, vol. 9, no. 7, pp. 1141–1156, Jul. 2014.

[42] X. Wang, Y. Mu, and R. Chen, "One-round privacy-preserving meeting location determination for smartphone applications," *IEEE Trans. Informat. Forensics Secur.*, vol. 11, no. 8, pp. 1712–1721, Aug. 2016.

[43] U. Chatterjee, R. Subhra Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Trans. Embedded Comput. Syst.*, vol. 67, pp. 1–25, 2017.

[44] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *Proc. 24th ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1465–1482.

[45] T. P. Pedersen, "Non-interactive and information theoretic secure verifiable secret sharing," in *Proc. 8th Annu. Int. Cryptol. Conf.*, 1991, pp. 129–140.

[46] F. Schuster et al., "VC3: Trustworthy data analytics in the cloud using SGX," in *Proc. IEEE 36th Symp. Secur. Privacy*, 2015, pp. 38–54.

[47] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," *Cryptol. ePrint Arch.*, 2006, Accessed: Jun. 1, 2022. [Online]. Available: https://eprint.iacr.org/2004/332

[48] Y. Lindell, "How to simulate it – A tutorial on the simulation proof technique," in *Proc. Tutorials Foundations Cryptography*, Berlin, Germany: Springer, 2017, pp. 277–346.

[49] R. Yang, M. H. Au, Q. Xu, and Z. Yu, "Decentralized blacklistable anonymous credentials with reputation," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2018, pp. 720–738.

[50] C. Liu et al., "Extending on-chain trust to off-chain – Trustworthy blockchain data collection using trusted execution environment," *IEEE Trans. Comput.*, to be published, doi: 10.1109/TC.2022.3148379.

[51] Z. Voell, "Ethereum classic Hit by third 51% attack in a month," 2021. Accessed: Jun. 28, 2022. [Online]. Available: https://www.coindesk.com/markets/2020/08/29/ethereum-classic-hit-by-third-51-attack-in–month

[52] B. Rirus, "Buterin helping to strategize against Ethereum 51% attack possibility," 2021. Accessed: 28, 2022. [Online]. Available: https://cointelegraph.com/news/buterin-helping-to-strategize-against-ethereum-51-attack-possibility

[53] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. 23th ACM Conf. Comput. Commun. Secur.*, 2016, pp. 17–30.

[54] S. Lee, M. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside SGX enclaves with branch shadowing," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 557–574.

[55] S. Zhao, Q. Zhang, Y. Qin, W. Feng, and D. Feng, "Sectee: A software-based approach to secure enclave architecture using tee," in *Proc. 26th ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1723–1740.

[56] G. Dessouky, T. Frassetto, and A.-R. Sadeghi, "Hybcache: Hybrid side-channel-resilient caches for trusted execution environments," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 451–468.

[57] I. Puddu, M. Schneider, M. Haller, and S. Capkun, "Frontal attack: Leaking control-flow in SGX via the CPU frontend," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 663–680.

[58] S. Matetic et al., "ROTE: Rollback protection for trusted execution," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 1289–1306.

[59] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, "Trusted computing meets blockchain: Rollback attacks and a solution for Hyperledger Fabric," in *Proc. IEEE 38th Symp. Reliable Distrib. Syst.*, 2019, pp. 324–333.

Meng Li (Senior Member, IEEE) received the PhD degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology, China, in 2019. He is an associate researcher and dean assistant with the School of Computer Science and Information Engineering, Hefei University of Technology, China. He is also a postdoctoral fellow with the Department of Mathematics and HIT Center, University of Padua, Italy, where he is with the Security and Privacy Through Zeal (SPRITZ) research group led by Prof. Mauro Conti. He was sponsored by ERCIM 'Alain Bensoussan' Fellowship Programme (from October 1, 2020 to March 31, 2021) to conduct postdoctoral research supervised by Prof. Fabio Martinelli at CNR, Italy. He was sponsored by China Scholarship Council (CSC) (from September 1, 2017 to August 31, 2018) for joint Ph.D. study supervised by Prof. Xiaodong Lin in the Broadband Communications Research (BBCR) Lab, University of Waterloo and Wilfrid Laurier University. His research interests include security, privacy, fairness, vehicular networks, applied cryptography, privacy computing, cloud computing, edge computing, blockchain. He has published more than 50 papers in international peer-reviewed transactions, journals, magazines, and conferences, including *IEEE Transactions on Dependable and Secure Computing*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Network Science and Engineering*, *IEEE Transactions on Green Communications and Networking*, *IoT Journal*, *Information Sciences*, FGCS, *IEEE Communications Magazine*, *IEEE Wireless Communication*, MobiCom, ICICS, SecureComm, TrustCom, and IPCCC.

Yifei Chen received the MS degree from the School of Computer Science and Information Engineering, Hefei University of Technology. He was rewarded the China National Graduate Student Scholarship Award. He is now a security researcher with the Solution and Architecture Research Department, NSFOCUS. His research interests include cloud computing, blockchain, and privacy computing.

Liehuang Zhu (Senior Member, IEEE) received the MS degree in computer science from Wuhan University, Wuhan, China, in 2001, and the PhD degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2004. He is currently a full professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include data security and privacy protection, and blockchain applications, AI Security. He has authored more than 150 journal and conference papers in these areas. He is an associate editor of *IEEE Transactions on Vehicular Technology*, *IEEE Network and IEEE Internet of Things Journal*, and a guest editor of special issue of *IEEE Wireless Communications*, *IEEE Transactions on Industrial Informatics*. He has served as Program co-chair of MSN 2017, IWWS 2018, and INTRUST 2014. He received the Best Paper Award at IEEE/ACM IWQoS 2017, IEEE TrustCom 2018, IEEE IPCCC 2014.

Zijian Zhang received the PhD degree from the School of Computer Science and Technology, Beijing Institute of Technology. He is now an associate professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He was a research fellow with the School of Computer Science, University of Auckland. He was a visiting scholar with the Computer Science and Engineering Department, State University of New York at Buffalo, in 2015. His research interests include design of authentication and key agreement protocol and analysis of entity behavior and preference.

Jianbing Ni (Senior Member, IEEE) received the BE and MS degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively, and the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2018. He is currently an assistant professor with the Department of Electrical and Computer Engineering, Queen’s University, Kingston, ON, Canada. His current research interests include applied cryptography and network security, with a focus on cloud computing, smart grid, mobile crowdsensing, and the Internet of Things.

Chhagan Lal (Member, IEEE) received the PhD degree in computer science and engineering from the Malaviya National Institute of Technology, Jaipur, India, in 2014. He is currently working as a postdoctoral research fellow with the Delft University of Technology, Netherland. Previously, he was a postdoctoral fellow with the Department of Mathematics, University of Padua, Italy, where he was part of the SPRITZ research group. He was a postdoctoral research fellow with Simula Research Laboratory, Norway. During his PhD, he has been awarded with the Canadian Commonwealth Scholarship under the Canadian Commonwealth Scholarship Program to work in University of Saskatchewan, Saskatoon, SK, Canada. His current research research interests include applications of blockchain technologies, security in software-defined networking, and Internet of Things networks.

Mauro Conti (Fellow, IEEE) received the PhD degree from the Sapienza University of Rome, Italy, in 2009. He is full professor with the University of Padua, Italy. He is also affiliated with TU Delft and University of Washington, Seattle. After his Ph.D., he was a postdoc researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as assistant professor with the University of Padua, where he became associate professor in 2015, and full professor in 2018. He has been visiting researcher with GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a fellowship by the German DAAD (2013). His research is also funded by companies, including Cisco, Intel, and Huawei. His main research interests include the area of Security and Privacy. In this area, he published more than 400 papers in topmost international peer-reviewed journals and conferences. He is editor-in-chief for *IEEE Transactions on Information Forensics and Security*, area editor-in-chief for *IEEE Communications Surveys & Tutorials*, and has been associate editor for several journals, including *IEEE Communications Surveys & Tutorials*, *IEEE Transactions on Dependable and Secure Computing*, and *IEEE Transactions on Network and Service Management*. He was Program chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, and General Chair for SecureComm 2012, SACMAT 2013, NSS 2021 and ACNS 2022. He is senior member of the ACM, and fellow of the Young Academy of Europe.