Prov-trust: towards a trustworthy SGX-based data provenance system

Nesrine Kaaniche, Sana Belguith, Maryline Laurent, Ashish Gehani, Giovanni Russello

Prov-Trust: Towards a Trustworthy SGX-based Data Provenance System

Nesrine Kaaniche[1][a], Sana Belguith[2][b], Maryline Laurent[3][c], Ashish Gehani[4] and Giovanni Russello [5]

[1]*Department of Computer Science, University of Sheffield, Sheffield, UK*

[2]*School of Science, Engineering and Environment, University of Salford, Manchester, UK*

[3]*Telecom SudParis, Institut Polytechnique de Paris, France*

[4]*Computer Science Laboratory, SRI International, USA*

[5]*Cyber Security Foundry, The University of Auckland, New Zealand*

Keywords: Intel SGX, privacy preserving, data provenance, blockchain, data integrity

Abstract: Data provenance refers to records of the inputs, entities, systems, and processes that influence data of interest, providing a historical record of the data and its origins. Secure data provenance is vital to ensure accountability, forensics investigation of security attacks and privacy preservation. In this paper, we propose Prov-Trust, a decentralized and auditable SGX-based data provenance system relying on highly distributed ledgers. This consensually shared and synchronized database allows anchored data to have public *witness*, providing tamper-proof provenance data, enabling the transparency of data accountability, and enhancing the secrecy and availability of the provenance data. Prov-Trust relies on Intel SGX enclave to ensure a trusted execution of the provenance kernel to collect, store and query provenance records. The use of SGX enclave protects data provenance and users' credentials against malicious hosting and processing parties. Prov-Trust does not rely on a trusted third party to store provenance data while performing their verification using smart contracts and voting process. The storage of the provenance data in Prov-Trust is done using either the log events of Smart Contracts or blockchain's transactions depending on the provenance change event, which enables low storage costs. Finally, Prov-Trust ensures an accurate privacy-preserving auditing process based on blockchain traces and achieved thanks to events' logs that are signed by SGX enclaves, transactions being registered after each vote session, and sealing the linking information using encryption schemes.

1 INTRODUCTION

Data provenance refers to the chain of ownership of a piece of data that covers who created it, who modified and accessed it (Moreau et al., 2010). Data provenance is important to investigate security incidents where these data are analysed to detect malicious actions, data breaches and access policy violations. Many solutions have been proposed to collect, store and manage provenance data (Zafar et al., 2017). However, many challenges have been raised by these systems as they need to ensure both (i) a secure collection and verifiability/integrity of provenance data, and (ii) the confidentiality and secure storage of collected data.

The increased use of cloud computing services adds new challenges to data provenance systems. Security of data stored in clouds is of great importance, as the user delegates the management of his data to a service provider who is not considered as a fully trusted entity. Therefore, data provenance is more and more needed in cloud computing services to ensure users' data auditing and accountability. Managing data provenance in clouds is challenging as these data may reveal private information about the stored data or the users. Thus, there is a need to protect the provenance data against unauthorized access while preserving the privacy of users. To address these challenges, several secure provenance systems have been introduced. Some systems have implemented cryptographic mechanisms such as homomorphic encryption to search over provenance data (Asghar et al., 2012) or attribute based encryption to ensure encrypted access control over provenance data (Belguith et al.,). Some other research works have addressed the secure collection of provenance data relying on trusted software and/or hardware tools such

[a] https://orcid.org/0000-0002-1045-6445
[b] https://orcid.org/0000-0003-0069-8552
[c] https://orcid.org/0000-0002-7256-3721

as Trusted Platform Module (TPM) (Taha et al., 2015; Ko and Will, 2014). Recently, Blockchain technology have been leveraged to ensure a tamper-proof and available data provenance systems (Li et al., 2019; Ramachandran and Kantarcioglu, 2018).

Contributions — In this paper, we present Prov-Trust, an Intel SGX and blockchain based data provenance architecture that provides a secure and privacy preserving data provenance generation and storage.

Prov-Trust relies on the distributed tamper-proof nature of the blockchain technology, cryptographic techniques and trusted hardware, i.e, Intel SGX to securely collect and store data provenance while preserving the privacy of sensitive information. Operations over data are monitored in real time to collect provenance data, that support the compliance of access control policy enforcement and intrusion detection based on provenance middlewares (Gehani and Tariq, 2012).

Prov-Trust relies on Intel SGX enclave to ensure a trusted execution of the provenance kernel to collect, store and query provenance records. The use of SGX enclave protects data provenance and users' credentials against malicious hosting and processing parties. The Prov-Trust framework does not rely on a trusted third party to store provenance data while performing their verification using smart contracts and voting process. Indeed, a smart contract is used to monitor the changes performed over a document while implementing access control policies and maintaining all users' accesses/modification to the document. The vote contract allows a set of *designated* users to vote for/against the change of a set of defined provenance change events, namely "download" and "transfer". The storage of the provenance data in Prov-Trust is done using either the log events of Smart Contracts or blockchain's transactions depending on the provenance change event, which enables low storage costs.

Finally, Prov-Trust ensures an accurate privacy-preserving auditing process based on blockchain traces and achieved thanks to events' logs are signed by SGX enclaves, transactions being registered after each vote session, and enciphering of linking information with respect to an attribute based encryption and re-randomizable encryption schemes (Kaaniche and Laurent, 2018).

Paper organization — This paper is organized as follows. Section 2 presents a literature review of data provenance solutions while defining security and functional requirements for the proposed architecture. An overview of Prov-Trust underlying technologies namely, Blockchain and Intel SGX is introduced in Section 3. Afterwards, we define the network model and we present an overview of Prov-Trust phases in Section 4. The detailed phases of Prov-trust are presented in Section 5 before introducing the security and privacy analysis in Section 6. The paper is concluded and future works are presented in Section 8.

2 Related Work and Requirements

In this section, we first review data provenance solutions. Then, we details the security and privacy requirements that have to be fulfilled by the proposed solution.

2.1 Data Provenance Solutions

Data provenance refers to the chain of ownership of a piece of data that covers who created it, who modified and accessed it (Moreau et al., 2010). Data provenance is used to investigate security incidents where data provenance is analysed to detect malicious actions, data breaches and access policy violations. Extensive research has been conducted on data provenance in database, file workflow and operating systems (Suen et al., 2013) and many provenance management systems have been proposed (Zafar et al., 2017). PASS is one of the first systems that have been designed to provide automatic collection and maintenance of provenance data (Muniswamy-Reddy et al., 2006). Improved versions of PASS that supports the integration of multiple levels of abstractions have been introduced after its first release (Muniswamy-Reddy et al., 2008). PASSv2 operates within the three levels of abstraction: the system call boundary, a workflow specification language, and a domain-specific application level. HiFi is a kernel level provenance system that uses Linux Security Modules to gather provenance data (Pohly et al., 2012). These generated data can be forensically analysed to detect malicious activities thanks to the tracking of the kernel and application actions within the Linux system. SPADE (Gehani, 2016) is a provenance middleware providing users with a framework to collect, store and query provenance data that are generally data graphs describing the history of processes ran over a piece of data. Open Provenance Model (OPM) is a methodology representing the provenance graphs in three types of nodes: artifacts, processes and agents (Moreau et al., 2011).

The emergence of cloud computing created a need to adapt/develop provenance systems to track data changes in such highly distributed systems. Collecting provenance data is particularly challenging in complex systems such as cloud computing that presents multiple layers of interacting software and

hardware. Cloud computing systems are dynamic and heterogeneous systems involving several components provided by different providers. Many data provenance systems have been designed for cloud computing. SPADEv2 (Gehani, 2016) has been introduced as an open source tool supporting both graph and relational database storage and querying in a distributed system which makes it adapted for cloud computing. S2Logger is a solution designed to capture and visualise end-to-end provenance data across all guest and host physical machines in distributed virtualised environments such as cloud computing (Suen et al., 2013).

As provenance data usually contain sensitive information that should be kept secure. Additionally, provenance data can be modified or forged by a malicious attacker to mislead investigation and analysis of events logs. Assuring a trusted data provenance will enable users to verify the operations performed over their data. Therefore, provenance data should fulfill a set of security criteria such as confidentiality, integrity, unforgeability,non-repudiation and availability (Zafar et al., 2017). The Secure Provenance (SPROV) scheme is one of the first provenance solutions that considered security features (Hasan et al., 2009). SPROV automatically collects data provenance besides providing security assurances of confidentiality and integrity of these data. Asghar et al. (Asghar et al., 2012) designed a secure data provenance storage system that enables secure and privacy preserving search over provenance data based on cryptographic algorithms.

To fulfill availability and integrity requirements, Blockchain systems have been used to store and manage provenance data. Blockchain has been first leveraged in data provenance systems in 2017 by introducing Provchain (Liang et al., 2017). Provchain is a data provenance system based on blockchain for a cloud applications. In Provchain, OPM has been used to collect provenance data in real-time. Afterwards, operations history are saved in blockchain transactions. This enables auditors to validate the operations by consulting the transactions receipts. BlockCloud (Shetty et al., 2017) monitors users activities in real time using special classes of events namely, hooks and listeners that enables the generation of provenance data. The provenance data are stored in transactions that are broadcasted to the core of blockchain network created by a specific set of validating virtual machines. Similar to Provchain solution, the provenance auditor validates the data using the transactions receipts. SmartProvenance (Ramachandran and Kantarcioglu, 2018) applies Ethereum Smart Contracts to provide data provenance. SmartProvenance is based on two smart contracts, namely Document Tracker contract and Vote contract. The Document Tracker smart contract is used to monitor the operations performed over a given document while the vote contract specifies the voting protocol. A voting process is used to validate every provenance change event.

These solutions do not address a trusted generation of the provenance data, yet unforgeability is a security requirement that needs to be achieved while generating and storing provenance data. The unforgeability means that an adversary cannot forge a valid provenance record either by modifying any existing one or directly introducing a new forged provenance record without being detected. Hence, unforgeability guarantees the tamper evidence (Asghar et al., 2012; Taha et al., 2015). A first attempt to use Trusted Platform Module to ensure trust in provenance generation had been introduced by Bock et al. (Böck et al., 2010). This solution presented a data provenance collection solution using a TPM chip and the AMD Secure Virtual Machine (SVM) technology. This solution has been enhanced further by Taha et al. (Taha et al., 2015) by adding the integrity and availability of the provenance data to the solution.

In our proposed solution Prov-Trust, we leverage Blockchain technology and trusted hardware; the Intel SGX to provide trustworthy and secure provenance data.

2.2 Security and Functional Requirements

Prov-Trust has to fulfill the following security and functional requirements, defined as:

- **data confidentiality of provenance data —** the proposed framework has to ensure the secrecy of collected provenance data as well as associated meta-data.
- **verifiability/integrity —** the integrity of provenance records is important during their storage, processing, and transport. Given the nature of most of the provenance data, integrity is generally considered as the most important assurance to achieve trustworthy provenance.
- **accountability —** accountability refers to the traceability capability. In fact, as data are stored and processed in remote servers, a data owner should be provided with a complete transparent view over data collection, access and processing operations.
- **auditability** — the proposed solution has to enable authorized auditing authorities to initiate an investigation and obtain consistent proofs when non-compliant activities have been recorded.

- **privacy preservation —** the privacy requirement mainly refers to the unlinkability. For instance, Prov-Trust has to guarantee that created smart contracts by the same data owner can not be linked by public verifiers as well as non-authorized service providers or/and users. Furthermore, BC-transactions associated to the same data should not be linked.

3 Background

In the following, we introduce the technologies that we use to develop Prov-Trust. We first detail the trusted hardware Intel SGX in subsection 3.1, then the Blockchain technology in subsection 3.2.

3.1 Intel SGX

The trusted hardware has been introduced to solve the issue of secure remote computation, which means that a user can securely execute a software on a distant untrusted party. The trusted hardware creates a secure container enabling a user to upload his software and data to the secure container where computations are performed while protecting their confidentiality and integrity. Trusted hardware solutions have been introduced such as TPM (Main, 2007) and TXT (Grawrock, 2009).

Intel's Software Guard Extensions (SGX) is the latest iteration of designing trusted hardware solutions (McKeen et al., 2013). Intel SGX is a set of CPU extensions enabling the creation of a trusted isolated execution environments. This isolated execution environment which is referred to as Enclave, enables running an application residing in the enclave while protecting its confidentiality and integrity from other software including an untrusted operating system. Physically, the enclave is located inside a hardware guarded area of memory called Enclave Page Cache (EPC).

Besides isolated code and execution, Intel SGX enables two more functionalities which are sealing and attestation. Sealing is defined as the procedures of storing and encrypting private data on a storage disk (Intel, 2016; Cui et al., 2018; Kaaniche et al., 2020). The SGX processor uses a secret key denoted by Root Seal Key that is physically embedded in the hardware by the manufacturer. Upon creating the enclave, a seal key is derived from the root seal key, which will be used to encrypt and store the existent data once the enclave is deleted. The second property is the attestation that is used to prove to a user that his communication is being performed with a software residing and running inside the enclave hosted by the Intel SGX hardware. This attestation is a cryptographic signature certifying the hash of the contents of the enclave. The user verifies the attestation key used to generate the signature against an endorsement certificate created by the trusted hardware's manufacturer. In a nutshell, the SGX attestation enables the user to verify the creation of the enclave in a trusted SXG, the integrity of the data and the code loaded into the enclave, and creating a secure communication channel between the enclave and an external user.

3.2 Blockchain Technology

Blockchain has been very popular since its introduction in 2008. Bitcoin[1] was the first blockchain technology appeared as a means to transfer cryptocurrency without reliance on a central entity which makes it the first fully decentralised transfer system.

Blockchain is a set of transactions grouped together in a distributed ledger that is relying on a combination of technologies such as digital signature, peer to peer networking and cryptographic proof of work (Swan, 2015). Blockchain is a number of transactions or event recorded and saved in a public ledger (Crosby et al., 2016). These records are *shared by all network nodes, updated by miners, monitored by everyone, and owned and controlled by no one* (Swan, 2015). These network nodes are end-users using their personal computers or mobiles while the miners are nodes that have an extensive computational resources used to validate the transaction.

In order to implement the decentralised services and applications of blockchain, two types of blockchain exist currently on the market: public or permissionless blockchain and private or permissioned blockchain. Permissionless blockchain is a framework built on the existent blockchain technology: Bitcoin. This enables the framework to be more resilient, transparent and secure as Bitcoin is already utilised by many users. However, permissionless blockchain needs to mine the blocks every 10 minutes and the used scripting language is not Turing-complete (Swan, 2015).

Permissioned blockchain is a blockchain enabling the same functionalities as the permissionless one such as mining the cyptocurrency, full decentralisation and transactions validation. On top of these functionalities, permissioned blockchain enables establishing contracts referred to as *smart contract* along with an automated and expressive language to write the smart contracts.

[1] https://bitcoin.org/en/

Recently, permissioned blockchains are witnessing an increased interest across multiple domains. This type of blockchain is considered as a promising solution to apply blockchain technology in multiple business applications where end-users do not need to trust each other neither to be identified. Ethereum[2] and the Hyperledger [3] are among the most known permissionless blockchains (Wood, 2014). In permissioned blockchain, there exists a central entity that manages read/write rights operations among participating peers.

In our proposed solution, Prov-Trust, we rely on the Hyperledger to store and manage the provenance data. The Hyperledger project is an initiative launched to facilitate the application of blockchain technologies in business applications. It provides a modular consensus protocol ensuring efficient scalability and enhanced performances while executing thousands of transactions per second (Vukolic, 2017). It is actually developed and supported by a large consortium of world-leading companies.

4 Prov-Trust: A SGX/Blockchain based Data Provenance Scheme

In this section, we first define the network model by detailing the different entities involved in Prov-Trust, in subsection 4.1. Afterwards, we present an overview of the proposed solution 4.2.

4.1 Network Model

As shown in Figure 1, Prov-Trust involves seven entities, defined as follows:

- **Data Owner (DO)** — who owns data contents and has sharing relationship on other users' data, may opt for the provenance service, where the provenance data is stored on the public blockchain. A data owner can be a data user in the case he accesses and makes changes on the document where these changes are logged as provenance data.
- **Data User (DU)** — has access to the document/-data granted by the owner. He is able to change the document and log the changes as provenance trails on the blockchain. Data changes made by the user can be monitored and validated by vote nodes, depending on requested changes, but the nodes may not know about details of other users' activities.
- **Provenance Auditor (PA)** — can retrieve all the provenance data from the blockchain transactions and registered events' logs, w.r.t. associated credentials. Indeed, only authorized PAs can decipher linking information, re-construct the whole provenance graphs and correlate the provenance entry to the data owner.
- **Validators (V)** — are participants, also referred to as voters, who vote for/against the change of a set of provenance change events, namely "download" and "transfer", using a vote protocol. We assume that validators may be chosen based on a reputation system.
- **Blockchain Network (BC)** — consists of globally participating nodes. All the provenance data will be recorded in form of blocks and verified by blockchain nodes.
- **SGX enclave** — is responsible for processing user queries and returning results to users without leaking any content and access pattern to the CS. SGX enclave is also considered to be trusted. In particular, both integrity and confidentiality of the code and data inside the enclave are protected with inherent cryptographic mechanisms. Based on the cryptographic signature provided by the SGX, information provided by SPADE client are certified to be generated by a trusted software.
- **Cloud Service Provider (CSP)** — offers a cloud storage service and is responsible for user registration. CSPs can benefit from the Prov-Trust system in the following aspects. First, they are able to audit the publicly available data changes. In addition, through provenance data, they can also detect intrusion from anomalous behaviours, when privileges are granted .

4.2 Overview

Prov-Trust relies on three main layers, supporting different execution steps (i) the system initialisation, denoted as SYS_INIT, (ii) the data provenance storage phase, i.e., STORAGE, and (iii) the auditing process, referred to as AUDIT. In the following, we provide an overview of different execution layers and detail the interaction flow between entities (Figure 2):

During the SYS_INIT phase, the system is set up and each entity is assigned with a pair public and private keys. Indeed, SGX enclaves are installed on their respective hosting devices and provided with associated keys.

The STORAGE phase encompasses two main protocols, namely the Create and Evt.

[2]https://www.ethereum.org/

[3]https://www.hyperledger.org/

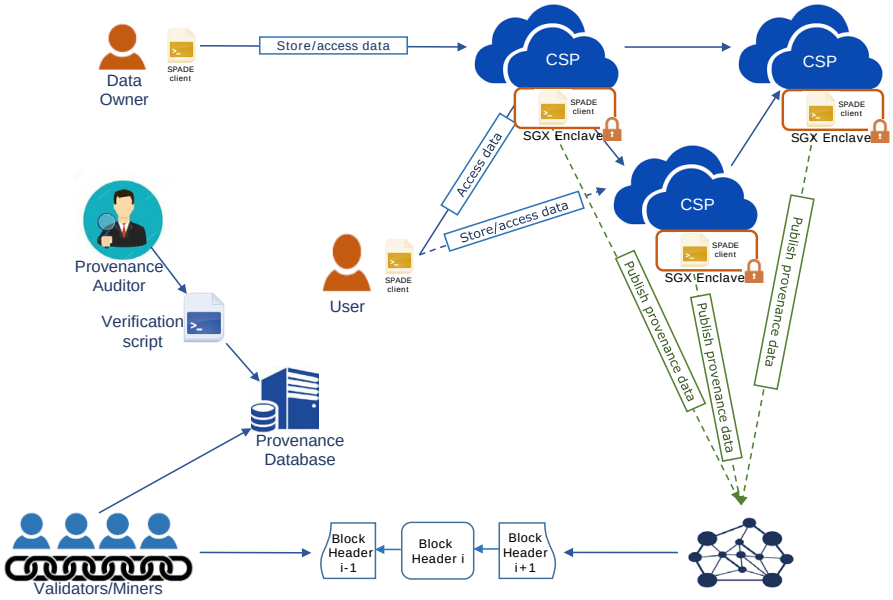

Figure 1: Prov-Trust Architecture

During the Create protocol, the DO has first to create a smart contract (SC), denoted by `prov-doc`, per data file that he intends to track its change/access events. The `prov-doc` contains the Access Control List (ACL) on the data content, i.e., which actions and/or entities are authorized. It also contains the ACL on the events' logs of the associated smart contract. Note that the creation of `prov-doc` has to be anchored into a BC transaction.
The DO may also –optionally– createa vote smart contract associated with the `prov-doc` smart contract. The vote SC, referred to as `vote-doc`, describes the voting protocol associated with the related data file. It specifies the process of voting and the choice of voters.
The `vote-doc` smart contract is mainly required to validate both the `download` and `transfer` actions. Note that the vote is not necessary as the data owner may directly grant the permission to the requesting entity.

For the Evt, two sub-cases are considered, depending on the design choice of the DO. On one hand, when a DU wants to perform some actions on a data file, excluding the `download` and `transfer` actions, an event log will be added to the thread of the `prov-doc` smart contract. Note that each event log has to be signed by the SGX enclave of the requesting entity. On the other hand, when DU wants to do either a `download` or a `transfer` action, the `vote-doc` smart contract is called. At the end of the vote process, a BC transaction needs to be added including the vote result, and signed by a chosen voter.

For the AUDIT phase, Prov-Trust considers two auditing processes. In fact, the auditing process can be performed either by the DO or an authorized provenance auditor. As several parts of the event logs and BC transactions are encrypted, the chaining process, permitting the PA to reconstruct the provenance graph associated to a specific data file, is ensured as described in (Kaaniche and Laurent, 2018).

5 Prov-Trust Phases

In this section, we present the concrete construction of Prov-Trust phases.

5.1 System Initialisation

The SYS_INIT phase includes two different algorithms, defined as follows:

- $\mathsf{Setup}(\xi) \rightarrow (\mathtt{pp}, \mathtt{msk})$ – given the security parameter ξ, this algorithm generates the public parameters pp.
- $\mathsf{KeyGen}(\mathtt{pp}, E_i) \rightarrow (pk_{E_i}, sk_{E_i})$ – derives the public and private keys of an entity E_i, where $E \in \{DO, DU, PA, V\}$, $i \in \{1, N\}$; N is the number of actors supported by the system. It takes as input pp and the entity's identifier E_i. The KeyGen algorithm returns the entity's pair of keys (pk_{E_i}, sk_{E_i}).

We assume that each device d_i, running the SPADE tool, i.e., data provenance algorithm, has a

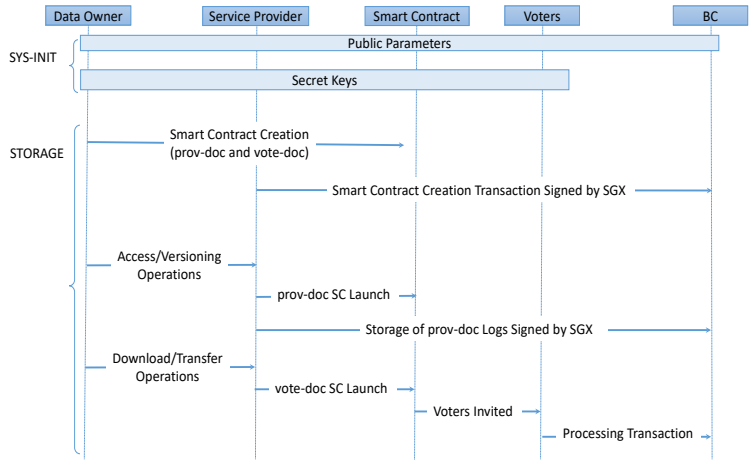

Figure 2: Prov-Trust Workflow

private key denoted by k_{E_i}. This proprietary key is used to sign events associated to DO's data files. Note that a DO may have several devices, thus running different enclaves on each device. Similarly, the CSP runs a different SGX enclave with respect to each storage server.

5.2 Data Provenance Storage

As introduced above, the data provenance STORAGE phase relies on the Create and Evt protocols.

5.2.1 The Create Process

The Create protocol defines the process of creating smart contracts associated to data contents. First, the DO first creates the `prov-doc` smart contract, per data file that he intends to track its access and processing events. The `prov-doc` contains two main parts : the Access Control List (ACL) on both the data content, i.e., which actions and/or entities are authorized, and the events' logs. Note that for ease of presentation, we refer to ACLs to any access control mechanism selected by the DO, to be enforced on his data. ACLs on data contents may be encrypted, however, they have to be accessible to the CSP, the PA as well as voters, if a voting smart contract was created. ACLs on provenance data should ensure the access to involved parties, with respective privileges, as implied by legislation, namely data owners, data processors and controllers, and auditing authorities.

For this purpose, DO establishes a direct secure channel with the CSP, where CSP and DO authenticate each other thanks to their respective (pk_{E_i}, sk_{E_i}) pair of keys (cf. Section 5.1). The creation of `prov-doc` has to be anchored into a BC transaction. The transaction is identified with a T_{id} parameter. Through this channel, DO might initiate a data storage request to the CSP, to infer some parameters for the smart contract prior to launching the smart contract into the blockchain, up-to the approval of the contract by the CSP and transmission of data to its servers.

This process is then registered in the blockchain by the involved CSP, such as a SC creation transaction is added. It is defined as follows:

$$T_C = \{T_{id,C}, C, SC_{id}, \sigma_{k_{CSP}}\}$$

- $T_{id,C}$ – denotes the transaction identifier, which contains a unique fresh random and a binary identifier associated with the SC generation process.
- C – refers to the type of the transaction: it corresponds here to the creation of a smart contract.
- SC_{id} – is the smart contract identifier.
- $\sigma_{k_{CSP}}$ – is the signature of all the above fields, by the SGX enclave of the storing server, approving the SC as well as the associated ACLs.

Note that the transaction identifier id_T is then shared with the server $\mathcal{S}$ and the corresponding user DO. As presented in section 4.2, the DO may also create a `vote-doc` contract that defines the voting protocol associated with the corresponding data file and the process of voting and the choice of voters [4] (Remark 1). The `vote-doc` smart contract is mainly required to validate both the `download` and `transfer` actions, detailed in subsection 5.2.2.

Remark 1. ***Blockchain-based reputation systems—*** *Reputation systems are important for distributed applications in which users have to be made accountable for their actions, namely e-commerce websites.*

[4] For ease of presentation, we will not detail the process of the voting process. Note that, in our proof of concept, we implemented a threshold-based voting protocols, while precisely identifying the voting nodes

The Blockchain technology is suitable for supporting reputation systems thanks to its intrinsic properties, namely integrity and availability of registered information (Li et al., 2018). Thus, Prov-Trust may employ a reputation system for validators, i.e., voters, where raters are BC clients. That is, each validator's rate depends on the set of performed honest rating and detected malicious actions.
Prov-Trust considers the behavior of all participating raters for the reliability of the data. If a validator has high reputation value, it will be frequently selected by owners, users and service providers. For this purpose, we set t as the threshold value of reputation, such that if the validator's reputation is greater than t, so it is considered as honest. The reputation value assigned to each validator is multi-dimensional, which is based on clients and providers' positive and negative rates as well as on rates given by authorized auditors (c.f., Table 1). As such, two weight values α *and* β *are defined to calculate the resulting rate for each validator as* $R_V = \alpha R_{aud} + \beta R_{int}$*, where* R_{aud} *is the score computed based on auditors rates and* R_{int} *is the score computed based on users and servers rates. The* R_{int} *is based on the positive activity,*

Table 1: Validator's Score Table

Type	Activity	Score
positive	continuous selection	1.0
	recommended selection	0.75
	favorite	0.25
negative	blacklist	−1.0
	not recommended	−0.5

denoted by pos *and negative activity, denoted by* neg*, computed as:*

$$\begin{cases} \texttt{pos} = \sum_{i=1}^{n_{DO}} \sum_{j=1}^{n_{CSP}} \sum_{k=1}^{n_p} \texttt{pos}_{ijk} \\ \texttt{neg} = \sum_{i=1}^{n_{DO}} \sum_{j=1}^{n_{CSP}} \sum_{k=1}^{n_n} \texttt{neg}_{ijk} \\ R_{int} = \frac{1}{2}[\frac{\texttt{pos}}{n_{DO}+n_{CSP}}] + \frac{1}{2}[\frac{\texttt{neg}}{n_{DO}+n_{CSP}}] \end{cases}$$

where: n_{DO} *is the number of participating owners,* n_{CSP} *is the number of participating servers,* n_p *is the number of positive activities,* n_n *is the number of negative activities*

5.2.2 The Evt Process

The Evt process defines the chronological steps for (i) endorsing transactions associated with the retrieval and transfer of data contents, and (ii) `prov-doc` logs associated with data files' access and versioning. Note that these logs are not anchored as transactions, however, they are pinned as SGX-based provenance data and associated to the corresponding SC.

The `vote-doc` SC is performed, once a `download` or `transfer` query is initiated. Note that both queries refer to the transfer of data contents to another entity, i.e., `download` corresponds to the transfer for data users, `transfer` is related to data processors' queries. The transfer query launches the execution of the `vote-doc` smart contract, w.r.t. the requesting CSP and a transaction is added by the CSP to the BC defined as:

$$T_T : (T_{id,T}; T; s_{id}, id_{CSP_B}; V_R; \sigma_V, \mathsf{enc}(M_{CSP_A}); \sigma_{k_{CSP_A}})$$

- $T_{id,T}$ – denotes the transaction identifier, which contains a unique fresh random and a binary identifier associated with the transfer process.
- T – refers to the type of the transaction, i.e., transfer query.
- (s_{id}, id_{CSP_B}) – s_{id} is the transfer session identifier with the corresponding processing entity, where id_{CSP_B} denotes its associated identifier.
- V_R – represents the result of the voting process.
- σ_V – is the signature of the voting process's result concatenated with the session identifier s_{id}.
- $\mathsf{enc}(M_{CSP_A})$ – represents the encryption of a message M_{CSP_A}, based on an ABE mechanism, i.e., enc_{ABE}, where M_{CSP_A} contains two different messages as $M_{CSP_A} = T_{id,C} || T_{id,T/D}$, where $T_{id,C}$ is the creation transaction identifier and $T_{id,T/D}$ is the previous transaction identifier added by the SGX enclave of the processing server, w.r.t. the process on the DO's outsourced or collected data. Notice that $T_{id,C}$ enables PA to check granted privileges w.r.t. outsourced data contents in CSP servers.
- $\sigma_{k_{CSP_A}}$ – is the signature of all the above fields, by the SGX enclave of the storing server.

Remark 2. ***Multi-level encryption of data usage transaction's history—*** *Note that CSPs may rely on a multi-level ABE scheme (Kaaniche and Laurent, 2017),(Kaaniche et al., 2018), to encrypt their messages* M_{CSP_A} *and* M_{CSP_B} *respectively. That is, depending on their associated credentials (i.e; certified attributes), authorized authorities can access to a sub-sets of the encrypted message* M_{CSP_i} *defined as* $M_{CSP_i} = T_{id,C} || T_{id,T/D}$*. For instance, the authorized provenance auditor PA may be allowed to only check the consistency of the actual transaction.*

Similarly, the `download` query is anchored in a transaction such as

$$T_D : (T_{id,D}; D; s_{id}, id_{DU}; V_R, \sigma_V, \mathsf{enc}(M_{CSP}); \sigma_{k_{CSP}})$$

- $T_{id,D}$ – denotes the transaction identifier, which contains a unique fresh random and a binary identifier associated with the download process.

- D – refers to the type of the transaction, i.e., download query.
- (s_{id}, id_{DU}) – s_{id} is the download session identifier with the corresponding user, where id_{DU} denotes its DU identity. Note that DU identity can be replaced by authorized users group identifier.
- V_R – represents the result of the voting process.
- σ_V – is the signature of the voting process's result concatenated with the session identifier s_{id}.
- $\mathsf{enc}(M_{CSP})$ – represents the encryption of M_{CSP}, i.e., containing two messages such as $M_{CSP} = T_{id,C}||T_{id,T/D}$, where $T_{id,C}$ is the creation transaction identifier and $T_{id,T/D}$ is the previous transaction identifier added by the SGX enclave of the processing server, w.r.t. the process on the DO's outsourced or collected data.
- $\sigma_{k_{CSP}}$ – is the signature of all the above fields, by the SGX enclave of the storing server.

Remark 3. ***Update of granted privileges —*** *When a data owner DO wants to change granted privileges to a given CSP, a consent negotiation should be conducted and a new transaction is added to the blockchain, such that* $\{T_{id,U}, U, su_{id}, \mathsf{enc}_{ABE}(ACL_{up}||T_{id,C}), \sigma_{k_{CSP}}\}$, *such that* $T_{id,U}$ *is the transaction update identifier,* su_{id} *is the session identifier,* $\mathsf{enc}_{ABE}(ACL_{up}||T_{id,C})$ *is the encryption of updated access privileges associated with the creation transaction identifier and* $\sigma_{k_{CSP}}$ *is the signature of all the previous fields, by the SGX enclave of the storing server.*

5.3 Auditing

Prov-Trust supports three levels of auditing capabilities, referred to as, public auditing, data owner auditing, data provenance auditing.

First, the public auditing may be run by end-users, while investigating the consistency between different BC transactions. The main public process relies on verifying enclaves' signatures across endorsed transactions. Note that relying on their granted privileges, certain users may access related data, for instance, by deciphering the first level of ABE scheme, associated with the previous process transaction.

Second, the data owner auditing consists of verifying the storage and processing of his own data contents, outsourced to different CSPs. Note that DOs are not able neither to link other DOs' transactions, nor identify them.

Finally, the data provenance auditing permits provenance auditors to audit service providers, i.e., verifying the fulfilment of ACLs' requirements, the compliance with legislation, the consideration of clients' given consents, $\cdots$. For this purpose, the authorized PA identify the creation transactions, as well as associated processing and granted privileges update transactions, and compare the consistency of information with SC history logs and provenance database. On the other side, when a data owner requests a private audit for a misuse of his personal data, he shares the creation identifiers of the suspected smart contracts with the PA. As such, the auditing authority is able to lead an investigation, i.e., interpret the content of blockchain transactions associated by the provenance database and to detect non-compliant activities, while crawling blockchain transactions corresponding to the provided smart contracts' identifiers, We note that private auditing has to be paid by the audited service provider, if non-compliant activities are reported.

6 Security and Privacy Discussion

This section first introduces the considered adversaries. Then, it discusses the security and privacy guarantees, as detailed in subsection 2.

6.1 Threat Model

The Prov-Trust system considers two types of attackers: an external adversary and an internal adversary, defined as follows:

- **The external adversary** is a user who is not authorized to the document in the system, but is able to modify the provenance data of the outsourced document, i.e., including SC events and history logs. The external adversary does not have neither access to the location of the storing server nor the deciphering key. The attacker can only know the document identifier and exploit it to launch an attack against the blockchain based data provenance system. We assume that the adversary can not corrupt the information transmitted between SGX enclave and users.
- **The internal adversary** is an entity who is granted access to the document by the owner in the Prov-Trust system. This adversary is able to makes changes over the document and therefore log these operations on the blockchain as provenance data. An internal attacker is unable to grant access to another user over this document which is assumed not his own document. However, this attacker can try to log corrupted provenance data in the blockchain.

We assume that the cloud provider is not fully trusted and all the outsourced data files are encrypted. Once the data provenance service is enabled, the user will be able to trace the data and the provenance auditor is allowed to access all the provenance trails.

6.2 Security and Privacy Analysis

This section discusses the security properties, with respect to the defined threat model. Indeed, the privacy preservation (subsection 6.2.1), confidentiality (subsection 6.2.2), auditability (subsection 6.2.3) and availability (subsection 6.2.4) are analysed while considering different adversaries.

6.2.1 Privacy Preservation

As detailed in subsection 2.2, the privacy property is beyond the confidentiality guarantee and mainly considers the unlinkability feature and is ensured in the Prov-Trust approach thanks to the following features:

- signed transactions with several SGX-enclaves – linking smart contracts in between as issued by the same owner is not possible, as smart contracts' creation transaction are signed by the SGX-enclaves of the processing servers. As such, a search over the blockchain ledger for the same signing enclaves (assumed to match the same owner) is inconclusive, as the processing server is storing data contents of different owners and the same data content can be stored by several servers (for replication purposes).
- one smart contract per a data file – each smart contract is specific to a data file and has its own identifier, a randomly generated value. Thus, it is even not possible to link two smart contracts provided with two different IDs to the same owner.

6.2.2 Confidentiality

Prov-Trust is resistant against data secrecy attacks against data provenance trials, as discussed here-after:

- only freshly and randomly generated identifiers and enciphered information are published in the BC infrastructure – the DO is in charge of specifying the ACLs w.r.t. the data file and its related provenance trials, while associating a random identifier. The data file identifier is published only once, in the creation transaction of the `prov-doc` smart contract. For other processing transaction, the file identifier is enciphered using a -multi-level- ABE scheme (Belguith et al., 2018a; Belguith et al., 2018c; Belguith et al., 2018b), and is only accessible by a group of authorized entities.
- fine-grained encrypted access to data – data contents and provenance trials are protected thanks to the usage of both ACLs and a -multi-level- ABE scheme (Kaaniche and Laurent, 2017). Recall that, on one side, ACLs refer to any access control mechanism selected by the DO, to be enforced on his data. ACLs on data contents may be encrypted -up-to the DO-, however, they have to be accessible to the CSP, the PA as well as voters, if a voting smart contract was created. ACLs on provenance data ensure the access to involved parties, with respective privileges. On the other side, multi-level ABE applied on processing transactions ensure fine-grained access to data provenance trails by authorized PAs, and enhance unlinkability of processing actions on DO's data by external adversaries.

As the public ledger only registers random values or encrypted information, no significant information can be learnt from the blockchain.

6.2.3 Auditability

The proposed approach ensures the auditability requirement as follows:

- signed transactions – the Prov-Trust solution relies on signed transactions, by the SGX-enclaves of the storing server. Thus, both smart contract creation and data processing transactions must be signed by the involved entities, i.e, SGX enclaves and validators respectively. Signed transactions ensure that each activity has been efficiently performed by the holder of the used private key, which is certified by the device's manufacturer. As such, the resistance of the chosen signature scheme against forgery attacks has a direct impact on the fulfillment of the auditability requirement (Intel, 2016), (Laurent et al., 2018).
- approval of smart contracts creation – the service provider is requested to approve each smart contract creation by the data owner DO, by using the secret key associated with the SGX enclave of the involved storing server. More precisely, the signature $\sigma_{k_{CSP}}$ associated with the creation of a particular smart contract, and to prove its authenticity and its legitimacy.
- tamper-proof architecture – as emphasized above, all blockchain-specific operations, such as transaction anchoring activities, are considered as secure and non-corruptible, thus ensuring non-tamper proofs of data processing and managing

events (Laurent et al., 2018).

- transparent usage – the proposed approach is based on a consortium blockchain infrastructure, that permits public access (i.e; read privilege) the contract and its associated transactions, to anyone. Thus, it provides a transparent view over how data are collected and accessed.

6.2.4 Availability

The Prov-Trust solution ensures availability assurance and liveness guarantees of data usage, as relying on a highly decentralized infrastructure. We also point out the similarity between the well known double spending problem in bitcoin architectures (Karame et al., 2015) and the attack aiming at preventing a valid transaction to be registered in the blockchain. Indeed, both assume that an adversary has control over more than a half of the blockchain nodes, the achievement of which is assumed difficult (Karame et al., 2015).

7 Proof of Concept

In order to evaluate the performances and the deployment of the proposed Prov-Trust solution, we first created a BC-testing environment for our provenance architecture and started by implementing the vote protocol along with the Access Control functionality. For our implementation, we had to choose one of two blockchain options: Ethereum or the Hyperledger. We have chosen the second as it enables a general description of transactions while providing users with tools to develop their own personalized blockchains adapted to the needs of their businesses. Hyperledger [5] is an open source collaborative project hosted by The Linux Foundation. It is not a blockchain technology on itself but it is more a strategy combining multiple blockchain platforms to develop enterprise-oriented solutions. We have chosen Hyperledger Composer to be deployed the Trust-Prov underlying technology as it is more adapted to create the access control functions and to evaluate the solution smoothly. Hyperledger Composer supports the existing Hyperledger Fabric blockchain infrastructure and runtime, which supports pluggable blockchain consensus protocols to ensure that transactions are validated according to policy by the designated business network participants. Hyperledger is built with *javascript* language and supports modern tools including node.js, npm, and CLI.

[5] https://www.hyperledger.org

In the following, we introduce the proof-of-concept implemented by developing the vote protocol based on the Hyperledger Composer framework while defining an access control list that is checked by the voters to grant access to the different users.

To implement the `vote-doc` smart contract, we start by defining the access rights for different types of entities where we differentiate between two types: data users and data owners, also called clients. Clients creates the data file and upload it on the cloud storage service while users access, read or modify it. Both are identified with two different identifiers: *clientId* for data owners and *userId* for data users. An example of the defined access control list is shown in Listing 1.

Listing 1: An example of the Access Control List

```
rule ParticipantSystem {
  description: "Grant any participants the right to see
  every resource"
  participant: "org.vote.Utilisateur"
  operation: READ
  resource: "**"
  action: ALLOW
}
rule ContributorHasRightToRead {
  description: "If a user contributes to a document, he has
  access to
  the document"
  participant(user): "org.vote.Utilisateur"
  operation: READ, UPDATE
  resource(doc): "org.vote.Document"
  condition: (doc in user.documentsModifiables)
  action: ALLOW
}
rule OwnerHasRightToReadAndUpdate {
  description: " The client has always the right to see and
  update to
  access its own document "
  participant(client): "org.vote.Client"
  operation: READ, UPDATE
  resource(doc): "org.vote.Document"
  condition:(client.getIdentifier()==doc.owner.
  getIdentifier())
  action: ALLOW
 }
rule UserHasRightToCreateABlock {
  description: "Users can right in the blockchain"
  participant: "org.vote.Utilisateur"
  operation: CREATE
  resource:"org.vote.modifierDocument"
  action: ALLOW
}
```

We define *assets* by the documents created by a user, where each document is identified by its unique identifier *documentId*. We also define two *events* related to the type of actions performed on the data file. The first event is related to the modification of the document while the second is related to the update of granted privileges, i.e., addition of a user to the list of participants authorized to access the data content. we also define two *transactions* related to either the modification of the assets, i.e., the document or the modification of the list of participants. Each transaction is followed by the related event. In other words, one transaction is related to the change in the access rights of users by adding new users. The second transaction initiates the vote protocol and implement the modification (i.e., download) of the data file based on

the vote result as shown in Figure2. If the vote is successful the event is published in the blockchain (c.f., Listing 3)

Listing 2: The event publication

```
 let documentModificationEvent = getFactory().newEvent(NS,
 'documentModification');
  documentModificationEvent.resultat=false;
  documentModificationEvent.protocoleVote=maj.protocoleVote;

  if(maj.protocoleVote=="MAJORITE"){
    if(nbOui>nbNon){
      documentModificationEvent.resultat=true;
  }
  }
  if(maj.protocoleVote=="UNANIMITE"){
    if(nbNon==0){
      documentModificationEvent.resultat=true;
  }
}
```

Listing 3: The vote protocol implementation

```
  if(documentModificationEvent.resultat==true){
  await documentRegistry.update(document);
  }
  documentModificationEvent.documentId=document.
  getIdentifier();
  documentModificationEvent.utilisateurId=contributeurId;
  documentModificationEvent.chaineRemplacee=chaineRemplacee;
  documentModificationEvent.chaineInseree=maj.chaineInseree;

  emit(documentModificationEvent);
}
```

The solution can be tested using the graphic interface provided by Composer Playground. It is necessary to load the different files such as permission, query and transaction files. For more information, the implementation files are accessible in Github[6].

8 Conclusion

Prov-Trust is a distributed data provenance architecture designed to ensure a secure, tamper-proof and privacy preserving provenance data in cloud computing systems. It is relying on a novel use of Intel SGX trusted hardware to collect provenance data while storing them in Blockchain using transactions and smart contracts. Prov-Trust provides a secure and authorized auditing function to enable an auditor or the data owner to verify the operations performed over the data stored in cloud storage services. A security and privacy analysis has been detailed to prove the fulfillment of the defined requirements. We have also presented a proof of concept of the proposed system by implementing the vote system.

As future work, we aim to finish the implementation of Prov-Trust and test the performance of the data provenance generation within the Intel SGX besides adding more sophisticated encryption and privacy preserving schemes while storing and auditing the data stored in blockchain.

[6]https://github.com/nathanael-denis/blockchain-vote/blob/master/blockchain-based-voting-system/lib/logic.js

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant ACI-1547467. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Asghar, M. R., Ion, M., Russello, G., and Crispo, B. (2012). Securing data provenance in the cloud. In *Open problems in network security*, pages 145–160. Springer.

Belguith, S., Kaaniche, N., and Hammoudeh, M. Analysis of attribute-based cryptographic techniques and their application to protect cloud services. *Transactions on Emerging Telecommunications Technologies*, page e3667.

Belguith, S., Kaaniche, N., Laurent, M., Jemai, A., and Attia, R. (2018a). Phoabe: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot. *Computer Networks*, 133:141–156.

Belguith, S., Kaaniche, N., Mohamed, M., and Russello, G. (2018b). C-absc: cooperative attribute based signcryption scheme for internet of things applications. In *2018 IEEE International Conference on Services Computing (SCC)*, pages 245–248. IEEE.

Belguith, S., Kaaniche, N., Russello, G., et al. (2018c). Lightweight attribute-based encryption supporting access policy update for cloud assisted iot. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications-Volume 1: SECRYPT*, pages 135–146. SciTePress.

Böck, B., Huemer, D., and Tjoa, A. M. (2010). Towards more trustable log files for digital forensics by means of "trusted computing". In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 1020–1027. IEEE.

Crosby, M., Pattanayak, P., Verma, S., and Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2:6–10.

Cui, S., Belguith, S., Zhang, M., Asghar, M. R., and Russello, G. (2018). Preserving access pattern privacy in sgx-assisted encrypted search. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE.

Gehani, A. (2016). https://github.com/ashish-gehani/spade.

Gehani, A. and Tariq, D. (2012). c. In *Proceedings of the 13th International Middleware Conference*, Middleware '12, pages 101–120, New York, NY, USA. Springer-Verlag New York, Inc.

Grawrock, D. (2009). *Dynamics of a Trusted Platform: A building block approach*. Intel Press.

Hasan, R., Sion, R., and Winslett, M. (2009). Preventing history forgery with secure provenance. *ACM Transactions on Storage (TOS)*, 5(4):1–43.

Intel (2016). Introduction to Intel SGX sealing. https://software.intel.com/en-us/blogs/2016/05/04/introduction-to-intel-sgx-sealing. Last accessed: September 10, 2018.

Kaaniche, N., Belguith, S., and Russello, G. (2018). Emalab: Efficient multi authorisation level attribute based access control. In *International Conference on Network and System Security*, pages 187–201. Springer.

Kaaniche, N. and Laurent, M. (2017). Attribute based encryption for multi-level access control policies. In *14th International Conference on Security and Cryptography (SECRYPT)*.

Kaaniche, N. and Laurent, M. (2018). Bdua: Blockchain-based data usage auditing. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 630–637. IEEE.

Kaaniche, N., Laurent, M., and Levallois-Barth, C. (2020). Id-based user-centric data usage auditing scheme for distributed environments. *Frontiers in Blockchain*, 3:17.

Karame, G., Androulaki, E., Roeschlin, M., Gervais, A., and Capkun, S. (2015). Misbehavior in bitcoin: A study of double-spending and accountability. *ACM Transactions on Information and System Security (TISSEC)*, 18(1):2.

Ko, R. K. and Will, M. A. (2014). Progger: An efficient, tamper-evident kernel-space logger for cloud data provenance tracking. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 881–889. IEEE.

Laurent, M., Kaaniche, N., Le, C., and Vander Plaetse, M. (2018). A blockchain-based access control scheme. In *15th International Conference on Security and Cryptography (SECRYPT)*, pages 168–176.

Li, H., Gai, K., Fang, Z., Zhu, L., Xu, L., and Jiang, P. (2019). Blockchain-enabled data provenance in cloud datacenter reengineering. In *Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pages 47–55.

Li, Y., Marier-Bienvenue, T., Perron-Brault, A., Wang, X., and Paré, G. (2018). Blockchain technology in business organizations: A scoping review. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., and Njilla, L. (2017). Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 468–477. IEEE Press.

Main, T. (2007). Part 2 tpm structures. *Specification version*, 1.

McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., and Savagaonkar, U. R. (2013). Innovative instructions and software model for isolated execution. *Hasp@ isca*, 10(1).

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., et al. (2011). The open provenance model core specification (v1. 1). *Future generation computer systems*, 27(6):743–756.

Moreau, L. et al. (2010). The foundations for provenance on the web. *Foundations and Trends® in Web Science*, 2(2–3):99–241.

Muniswamy-Reddy, K.-K., Barillari, J., Braun, U., Holland, D. A., Maclean, D., Seltzer, M. I., and Holland, S. D. (2008). Layering in provenance-aware storage systems.

Muniswamy-Reddy, K.-K., Holland, D. A., Braun, U., and Seltzer, M. I. (2006). Provenance-aware storage systems. In *Usenix annual technical conference, general track*, pages 43–56.

Pohly, D. J., McLaughlin, S., McDaniel, P., and Butler, K. (2012). Hi-fi: collecting high-fidelity whole-system provenance. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 259–268.

Ramachandran, A. and Kantarcioglu, M. (2018). Smart-provenance: A distributed, blockchain based data-provenance system. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 35–42. ACM.

Shetty, S., Red, V., Kamhoua, C., Kwiat, K., and Njilla, L. (2017). Data provenance assurance in the cloud using blockchain. In *Disruptive Technologies in Sensors and Sensor Systems*, volume 10206, page 102060I. International Society for Optics and Photonics.

Suen, C. H., Ko, R. K., Tan, Y. S., Jagadpramana, P., and Lee, B. S. (2013). S2logger: End-to-end data tracking mechanism for cloud data provenance. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 594–602. IEEE.

Swan, M. (2015). *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc.

Taha, M. M. B., Chaisiri, S., and Ko, R. K. (2015). Trusted tamper-evident data provenance. In *2015 IEEE Trustcom/bigdatase/ispa*, volume 1, pages 646–653. IEEE.

Vukolic, M. (2017). Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, BCC '17, pages 3–7, New York, NY, USA. ACM.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151.

Zafar, F., Khan, A., Suhail, S., Ahmed, I., Hameed, K., Khan, H. M., Jabeen, F., and Anjum, A. (2017). Trustworthy data: A survey, taxonomy and future trends of secure provenance schemes. *Journal of Network and Computer Applications*, 94:50–68.