Enforcing Private Data Usage Control with Blockchain and Attested Off-chain Contract Execution

Yang Xiao[†], Ning Zhang[*], Wenjing Lou[†], Y. Thomas Hou[†]
[*]Washington University in St. Louis, MO
[†]Virginia Polytechnic Institute and State University, VA

***Abstract*—The abundance of rich varieties of data is enabling many transformative applications of big data analytics that have profound societal impacts. However, there are also increasing concerns regarding the improper use of individual users' private data. Many argue that the technology that customizes our experience in the cyber domain is threatening the fundamental civil right to privacy.**

In this paper, we propose PrivacyGuard, a system that leverages smart contract in blockchain and trusted execution environment to enable individuals' control over other parties' access and use of their private data. In our design, smart contracts are used to specify data usage policy (i.e. who can use what data under which conditions along with how the data can be used), while the distributed ledger is used to keep an irreversible and non-repudiable record of data usage. To address the contract execution efficiency problem, as well as to prevent exposing user data on the publicly viewable blockchain, we construct a novel off-chain contract execution engine which realizes trustworthy contract execution off-chain in an trusted execution environment (TEE). By running the contract program inside a hardware-assisted TEE, the proposed off-chain trustworthy contract execution improves system efficiency significantly, as its correctness does not rely on distributed consensus which essentially requires the contract program be executed on all miner nodes. In order to leverage TEE in off-chain execution, PrivacyGuard has to several technical challenges such as synchronous function completion and scalability mitigation in blockchain platform. We build and deploy a prototype of PrivacyGuard using Ethereum and Intel SGX, and our experiments demonstrate the feasibility to support data-intensive applications using data from a large number of users.

I. Introduction

The recent emergence of big data analytics and artificial intelligence has made life-impacting changes in many sectors of society. For example, deep-learning trained physical assistant has demonstrate higher accuracy in pneumonia diagnosis than expert radiologists working alone [1]. One of the fundamental enabling components for the recent advancements in artificial intelligence is the abundance of data. However, as more and more information on individuals is collected, shared, and analyzed, there is an increasing concern on the privacy implication. In the 2018 Facebook-Cambridge Analytica incident, an API, originally designed to allow a third party app to access the personality profile of participating users, was misused by Cambridge Analytica to collect information on 50 million of Facebook profiles without the consent of the users [2]. These illicit harvested private data are later used to create personalized psychology profiles for political purposes [2]. With increasing exposure on the privacy risks of big data, many now consider the involuntary collection of personal information a step backward in fundamental civil right of privacy, or even in humanity [3], [4]. Yet, driven by economic force, the collection and analysis of the personal data continue to grow at an amazing pace.

Individuals share personal information with people or organizations within a particular community for specific purposes; this is often referred to as the context of privacy [5]. For example, individuals may share their medical status with healthcare professionals, product preferences with retailers, and real-time whereabouts with their loved ones. When information shared within one context is exposed in another unintended one, people may feel a sense of privacy violation [6]. The purposes and values of those contexts are also undermined. The *contextual nature of privacy* implies that privacy protection techniques need to address at least two aspects: 1) what kind of information can be exposed to whom, under what conditions; and 2) what is the "intended purpose" or "expected use" of this information. Much research has been done to address the first privacy aspect, focusing on *data access control* [7], [8], [9], [10] and *data anonymization* [11], [12], [13], [14]. Only recently, there have been a few works that attempted to address the second aspect of privacy from the architecture perspective [15], [16], [17], [18], [19], [20]. In fact, many people used to believe that the prevention of this kind of "second-hand" data (mis)use can only be enforced by legal methods [21]. With the current practice, once an authorized user gains access to the data, there is little control over how this user would use the data. Whether or not he/she would use the data for purposes not consented by the user, or pass the data to another party (i.e., data monetization) is entirely up to this new "data owner", and is no longer enforceable by the original data owner.

In this paper, we propose PrivacyGuard to address this very challenging privacy problem—how to empower individual users with full control of the use of his/her personal data. The data owner should not only be able to control who can have what access to his/her data, but also be ensured that the data are used only for the intended purposes. To realize the envisioned functionality of PrivacyGuard, there are three key requirements. First, users should be able to define their own data access policy in terms of to whom they will share the data, at what price, and for what purpose. Second, data usage should be recorded in a platform that offers non-repudiation [22]. Third, actual use of user data should have a verifiable proof that it is compliant to the policy defined by the user.

Blockchain, the technology behind *Bitcoin* [23] and *Ethereum* [24], has emerged as a popular technology in recent years. To provide these security services, smart contract on blockchain platform is adapted in PrivacyGuard. The distributed ledger enabled by distributed consensus in blockchain has the unique time-stamped property that is non-repudiable once recorded. *Smart contract* is a program that runs on the blockchain and has its correct execution enforced by the consensus protocol. PrivacyGuard leverages the distributed ledger to provide transparent and tamper-proof recording of data usage. Furthermore, smart contract is adopted in PrivacyGuard to enable individual users to control their data access and use, by establishing a data usage contract between the users and potential data consumers.

While smart contract and blockchain appear to be the perfect solution logically, there are fundamental limitations if we directly apply those technologies to implement the proposed framework. First of all, data used by smart contracts have to be on the blockchain and recorded as a transaction. Yet, Bitcoin network produces one block with size of 1 MB every ten minutes [23]. Furthermore, the transaction capacity is not the only forbidden factor. Smart contracts are small programs that have to be executed by all participants in the network, which raises serious efficiency concern. Platforms such as Ethereum [24] often require payment in the form of gas for executing a smart contract that is metered based on the complexity of the invoked function. This implies that executing a large, complex data analytics program as a smart contract will not only be a waste of energy for the network but also economically infeasible. Last but not least, data used by smart contracts are available to every participant on blockchain by design. If user's private data are used directly in smart contract, existing secure computation techniques to preserve confidentiality and utility of data, such as functional encryption [25], can be prohibitively expensive for the network.

To tackle data and computation scalability problems, we build on the intuition that blockchain is a powerful primitive to enable users to transact based on mutually agreed contracts, but not a platform for heavy computation on large volumne of data. PrivacyGuard splits the private data use enforcement problem into two domains: the control plane and the data plane. Individuals publish the availability as well as the use policy of their private data as smart contracts on Blockchain, which is part of the control plane in the PrivacyGuard system. The policy can include authorized users, intended computation, triggering condition, as well the price, etc. Data consumers interact with the smart contract to obtain authorization to use the data. However, the actual data of the users are never exposed on the blockchain. Instead, they are stored in the Cloud in the encrypted form. The computation on those private user data as well as the provision of secrets are accomplished off-chain in the data plane using a trusted execution environment [26], [27] on the Cloud.

When a data contract is split into control and computation, where the computation actually takes place off-chain, there are several challenges that need to be addressed. First, the correctness of the contract execution can no longer be guaranteed by the consensus of the network. To this end, we propose ”local consensus” using remote attestation among contracting parties. Second, the execution of contract is no longer atomic when the computation slice is executed off-chain. We design a multi-step commitment protocol to ensure that result release and data transaction remain an atomic operation, where if the computation results were tampered with, the data transaction would abort gracefully. Lastly, using a secure container, private data are protected inside the trusted execution environment and secrets are only provisioned when approved according to the contract binding.

Finally, contract execution is not the only limiting factor. Even when the decentralized blockchain platform is used in the control plane, the current transaction rate remains too low to accommodate studies that involve data from millions of users. To that end, we propose *data broker*, to which individual data owners can delegate the control of their data. Individual data owners' trust on the data broker can be bootstrapped by remote attestation on the data broker system and examination of broker smart contract. To reduce the transaction volume on the main chain, data broker records individual data uses in off-chain storage backed by recorded hash value in the main chain. Second-level chain or payment channel can then be used to distribute the reward back to individual users. The delegation system offers the trade-off between decentralization and performance without degrading trust, as we move towards highly scalable blockchain solutions.

We implemented a prototype of PrivacyGuard using Intel SGX as the trusted execution environment and Ethereum as the smart contract platform. We chose these two technologies for implementation due to their wide adoption. Our design generally applies to other types of trusted execution environments and blockchain platforms. We show that it is possible to enable large complex data computations automatically with the security protection as specified by the smart contract. Furthermore, we also analyze the security of the system and share our empirical measurement of the platform.

We make the following contributions in this paper:

- We propose PrivacyGuard, a framework for trustworthy private data computation and data usage tracking that aims to address one of the most pressing problems in big data analytics—private data usage control. PrivacyGuard builds on latest advancements in trustworthy computing and decentralized platforms. PrivacyGuard is an open platform that allows any user to contribute his/her data into the ecosystem, and allows the user to specify the context under which his/her data can be used, i.e. what the data can be used for, when it can be used, where it can be used, and by whom it can be used.

- We propose a novel construction of off-chain contract execution engine to support the vision of PrivacyGuard. The proposed off-chain trustworthy contract execution is the key to improve the execution efficiency of smart contract technology, enabling trustworthy execution of more complex contract program without solely relying on costly network consensus. Private data computation in PrivacyGuard is split into the control plane and data plane. The control plane on blockchain provides high-level data exchange negotiation and control, while computationally expensive tasks are off-loaded to trusted environments for confidential data processing in the data plane. Remote attestation is used to create local consensus among the contract parties.
- We implemented a prototype of PrivacyGuard using Intel SGX and Ethereum smart contract. Our evaluation of the system shows that PrivacyGuard is capable of processing large volumes of data without incurring significant overhead.

II. Background

A. Blockchain and Smart Contract

Blockchain is a recently emerged technology used in popular cryptocurrencies such as Bitcoin [23] and Ethereum [24]. It enables a wide range of distributed applications as a powerful primitive. With a blockchain in place, applications that could previously run only through a trusted intermediary can now operate in a fully decentralized fashion without the need for a central authority, and achieve the same functionality with comparable realibiity. When the majority of the nodes in the network follow the protocol honestly, the shared blockchain becomes an authenticated and timestamped record of the network's activities. The conceptual idea of programmable electronic "smart contracts" dates back nearly twenty years [28]. When implemented in the blockchain platform, smart contracts are account-holding objects that can receive transfers, make decisions, store data or even interact with other contracts. Due to its ability to enable various forms of business logic, smart contract has become a newly-emerged killer application of blockchain technology. Ethereum is among the firsts to offer a Turing-complete decentralized smart contract platform. The blockchain and the smart contract platform however have several drawbacks in transaction capacity [29], computation cost [30], [31], as well as privacy of user and data [32], [31].

B. Trusted Execution Environment

Creating vulnerability-free software has long been considered a very challenging problem [33]. To tackle this problem, researchers in the architecture community in both academia [34] and industry [27], [26] have shifted a new paradigm of limiting the trusted computing base (TCB) to only the hardware. The Intel SGX is a newly developed instruction set extension to provide secure execution [26]. Applications are executed in secure containers, called *enclaves*. The hardware guarantees the integrity and confidentiality of the protected application, even if the privileged software is compromised. This protection is accomplished by integrating multiple technologies into processor, such as *software attestation* and *memory encryption*. Programs executed within the enclave are executed as user space programs. Memory of the enclave is encrypted automatically by the hardware, while I/O functionalities are supported by the potentially untrusted host operating system. The Intel SGX has recently been adapted as a powerful primitive to build secure systems [35], [36], [37], [30].

III. PrivacyGuard Overview: A Framework Enabling User Control on Private Data

The abundance of data has been the catalyst for recent transformative changes to society with the application of artificial intelligence and machine learning. However, at the same time, there is an increasing concern on the use/misuse of the data. PrivacyGuard aims to provide a platform to enable end-to-end control of data for individual data owners, including all subsequent use of data after collection.

A. System Goal

1) Confidentiality Protection on User Data: When data is generated under the current paradigm, it is stored in the vendor's cloud storage. Access control on user data generally relies on the vendor system. This lack of confidentiality guarantee often discourages individual data sharing among individual users. User-controlled, rather than service provider-controlled, encryption/decryption is fundamental for broad participation in data contribution.

2) User-controlled Fine-grained Verifiable Data Access and Usage Recording: Under the current paradigm, once the data is uploaded to the vendor cloud, it belongs to the vendor under a service agreement. A user might be able to grant or deny others access to her data. There is, however, no way for the user to verify who actually accessed her private data, not to mention the purpose of the access. The lack of transparency and verifiability on data access often discourages users from sharing data [38]. A public service that keeps track of user data usage and makes it auditable by data owners is therefore essential to not only protecting user privacy but also promoting data sharing in the community.

3) Enforceable Legal Binding on User Data Usage: Service-level agreements and legal contracts are often the only protection a user has for how his/her data is stored, shared, and used. However, these agreements are often very difficult to understand, not to mention enforce. A security system that can capture user-defined privacy policies and then enforce the compliance of the policy during the execution of the data access is instrumental in enabling broader data sharing.

B. System Overview

Fig. 1 shows the system architecture of *PrivacyGuard*. Although we have been using the term *users* to refer to both individuals and organizations, we differentiate two roles that

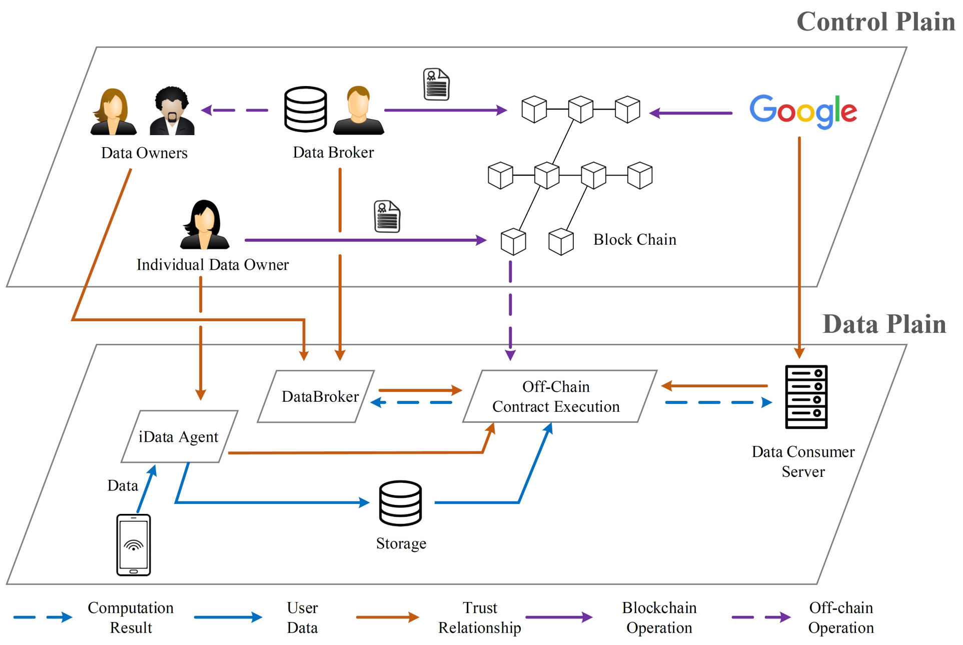


Fig. 1. System Architecture for PrivacyGuard Framework

an user in the data market can take. We refer to the individual or organization that owns the data as *data owner* (DO) and the entity that needs to access the data as *data consumer* (DC). We call the entity trusted by an individual data owner for data management the *iDataAgent*. We call the entity that acts as broker for a collection of users the *data broker* (DB). Individual data owners extend their trust to the data broker via a smart contract along with remote attestation.

Classified by the assigned responsibility, there are three main components in the PrivacyGuard framework: 1) data market, 2) iDataAgent and data broker, 3) off-chain contract execution.

1) Data Market: Essential to PrivacyGuard, data market is the subsystem that will support the supply, demand and exchange of data on top of blockchain. For data access control, a data owner can encode the terms and conditions regarding the access to his/her personal data in a smart contract, and publish it on a blockchain platform such as Ethereum [24]. Data usage by DC is recorded via a transaction that interacts with the data smart contract.

2) iDataAgent and Data Broker: Individual data agent (iDataAgent) is a trusted program that represents the data owner. The iDataAgent is responsible for key management for the user. It also participates in contract execution by only provisioning the data key material to attested remote entities. The iDataAgent can be the user's smart home hub or a trusted program running in a TEE on the Cloud. Since it is often not realistic to expect individual data owner to be connected all the time, iDataAgent can also be a trusted program hosted by the Cloud platform. To address the inherent transaction bandwidth of current blockchain implementation, data broker is introduced to collectively represent a group of users.

3) Trusted Off-chain Contract Execution: This component executes data operations that were negotiated between the data owner and data consumer in smart contracts from the control plane. The off-chain contract execution should also provide the correctness guarantee as that of on-chain, which turns out to be quite challenging.

C. PrivacyGuard High-Level Workflow

In what follows we outline the workflow of PrivacyGuard. The workflow has three stages: 1) data generation, encryption, and key management, 2) data access policy generation and contract negotiation, 3) data utilization and contract execution.

1) Data Generation, Encryption, and Key Management: In this stage, user data are collected and uploaded to the cloud storage. In PrivacyGuard, user data are generated by user device and encrypted by iDataAgent, who then passes the encrypted data to the Cloud for storage. The iDataAgent program can either be hosted in a trusted Edge node or in a TEE hosted by the Cloud. Keys used for data encryption are generated by users via interface to iDataAgent, but they will be managed by iDataAgent.

2) Policy Generation and Contract Negotiation: In PrivacyGuard, individual owners can define their own access and usage policies for their private data. The policy that is encoded in the smart contract on the blockchain usually includes the essential components for privacy context. For example, a policy can contain [*data type, data range, operation, consumer, expiration, cost*], where the intended use of data of certain *type, range* is encoded as *operation*, which can be arbitrary computer programs attestable by iDataAgent. While this construction provides a mechanism for an individual to contribute data with fine-grained access and usage policy, the deployment requires a highly scalable blockchain platform that can handle millions of transaction per second. To mitigate this scalability constraint, PrivacyGuard makes use of data broker as an intermediate surrogate for groups of users.

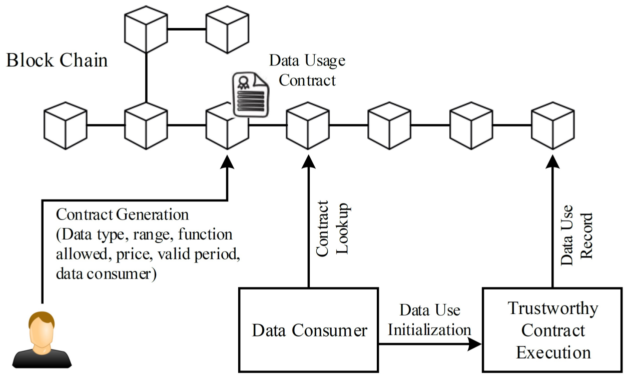
3) Data Utilization—Contract Execution: Smart contract, by design, can only embed simple logic operations (functions) that compute on data on blockchain. However, the intended use of the data often involves complex computer programs consuming a vast amount of data. To tackle this challenge, the smart contract is broken into control segments and data computation segments. The computation segments are then executed off-chain, while the control segments remain on the chain. By executing the data computation off-chain in a TEE such as Intel SGX, it is possible to not only process a large volume of off-chain data, but also handle these private user data confidentially. The correctness of the off-chain execution is achieved via local consensus by leveraging remote attestation and trusted computing, where the trust on the contract execution container is established via attestation.

Therefore, to use the data owner's private data, the data consumer will invoke the data owner's contract for permission to use and pay the deposit. If the permission is granted on the blockchain platform, both the data consumer and the data owner would perform remote attestation on a TEE that loads the program whose checksum is specified in the contract. The data owner could either be represented by the iDataAgent or the data broker. Only when the remote container is successfully attested, will keys be loaded into the container to enable decryption and utilization of private data within the TEE. When the computation finishes, the container will erase all the associated data and keying materials provisioned to the container. At the end, the results need to be released to the data consumer at the same time when cryptocurrency is transferred to the data owner. To enable this atomic operation that combines the disconnected data plane and control plane, we make use of a commitment protocol. More specifically, the data consumer will invoke the data contract with the commitment hash after obtaining the encrypted result. The data transaction will only complete when both data consumer and data owner agree. The detailed design of off-chain execution will be described in Section V.

D. Threat Model and Assumptions

The end goal of PrivacyGuard is to provide a methodology using technology rather than ambiguous legal agreements for users to understand and control how their data can be used, and to enable a market of fair and transparent information exchange. As a result, we assume that the data owners are honest and trustworthy, while all the other entities act based on self-interest and may not follow the protocol. However, we do assume the security system, i.e. the blockchain and TEE are trustworthy and are free of vulnerability. More specifically, our model for the control plane and data plane are as follows.

In the control plane, we assume the blockchain infrastructure is secure that adversaries do not control enough resources to disrupt distributed consensus. It is also assumed that smart contract implementations are free of software vulnerabilities. Data consumers might attempt to use the data without paying or leaving a record of use. They may also attempt to perform operations that are not allowed by the smart contract. We also assume that the data owner will not attempt to sell falsified data to the data consumer. There are mechanisms to build penalize dishonest data owners using smart contracts, but is out of the scope of this paper.



Block Chain
Data Usage Contract
Contract Generation (Data type, range, function allowed, price, valid period, data consumer)
Contract Lookup
Data Use Record
Data Consumer
Data Use Initialization
Trustworthy Contract Execution

Fig. 2. Data Market Workflow

In the data plane, we assume trusted execution environment is updated, particularly, Intel SGX, is secure against malicious attack from the operating system. We recognize that trusted execution environment is not always perfect [39], [40], [41], [42], [43], [44], [45], and there has been numerous work demonstrating side channel information leakage on the SGX platform alone [39], [40], [43], [44], we believe our design in this work leverages SGX as a generic trusted execution environment, and generally applies to other implementation of the execution environment. Preventing attacks against specific platform is an important but orthogonal challenge. Furthermore, our work relies on TEE to provide data confidentiality protection as well as computation integrity, it inherits certain limitations of TEE while using it as primitive. We also assume the cloud platform outside the TEE can be compromised, and collude with either the data consumer or the data owner. All the communication links are assumed to be secure. We also assume that there exists a node in the blockchain network that accepts secure connection for data broker to submit transactions. However, we do not make any other assumption on the availability of the network for the TEE.

IV. Enabling Free Market of User-defined Access to Data with Blockchain

The high level idea of data market is illustrated in Fig. 2. The intuition behind is to leverage the Blockchain platform to enable *autonomous* and *transparent* data transactions. The sales of the knowledge extracted from the data are traded autonomously via smart contracts in our system such that the data owner obtains the negotiated compensation immediately. Furthermore, the recording of the data transaction on the blockchain makes the data use transparent. More specifically, smart contract detailing how data can be used by data consumers are generated by either individual data owner or data broker. To search for data availability, data consumers such as Netflix or Google can parse the data contracts on the chain. Base on the data availability and operations permitted at the associated price, they can select the data set for the study. To start the data operation, data consumer pays a deposit to the data broker by invoking the data contract. Instead of completing the data transaction on the blockchain, the computation is off-loaded to a trusted contract execution environment (CEE) that is off-chain. At the end of data computation, CEE releases the result to the data consumer while completing the data transaction via an on-chain smart contract invocation.

A. Encoding Data Access Policy with Smart Contract

1) Basic Data Contract: While blockchain and smart contracts are both relatively mature technology, how to make use of these primitives requires careful considerations, since different design and implementation of the data policy could lead to significantly overhead costs. From the data sharing perspective, a rule in the data access policy often includes attributes such as type of the data, range or repository of the data, owner of the data DO and consumer of the data DC. For example, if a patient X with public key pair (pk_X,sk_X) has three types of medical records, radiology record M_r, blood test record M_b, and psychological mental record M_p, and he is only willing to share his radiology record and blood test with urology specialist S with public key pk_S, he can specify such an access policy P in a data contract C_{DA} that includes at least the following information: $C_{DA} = \{P = \{M_r, M_b, pk_s\}, Sig_{sk_X}(P)\}$. However, the rule above specifies only access control but not the obligations of the data consumer once access is granted. The data consumer could share the data with other parties against the original intention of the data owner. To enable fine-grained control of how user data is used, obligations such as allowable function f can be included in the policy. If patient X only wants the specialist to run anomaly detection operation Op on the data, then the new data contract will be $C_{DA} = \{P = \{M_r, M_b, Op, pk_s\}, Sig_{sk_X}(P)\}$.

2) Transparent Tracking of Data Consumption: One of the main objectives is to support trustworthy tracking of data consumption. To achieve this goal, the system needs to enforce that only authorized operations in the data usage contract can be executed, and each data transaction is recorded in a tamper-resistant storage. In PrivacyGuard, transactions in blockchain are used to facilitate the recording of data utilization. Since the ledger is publicly verifiable and unforgeable, data function invocations encoded in the transactions can provide non-repudiation on data consumption.

3) Enabling Data Market Economy—Addressing the Incentive: Data is the source of knowledge for many recent advances in data mining and artificial intelligence. The ownership of data is often considered an asset of great monetary value in modern society. It is therefore important to provide adequate incentives for both the data owners and the data consumers to participate in the platform. Through PrivacyGuard, users can reclaim the ownership of their data by enforcing who and how other entities can use the private data. This ownership requires additional infrastructure for storage and access control mechanisms, ultimately at an additional monetary cost. On the other hand, the platform also encourages more individual users to share their private data for the greater good and personal benefit without any concerns regarding privacy leakage or data misuse. Building on top of the success of cryptocurrency, data owner and data consumer cannot only transact on the use of data, the platform can also enable financial purchase for the use of the data. This can create a vibrant economic sustainable environment to support the continuous operation of the platform. To add financial value for data, we further augment our model with cryptocurrency denoted by $\$f$, and the policy can be defined as $\{M_r, M_b, Op, \$f, pk_s\}, Sig_{sk_X}(DataUse)\}$. Using the construction above, data owners can specify a price for data based on the expected utility of the data as well as its demand.

4) Encoding the Data Contract: The data contract captures the concepts discussed in earlier in this section, such as the information that who can access which dataset for what purpose at what price. The pseudo code of data contract C_{DA} is shown in Algorithm 1. Let Op be the operation requested by the data consumer, D_{target} is the data requested by DC. Additionally, $\$f$ denotes the currency a DC deposits to refund gas expenditure for data use. W_{DO} is the wallet of DO, and W_{DC} is the wallet of DC. A contract should have several key functions such as *Init, RequestDataUse, ComputationComplete, and CompleteTransaction*, where *Init* initializes the policy, *RequestDataUse* takes a frozen deposit of amount $\$f$ from the DC and proceeds to the next stage of contract execution. *ComputationComplete* is used by DC to signal the completion of an off-chain data operation, and *CompleteTransaction* is called by DO to record the data usage and completes the cryptocurrency transaction. Lastly, *Revoke* invalidates the contract and can be called only by DO.

Note that privacy policy writing by itself is a research topic. Although a relevant topic, it is not the focus of this paper. Our focus in this paper is the enforcement mechanism. We use the above-described simple privacy policy to demonstrate our enforcement mechanism. We envision data consumers who need to use individual users' private data can use services similar to Google Play or Apple App Store to publish their applications (such as big data analytic app) together with the applications' hash values. Individual users can then request that only the program with the specific hash value, from certain entity, could execute on their private data, when satisfying certain conditions. Furthermore, a user can also specify that his data can only be used if only a certain level of differential privacy is satisfied, but implementation of the differential privacy mechanism has to be built into the data analytic app and become approved but this policy can be enforced. Ultimately, PrivacyGuard binds the expected behavior of the data use to the computer program.

Algorithm 1: Data Access and Utilization Smart Contract C_{DA} for DO

```
Init: Initialize with $policy$ from $DO$
parse $policy$ as (pDataset, price, operation, authorizedList) ;
dataSet ← pDataset ;
dataPrice ← price ;
authorizedOperation ← operation ;
authorizedDataConsumer ← authorizedList ;

/* Handling datause request (callable by $DC$)              */
RequestDataUse: Receive the authorization request $req$
parse $req$ as ($\$f$,$Op$,$DC$,$D_{target}$) ;
if isAuthorized($Op$,$DC$,$D_{target}$) then
    frozenCashForData ← $\$f$ ;
else
    terminate ;

/* Atomic result release & commitment                      */
/* Signaling completion (callable by $DC$)                  */
ComputationComplete:
Receive $Hash(K_{result})$ from DC
Save $Hash(K_{result})$ internally ;

/* Recording usage & finish (callable by $DO$)              */
CompleteTransaction: Receive $K_{result}$ from DO
if $K_{result}Hash = Hash(K_{result})$ then
    send deposit to DO ;
else
    exit ;

/* Callable by $DC$                                         */
Cancel: parse $req$ as ($ID_{transaction}$)
if $\neg ID_{transaction}$ valid then
    exit ;
if sender = DB then
    cancelTransaction($ID_{transaction}$) ;
else if sender = DC && requestTimeoutFlag = true then
    cancelTransaction($ID_{transaction}$) ;

/* Callable by $DO$                                         */
RevokeContract: if sender = owner then
    selfdestruct ;
```

B. Using Data Broker To Mitigate Scalability Challenge

While the blockchain technology is the core security building block behind the world's largest cryptocurrency networks, it faces a significant scalability barrier in terms of transaction capacity. For comparison, popular blockchain systems such as Bitcoin and Ethereum can process an average of 3–20 transactions per second [46], while VISA can handle an average of 24,000 transactions per second [47]. If Bitcoin were to replace VISA today, the sheer number of transactions would bring the blockchain network to a halt. The maximum rate of transaction processing by blockchain is fundamentally limited by the choice of the rate of block generation as well as the size of the block [23], [29]. Increasing the block size could lead to longer block propagation delays while increasing the block generation rate could lead to instability (high fork rate, thus decreased consensus security) in the network [48]. Recognizing this challenge, there have been several proposals to solve this problem [48], [49], [50], [51]. One of the promising approaches is the off-chain processing [49], [51], which aims to shift transactions off the main blockchain by creating a network of payment channels among individual parties and only updating the main chain upon consolidation. However, this technique cannot be directly applied to data consumption tracking in PrivacyGuard since we want to record all instances of data usage.

While there are many ongoing efforts to improve the scalability of distributed consensus platform, we take a different but complementary approach in PrivacyGuard to address this issue. A trusted delegate, which we call data broker, is used to represent a group of users. A data contract is also generated on the control plane for the data broker. However, different from a single data owner contract, the data broker contract also provides the functionality to allow individual users to delegate secret provisioning to the data broker. The related functions are listed in Algorithm 2. When a data owner wants to make use of the data broker, he will first invoke *RegisterUserDataSource* to initiate the registration of his data in the data broker. In the data plane, the data owner will need to remotely attest the data broker. If the data broker can be trusted, the data owner will then provision the data secret key to the data broker. This, however, is not the end of data registration, because the data source and quality still need to be verified by the data broker. Once verified, the data broker invokes *ConfirmUserDataSourceRegistry* to complete the data registration. Furthermore, certain functions used in result commitment are also different for data broker. *CompleteTransaction* now needs to record the data usage for all the data owners for which the data transaction covers. And the financial reward should be redistributed to the data owner. The other data related functions are identical to that presented in Algorithm 1. Another motivation for using data broker is to address the scalability challenge in remote attestation for establishing local consensus among a large group of data owners, which will be discussed in detail in the next section.

Algorithm 2: Data Broker (DB) Smart Contract C_{DB}

```
Init: Initialize from DB the contract creator
pendingDataSourceList[] ← null ;
dataSourceList[] ← null ;

/* Callable by $DO$                                         */
RegisterUserDataSource:
Receive registerRequest(operation, price, authorizedList)
add ($ID_{dataOwner}$, registerRequest) to pendingDataSourceList ;

/* Callable by $DB$                                         */
ConfirmUserDataSourceRegistry:
if sender=$ID_{databroker}$ then
    add pendingDataSourceList[$ID_{dataOwner}$] to dataSourceList

/* Callable by $DB$                                         */
CompleteTransaction: Receive $K_{result}$ from DB
if $K_{result}Hash = Hash(K_{result})$ then
    Record data usage [$Op, DC, D_{target}, timestamp$] ;
    Distribute $\$f$ to $W_{DO}$ for all involved DOs ;
else
    exit ;

RequestDataUse: (same as in $C_{DA}$)
ComputationComplete: (same as in $C_{DA}$)
Cancel: (same as in $C_{DA}$, callable by $DB$)
RevokeContract: (same as in $C_{DA}$, callable by $DB$)
```

V. Off-Chain Contract Execution

PrivacyGuard leverages blockchain to provide the control mechanisms for valued data exchanges. While the technology offers a distributed time-stamped ledger which is ideal in providing a transparent recording of individual data use, smart contract suffers from several prohibiting drawbacks when it comes to confidential data computation. First, data has to be decrypted and stored on the chain. Second, a smart contract is executed by all participating nodes in the network, therefore the cost to run complex algorithm is prohibitive even assuming data storage is not an issue.

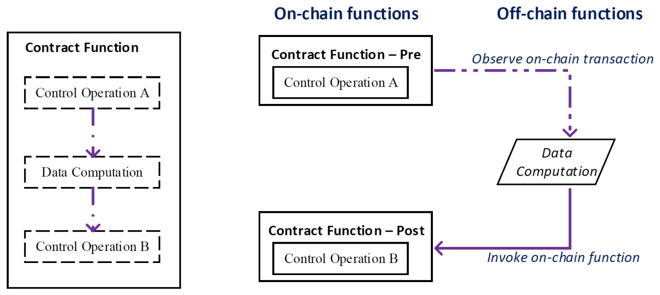


Fig. 3. Off-chain Contract Execution Overview

To tackle this problem, we introduce the concept of off-chain contract execution in PrivacyGuard to bring both the computation and data provisioning off-chain. Intuitively, we decompose smart contract into two portions, the control part and the computation part. The execution of a contract function will, therefore, be broken into multiple segments. As shown in Fig. 3, control flow will stop at the computation task and continue off-chain. The function is then resumed with another smart contract invocation when the off-chain computation task is finished. While the idea is simple, there are several unique challenges in maintaining the security of the decomposed smart contract functions, especially when the availability of the off-chain execution can be manipulated by the adversary.

A workflow of off-chain contract execution is shown in Fig. 4, the system has three stages from the high level, the off-chain contract initialization (step 1 to 4) that takes contract execution flow from the blockchain to trusted execution environment, off-chain contract execution (step 5), and lastly the off-chain contract completion with the result release (step 6 – 10) that takes contract execution flow from trusted execution environment back to the blockchain. In the remainder of the section, instead of elaborating on the workflow details, we will highlight our design principles in addressing challenges in executing contracts off-chain.

A. Establishing Trust with Local "Consensus"

The first challenge is correct execution of the contracted function. When smart contracts are executed on blockchain platform, the correctness of the execution is guaranteed by the entire network. Every node in the network executes the same contract and reaches consensus on the result. The fundamental problem is the method of consensus. If the computation task

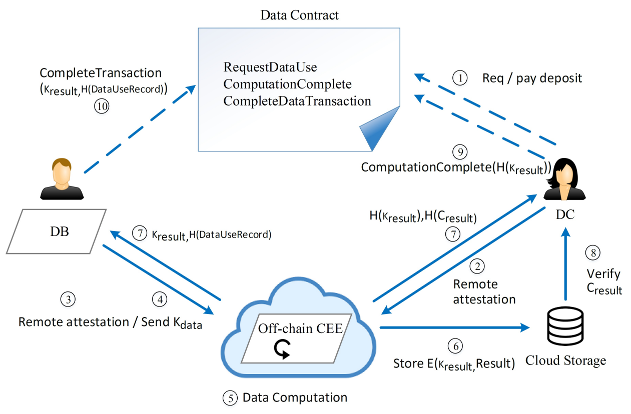

Data Contract
RequestDataUse
ComputationComplete
CompleteDataTransaction
CompleteTransaction (K_{result},H(DataUseRecord))
10
1 Req / pay deposit
9 ComputationComplete(H(K_{result}))
DB
DC
7 K_{result},H(DataUseRecord)
H(K_{result}),H(C_{result})
7
2 Remote attestation
8 Verify C_{result}
3
4
Remote attestation / Send K_{data}
Off-chain CEE
6 Store E(K_{result},Result)
Cloud Storage
5 Data Computation

Fig. 4. Off-chain Contract Execution for Data Transaction

has to be repeated by all nodes to reach consensus, then not only will there be prohibitive cost, but there will also be confidentiality problem. Furthermore, not all the nodes are equipped with the right hardware to perform those data computation at a reasonable cost.

Our observation is that the correctness of one particular computation instance only matters to the participating parties of the data transaction, i.e. the data provider (data owner or data broker) and the data consumer. Suppose if a data consumer, such as an election strategy office, is interested in analyzing the demographic data from a voting district which is polled by a dedicated data broker, then the correctness of the analysis (e.g. voting tendency) matters to the election office, since he is the direct consumer of the data. On the other hand, the correctness of the analysis also matters to the data broker, because he wants to make sure it was only the analysis result that's released, not individual users' data. Other than the data consumer and the data broker, no one needs to be concerned about the analysis. As a result, we don't need the entire network to acknowledge the correctness of the data computation and the consensus can be narrowed to only the participants of the data transaction.

In the conventional setting of distributed consensus, both the data consumer and data provider will perform the data operation and expect the same result from each other. However, it contradicts our initial goal of fine grain control on data usage if the data are directly provided to data consumers. To tackle this problem, we rely on the software remote attestation, which is widely available as primitive in trusted execution environments [26], [27]. By specifying the configuration hash in the smart contract, all of the participating parties can remotely attest the program in trusted execution environment to verify the authenticity of the loaded program, and gain confidence in the correctness of the execution. As a result, as shown in Fig. 4, the first step immediately after data transaction request is to perform remote attestation on the off-chain contract execution environment (CEE). Once correctly verified, both sides of the contracting parties can then extend their trust to CEE, knowing the environment will allow the attested program to execute securely till termination, and therefore results produced by this entity would be the consensus of both the data consumer and data broker.

B. Enforcement of Contracted Data Operation

Enforcement of data operation is the second challenge we address in our design. One of the primary motivations to employ trusted execution environment for the off-chain contract execution environment is its ability to allow a remote party to measure the computation to be performed inside the container. Furthermore, the computation will be executed with minimal influence from external environment even if the adversary can control the operating system. The measurement process is often referred to as *remote attestation*. More specifically, as one of the building blocks for TEEs, remote attestation allows a *verifier* to authenticate the enclave configuration of the *prover* remotely, including code and data of the software as well as the platform configuration. During this process, the prover generates a unique cryptographically signed proof of the current configuration. The verifier first verifies the authenticity of the proof by examining the signature, then checks to see if the configuration is as expected.

In PrivacyGuard, we propose to employ the remote attestation capability in Intel SGX to verify that the binary loaded in the enclave (i.e. the operation to be executed) is compliant with the contract. The configuration checksum, which represents the type of computation that can be performed on data is stored in the smart contract as a variable. During the initialization of the off-chain contract execution, each contracting party individually verifies that the configuration of the program loaded in the CEE enclave is indeed the one that is specified by the contract. Only when all entities involved in the contract reach the local consensus via remote attestation, will the execution of off-chain smart contract continue. As shown in Figure 4, only after the successful remote attestation of the off-chain CEE by DC in step 2 and by DO in step 4, will the DB provision to off-chain CEE the data decryption key, which enables the data operation program inside CEE to decrypt the data and continue.

C. Enforcement of Data Obligation

With the two components above, the data intensive computation can be offloaded to off-chain computing platform while maintaining the correctness of computation. However, in order to achieve the privacy goals of PrivacyGuard, computation itself has to fulfill the *data obligation*, which we refer to as the obligations of the data consumer for utilizing the user data. More specifically, it follows the general requirement of secure computation, wherein only the computation result is accessible by the data consumer, not the original source data. This requirement has several implications on the data computation program. First, it should not output any decrypted source data or any intermediate results that are derived from the source data. Second, at the end of the computation, all decrypted data and intermediate results should be sanitized.

Despite recent breakthrough in fully homomorphic encryption, performing arbitrary computation over encrypted data remains impractical for generic computation. In PrivacyGuard, we make use of trusted execution environment, such as Intel SGX, to create the environment for confidential computing, where the k_{data} will only be provisioned to the container if it can be cryptographically verified via remote attestation. The hardware, the processor specifically, enforces the isolation between the platform and the container. Since memory contents are encrypted in Intel SGX, once the keying material is removed, the data can effectively be considered sanitized. Therefore, we want to make sure that the program inside the enclave container will terminate once the contracted task is completed.

Algorithm 3: Program for Off-chain Contract Execution ($CExeProg_{enclave}$)

```
Function Initialize ($Req_{DataOp}$)
    parse request as ($DC$,$DO$,$Op$,data,sig) ;
    send $Req_{DataOp}$ and $\sigma_{att}$ to $DO$ ;
    send $Req_{DataOp}$ and $\sigma_{att}$ to $DC$ ;

Function OnDataOpeationApproval (approval)
    Parse approval as (pk$_{approver}$, OpUid, sk$_{op}$, sig$_{approver}$) ;
    if $\neg$SigVerify(sig$_{approver}$) then
        terminate ;
    if approver$_{DC}$ then
        sk$_{r}ec$ $\leftarrow$ sk$_{op}$
    if approver$_{DO}$ then
        sk$_{d}ec$ $\leftarrow$ sk$_{op}$
    if $\neg$ approval$_{DO}$ $\vee$ $\neg$ approval$_{DC}$ then
        terminate ;
    C$_{Data}$ $\leftarrow$ loadFromStorage ;
    Data $\leftarrow$ $\Sigma$.Decrypt(C$_{Data}$,sk$_{d}ec$) ;
    result := Op(Data) ;
    $g_{rec}$ := $G_{max}$ ;
    $\sigma$ := $\Sigma$.Sign(sk$_{dc}$) ;
```

The pseudo-code for off-chain CEE is listed in Algorithm. 3. $CExeProg_{enclave}$ denotes the contract execution enclave program. Additionally, k_{data} is the decryption key from DO for data under contract. Upon initialization, CEE will receive key materials from DO providing the attestation report σ_{att} is successfully verified. The program will continue to run using the encrypted data from Cloud storage, and k_{data} from DO. When the data operation is completed, the results will be written to Cloud storage and the enclave program will terminate, thus sanitizing all the original data and immediate results.

D. Ensuring Atomicity in Off-Chain Contract Execution

The last challenge is ensuring the atomicity of the contract, which arises from the off-chain contract loading. Contract functions that were previously executed in a single block are now completed via multiple function invocations that are executed in multiple blocks. Furthermore, there is no guarantee on the execution time of the off-chain computation, because an adversary controlling the platform can always interrupt the computation. This is one of the challenges for our proposed method of "local consensus". Specifically, two issues need to be addressed: contract function runtime and atomic result release.

The first issue is the contract function runtime. When the adversary has control of the off-chain computation platform, he can pause or delay the computation. For many data computations, the result can be time-sensitive. For example, in real-time location-aware advertising, if the advertisement is not delivered to the consumer while she is still in the vicinity, the advertising effectiveness can be significantly reduced. To tackle this problem, we add a time limit data operation function in the data contract. After the time limit, either of the participating parties, namely DC or DO can cancel the contract after this time limit, thus protecting their investment.

The second issue is the atomic completion of the contract, in which we would like to allow both the DO to get the payment in the control plane while allowing DC to get the computation results. This particularly challenging due to the lack of availability guarantee on the platform for off-chain computation. Using local consensus where correctness is now guaranteed using remote attestation from participating parties, computation are only executed at one location. When that platform is compromised, the availability can no longer be guaranteed. The implication of this is that the adversary can intercept and modify any external I/O from the container, including both the network and storage. Our design for the atomic completion can be observed from the last two steps shown in Fig. 4. The key idea is that result release and contract completion should be done as a single message in the control platform. To prevent DC from getting the result and not complete the payment to DO. The results are encrypted and signed before being written to the Cloud storage. Since the platform can corrupt any output from the TEE, the results include cryptographic checksum that can be authenticated. The DC obtains this result and verify its integrity. To make sure that DO will not be able to complete the transaction with an arbitrary key, DC will also receive a commitment of the key that is used to encrypt the data operation results. In PrivacyGuard, we use a hash as commitment, however, any type of commit protocol should apply. DC then invokes the smart contract in the control plane stating that it has the encrypted result and is ready to finish the data transaction if and only if the final result key that is bound by the commitment is released. Upon observing the proof of publication of the message from DC, DO will then send a message to the data contract completing the transaction including the result key k_{result} in the function invocation. Only when the hash of the key matches the commitment, will the contract write the data usage into records, release the payment to DO, and finally conclude the data transaction.

E. Using Data Broker for Scalability

One of the challenges in PrivacyGuard is the scalability limitation in the blockchain platform, which we discussed in the previous section. To alleviate the pressure on the control infrastructure, we introduced data broker. Another scalability challenge originates from the need to use a large amount of different types of data from different users, which results in a large number of remote attestations for the local consensus. When CEE needs to use data from a large number of users, naive use of remote attestation would require each iDA to individually attest and verify CEE, resulting in significant cost in both computation and network throughput. One solution to address this scalability problem is to create a trusted intermediate using SGX enclave to attest CEE for all the associated iDAs. The cost is then constant instead of linear to the number of DOs for data DC uses. We will incrementally experiment with prototypes and designs to enable efficient and scalable remote attestation.

Algorithm 4: Program for data broker TEE

```
Function OnRecvDataUseRequest (request)
    Parse request as (pk$_{DC}$,data, op, $\sigma_{att}$, pk$_{ce}$, $T$) ;
    /* verify the access right on the contract */
    if $\neg$CheckAccessRight(pk$_{DC}$,data) then
        terminate ;
    if $\neg$VerifyAttestation(pk$_{DC}$,$\sigma_{att}[CExeProg_{enclave}]$) then
        terminate ;
    transmit(list($k_{data}$),Addr$_{CEE}$) ;
    storeDataAccessOffChainForAllUsers() ;
    storeDataAccessOnChainForTheContractedUsage() ;
Function OnRegisteringForData (request)
    Parse request as (pk$_{DC}$, link$_{data}$, op list, sk$_{data}$,$\sigma_{att}$, $T$) ;
    if $\neg$CheckDataRegisterOnChain(pk$_{DC}$,link$_{data}$) then
        terminate ;
    if $\neg$VerifyData(link$_{data}$,sk$_{data}$) then
        terminate ;
    completeOnChainDataRegister() ;
Function OnCompletionOfDataTransaction (request)
    parse request as ($k_{result}$)
```

VI. Implementation and Evaluation Setting

We implemented a prototype of PrivacyGuard. There are five main applications, namely *iDataAgent (iDA)*, *Data Broker (DB)*, *Data Consumer (DC)*, *Contract Execution Environment (CEE)*, *Data Owner (DO)*. All of them are implemented in C++ using Intel SGX SDK v2.3.1 on top of Ubuntu 16.04 LTS. The total software lines of code (SLOC) for C++ components is 37,788. The on-chain components, namely the DO contract and the DB contract, are implemented in Solidity with 144 and 162 SLOC respectively. However, even though our system is fully compatible with Ethereum, we implemented and tested our prototype in the Rinkeby testnet, a proof-of-authority based blockchain offering faster and more stable transaction confirmation than the proof-of-work based main chain [52]. Table I summarizes the system implementation and evaluation parameters. The source code of the project is available at XXX[1].

[1]Will be revealed at publication.

TABLE I
A SUMMARY OF SYSTEM SETTINGS FOR EVALUATION

Item	Description	Value/Size
N	Number of DOs	$1 \rightarrow 128$
P_1	Price of one data item	0.01 ethers
P_g	Gas price	1e-9 ethers
$\mathcal{M}_{NN}$	Neural Net Classifier Model	$14 \times 8 \times 8 \times 2$
$\mathcal{M}_{NN}.EPC$	Max training epoch number	10,000
$\mathcal{M}_{NN}.ERR$	Desired training error	0.001
$\mathcal{DS}_{adult}$	UCL ML Repo's Adult dataset	48842 data points
d_size	# of data points for each DO	500

TABLE II
HARDWARE SPECIFICATION

	SGX Machine	Non-SGX Machine
CPU	Intel Core i5-7260U (2 cores 4 threads, 3.5GHz)	AMD FX-8320 (4 cores 8 threads, 3.5GHz)
Memory	16GB	32 GB
OS	Ubuntu 16.04 LTS	Ubuntu 16.04 LTS
Apps	1 DataBroker, 1 iDataAgent, 1 CEE	128 DataOwners, 1 DataConsumer

TABLE III
AVERAGE RUNTIMES OF KEY CONTROL PLANE OPERATIONS

Operation	Avg. Time (s)	STD
DB attesting enclave to DO	1.068	0.010
DB receiving & processing DO's data	0.003	0.001
DO calling DB contract's Register()	13.141*	4.008
DB calling DB contract's Confirm()	10.385*	5.237
DC calling DB contract's Request()	13.067*	4.036
DC getting iDA's approval	0.488	0.029
CEE attesting enclave to DB	1.074	0.012
CEE attesting enclave to DC	1.072	0.015
DC calling contract's ComputationComplete()	12.245*	4.298
DB calling contract's CompleteTransaction()	15.888*	5.605

*Time until the contract function call succeeds. Note that there are different finality criteria in Ethereum blockchain to judge the success of a transaction. The commonly used are *receipt* (once being mined in a block) and *confirmation* (once being confirmed by 12 subsequent blocks). Here we use *receipt* as the success event of the transaction that makes the function call. This is also used in Fig. 5(c).

A. Multi-thread Implementation

Enclave based applications often need to handle concurrent enclave operations. For example, DB may need to attest to multiple DOs at the same time, while the CEE may be required to execute parallel data operation tasks. Our implementation achieves these by harnessing the multi-threading capability of SGX enclave. The maximum thread number for CEE enclave (TCSNUM) is configured accordingly to accommodate the expected load [53].

Parallel Attestation: In the remote attestation protocol provided by Intel, a variable named attestation context operates in a state-machine manner to mark the progress of the protocol. In our implementation of DB, every attestation thread creates its own attestation context. In the end, every thread will result in the same attestation context and establish a secure channel between the enclave and the corresponding DO. Note that although CEE is primarily responsible for executing data operations, it is also capable of handling concurrent attestation requests from DOs if they choose not to rely on DB.

B. Data Operation

We use the *adult* dataset $\mathcal{DS}_{adult}$ from UCI Machine Learning Repository [54] which contains 48,842 records of consensus income data across 40 countries. Each data record contains 14 attributes and an income label of either $<= 50K$ or $> 50K$. To simulate the data generation of DOs, each DO randomly sampled 500 data points from $\mathcal{DS}_{adult}$ to be its private data.

We designed a simple data operation task for the CEE: training a $14 \times 8 \times 8 \times 2$ neural network classifier with training epoch number $\mathcal{M}_{NN}.EPC = 10,000$ and the desired training error $\mathcal{M}_{NN}.ERR = 0.001$, which makes the training task moderately time consuming. Other detailed model parameters can be found in our source code.

VII. EVALUATION RESULT

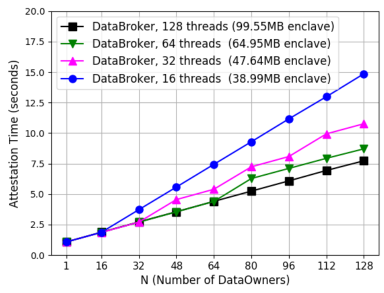
We aim to analyze the high level performance cost and overhead of conducting data operations using PrivacyGuard. Up to 160 DOs and the DC are running on a non-SGX machine while the Data Broker, iDataAgent, and CEE are running on an SGX-machine (Intel NUC KIT NUC7i5BNH). Table II summarizes the detailed hardware specification. The two machines are connected in a gigabyte LAN. The evaluation was conducted in three parts: control plane runtimes, control plane costs, and data plane runtimes.

A. Control Plane Runtimes

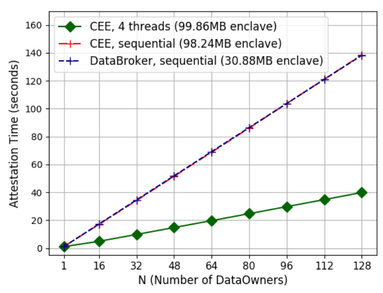
We ran the control flow of PrivacyGuard ten times to obtain the average runtime of each control operation. The ten runs were evenly spread out from 8:00am to 5:00 pm on the day of 11/1/2018 to reflect the varying nature of the response times of the underlying blockchain. From the results shown in Table III, we find that non-contract-call operations typically have much shorter runtimes, meanwhile the contract calls may take up to 20 seconds to be written in the blockchain, which is consistent with the 15-second block interval of Rinkeby. Due to the varying network conditions, contract call runtimes also tend to have higher variations.

To evaluate the scalability of the remote attestations to multiple DOs, we tested the scenario where N DOs simultaneously attest the DB or CEE. The experiment was repeated under different maximum thread numbers (TCSNUM). The attestation becomes a sequential process with no parallelism when TCSNUM=1. The result is shown in Fig. 5(a) and 5(b). For both DB or CEE, multi-thread attestation is significantly faster than sequential attestation. When N=128, the 128-thread DB and the 4-thread CEE take less than 1/18 and 1/3 attestation time of their sequential counterparts respectively.

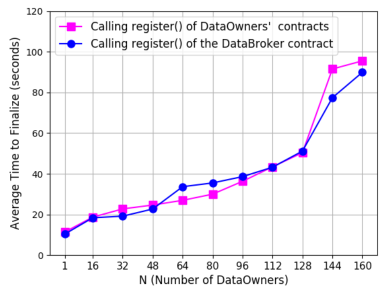
To further evaluate the performance constraints imposed by the underlying blockchain network, we measured the average transaction finalization delay in a congested environment. To



(a)



(b)



(c)

Fig. 5. (a,b) Attestation times of DB and CEE when N DOs simultaneously starts the attestation. The zig-zags in a curve are the result of batch operations when the requests outnumber the available threads. (c) Average time to finalize when N DOs simultaneously call the **register()** function in their own DO contracts or the DB contract.

simulate the congestion as much as possible, we set up 160 DOs to simultaneously send out a transaction calling the *Register()* function in the DB contract and their own DO contracts. The result is shown in Fig. 5(c). As more DOs send transactions at the same time, the average time to finalize a transaction increases dramatically. Since the scalability of the underlying blockchain is an orthogonal problem, an alternative solution is to require DOs to call the *Register()* function according to a time schedule that minimizes congestion.

B. Control Plane Cost

The monetary cost of the control plane mainly comes from the gas cost of operating smart contracts. At the start, every DO registers its data item on its own contract and the DB contract. The DB fetches data from whoever registered with its contract and routinely confirms all new registries. The Data Consumer then requests for the data items from all N DOs by sending a request transaction to the DB contract (or separate requests to the DO contracts) with a sufficient deposit to cover the price before proceeding to enclave attestation and data operation. We repeated the experiment for $N = 1 \rightarrow 10$ and obtained the gas costs and dollar equivalents for each contract function call, based on the fixed gas price $P_g = 10^{-9}$ ethers and the price of ether on 10/31/2018, which was \$197.85[2].

We find that in both DB and DO contracts the costs of calling *register()* and *cancel()* do not depend on the number of registered DOs. We call these type of function calls *scale-independent*, which also include calling *constructor()* (contract creation). In contrast the function calls whose costs depend on the number of registered DOs are called *scale-dependent*. The costs of scale-independent function calls and scale-dependent functions calls are shown in Table IV and Fig. 6(a) respectively. The cost of every scale-dependent call grows near linearly with the number of DOs. Notably, the costs of *request()* and *ComputationComplete()* called by the Data Consumer grow faster than the costs of *Confirm()* and *CompleteTransaction()* called by the DB. This implies the total cost will increasingly become the burden of the

[2]Source: https://coinmarketcap.com/

TABLE IV
COST OF THE DATA BROKER CONTRACT'S SCALE-INDEPENDENT FUNCTION CALLS

Contract	Function	Caller	Cost (gas)	Dollar Equiv.*
Data Broker	Constructor()	DB	951747	\$0.18830
	Register() (new)	DO	156414	\$0.03095
	Register() (update)	DO	30121	\$0.00596
	Cancel()	DC	81998	\$0.01622
Data Owner	Constructor()	DO	846794	\$0.16754
	Register() (new)	DO	125392	\$0.02481
	Register() (update)	DO	45177	\$0.00894
	Cancel()	DC	66954	\$0.01325

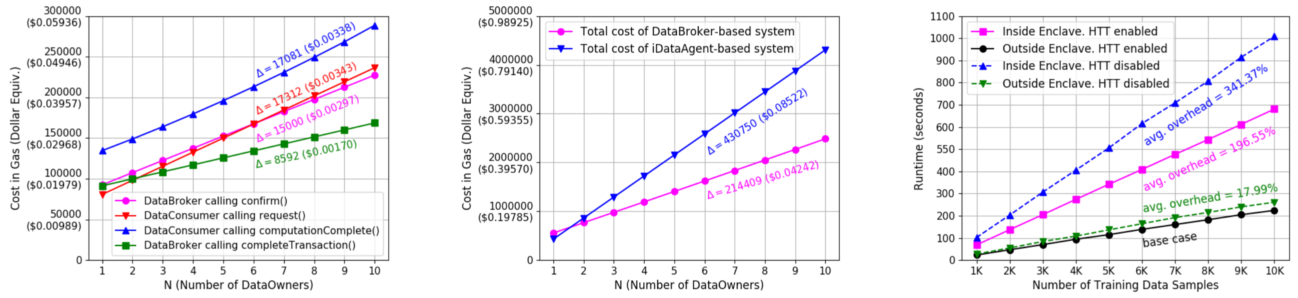
*Based on the gas price of 10^{-9} ethers and the ether price of \$197.85 on 10/31/2018.

DataConsumer, which is a scalable trend for the system, as the Data Consumer always has more incentives to pay for data usage.

To evaluate the scalability gain brought by the DB, here we compare the base case where individual DOs share individual data via their own contracts versus via the DB contract. In both cases, the total amount of data requested by the DC and subsequently operated with by the CEE are the same. We counted the costs of all function calls except for the contract creation (calling *Constructor()*) and computed the total cost. The result in Fig. 6(b) shows that it costs the DB based system much less gases to accommodate one extra DO (\$0.04242) compared to that of the iDataAgent based system (\$0.08522). This result together with the result of control plane runtimes (Fig. 5) have verified DB's ability to provide financial and performance scalability to PrivacyGuard in face of a growing number of DOs.

C. Data Plane Cost

The security provided by SGX enclave comes at a cost of efficiency. To evaluate such cost, we implemented an untrusted version (outside enclave) of the data operation task that runs on the same Linux machine. Since in the trusted version the enclave takes in encrypted data and decrypts it before data operation, in this test we directly fed unencrypted data to



(a) (b) (c)

Fig. 6. (a) Gas costs of the DB contract's scale-dependent function calls. (b) Total gas costs of the DB based system and the iDataAgent based system. The dollar equivalents are computed based on the gas price of 10^{-9} ethers and the ether price of \$197.85 on 10/31/2018. (c) Runtimes of training a $14\times8\times8\times2$ neural network classifier for different training data sizes under four hardware settings.

the enclave to make it a fair comparison, as the untrusted version doesn't have to deal with encrypted data. Moreover, recent works show Intel's Hyperthreading Technology (HTT) has flaws that may impair the security of SGX enclaves [39].

To evaluate the impact of enclave and HTT on computation time, we first disabled the CPU's TurboBoost feature to avoid nonlinear CPU performance boost and carried out our data plane experiment under four different hardware options: 1) *inside-enclave-HTT-enabled*, 2) *outside-enclave-HTT-enabled*, 3) *inside-enclave-HTT-disabled*, 4) *outside-enclave-HTT-disabled*.

Figure 6(c) shows time consumption of training the mentioned neural network classifier under four hardware options. Table V summarizes the average overheads and shows the marginal overhead caused by either enclave or disabling HTT. Column-wisely, we can see the overheads caused by disabling HTT are 17.99% and 48.84% for outside-enclave and inside-enclave respectively. This implies disabling HTT will significantly increase the runtime inside enclave. Row-wisely, the overheads caused by enclave are 196.55% and 274.13% for HTT-enabled and HTT-disabled respectively. This means running programs in enclave is significantly more time consuming than outside enclave, at least for our this four-layer neural network training. And this overhead exacerbates when HTT is disabled. A possible explanation for this overhead is the mismatch between the machine learning program's high need for parallelism and the limited multi-threading capability of SGX enclaves, which assign logical threads to enclave applications up to the processor limit of the CPU. Another possible factor is memory usage of SGX. For example the L3 cache accessible to SGX may be far from enough to accommodate the machine learning task. We will study the performance caveats of Intel SGX and possible solutions in future works.

VIII. Security Analysis

PrivacyGuard has two key components, namely blockchain in the control plane and the trusted execution environment in the data plane. The security properties of these two primitives have received significant amount of attentions recently [39], [44], [45], [55], [50], [36], [32], [30]. In the discussion below, the focus will be on issues in harmonizing the trusted execution environment and blockchain. More specifically, in PrivacyGuard, a single on-chain contract function is partitioned into multiple functions with both on-chain and off-chain components with the proposed off-chain execution technique. This technique has opened new avenues for adversaries that are not previously available on conventional blockchain platform.

TABLE V
Average runtime overhead caused by SGX enclave and disabling HTT

Avg. Runtime Overhead	HTT Enabled	HTT Disabled	*Marginal
Outside Enclave	0	17.99%	–
Inside Enclave	196.55%	341.37%	48.83%
*Marginal	–	274.13%	–

*Marginal overhead is the relative runtime overhead for the specific row/column.

A. Exploiting Control of Cloud Platform

Off-chain contract execution significantly reduces the computation load on the network, but at the same time, also provides a single point of failure for the adversary to launch attack. When the attacker is able to compromise the cloud platform, there are several attacks she can launch. During the execution of the off-chain contract stage, the adversary can stop the program at any point of time, and also drop or reorder any message. However, he cannot tamper with the content of the network message as long as the connection is secured. Anticipating the execution of the data contract can be halted by the adversary at any stage, our design on the data contracts allows any party to abort the data transaction without incurring financial loss. The attacker may also attempt to launch side channel attack or directly attempt to exploit a software vulnerabilityt in the data analytic program within the SGX enclave, and is out of the scope of this paper. It should however be noted that, system implementations are often imperfect. Due to the use of TEE to provide secure execution environment for the data analytic program, our system inherits vulnerabilities of in implementation of both TEE and data analytic program.

B. Exploiting the Incoherence Between Data Plane and Control Plane in Result Commitment

There are three main players in the last stage (result commitment and release) of the data operation contract: DO, DB and the Cloud. Ideally, the release of the data and the deposit should be an atomic decision. However, with the separation of control plane and the data plane, these operations have become inherently non-atomic. This is further complicated by the fact that there is lack of availability guarantee for the TEE technology we choose to implement PrivacyGuard, Intel SGX. Transactions on blockchain is used to signal the completion of the data operations and enable the completion of the financial transaction for the data use. On the other hand, this does not necessarily guarantee the delivery of result to the data consumer who has already paid for the service using the control plane operation on the blockchain. There are several ways an adversary can attack our result commitment and release protocol.

1) DO controls the cloud: The DO or DB can attempt to complete the data transaction without uploading the experiment result. Using the control of the Cloud platform, the DO can either maliciously modify the data operation result or he can simply deny the output of results. This attack is thwarted by having the *dataOperationComplete* contract function to enforce the result commitment before completing the data transaction.

2) DC controls the cloud: Under unlikely circumstances, if a malicious DC is powerful enough to compromise the platform which DB and DO are running on, he might be able to capture the last commitment message to get the result release key, but never forward the blockchain invocation message to the network. This way, the DC can obtain the result without completing the payment. However, with an increasing number of nodes accepting transaction broadcasts in secure manner, even though the adversary might be able to block all the connection, he cannot break the secure connection by launching man-in-the-middle attack. This attack is thwarted by our system sending the *CompleteTransaction* message to a collection of transaction recipients in secure channels. The *CompleteTransaction* message, which contains the result release key, will only be transmitted if the secure channel can be established. Therefore the adversary would have to block all of the traffic, otherwise he cannot block the transaction broadcasts. When all the network connections are blocked, the result will never be released, which is not a problem for DB and DO. When the adversary can only block a subset of the result broadcast messages, the data transaction will continue to complete and thus is not an issue either. Furthermore, it is also possible to add functionality to verify the proof of publication in data broker instead of relying on broadingcasting via secure channel.

IX. Related Work

A. Privacy Protection

Privacy-preserving computation has been an active area of research in the past decade [37], [56], [57], [58], [59], [60]. With the increasing reliance on rich data, there has been a significant amount of research on applying cryptographic techniques to perform privacy preserving computation [61], [25], [58], [59]. Recently, hardware-assisted secure environment has also become a popular component in these privacy-preserving computation system due to its attractive security features, such as enabling application level secure container in the Intel SGX. Secure enclave technology [26] has been adapted in numerous works to achieve privacy-preserving computation [57], [37], [56], [62], [63], [64]. Ryoan [37] is closely related to PrivacyGuard. It combines native client sandbox and Intel SGX to confine data processing module. Similar to PrivacyGuard, it protects the confidentiality of data using trusted execution environment. However, Ryoan aims to achieve data confinement with a user-defined directed acyclic graph that specifies information flow. In comparison, PrivacyGuard allows data user and consumer to negotiate data usage using smart contract, and provides transparent recording on the data usage.

When the data processing system can be trusted, there is a large body of work studying privacy languages and their enforcement systems [17], [18], [19]. Unlike the aforementioned systems, PrivacyGuard aims to enable secure processing without relying on the integrity of the platform.

B. Blockchain Application

The idea of moving computation off-chain to improve the performance and security is mentioned in [65], [31], [20], [30]. Choudhuri et al. [65] combines blockchain with TEE to build one-time programs that resemble to smart contracts but only aim for a restricted functionality. Arbitrum [31], is a system that delegates the task of smart contract verification to a small subset of managers that are incentivized to execute the virtual machines honestly. Different from Arbitrum, PrivacyGuard focuses on moving computation off-chain using trusted execution and local consensus that involve only the contract participating parties. The Intel Private Data Object (PDO) project [20] and the Ekiden [30] are two concurrently developed projects that are closely related to PrivacyGuard. Similar to PrivacyGuard, Ekiden aims to combine trusted computing and distributed ledger to enable confidential contract execution. However, Ekiden [30] aims to modify the blockchain ecosystem to enable the off-chain execution in a collection of computer nodes. In comparison, PrivacyGuard is designed, implemented and evaluated on existing blockchain infrastructure. The Intel PDO [20] is another project that aims to combine Intel SGX and distributed ledger to allow distrusting parties to work on the data in a confidential manner. However, the system focuses heavily on a permission-based model with significant overhead for bootstrapping trust.

Several other related works aim to guarantee the computation integrity but not the privacy [66], [67]. Others use blockchain as a platform for incentives and penalties [68], [69], [70], [31]. Compared to the aforementioned methods, PrivacyGuard is designed to enable large-scale and complex data market operations using TEE. There have also been several research work that utilize blockchain as a platform to enable flexible data access control [71], [72], [16]. However, none of them provide a concrete design and implementation that would allow fine-grained control of data usage.

The confidentiality issue of smart contracts has also attracted a significant amount of attention in the past few years. Researchers attempt to use cryptographic techniques to fix the problems [65], [32], [73], [30], [74]. The possibility of using trusted hardware for blockchain process is discussed in Hawk [32], and also realized in [30], [65], [75], [20], [76]. The Coco framework [76] is proposed to use trusted execution environment to protect the confidentiality of smart contracts in Ethereum. However, there is only a high-level description in its whitepaper.

X. Conclusion

In this paper, we proposed PrivacyGuard, a data exchange platform that combines blockchain and secure execution environment to enable fine-grained data use control and tracking. Our system provides a technical solution to address one of the most important privacy issues in data analytics—transparent enforcement of data use. Blockchain can not only be used as a tamper-proof platform that records data use, but also facilitate financial transactions to incentivize data sharing. To enable complex and confidential operations on private data, PrivacyGuard splits smart contract function into control operations and data operations. Remote attestation and trusted execution environment are used to achieve local consensus of all contract participating parties on the a data operating platform. Atomicity of the contract completion is facilitated by a commitment protocol before the result release. We implemented and open-sourced our PrivacyGuard platform and evaluated it in a simulated data market. Our evaluation result shows the feasibility of executing complex data operations in a confidential manner using the platform. In the future, we plan to further investigate the economic aspect of the private data market enabled by PrivacyGuard.

References

[1] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, *et al.*, “Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning,” *arXiv preprint arXiv:1711.05225*, 2017.

[2] “Facebookcambridge analytica data scandal.” https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal.

[3] “How tech companies deceive you into giving up your data and privacy.” https://goo.gl/hSfaUX.

[4] “Tim cook: Personal data collection is being ’weaponized against us with military efficiency’.” https://goo.gl/BsWB3k.

[5] H. Nissenbaum, “Privacy as contextual integrity,” *Wash. L. Rev.*, vol. 79, p. 119, 2004.

[6] “National privacy research strategy.” https://www.nitrd.gov/PUBS/NationalPrivacyResearchStrategy.pdf.

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS ’06, (New York, NY, USA), pp. 89–98, ACM, 2006.

[8] A. Sahai, “Ciphertext-policy attribute-based encryption,” in *In Proceedings of the IEEE Symposium on Security and Privacy*, pp. 321–334, 2007.

[9] G. Wang, Q. Liu, and J. Wu, “Hierarchical attribute-based encryption for fine-grained access control in cloud storage services,” in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS ’10, (New York, NY, USA), pp. 735–737, ACM, 2010.

[10] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in *Infocom, 2010 proceedings IEEE*, pp. 1–9, Ieee, 2010.

[11] C. Dwork, “Differential privacy,” in *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP’06, (Berlin, Heidelberg), pp. 1–12, Springer-Verlag, 2006.

[12] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “L-diversity: Privacy beyond k-anonymity,” *ACM Trans. Knowl. Discov. Data*, vol. 1, Mar. 2007.

[13] L. Sweeney, “K-anonymity: A model for protecting privacy,” *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, pp. 557–570, Oct. 2002.

[14] N. Li, T. Li, and S. Venkatasubramanian, “t-closeness: Privacy beyond k-anonymity and l-diversity,” in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 106–115, IEEE, 2007.

[15] G. Zyskind, O. Nathan, and A. S. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in *Security and Privacy Workshops (SPW)*, IEEE, 2015.

[16] G. Zyskind, O. Nathan, and A. Pentland, “Enigma: Decentralized computation platform with guaranteed privacy,” *arXiv preprint arXiv:1506.03471*, 2015.

[17] S. Sen, S. Guha, A. Datta, S. K. Rajamani, J. Tsai, and J. M. Wing, “Bootstrapping privacy compliance in big data systems,” in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 327–342, IEEE, 2014.

[18] E. Elnikety, A. Mehta, A. Vahldiek-Oberwagner, D. Garg, and P. Druschel, “Thoth: Comprehensive policy compliance in data retrieval systems.,” in *USENIX Security Symposium*, pp. 637–654, 2016.

[19] A. Datta, M. Fredrikson, G. Ko, P. Mardziel, and S. Sen, “Use privacy in data-driven systems: Theory and experiments with machine learnt programs,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1193–1210, ACM, 2017.

[20] M. Bowman, A. Miele, M. Steiner, and B. Vavala, “Private data objects: an overview,” *arXiv preprint arXiv:1807.05686*, 2018.

[21] B. Custers and H. Uršič, “Big data and data reuse: a taxonomy of data reuse for balancing big data benefits and personal data protection,” *International Data Privacy Law*, vol. 6, no. 1, pp. 4–15, 2016.

[22] “Why transparency is critical in this data economy.” https://www.forbes.com/sites/michaelmarrale/2018/07/23/the-almighty-data/#718653b14bd2.

[23] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[24] “Ethereum: Blockchain app platform.” https://www.ethereum.org/.

[25] D. Boneh, A. Sahai, and B. Waters, “Functional encryption: Definitions and challenges,” in *Theory of Cryptography Conference*, pp. 253–273, Springer, 2011.

[26] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, “Innovative instructions and software model for isolated execution.,” in *HASP@ ISCA*, p. 10, 2013.

[27] “ARM Security Technology, Building a Secure System using TrustZone Technology,” apr 2009.

[28] N. Szabo, “Formalizing and securing relationships on public networks,” *First Monday*, vol. 2, no. 9, 1997.

[29] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, *et al.*, “On scaling decentralized blockchains,” in *International Conference on Financial Cryptography and Data Security*, pp. 106–125, Springer, 2016.

[30] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, “Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution,” *arXiv preprint arXiv:1804.05141*, 2018.

[31] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, “Arbitrum: Scalable, private smart contracts,” in *Proceedings of the 27th USENIX Conference on Security Symposium*, pp. 1353–1370, USENIX Association, 2018.

[32] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 839–858, IEEE, 2016.

[33] D. Song, J. Lettner, P. Rajasekaran, Y. Na, S. Volckaert, P. Larsen, and M. Franz, "Sok: Sanitizing for security," in *2019 2019 IEEE Symposium on Security and Privacy (SP)*, vol. 00, pp. 187–207.

[34] V. Costan, I. A. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation.," in *USENIX Security Symposium*, pp. 857–874, 2016.

[35] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "Iron: functional encryption using intel sgx," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 765–782, ACM, 2017.

[36] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 270–282, ACM, 2016.

[37] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, "Ryoan: A distributed sandbox for untrusted computation on secret data.," in *OSDI*, pp. 533–549, 2016.

[38] L. Rainie and M. Duggan, "Privacy and information sharing," *Pew Research Center*, vol. 16, 2016.

[39] J. Van Bulck, F. Piessens, and R. Strackx, "Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018.

[40] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, and C. A. Gunter, "Leaky cauldron on the dark land: Understanding memory side-channel hazards in sgx," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2421–2434, ACM, 2017.

[41] N. Zhang, H. Sun, K. Sun, W. Lou, and Y. T. Hou, "Cachekit: Evading memory introspection using cache incoherence," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 337–352, IEEE, 2016.

[42] N. Zhang, K. Sun, D. Shands, W. Lou, and Y. T. Hou, "Trusense: Information leakage from trustzone," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 1097–1105, IEEE, 2018.

[43] J. Van Bulck, N. Weichbrodt, R. Kapitza, F. Piessens, and R. Strackx, "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1041–1056, 2017.

[44] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *2015 IEEE Symposium on Security and Privacy*, pp. 640–656, IEEE, 2015.

[45] N. Zhang, K. Sun, W. Lou, and Y. T. Hou, "Case: Cache-assisted secure execution on arm processors," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 72–90, IEEE, 2016.

[46] V. Buterin, "Privacy on blockchain." https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/.

[47] "Smart business retail — visa." https://usa.visa.com/run-your-business/small-business-tools/retail.html.

[48] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol.," in *NSDI*, pp. 45–59, 2016.

[49] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *Technical Report (draft)*, 2015.

[50] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 279–296, USENIX Association, 2016.

[51] "Raiden network." https://https://raiden.network/.

[52] "Rinkeby: Ethereum testnet." https://www.rinkeby.io/.

[53] Intel, "Intel software guard extensions (intel sgx) sdk for linux os - developer reference," 2018.

[54] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017.

[55] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 254–269, ACM, 2016.

[56] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "Vc3: Trustworthy data analytics in the cloud using sgx," in *Security and Privacy (SP), 2015 IEEE Symposium on*, pp. 38–54, IEEE, 2015.

[57] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors.," in *USENIX Security Symposium*, pp. 619–636, 2016.

[58] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2014.

[59] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *ACM Sigmod Record*, vol. 33, no. 1, pp. 50–57, 2004.

[60] N. Zhang, J. Li, W. Lou, and Y. T. Hou, "Privacyguard: Enforcing private data usage with blockchain and attested execution," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp. 345–353, Springer, 2018.

[61] R. Pass, E. Shi, and F. Tramer, "Formal abstractions for attested execution secure processors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 260–289, Springer, 2017.

[62] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Opaque: An oblivious and encrypted distributed analytics platform," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, (Boston, MA), pp. 283–298, USENIX Association, 2017.

[63] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," *arXiv preprint arXiv:1803.05961*, 2018.

[64] B. A. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "Iron: Functional encryption using intel sgx," 2017.

[65] A. R. Choudhuri, M. Green, A. Jain, G. Kaptchuk, and I. Miers, "Fairness in an unfair world: Fair multiparty computation from public bulletin boards," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 719–728, ACM, 2017.

[66] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 706–719, ACM, 2015.

[67] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," 2017.

[68] R. Kumaresan and I. Bentov, "Amortizing secure computation with penalties," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 418–429, ACM, 2016.

[69] R. Kumaresan, T. Moran, and I. Bentov, "How to use bitcoin to play decentralized poker," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 195–206, ACM, 2015.

[70] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 443–458, IEEE, 2014.

[71] G. Zyskind, O. Nathan, *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *Security and Privacy Workshops (SPW), 2015 IEEE*, pp. 180–184, IEEE, 2015.

[72] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Open and Big Data (OBD), International Conference on*, pp. 25–30, IEEE, 2016.

[73] I. Bentov, Y. Ji, F. Zhang, Y. Li, X. Zhao, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware.," *IACR Cryptology ePrint Archive*, vol. 2017, p. 1153, 2017.

[74] F. Tramer, F. Zhang, H. Lin, J.-P. Hubaux, A. Juels, and E. Shi, "Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 19–34, IEEE, 2017.

[75] G. Kaptchuk, I. Miers, and M. Green, "Giving state to the stateless: Augmenting trustworthy computation with ledgers," tech. rep.

[76] "The confidential consortium blockchain framework." https://github.com/Azure/coco-framework/blob/master/docs/Coco%20Framework%20whitepaper.pdf.