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

Yang Xiao[1]*, Ning Zhang[2], Jin Li[3], Wenjing Lou[1], and Y. Thomas Hou[1]

[1] Virginia Polytechnic Institute and State University, VA, USA
[2] Washington University in St. Louis, MO, USA
[3] Guangzhou University, Guangzhou, China

Abstract. The abundance and rich varieties of data are 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 data owner's private data. In this paper, we propose PrivacyGuard, a system that leverages blockchain smart contract and trusted execution environment (TEE) to enable individual's control over the access and usage of their private data. Smart contracts are used to specify data usage policy, i.e., who can use what data under which conditions and what analytics to perform, while the distributed blockchain ledger is used to keep an irreversible and non-repudiable data usage record. To address the efficiency problem of on-chain contract execution and to prevent exposing private data on the publicly viewable blockchain, PrivacyGuard incorporates a novel TEE-based off-chain contract execution engine along with a protocol to securely commit the execution result onto blockchain. We have built and deployed a prototype of PrivacyGuard with Ethereum and Intel SGX. Our experiment result demonstrates that PrivacyGuard fulfills the promised privacy goal and supports analytics on data from a considerable number of data owners.

Keywords: Privacy, data access and usage control, trusted execution, blockchain, smart contract

1 Introduction

The recent emergence of big data analytics and artificial intelligence has made life-impacting changes in many sectors of society. One of the fundamental enabling components for the recent advancements in artificial intelligence is the abundance of data. However, as more information on individuals is collected, shared, and analyzed, there is an increasing concern on the privacy implication. In the 2018 Facebook-Cambridge Analytica data scandal, an API, originally designed to allow a third party app to access the personality profile of limited participating users, was misused by Cambridge Analytica to collect information on 87 million of Facebook profiles without the consent of the users. These illicitly harvested private data were later used to create personalized psychology profiles for political purposes [21]. With increasing exposure to the

* To present in the 25th European Symposium on Research in Computer Security (ESORICS 2020), September 14 - 18, 2020.

privacy risks of big data, many now consider the involuntary collection of personal information a step backward in the fundamental civil right of privacy [19], or even in humanity [43,44]. Yet, driven by economic incentives, 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 [33]. 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 [32]. 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 [23,4,49,53] and data anonymization [16,29,41,28]. Only recently, there have been a few works that attempted to address the second aspect of privacy from the architecture perspective [61,60,38,17,14,6]. In fact, many believe that the prevention of this kind of "second-hand" data (mis)use can only be enforced by legal methods [13]. Under 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 he would use the data for purposes not consented by the original data owner, 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.

Our Contribution Building upon our previous work [55], we present the design, implementation and evaluation of *PrivacyGuard* in this paper. PrivacyGuard empowers individuals with full control over the access and usage of their private data in a data market. The data owner is not only able to control who can have access to their private data, but also ensured that the data are used only for the intended purpose. To realize this envisioned functionalities of PrivacyGuard, three key requirements need to be met. First, users should be able to define their own data access and usage 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. Third, the actual usage of data should have a verifiable proof to show its compliance to the policy.

Blockchain, the technology behind Bitcoin [31] and Ethereum [18], has exhibited great potential in providing security and privacy services. Smart contract is a program that realizes a global state machine atop the blockchain and has its correct execution enforced by the blockchain's consensus protocol. PrivacyGuard enables individual users to control the access and usage of their data via smart contract and leverages the blockchain ledger for transparent and tamper-proof recording of data usage.

While smart contract and blockchain appear to be the perfect solution, there are fundamental limitations if applied directly. First, data used by smart contracts are uploaded in the form of blockchain transaction payload, which is not designed to hold arbitrarily large amount of data due to communication burden and scalability concerns [31,12]. Second, smart contracts are small programs that have to be executed by all participants in the network, which raises serious computational efficiency concerns. For the same reason, existing platforms such as Ethereum are not purposed to handle complex contract programs [51]. Last but not least, data used by smart contracts are available to every participant on blockchain by design, which conflicts with the confidentiality requirement of user data. Existing secure computation techniques for preserving confidentiality and utility of data, such as functional encryption [5], can nonetheless be prohibitively expensive for the network.

To tackle data and computation scalability problems, PrivacyGuard splits the private data usage enforcement problem into two domains: the control plane and the data plane. In the control plane, individual users publish the availability as well as the usage policy of their private data as smart contracts on blockchain. Data consumers interact with the smart contract to obtain authorization to use the data. Crucially, the actual data of the users are never exposed on the blockchain. Instead, they are stored in the cloud in encrypted forms. Computation on those private user data as well as the provision of secret keys are accomplished off-chain in the data plane with a trusted execution environment (TEE) [30,2] on the cloud.

When a data contract's execution is split into control and computation, where the computation actually takes place off-chain, several challenges occur. First, the correctness of the contract execution can no longer be guaranteed by the blockchain consensus. To this end, we propose "local consensus" for the contracting parties to establish trust on the off-chain computation via remote attestations. Second, the execution of contract is no longer atomic when the computation part 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, private data are protected inside the TEE enclave and secrets are only provisioned when approved according to the contract binding.

We implemented a prototype of PrivacyGuard using Intel SGX as the TEE technology 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 smart contract platforms. The platform fulfills the goal of user-define data usage control at reasonable cost and we show that it is feasible to perform complex data operations with the security and privacy protection as specified by the data contract.

To summarize, we make the following contributions in this paper:

- We propose PrivacyGuard, a platform that combines blockchain smart contract and trusted execution environment to address one of the most pressing problems in big data analytics—trustworthy private data computation and usage control. PrivacyGuard essentially allows data owners to contribute their data into the data market and specify the context under which their data can be used.
- We propose a novel construction of off-chain contract execution environment to support the vision of PrivacyGuard, which is the key to improving the execution efficiency of smart contract technology and enabling trustworthy execution of complex contract program without solely relying on costly network consensus.
- We implemented a prototype of PrivacyGuard using Intel SGX and Ethereum smart contract and and deployed it in a simulated data market. Our evaluation shows that

PrivacyGuard is capable of processing considerable volumes of data transactions on existing public blockchain infrastructure with reasonable cost.

2 Background

Blockchain and Smart Contract Blockchain is a recently emerged technology used in popular cryptocurrencies such as Bitcoin [31] and Ethereum [18]. 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 and achieve the same functionality with comparable reliability. When the majority of the network's voting power (hashing power, stake value, etc.) are controlled by honest participants, the shared blockchain becomes a safe and timestamped record of the network's activities. The conceptual idea of programmable electronic "smart contracts" dates back nearly twenty years [42]. When implemented in the blockchain platform (eg. Ethereum), smart contracts are account-like entities that can receive transfers, make decisions, store data or even interact with other contracts. The blockchain and the smart contract platform however have several drawbacks in transaction capacity [12], computation cost [9,26], as well as privacy of user and data [27,26].

Trusted Execution Environment Creating vulnerability-free software has long been considered a very challenging problem [40]. Researchers in the architecture community in both academia [11] and industry [2,30] have embraced a new paradigm of limiting the trusted computing base (TCB) to only the hardware, realizing a trusted execution environment (TEE). The well-known Intel SGX [30] is an instruction set extension to provide TEE functionalities. Applications are executed in secure containers called *enclaves*. The hardware guarantees the integrity and confidentiality of the protected application, even if the platform software is compromised. TEE has recently been adapted as a powerful tool to support blockchain-based applications [22,54,25,9].

3 PrivacyGuard Overview

3.1 System Goal and Architecture

The vision of PrivacyGuard is to not only protect data owner privacy but also promote a vibrant data sharing economy, in which data owners can confidently sell the right to use their data to data consumers for profits without worrying about data misuse. To realize this, there are three specific goals. First, data encryption/decryption are fully controlled by data owners. Untrusted parties (eg. cloud storage and data consumers) can not obtain or possess data owners plaintext data. Second, Data owners are able to control who can access which data items under what conditions for what usage. The data usage records should be non-repudiable and auditable by data owners. Third, the security mechanism of our system should be able to capture user-defined policies and enforce the compliance of the policies during the execution of data access.

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

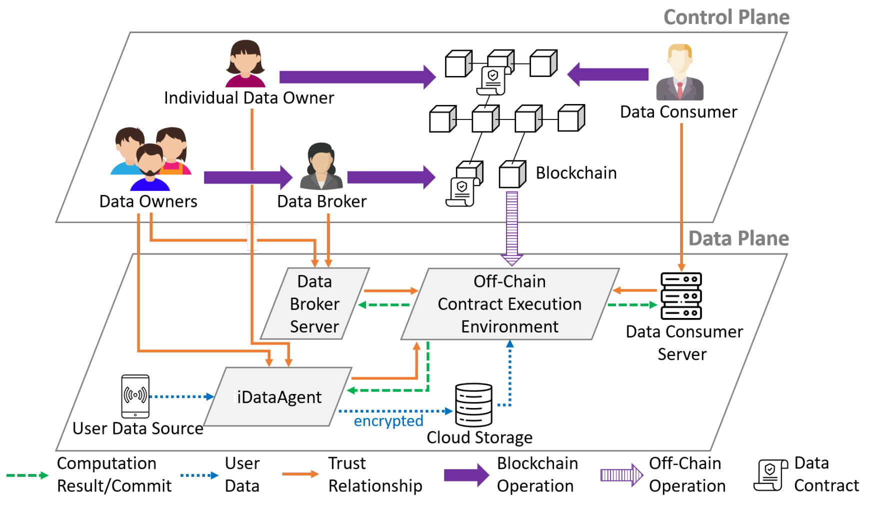


Fig. 1. System architecture for PrivacyGuard framework.

that 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)*. Classified by the assigned responsibility, there are three main functional components in the PrivacyGuard framework:

Data Market Data market is an essential PrivacyGuard subsystem that supports the supply, demand and exchange of data on top of blockchain. For data access and usage control, DO can encode the terms and conditions pertaining to her personal data in a data contract, and publish it on a blockchain platform such as Ethereum. Data usage by DC is recorded via transactions that interact with the data contract.

iDataAgent (iDA) and Data Broker (DB) iDA is a trusted entity representing an individual DO and responsible for key management for the DO. It also participates in contract execution by only provisioning the data key material to attested remote entities. Since it is often not realistic to expect individual DO to be connected all the time, iDA can also be instantiated as a trusted program in a TEE-enabled cloud server. To address the inherent transaction bandwidth limit of the blockchain network, DB is introduced to collectively represent a group of users.

Off-chain Contract Execution Environment (CEE) This off-chain component executes data operations contracted between DO(s) and DC in a TEE enclave. The trusted execution guarantees correctness as if it was executed on-chain. The computation result is securely committed to DC while enforcing the contract obligation.

3.2 PrivacyGuard High-Level Workflow

The workflow of PrivacyGuard proceeds in three stages which can function concurrently. Stage 1 and 2 involve the supply side (DO, iDA, DB) that prepares the data items and usage contracts while stage 3 characterizes the regular operation.

Stage 1: Data Generation, Encryption, and Key Management In this stage, a DO's data are generated by its data sources and collected by its iDA, who passes the encrypted data to the cloud storage. Keys for data en/decryption are generated by the DO via interface to iDA and managed by iDA. For a group of users with common data types, they can delegate their trust to a DB by remote-attesting the DB's enclave and provision data keys to the enclave.

Stage 2: Policy Generation with Smart Contract In PrivacyGuard, individual DOs can define their own usage policies for their private data in DO contract ($\mathbf{C_{DO}}$). The policies encoded usually includes the essential components for privacy context, such as *data type, data range, operation, cost, consumer, expiration*, etc. The operation, which specifies intended usage of the targeted data, can be an arbitrary attestable computer program. This paradigm grants DOs fine-grained control on the data usage policy and the opportunity to participate in the data market independently. However, it requires ample transaction processing capacity from the blockchain network that scales in the number of DOs. Alternatively, the DB-based paradigm uses DB as a trusted delegate for a large number of DOs. DB represents the DOs in the blockchain by curating a DB contract ($\mathbf{C_{DB}}$) that accepts data registries from DOs and advertising their data in bundles. The encoding of $\mathbf{C_{DO}}$ and $\mathbf{C_{DB}}$ will be elaborated in Section 4.

Stage 3: Data Utilization and Contract Execution DC invokes a $\mathbf{C_{DO}}$ (or $\mathbf{C_{DB}}$) for permission to use certain private data of the targeted DO(s) for a specific operation, and deposits payment onto the contract. If permission is granted on the blockchain, DC instructs CEE to load the enclave program for the contracted operation whose checksum is specified in the contract. Then both the DC and iDA (or DB) proceed to remote-attest the CEE enclave. This essentially allows the two parties to reach a "local consensus" on CEE's trustworthiness that enables the *off-chain execution* of the on-chain contract. When the attestations succeed, iDA (or DB) provisions data decryption keys to the CEE enclave to enable data operation within the enclave. When the operation finishes, the enclave releases the result in encrypted form and erases all the associated data and keying materials. To achieve a fair and atomic exchange that DC gets the decrypted result while DO(s) get the payment, we propose a commitment protocol for the two sides which ensures the atomic exchange only when they agree upon each other. The detailed design of the commitment protocol will be explained in Section 5.

3.3 Threat Model and Assumptions

We assume all entities act based on self-interest and may not follow the protocol. However, to maintain a reasonable scope for the paper, we assume DO will not provide meaningless or falsified data intentionally. It is possible to encode rules in smart contract to penalize DOs for abusing the system with bad data. Furthermore, we assume the security systems, i.e. the blockchain and TEE, are trustworthy and are free of vulnerability. Specifically, in the control plane, we assume the blockchain infrastructure is secure that adversaries do not control enough resources to disrupt distributed consensus. We also assume smart contract implementations are free of software vulnerability. In the data plane, we assume the TEE is up to date, and particularly, Intel SGX, is secure against malicious attack from the operating system. We recognize that TEE implementations are not always perfect, and previous work has demonstrated side channel information leakage on the SGX platform alone [45,50,56,58,46,52,57], preventing such attacks is an important but orthogonal task. We also assume that all data operations requested by DC have been ratified by trusted sources and a cryptographic checksum of the program binary is sufficient for PrivacyGuard to check the data operation integrity.

4 Data Market of User-Defined Usage with Blockchain

The intuition behind the data market is to enable fair and transparent data transactions between DO and DC. In PrivacyGuard, DOs advertise private data items available for knowledge extraction on blockchain smart contracts. DC shops for a desirable data set and contract for his analytics. To start the data transaction, DC invokes the data contract and deposits a payment. The sales of knowledge extraction rights on private data are fulfilled that DO obtains the payment while the DC obtains the knowledge. The data transaction is then recorded in the blockchain with transparency. To enable user-defined access and usage control, the data contract, needs to encode DO's data usage policy including how data can be used by which DC at what cost. Next we present the our data contract design in PrivacyGuard in a constructive manner.

4.1 Encoding Data Usage Policy with Smart Contract

Basic Data Usage Contract In the conventional data sharing scenario, the data access policy often includes attributes such as type of the data, range or repository of the data, DO and DC credentials. For example, we assume patient X with public key pair (pk_X,sk_X) has three types of medical data: radiology data, blood test data and mental record data. X is only willing to share his radiology data (with descriptor $pData$) with urology specialist S with public key pk_S. X can treat S as a DC and specify an access policy P in a *data access* contract: $\mathbf{C_{X(DA)}} = \{P = \{pData, pk_S\}, Sig_{sk_X}(P)\}$. This encoding, however, specifies only data access but no obligation of the DC once access is granted. The DC could share the data with other parties against the original intention of the DO. To enable fine-grained control on how data is used, obligations need to be attached to the policy. For instance, if X only wants S to run a certain operation *op* on the data, then X can encode a new *data usage* contract in the following form: $\mathbf{C_{X(DU)}} = \{P = \{pData, op, pk_S\}, Sig_{sk_X}(P)\}$.

Enabling Data Market Economy A key feature of PrivacyGuard is to encourage DOs to share private data for public welfare as well as financial rewards without concerning privacy leakage or data misuse. Building on top of the success of cryptocurrency, the blockchain smart contract platform allows DO and DC to transact on the usage of data with financial value attached. DO can specify a price tag $\$pr$ (in cryptocurrency) in the policy. To further ensure a fair exchange that DC gets the knowledge and DO gets the payment, certain control logic should be instated in the form of smart contract functions. We call these functions and other contract metadata the contextual information, denoted ctx. Back to the previous example, we now have $\mathbf{C_{X(DU)}} = \{P = \{pData, op, \$pr, pk_S\}, ctx, Sig_{sk_X}(P||ctx)\}$. In blockchain domain, the signature is conveniently fulfilled by X's signature in the contract creation transaction.

Transparent Tracking of Data Utilization For the system to provide transparent data utilization tracking and policy compliance auditing, each data transaction needs to be recorded in a tamper-resistant and non-repudiable manner. In PrivacyGuard, contract functions (part of ctx, invoked via blockchain transactions) are used to facilitate the recording of data utilization. Since the blockchain ledger is publicly managed via global consensus and unforgeable, contract function invocations in blockchain transactions can provide non-repudiable records on data utilization.

Algorithm 1: Data Owner's Smart Contract $\mathbf{C_{DO}}$ Pseudocode

```
Function Constructor()    // Contract creation by DO with a policy
    Parse $policy$ as $(dataset, price, operation, DCList, requestTimeout)$ ;
    $pDS \leftarrow policy.dataset$ ;
    $pPrice \leftarrow policy.price$ ;
    $pOP \leftarrow policy.operation$ ;
    $pDCL \leftarrow policy.DCList$ ;
    $pRTO \leftarrow policy.requestTimeout$ ;
    $R \leftarrow [\,]$   // Usage records ;
    $DO \leftarrow creator$ ;
Function Request($op, data, \$f$)    // Callable by DC
    if $op = pOP$ and $sender \in pDCL$ and $data \subset pDS$ and $f \geq pPrice$ then
        Create a record entry $R[idx]$ with index $idx$ for this new data transaction ;
        $R[idx].\{data, DC, reqTime\} \leftarrow \{data, sender, sys.time\}$ ;
        $R[idx].status \leftarrow$ WAIT_COMPUTATION ;
    else
        Return $\$f$ to $sender$ and terminate ;
Function ComputationComplete(idx, $K_{result}Hash$)    // Callable by DC
    $R[idx].krHash \leftarrow K_{result}Hash$ ;
    $R[idx].status \leftarrow$ WAIT_COMPLETE ;
Function CompleteTransaction(idx, $K_{result}$)    // Callable by DO
    if Hash$(K_{result}) = R[idx].krHash$ then
        Send $\$f$ to $DO$ ;
        $R[idx].kr \leftarrow K_{result}$ ;
        $R[idx].status \leftarrow$ COMPLETE   // Data transaction complete ;
Function Cancel($idx$)    // Callable by DC
    if $sender = R[idx].DC$ and $(sys.time - R[idx].reqTime) > pRTO$ then
        Return $\$f$ to $R[idx].DC$; ;
        $R[idx].status =$ CANCELED ;
Function Revoke()    // Callable by DO
    if $sender = DO$ then
        contract selfdestruct ;
```

Data Owner's Smart Contract $\mathbf{C_{DO}}$ We design $\mathbf{C_{DO}}$ to capture the functionalities discussed above. The pseudo code of $\mathbf{C_{DO}}$ in shown in Algorithm 1. In addition to the policy variables, $\mathbf{C_{DO}}$ encodes functions for enforcing the control logic. *Constructor* initializes the policy at contract creation. *Request* takes a payment deposit from DC along with the requested operation *op*, the requested data descriptor D_{target}, and authorizes this data transaction. *ComputationComplete* is called by DC to signal the completion of the off-chain data execution. *CompleteTransaction* is called by DO to record the data usage and completes the transaction. The deposited payment is then redistributed to DO. We will cover more details on them along with the result commitment process in Section 5. *Cancel* is called by DC to abort the current transaction if the timeout passes. Lastly, *Revoke* invalidates the contract and can be called only by DO.

4.2 Using Data Broker to Address the On-Chain Scalability Challenge

While $\mathbf{C_{DO}}$ allows individual DOs to have fine-grained control over data usage policy and participate in data market independently, this paradigm puts heavy pressure on the blockchain transaction processing capability when the number of DOs is huge. In the meantime limited transaction throughput is a known problem for major public blockchains [7,12,20]. While there are many ongoing efforts to scale up transaction throughput [36,1], we take a different but complementary approach to address this issue in PrivacyGuard's scenario. A trusted delegate, namely data broker (DB), is used to represent a group of users and curates a DB's contract ($\mathbf{C_{DB}}$). $\mathbf{C_{DB}}$ allows individual DOs to register data entries and operations for DB to moderate. DB then participates in the data market on behalf of the registered DOs. We call this paradigm the DB-based system in our later implementation, in contrast to the iDA-based system.

The pseudo code of $\mathbf{C_{DB}}$ is provided in Appendix A. $\mathbf{C_{DB}}$ emulates $\mathbf{C_{DO}}$ for most parts but with extra global variables for data source management and two more functions: *Register* and *Confirm*. When a DO wants to make use of the DB, she first invokes *Register* function to register her data with the $\mathbf{C_{DB}}$. In the data plane, the DO needs to remotely attest the DB to establish trust, then provisions the data keys to the DB enclave. This, however, is not the end of data registration, because the data source and quality still need to be verified by the DB. Once verified, DB invokes *Confirm* function to complete the data registration. Furthermore, result commitment is also slightly different for $\mathbf{C_{DB}}$. The *CompleteTransaction* function is now callable by DB and needs to distribute payments to all involved DOs.

5 Off-Chain Contract Execution

PrivacyGuard leverages blockchain smart contract 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 data usage, smart contract suffers from several prohibiting drawbacks when it comes to confidential data computation purely on-chain. First, the smart contract invocation and the ensuing computation is executed and repeated by all nodes in the blockchain network. The cost to run complex algorithms on-chain can be prohibitive even assuming data storage is not an issue. Second, data has to be decrypted and stored on the chain, causing confidentiality problems.

To tackle this problem, we introduce the concept of *off-chain contract execution* in PrivacyGuard and introduce an entity called off-chain contract execution environment (CEE) to bring both the computation and data provisioning off-chain. Particularly, we decompose a data usage contract into two portions, the control part and the computation part. The control flow starts with invoking the contract and stops at the contracted computation task which switched to off-chain. The control flow is resumed with another contract invocation when the off-chain computation task is finished. Accordingly, we propose a novel off-chain contract execution and result commitment protocol, as is shown in Fig. 2. Note that both DB and iDA can represent a DO. Here we resort to the DB-based paradigm for convenience of presentation. We defer the discussion on DB's role in the data plane to the end of this section. Next we elaborate on the important features of off-chain contract execution in a constructive manner.

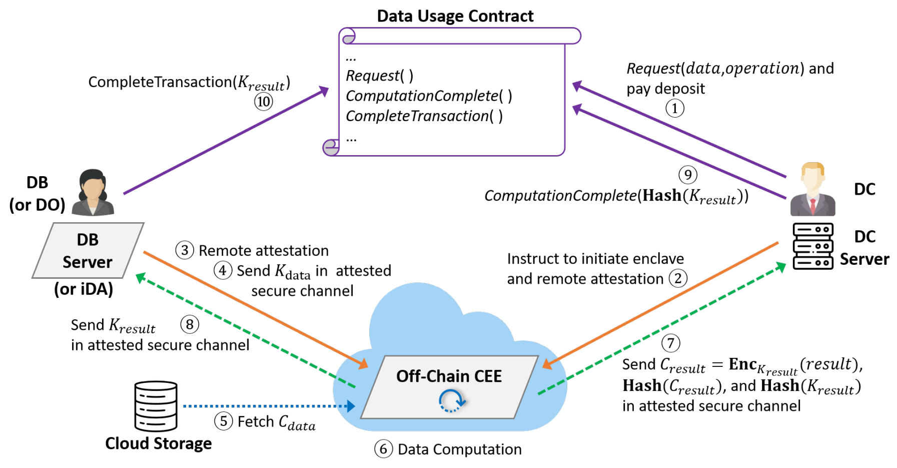

Data Usage Contract
...
Request( )
ComputationComplete( )
CompleteTransaction( )
...
CompleteTransaction(K_{result}) ⑩
Request(data,operation) and pay deposit ①
⑨ ComputationComplete(Hash(K_{result}))
DB (or DO)
DC
DB Server (or iDA)
DC Server
③ Remote attestation
④ Send K_{data} in attested secure channel
Instruct to initiate enclave and remote attestation ②
Send K_{result} ⑧ in attested secure channel
Off-Chain CEE
⑦ Send $C_{result} = \mathbf{Enc}_{K_{result}}(result)$, $\mathbf{Hash}(C_{result})$, and $\mathbf{Hash}(K_{result})$ in attested secure channel
Cloud Storage
⑤ Fetch C_{data}
⑥ Data Computation

Fig. 2. Off-chain Contract Execution and Result Commitment

5.1 Establishing Trust on the Execution of Contracted Operation through "Local Consensus"

The first challenge is the correct execution of the contracted task. As we have mentioned, when smart contracts are executed on blockchain platform, the correctness of the execution is guaranteed by the entire network through global consensus, which suffers from high on-chain cost. Our observation is that the correctness of one particular computation instance only matters to the stakeholders of the data transaction, i.e. the DOs, DB, and DC. And we do not need the entire network to verify the correctness.

In the conventional setting of distributed computing, both the DC and DO would perform the data computation task and expect the same result from each other. However, it contradicts DO's goal of fine-grained control on data usage if the data are directly provided to DC. Instead, we rely on software remote attestation, which is a widely available primitive with TEEs [30,2], for securely delegating the computation task to CEE. In this paper we opt for Intel SGX [30]. First of all, the designated computation program should pre-ratified with its program (binary) hash published in the data contract along side "authorized operations". When instructed by DC for a specific computation task, CEE loads the corresponding enclave program for that task. Then the two transacting sides in the data plane, DB and DC, remotely attest the enclave program to verify its authenticity and integrity with the program hash in the contract. As a result, as shown in Fig. 2, the immediate steps after data transaction request is to have CEE load the enclave program and DC and DB remotely attest the CEE enclave. Once correctly CEE enclave is verified with attestation reports, both sides of the contract can then extend their trust to CEE, knowing the attested program will execute securely in the enclave till termination, and the computation result will be genuine even if an adversary compromises CEE's untrusted platform (i.e., "normal world" in TEE terminology, which includes the operating system and non-enclave programs). And finally the result produced by CEE will be the "local consensus" between the two sides.

5.2 Enforcing Data Obligation and Confidentiality

The local consensus mechanism guarantees the data intensive computation task can be offloaded to the off-chain entity CEE for execution 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 DC for utilizing DO's data. More specifically, it follows the general requirement of secure computation, wherein only the computation result is accessible by the DC, not the plaintext source data. First, the computation process should not output any plaintext 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 TEE enclaves to create the environment for confidential computing. As is illustrated by step 3 and 4 in Fig. 2, DO's data en/decryption key K_{data} can be provisioned to CEE's enclave only if the latter can be cryptographically verified via remote attestation and a secure channel is established. This comes as an integral part of the local consensus. The hardware of CEE, the processor specifically, enforces the isolation between the untrusted platform and the enclave. We require the enclave program to include steps to sanitize intermediate results and keying materials. Since memory contents are encrypted in Intel SGX, once the keying material is removed, the data can be considered effectively sanitized. This also ensures that the program inside the enclave will terminate once the contracted task is completed.

5.3 Ensuring Atomicity in Contract Execution and Result Commitment

The last challenge is ensuring the atomicity of the contract, which arises from the split of control between on-chain off-chain. 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 interrupt the computation and cause delays. Specifically, two issues need to be addressed.

The first issue is the contract function runtime. When the adversary has control of the off-chain computation platform of CEE, he can pause or delay the computation. For many data computations, the result can be time-sensitive. To tackle this problem, we add a timeout mechanism in the data contract to allow DC to cancel the request after timeout and have the deposit refunded (see Algorithm 1).

The second issue is the atomic completion of the contract. We want both the DOs to get the payment in the control plane while allowing DC to get the computation results in the data plane. This is particularly challenging due to the lack of availability guarantee on the CEE platform. When the platform is compromised, the adversary can intercept and modify any external I/O from the enclave, including both the network and storage. Our design for the atomic completion and result commitment can be observed from step 7 to 10 in Fig. 2. The key idea is that result release and contract completion should be done as a single message in the control plane. To prevent DC from getting the result without completing the payment to DOs, the result are encrypted into C_{result}

with a random result key K_{result}, before being sent to DC in the attested secure channel. Since the platform can corrupt any output from CEE, the CEE enclave also sends DC the hash of the encrypted result and key, i.e., $\mathbf{Hash}(C_{result})$ and $\mathbf{Hash}(K_{result})$, which will be later used by DC for integrity check on the result and the key. K_{result} is passed to DB in the attested secure channel. To prevent DB from completing the transaction without releasing the correct result key, DC needs to initiate the commitment procedure in the control plane by invoking the contract function *ComputationComplete* with $\mathbf{Hash}(K_{result})$, indicating it has the encrypted result and is ready to finish the data transaction if and only if the correct result key K_{result} is released. Upon observing the message from DC, DB then invokes the smart contract function *CompleteTransaction* with the result key K_{result}. Only when the hash of K_{result} matches the previously received $\mathbf{Hash}(K_{result})$, will the contract write the data usage into records, release the payment to DOs, and finally conclude the data transaction. Note that our commitment protocol design does not need to protect the confidentiality of K_{result} (thus enabling the on-chain hash check). This is because the encrypted result C_{result} is passed directly from CEE enclave to DC via the attested secure channel. Finally, DC has the full discretion in deciding whether to publish the computation result afterwards.

5.4 Data Broker for Scalability in the Data Plane

In the iDA-based paradigm, when DC needs to use the data from a large number of DOs, the naive use of remote attestation on the CEE would require each iDA to individually attest and verify the CEE enclave, resulting in linearly growing computation overhead and network traffic. To address this challenge, in the DB-based paradigm, DB can be re-purposed as a trusted intermediary between the CEE and all relevant DOs in the data plane during the preparation stage, similar to its control plane role. Essentially, DB is also deployed on a TEE-enabled machine and instantiates an enclave for secure handling of DOs' data. The enclave is attested to every new DO only once after the DO registers with DB. During the normal operation, DB attests CEE on behalf of all relevant DOs for each DC request, saving the need for individual DOs to attest CEE. To accommodate the extreme case when a large number of DOs registers with DB simultaneously, we will explore parallel remote attestation solutions in Subsection 6.1.

6 Implementation and Evaluation

We implemented a prototype of PrivacyGuard using Intel SGX as the TEE technology and Ethereum as the smart contract platform. Source code with documentation is available at `https://github.com/yang-sec/PrivacyGuard`. The on-chain components, namely the DO contract and the DB contract, were implemented in Solidity with 144 and 162 software lines of code (SLOC) respectively. The data usage price was set at 0.01 ethers per user data. The off-chain components include five PrivacyGuard applications, namely *iDataAgent (iDA)*, *Data Broker (DB)*, *Data Owner (DO)*, *Data Consumer (DC)*, *Contract Execution Environment (CEE)*. They were implemented in C++ with Intel SGX SDK v2.3.1 on top of Ubuntu 16.04 LTS. The total SLOC for off-chain components is about 37,000.

We deployed the contracts onto Ethereum Rinkeby testnet for evaluation, though our system is fully compatible with Ethereum mainnet. We used a fixed gas price of 10^{-9} ethers. PrivacyGuard applications were deployed in a LAN scenario. 1 DB, 1 iDataAgent, and 1 CEE ran on a SGX-enabled Linux machine with Intel Core i5-7260U CPU (2 cores 4 threads, 3.5 GHz). Up to 160 DOs and 1 DC ran on a Linux machine with AMD FX-8320 CPU (4 cores 8 threads, 3.5 GHz). We note that this setup aims for feasibility demonstration; in real-world deployment each application will most likely reside in a different machine. We used the *adult* dataset from UCI Machine Learning Repository [15] to simulate the data source. Each DO randomly drew 500 data points from the dataset as its private data. We have tested the entire PrivacyGuard workflow in multiple runs and the data usage history has been recorded in the deployed contracts. Our evaluation focuses on the system's scalability and consists of three parts: control plane runtimes, control plane costs, and data plane runtimes.

6.1 Control Plane Runtimes

To accommodate the scenario where N DOs simultaneously attest the DB enclave in the DB-based system, we experimented with a parallel attestation scheme in DB that each of the N attestation instances is handled by one of the T software threads, which invokes a new attestation context of the enclave and a dedicated enclave thread control structure (TCS) (thus TCSNUM = T). The experiment was repeated under different T. To avoid congesting the Intel Attestation Service (IAS) which may violate the terms of service, we instead used a simulated IAS that responds to EPID signature revocation list request and attestation report request with 0.1s and 0.5s delays respectively. The result is shown in Fig. 3(a). We observe that the parallel scheme is indeed a promising solution for scaling up attestation capacity, at the cost of enlarged enclave memory footprint. When $N = 160$, it takes the 64-thread DB about a tenth the attestation time of its sequential counterpart. We remark that efficient and scalable remote attestation is an interesting standalone topic to explore in future work.

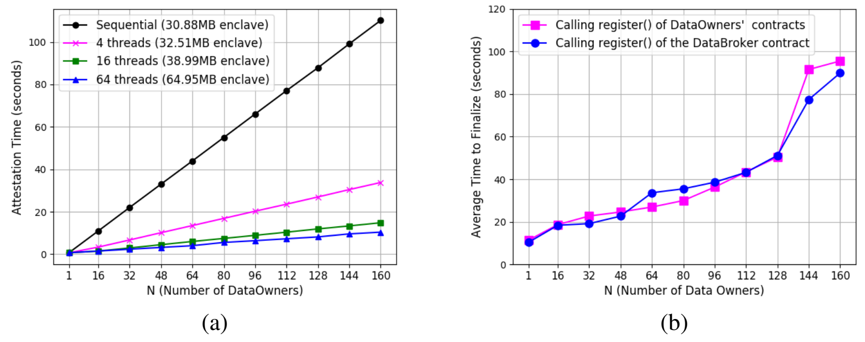


Fig. 3. (a) Attestation times of DB when N DOs simultaneously initiate attestation. (b) Average transaction finalization delay when N DOs simultaneously call a contract function.

To further evaluate the performance constraints imposed by the blockchain network, we measured the average transaction finalization delay in a congested environment. We set up 160 DOs to simultaneously send out a transaction calling the *Register()* function in the DB contract and their own DO contracts. we use *receipt* as the finalization response of the Ethereum transaction that makes the function call. The result is shown in Fig. 3(b). As more DOs send transactions at the same time, the average time to finalize a transaction increases dramatically. A straightforward workaround is to require DOs to call *Register()* according to a time schedule that minimizes congestion.

6.2 Control Plane Cost

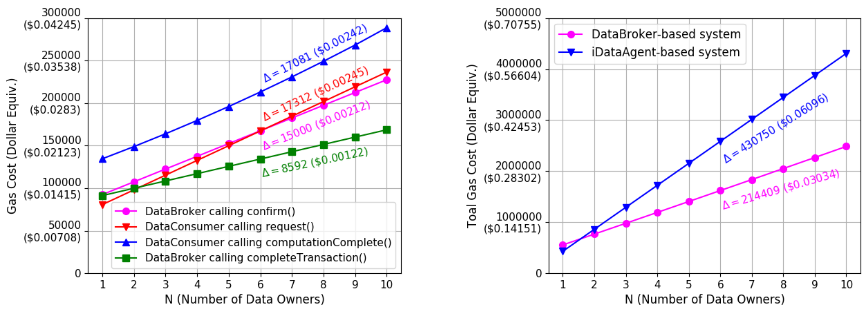
The monetary cost of the control plane mainly comes from the gas cost of operating smart contracts in Ethereum. At the beginning, every DO registers its data items on its own DO contract and the DB contract. DB fetches data from whoever registered with its contract and routinely confirms new registries. DC then requests for the data items from N DOs by sending a request transaction to the DB contract (or separate requests to all related DO contracts) with a sufficient deposit to cover the price before proceeding to attesting CEE. 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 ether price on 03/31/2019, which was $141.51 (source: `https://coinmarketcap.com/`).

Table 1. Cost of the data contract's scale-independent functions

	DO Contract		DB Contract	
Function	Gas Cost	USD Equiv.	Gas Cost	USD Equiv.
constructor()	951747	0.13468	846794	0.11983
Register() (new)	156414	0.02213	125392	0.01774
Register() (update)	30121	0.00426	45177	0.00639
Cancel()	81998	0.01160	66954	0.00947

We find that in both DB and DO contracts the costs of calling *constructor()* (contract creation), *Register()* and *Cancel()* do not depend on the number of registered DOs. We call these type of function calls scale-independent; otherwise scale-dependent. The costs of calling scale-independent functions and scale-dependent functions are shown in Table 1 and Fig. 4(a). Notably, the costs of calling *Request()* and *ComputationComplete()* grow faster than the costs of calling *Confirm()* and *CompleteTransaction()*. This implies the total cost will increasingly shift to the DC side, which is a scalable trend for the system as the DC has incentives to pay for more data usage.

To evaluate the scalability gain brought by DB, we compare the case wherein individual DOs share data via their own DO 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 summed the costs of all function calls except for the contract creation (calling *constructor()*) and extrapolated over different N. Fig. 4(b) shows that it costs the DB-based system much less to accommodate one extra DO ($0.0304)



(a) (b)

Fig. 4. (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.

compared to the iDataAgent-based system ($0.06096). This result together with control plane runtimes (Fig. 3(a)) demonstrate DB's ability to provide PrivacyGuard with financial and performance scalability when facing a growing number of DOs.

6.3 Data Plane Runtimes

To evaluate CEE's performance in off-chain contract execution, we experimented with a demonstrative, reasonably complex computation task: training four parallel instances of a neural network classifier. Detailed hyperparameters can be found in our source code. The training functions were ported to the SGX enclave from the Fast Artificial Neural Network (FANN) Library (`https://github.com/libfann/fann`). To evaluate enclave overhead, we also implemented an untrusted version (executed outside enclave) of the computation task that ran on the same machine. We noticed that recent work showed Intel's Hyperthreading Technology (HTT) has flaws that may impair the security of SGX enclaves [45]. Therefore, we tested the computation task under different hardware options with respect to the usage of SGX enclave and HTT. The Intel CPU's TurboBoost feature was turned off to avoid unexpected performance gain.

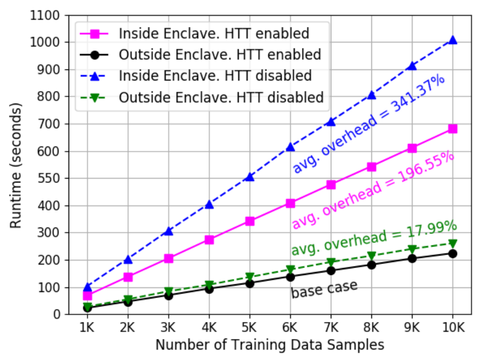


Fig. 5. Runtimes of training an example neural network classifier under four hardware options.

The experiment results is shown in Fig. 5. We find that the overhead caused by disabling HTT is 48.84% for inside enclave and 17.99% for outside-enclave. This indicates disabling HTT will drag down in-enclave performance more significantly. The overheads caused by enclave are 196.55% and 274.13% for HTT-enabled and HTT-disabled respectively. We speculate that the big enclave overhead is related to the enclave's secure function calls and our imperfect porting of the training program. We leave the performance caveats of Intel SGX and possible solutions to future work.

7 Related Work

Privacy Protection Privacy-preserving computation has been an active area of research in the past decade [25,37,34,8,47,55]. With the increasing reliance on rich data, there has been a significant amount of research on applying cryptographic techniques to perform privacy preserving computation and data access control [35,5,8,47,48,3]. Recently, hardware-assisted TEE has been adapted in numerous works to achieve privacy-preserving computation [34,25,37,59,24,22]. Specially, Ryoan [25] is closely related to PrivacyGuard. It combines native client sandbox and Intel SGX to confine data processing module and provide confidentiality. 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 with non-repudiable usage recording.
Blockchain and TEE The idea of moving computation off-chain to improve the performance and security is mentioned in [10,6,9,26,39,51]. Choudhuri et al. [10] combines blockchain with TEE to build one-time programs that resemble to smart contracts but only aim for a restricted functionality. Ekiden [9] and the Intel Private Data Object (PDO) project [6] are two concurrently developed projects that are closely related to our work. Similar to PrivacyGuard, Ekiden harmonizes trusted computing and distributed ledger to enable confidential contract execution. Ekiden offloads computation from consensus nodes to a collection computing nodes in the aim of improving the ecosystem. In comparison, PrivacyGuard is designed to fit existing blockchain infrastructure. The Intel PDO project aims to combine Intel SGX and distributed ledger to allow distrustful parties to work on the data in a confidential manner. However, the system focuses heavily on a permissioned model with significant overhead for bootstrapping trust.

8 Conclusion

In this paper, we proposed PrivacyGuard, a platform that combines blockchain smart contract and TEE to enable transparent enforcement of private data computation and fine-grained usage control. Blockchain can not only be used as a tamper-proof distributed ledger that records data usage, but also facilitate financial transactions to incentivize data sharing. To enable complex and confidential operations on private data, PrivacyGuard splits smart contract functionalities into control operations and data operations. Remote attestation and TEE are used to achieve local consensus of the contract participants on the trustworthiness of the off-chain contract execution environment. Atomicity of the contract completion and result release is facilitated by a commitment protocol. We implemented a prototype of PrivacyGuard platform and evaluated it in a simulated data market. The results show the reasonable control plane costs and feasibility of executing complex data operations in a confidential manner using the platform.

ACKNOWLEDGMENT

This work was supported in part by US National Science Foundation under grants CNS-1916902 and CNS-1916926.

A Data Broker Contract $\mathbf{C_{DB}}$

Algorithm 2: Data Broker's Smart Contract $\mathbf{C_{DB}}$ Pseudocode

```
Function Constructor()   // Contract creation by DB with config
    Parse $config$ as ($operationList, requestTimeout$) ;
    $cOPL \leftarrow config.operationList$ ;
    $cRTO \leftarrow config.requestTimeout$ ;
    $\{DO, DS, R\} \leftarrow \{[[\,]], [\,], [\,]\}$   // DOs, data sources, data usage records ;
    $DB \leftarrow creator$ ;
Function Register($op, DC, price$)   // Callable by a DO
    Create a DO entry $DO[ido, op]$ with index $ido$ for this new DO;
    $DO[ido, op].\{DO, DC, price\} \leftarrow \{sender, DC, price\}$ ;
Function Confirm($cfDOs$)   // Callable by DB
    for all $\{ido, op\}$ that $ido \in cfDOs$ and $op \in cOPL$ and $DO[ido, op] \neq null$ do
        Append $ido$ to $DS[op].DOList$ ;
        Append $DO[ido, op].DC$ to $DS[op].DCList$ ;
        $DS[op].price \leftarrow DS[op].price + DO[ido, op].price$ ;
Function Request($op, targetDOs, \$f$)   // Callable by DC
    if $op \in cOPL$ and $sender \in DS[op].DCList$ and $targetDOs \subset DS[op].DOList$ and
      $f \geq DS[op].price$ then
        Create a record entry $R[idx]$ with index $idx$ for this new data transaction ;
        $R[idx].\{targetDOs, DC, reqTime\} \leftarrow \{targetDOs, sender, sys.time\}$ ;
        $R[idx].status \leftarrow$ WAIT_COMPUTATION ;
    else
        Return $\$f$ to $sender$ and terminate ;
Function ComputationComplete($idx, K_{result}Hash$)
    (same as in $\mathbf{C_{DO}}$, see Algorithm 1)
Function CompleteTransaction($idx, K_{result}$)   // Callable by DB
    if Hash$(K_{result}) = R[idx].krHash$ then
        for all $ido \in DS[R[idx].op].DOList$ do
            Send $\$DO[ido, R[idx].op].price$ to $DO[ido].DO$;
        $R[idx].kr \leftarrow K_{result}$ ;
        $R[idx].status \leftarrow$ COMPLETE   // Data transaction complete ;
Function Cancel($idx$)
    (same as in $\mathbf{C_{DO}}$)
Function Revoke()
    (same as in $\mathbf{C_{DO}}$, except callable by DB)
```

References

1. brainbot technologies AG: Raiden network. https://https://raiden.network/

2. ARM: Security technology building a secure system using trustzone technology (2009)
3. Bacis, E., De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Rosa, M., Samarati, P.: Mix&slice: Efficient access revocation in the cloud. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 217–228 (2016)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07). pp. 321–334. IEEE (2007)
5. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011)
6. Bowman, M., Miele, A., Steiner, M., Vavala, B.: Private data objects: an overview (2018), https://arxiv.org/pdf/1807.05686.pdf
7. Buterin, V.: Privacy on blockchain. https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/
8. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on parallel and distributed systems **25**(1), 222–233 (2014)
9. Cheng, R., Zhang, F., Kos, J., He, W., Hynes, N., Johnson, N., Juels, A., Miller, A., Song, D.: Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 185–200. IEEE (2019)
10. Choudhuri, A.R., Green, M., Jain, A., Kaptchuk, G., Miers, I.: 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)
11. Costan, V., Lebedev, I.A., Devadas, S.: Sanctum: Minimal hardware extensions for strong software isolation. In: USENIX Security Symposium. pp. 857–874 (2016)
12. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., et al.: On scaling decentralized blockchains. In: International Conference on Financial Cryptography and Data Security. pp. 106–125. Springer (2016)
13. Custers, B., Uršič, H.: Big data and data reuse: a taxonomy of data reuse for balancing big data benefits and personal data protection. International Data Privacy Law **6**(1), 4–15 (2016)
14. Datta, A., Fredrikson, M., Ko, G., Mardziel, P., Sen, S.: 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)
15. Dheeru, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
16. Dwork, C.: Differential privacy: A survey of results. In: International conference on theory and applications of models of computation. pp. 1–19. Springer (2008)
17. Elnikety, E., Mehta, A., Vahldiek-Oberwagner, A., Garg, D., Druschel, P.: Thoth: Comprehensive policy compliance in data retrieval systems. In: USENIX Security Symposium. pp. 637–654 (2016)
18. Ethereum: Blockchain app platform. https://www.ethereum.org/
19. General data protection regulation (GDPR) (2016), https://eur-lex.europa.eu/eli/reg/2016/679/oj
20. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-NG: A scalable blockchain protocol. In: NSDI. pp. 45–59 (2016)
21. Facebookcambridge analytica data scandal. https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal
22. Fisch, B., Vinayagamurthy, D., Boneh, D., Gorbunov, S.: Iron: functional encryption using intel sgx. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 765–782. ACM (2017)

23. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 89–98 (2006)
24. Hunt, T., Song, C., Shokri, R., Shmatikov, V., Witchel, E.: Chiron: Privacy-preserving machine learning as a service (2018), https://arxiv.org/pdf/1803.05961.pdf
25. Hunt, T., Zhu, Z., Xu, Y., Peter, S., Witchel, E.: Ryoan: A distributed sandbox for untrusted computation on secret data. In: OSDI. pp. 533–549 (2016)
26. Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S.M., Felten, E.W.: Arbitrum: Scalable, private smart contracts. In: Proceedings of the 27th USENIX Conference on Security Symposium. pp. 1353–1370. USENIX Association (2018)
27. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: 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)
28. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd International Conference on Data Engineering. pp. 106–115. IEEE (2007)
29. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) **1**(1), 3–es (2007)
30. McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative instructions and software model for isolated execution. In: HASP@ ISCA. p. 10 (2013)
31. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
32. National Science and Technology Council: National privacy research strategy, https://www.nitrd.gov/PUBS/NationalPrivacyResearchStrategy.pdf
33. Nissenbaum, H.: Privacy as contextual integrity. Washington Law Review **79**, 119 (2004)
34. Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M.: Oblivious multi-party machine learning on trusted processors. In: USENIX Security Symposium. pp. 619–636 (2016)
35. Pass, R., Shi, E., Tramer, F.: Formal abstractions for attested execution secure processors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 260–289. Springer (2017)
36. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016), https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf
37. Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., Russinovich, M.: Vc3: Trustworthy data analytics in the cloud using sgx. In: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 38–54. IEEE (2015)
38. Sen, S., Guha, S., Datta, A., Rajamani, S.K., Tsai, J., Wing, J.M.: Bootstrapping privacy compliance in big data systems. In: Security and Privacy (SP), 2014 IEEE Symposium on. pp. 327–342. IEEE (2014)
39. Sinha, R., Gaddam, S., Kumaresan, R.: Luciditee: Policy-compliant fair computing at scale (2019), https://eprint.iacr.org/2019/178.pdf
40. Song, D., Lettner, J., Rajasekaran, P., Na, Y., Volckaert, S., Larsen, P., Franz, M.: Sok: sanitizing for security. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1275–1295. IEEE (2019)
41. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(05), 557–570 (2002)
42. Szabo, N.: Formalizing and securing relationships on public networks. First Monday **2**(9) (1997)

43. TED Talk: How tech companies deceive you into giving up your data and privacy. https://goo.gl/hSfaUX
44. Tim cook: Personal data collection is being 'weaponized against us with military efficiency'. https://goo.gl/BsWB3k
45. Van Bulck, J., Piessens, F., Strackx, R.: Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient out-of-order execution. In: 27th USENIX Security Symposium (USENIX Security 18) (2018)
46. Van Bulck, J., Weichbrodt, N., Kapitza, R., Piessens, F., Strackx, R.: 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)
47. Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. ACM Sigmod Record **33**(1), 50–57 (2004)
48. Vimercati, S.D.C.D., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM Transactions on Database Systems (TODS) **35**(2), 12 (2010)
49. Wang, G., Liu, Q., Wu, J.: 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. pp. 735–737 (2010)
50. Wang, W., Chen, G., Pan, X., Zhang, Y., Wang, X., Bindschaedler, V., Tang, H., Gunter, C.A.: 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)
51. Wüst, K., Matetic, S., Egli, S., Kostiainen, K., Capkun, S.: Ace: Asynchronous and concurrent execution of complex smart contracts. (2019), https://eprint.iacr.org/2019/835.pdf
52. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy. pp. 640–656. IEEE (2015)
53. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Infocom, 2010 proceedings IEEE. pp. 1–9. Ieee (2010)
54. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: 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)
55. Zhang, N., Li, J., Lou, W., Hou, Y.T.: Privacyguard: Enforcing private data usage with blockchain and attested execution. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology, pp. 345–353. Springer (2018)
56. Zhang, N., Sun, H., Sun, K., Lou, W., Hou, Y.T.: Cachekit: Evading memory introspection using cache incoherence. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 337–352. IEEE (2016)
57. Zhang, N., Sun, K., Lou, W., Hou, Y.T.: Case: Cache-assisted secure execution on arm processors. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 72–90. IEEE (2016)
58. Zhang, N., Sun, K., Shands, D., Lou, W., Hou, Y.T.: Trusense: Information leakage from trustzone. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. pp. 1097–1105. IEEE (2018)
59. Zheng, W., Dave, A., Beekman, J.G., Popa, R.A., Gonzalez, J.E., Stoica, I.: Opaque: An oblivious and encrypted distributed analytics platform. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). pp. 283–298. USENIX Association, Boston, MA (2017)
60. Zyskind, G., Nathan, O., Pentland, A.: Enigma: Decentralized computation platform with guaranteed privacy (2015), https://arxiv.org/pdf/1506.03471.pdf

61. Zyskind, G., Nathan, O., Pentland, A.S.: Decentralizing privacy: Using blockchain to protect personal data. In: Security and Privacy Workshops (SPW). IEEE (2015)