Secure and Efficient Distributed Network Provenance for IoT: A Blockchain-Based Approach

Dongxiao Liu, *Student Member, IEEE*, Jianbing Ni, *Member, IEEE*, Cheng Huang, *Student Member, IEEE*, Xiaodong Lin, *Fellow, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

***Abstract*—Network provenance is essential for Internet-of-Things (IoT) network administrators to conduct the network diagnostics and identify root causes of network errors. However, the distributed nature of the IoT network results in the management of the provenance data at different trust domains, which poses concerns on the security and trustworthiness of the cross-domain network diagnostics. In this article, we propose a blockchain-based architecture for secure and efficient distributed network provenance (SEDNP) in the IoT. Instead of directly storing and querying the whole provenance data on the blockchain with prohibitive implementation cost, we introduce a unified provenance query model and develop a provenance digest strategy that: 1) enables compact (constant size) on-blockchain digests of provenance data and a multilevel index regardless of provenance data volume and 2) ensures the correctness and integrity of provenance query results through the verification of the on-blockchain digests. We formally define the security requirements as Archiving Security along with thorough security analysis. Moreover, we conduct extensive experiments with the integration of a verifiable computation (VC) framework and a blockchain testing network. The experimental results are provided as performance benchmarks to demonstrate the application feasibility of SEDNP.**

***Index Terms*—Blockchain, distributed network provenance, Internet of Things (IoT), trust.**

I. Introduction

INTERNET of Things (IoT) [1] is a network of smart devices with ubiquitous and seamless connections. In the IoT network, network provenance plays a critical role in IoT network diagnostics and forensics, by collecting, storing, and analyzing the network logging data, e.g., runtime network state changes and events [2]. Specifically, network provenance maintains the detailed history of the information exchange in the IoT network [3] to answer the essential questions for the IoT network administrators: what has happened and why it has happened. For example, unexpected network behaviors or errors, such as the packet misrouting or an IoT node failure in a remote area [4], can be detected and traced to their original causes or inputs (*root causes*) [5]. Thus, network provenance has been extensively studied and applied in a wide range of IoT applications, such as wireless sensor network [6], [7] and smart home network [8].

IoT network is operated in a distributed manner. In specific, the network is controlled by distributed protocols, e.g., border gateway protocol (BGP), and managed by the IoT network administrators of different trust domains [5]. In a distributed IoT network, a global provenance diagnostics could collect provenance data from different network domains for global root cause analysis and performance optimizations. For example, the global diagnostics for packet routing protocols assist the administrators to better manage their local routing tables and identify cross-domain faults [9]. However, ensuring *secure* and *efficient* distributed network provenance management for the IoT is a nontrivial task. First, a cross-domain network diagnostics requires a provenance model that supports flexible and efficient queries (such as range or keyword query [10]) for different use cases [3] in the IoT network. Second, network provenance data are distributed and maintained among different administrative domains in the IoT network since there is a lack of a trusted centralized database [11]. As a result, a cross-domain network diagnostics demands the query and access of the provenance data cross trust boundaries, which raises concerns on the security and trustworthiness of the distributed provenance storage. At the same time, it remains a very challenging issue to guarantee the correctness and integrity of provenance query results.

By viewing the provenance storage for the IoT network as a distributed database, the emerging blockchain technique [12] promises to resolve the security and trust challenges. Essentially, the blockchain is a distributed ledger maintained by peer-to-peer nodes, that consists of an increasing number of blocks of transactions. Empowered by the lightweight cryptography and the distributed consensus protocol, the blockchain guarantees storage immutability and the consistent shared view of the distributed database among peer nodes without any mutual trust. Therefore, it is tempting to build the distributed network provenance atop a blockchain architecture, such that IoT network administrators of different domains can agree on a single and trusted provenance storage and conduct network diagnostics in a trustworthy manner. Unfortunately, it may not be feasible to store all provenance data onto the blockchain and use a smart contract to query the on-blockchain storage [13]–[16] in the real-world

Manuscript received January 12, 2020; revised March 14, 2020; accepted April 1, 2020. Date of publication April 17, 2020; date of current version August 12, 2020. *(Corresponding author: Jianbing Ni.)*

Dongxiao Liu, Cheng Huang, and Xuemin (Sherman) Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: dongxiao.liu@uwaterloo.ca; cheng.huang@uwaterloo.ca; xshen@bbcr.uwaterloo.ca).

Jianbing Ni is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: jianbing.ni@queensu.ca).

Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: xlin08@uoguelph.ca).

Digital Object Identifier 10.1109/JIOT.2020.2988481

implementations. Specifically, the network provenance data demands a huge volume of the on-blockchain storage and will be frequently accessed and queried for the cross-domain network diagnostics. At the same time, on-blockchain storage and computation resources are prohibitively expensive, since transactions must reach to all blockchain miners to be verified and stored at each miner's local storage [17]–[19]. As a result, there exist the following technical challenges for building a blockchain-based architecture for the distributed network provenance: 1) the design of a unified model that empowers flexible and efficient provenance query modules; 2) the development of a provenance digest strategy to authenticate both the correctness and the integrity of the query results; and 3) the optimization of the on-blockchain digest complexity for efficiently operating over the blockchain architecture.

In this article, we propose the blockchain-based secure and efficient distributed network provenance (SEDNP) for the IoT, by leveraging zero-knowledge succinct noninteractive arguments of knowledge (zk-SNARK)-based verifiable computation (VC) framework with detailed designs for rich provenance query functionalities. SEDNP allows IoT network administrators to maintain their provenance data locally with compact on-blockchain digests of the provenance logging data and index, which can later be queried with correctness and integrity guarantees. In particular, the contributions of this article are as follows.

1) We propose a blockchain-based architecture for the network provenance in the IoT with the design of a multilevel provenance query index. The architecture unifies keyword, range, and *K*-hop ancestor query in a single model for efficient instantiations in the VC framework.
2) We develop a digest strategy that authenticates both the provenance log and index. We minimize the on-blockchain storage and computation overhead with constant-size digests regardless of data volume.
3) We formalize *Archiving Security* to capture the correctness and integrity requirements of the query results along with thorough security analysis. We explore the implementation challenges with a proof-of-concept system that integrates the Pinocchio VC framework with a testing blockchain network. We conduct extensive experiments to provide real-world performance benchmarks and demonstrate the application feasibility of SEDNP.

The organization of this article is summarized as follows. In Section II, we formulate the system model of SEDNP with security models and design goals. In Section III, we introduce the building blocks: zk-SNARK-based VC and the blockchain technique. In Section IV, we propose SEDNP. Security analysis is presented in Section V and the performance evaluation is presented in Section VI. We summarize related work in Section VII. Finally, we conclude this article in Section VIII.

II. Problem Formulation

In this section, we review the graph-based model of network provenance for the IoT and formulate the system model of SEDNP with security models and design goals.

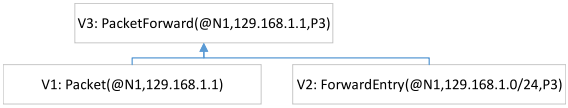


Fig. 1. Packet routing provenance.

TABLE I
NOTATIONS

$\mathbb{G}$	Multiplicative groups
$\mathcal{A}_I$, $\mathcal{A}_S$	Investigating administrator $\mathcal{A}_I$ Source administrator $\mathcal{A}_S$
$G = (V, E)$	Provenance local graph G
v_i	Provenance vertex $v_i \in V$
λ	System security parameter
$W = \{a_i\}_{i\in[1,n]}$	Attribute dictionary W Vertex attribute a_i
Q	Provenance query Q
$I = (I_F, I_S)$	Provenance query index I First level I_F, second level I_S
I_{k_i}	Keyword index $I_{k_i} \in I_F$
I_{v_i}	Event index $I_{v_i} \in I_S$
$n = n_1 + n_2$	Provenance index dimension n Range value dimension n_1 Keyword dimension n_2
$T_i = (vid_i, I_{v_i}, c_i)$	Tuple storage T_i for v_i ID vid_i, index I_{v_i}, log c_i
$R = (R_I, R_L)$	Provenance query result R Index result R_I, log result R_L
$\mathcal{F}(I, Q) \to R_I$	Index query function $\mathcal{F}$ Inputs I, Q and an output R_I
$\pi = (\pi_I, \pi_L, \pi_D)$	Provenance query proof π Index proof π_I, log proof π_L digest proof π_D
$CRS = (ek, vk)$	Common Reference String CRS Evaluation ek, verification vk
$\mathcal{D} = (\mathcal{D}_I, \mathcal{D}_L)$	Provenance digest $\mathcal{D}$ Index digest $\mathcal{D}_I$, log digest $\mathcal{D}_L$
$DK = (DK_E, DK_V)$	Digest key DK Evaluation DK_E, verification DK_V

A. Network Provenance Model

Based on the observations of network events, a directed graph $G = (V, E)$ is derived using the network dialog (NDlog) model [9], [11], which records the provenance relations of observed network events. In specific, each vertex $v_i \in V$ denotes an event (IoT node state or state change). An edge $e = (v_1, v_2) \in E$ indicates v_1 is the provenance (cause) of the event v_2. In Fig. 1, a simple provenance graph of packet routing consists of three vertexes. A packet with destination 129.168.1.1 is denoted as a packet arrival event $V1$. It is forwarded to port 3 at router N1 according to the packet forwarding entry $V2$. As a result, a packet forwarding instance is generated as $V3$. Notations are shown in Table I.

B. System Model

The system model of SEDNP is shown in Fig. 2, where N IoT network administrators $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$ from different trust domains agree on the same provenance graph

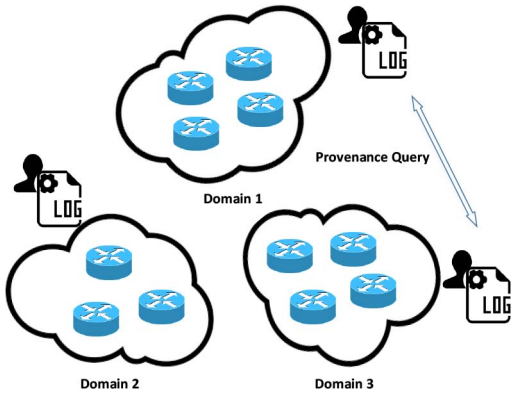


Fig. 2. Distributed network provenance.

model (i.e., NDlog). Each IoT network administrator maintains his/her local provenance graph $G = (V, E)$. It is worth noting that different administrators maintain different local provenance graphs based on their own observations of network events. Each vertex $v_i \in V$ is stored as a tuple $T_i = (vid_i, I_{v_i}, c_i)$. vid_i is the unique ID of vertex v_i. I_{v_i} is the event index, that consists the main attributes of the event (such as event types, location, and time). c_i denotes the detailed event log with the detailed descriptions of the event. For the local graph G, each administrator constructs a whole provenance query index I for all $v_i \in V$ and stores a succinct digest $\mathcal{D}$ of both the index and logs onto the blockchain. For a cross-domain network diagnostics instance, an investigating administrator $\mathcal{A}_I$ queries the provenance graph of a source administrator (denoted as $\mathcal{A}_S$) with a query Q. It is worth noting that any $\mathcal{A}_i \in \mathcal{A}$ can be an investigating administrator. $\mathcal{A}_S$ answers the query with vertex tuples $(vid_i, c_i) \in G$ that satisfies the query Q and a proof π. Finally, $\mathcal{A}_I$ collects provenance query results from all source administrators, verifies the correctness of the proofs and on-blockchain digests and reconstructs the whole provenance subgraph for the diagnostics. Formally, we define SEDNP as follows.

Definition 1: SEDNP consists of four algorithms

$$\text{SEDNP} = \{Setup,\ ProvCon,\ Query,\ ProvVer\}.$$

1) *Setup($1^\lambda, \mathcal{F}$):* The algorithm takes into the security parameter λ and an index query function $\mathcal{F}$. It outputs the system public parameter pp, common reference string CRS, and a digest key DK.
2) *ProvCon(G, pp, CRS, DK):* The algorithm takes into the provenance graph G, pp, CRS, and DK. It outputs a provenance query index $\mathcal{I}$ and a provenance digest $\mathcal{D}$.
3) *Query($Q, G, pp, CRS, DK, \mathcal{I}$):* The algorithm takes into a query Q, G, pp, CRS, DK, and $\mathcal{I}$. It outputs query result R and a correctness proof π.
4) *ProvVer($Q, R, \pi, CRS, DK, \mathcal{D}$):* The algorithm takes into Q, R, π, CRS, DK, and $\mathcal{D}$. It outputs either $\perp$ if the returned result does not meet the query; otherwise, it outputs the provenance graph G_Q for the query Q.

Note that the query result $R = (R_I, R_L)$, consisting of an index query result R_I and log query result R_L. R_I is the execution result of the index query function $\mathcal{F}(I, Q)$. Based on the provenance query model [5], we introduce the microquery modules for SEDNP.

Definition 2: Given a query Q over provenance index I, the query algorithm in SEDNP consists of three microquery modules.

1) *Keyword Query:* Given a keyword k_i, it returns vertexes $v_j \in G$ that k_i exists in I_{v_j}.
2) *Range Query:* Given a numerical range $(r_l, r_r)_j$, it returns vertexes $v_i \in G$ with the numeric value $r_j \in I_{v_i}$ that lies in the range (r_l, r_r).
3) *K-Hop Ancestor Query:* Given vid_j, it returns tuple IDs of its ancestors that are K-hop away.

We unify different provenance query modules in a single model for efficient instantiations in the VC framework to reduce the initialization cost. Complex provenance queries can be instantiated by the combinations of the microquery modules. We use the routing management as an example in Fig. 1. A provenance query can be represented as $Q = (N1, PacketForward, 129.168.0.0/16)$. This query combines two query modules *keyword* and *range*. The query returns the instance $V3$ with keyword "PacketForward" and "N1" and a destination IP range 129.168.0.0/16.

C. Security Model

We assume an individual IoT network administrator to be *rational*. That is, IoT network administrators honestly observe and archive their local provenance data. However, they are motivated to later tamper with or withhold *archived* provenance data that are unfavorable for them in a cross-domain network diagnostics. Since global provenance diagnostics for the IoT requires post-event analysis and usually finds cross-domain network misconfigurations, it is hard for administrators to determine at runtime if a network event will be a provenance cause of a future diagnostics. We define *Archiving Security* in Definition 3, that ensures any provenance query is correctly executed over the archived provenance data.

Definition 3: Following Definition 1, Archiving Security consists of two properties given as follows.

1) *Correctness:* An honest $\mathcal{A}_I$ will always accept (R, π) iff a) Setup, ProvCon, and Query functions are correctly executed and b) $\mathcal{D}$ is the digest of I, G.
2) *Integrity:* a) once the provenance data are archived, it cannot be later modified; b) if $vid_i \in R$, $vid_i \in G$ and I_{v_i} satisfies the query Q; c) there is no $vid_i \in G$, such that I_{v_i} satisfies the query Q and $vid_i \notin R$; and d) if log $c_i \in R$, $c_i \in G$.

D. Design Goals

1) *Functionality:* SEDNP should support all microquery modules in terms of keyword query, range query, and K-hop ancestor query in a unified model.
2) *Security:* SEDNP should achieve *Archiving Security* in terms of correctness and integrity.
3) *Efficiency:* SEDNP should be efficient for real-world implementations in terms of communication and computation overhead.

III. Building Blocks

A. Blockchain and Smart Contract

A public blockchain is a shared distributed ledger maintained by peer nodes. It consists of an increasing number of blocks of transactions [12], [19], and is secured by the cryptography (digital signatures and hash functions). Empowered by the consensus protocols, peer nodes are motivated to maintain the correctness and consistent view of the shared ledger. The public blockchain has the following properties.

1) *Transparency:* Transactions and block generations are open to public and verifiable.
2) *Tamperproof:* Once a block is approved and appended to the ledger, it cannot be deleted or modified.
3) *Decentralization:* The ledger is maintained by distributed peer nodes without any mutual trust.

A smart contract is a computer program executed over the public ledger by all peer nodes. A smart contract specifies terms and actions for involved parties and enforces obligations when terms are met. In particular, a smart contract can read/write blockchain storage and transfer cryptocurrencies to blockchain accounts.

B. Cryptographic Background and Assumptions

We define a security parameter λ and multiplicative groups [20], [21] $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ with a prime order p and an asymmetric bilinear paring $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ [22]. We define a collision-resistant hash function Hash : $\{0,1\}^* \to \{0,1\}^{256}$. We also define a set of cryptographic assumptions: q-power Diffie–Hellman (q-PDH), q-power knowledge of exponent (q-PKE), q-strong Diffie–Hellman (q-SDH), and strong external Diffie–Hellman (SXDH) assumptions, where $q = \text{poly}(\lambda)$. The details of the assumptions are in [23].

C. Verifiable Computation Framework

A Pinocchio [23] VC framework enables a party (either public or delegated) to verify an index function $\mathcal{F}$ is correctly executed with inputs Q and I. Formally, $\mathcal{F}$ is defined as follows:

$$\mathcal{F}(Q, I) = R_I. \tag{1}$$

It takes inputs Q and I, and outputs result R_I. $\mathcal{F}$ is equivalent to an NP-complete relation $\mathcal{R}$. The function is correctly computed iff $\mathcal{R}(Q, I, R_I) = 1$. Briefly speaking, $\mathcal{F}$ is first translated to a quadratic arithmetic program (QAP). Combined with the zero-knowledge succinct arguments (ZK-SNARK) [24], the evaluation of $\mathcal{R}(Q, I, R_I) = 1$ is achieved in a verifier-efficient manner. In the following, we present black-box definitions of the Pinocchio VC framework.

Definition 4: A Pinocchio [23] VC framework consists of three algorithms

$$vc = \{KeyGen,\ Eval,\ Verify\}.$$

1) *KeyGen(pp, $\mathcal{F}$):* The algorithm takes into the public parameters *pp* and an index function $\mathcal{F}$. It outputs common reference strings $CRS = (ek, vk)$, including an evaluation key *ek* and a verification key *vk*.
2) *Eval(ek, Q, I):* The algorithm takes into I, Q, and *ek*. It evaluates the function $\mathcal{F}$ with I, Q as inputs and outputs result R_I with a correctness proof π_I.
3) *Verify(vk, Q, R_I, π_I):* The algorithm takes into *vk*, Q, R_I, and π_I. It outputs true if $\mathcal{R}(Q, I, R_I) = 1$; otherwise, it outputs false.

In the network provenance, the provenance query function is instantiated with the combination of index query function $\mathcal{F}$ and a consistency check function with the on-chain digests. Index query execution is implemented as a computing program of $\mathcal{F}$ that is embedded in the *CRS*. The detailed designs of the function $\mathcal{F}$ are discussed in Section IV.

IV. Secure and Efficient Distributed Network Provenance

Following Definition 1, we present the detailed designs of SEDNP with the following phases.

1) A trusted authority (*TA*) runs Setup to generate the public parameters *pp*, *CRS*, and *DK*. It is worth noting that the role of *TA* can be replaced with a multiparty computation protocol [25] among IoT network administrators.
2) Each administrator $\mathcal{A}$ maintains a local provenance graph $G = (V, E)$. $\mathcal{A}$ runs ProvCon to generate a multilevel query index I and a provenance digest $\mathcal{D}$. $\mathcal{A}$ outsources $\mathcal{D}$ onto the blockchain.
3) An investigating administrator $\mathcal{A}_I$ queries provenance graphs from a source administrator $\mathcal{A}_S$ with a query Q. $\mathcal{A}_S$ runs the Query function over their local provenance graph and outputs the query result R along with a correctness proof π.
4) $\mathcal{A}_I$ runs the ProvVer function to verify the correctness and integrity of the query result from $\mathcal{A}_S$ and checks the consistency with the on-blockchain digest. $\mathcal{A}_I$ accepts the result if the verification passes.

A. Setup

TA selects a security parameter λ and outputs a multiplicative group $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$. *TA* randomly chooses $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. *TA* defines an n-dimensional index dictionary $W = (a_1, a_2, \ldots, a_n)$, which is further divided into two sets: $(\{a_x\}_{x\in[1,n_1]}, \{a_y\}_{y\in[n_1+1,n]})$. $\{a_x\}_{x\in[1,n_1]}$ denotes a numeric attribute, such as IP address or port number. $\{a_y\}_{y\in[n_1+1,n]}$ indicates a keyword attribute, such as protocol type. Considering local provenance graph $G = (V, E)$, *TA* defines the form of provenance index $I = (I_F, I_S)$

$$\begin{aligned} &I_F = \left\{I_{k_i}\right\}_{i\in[1,n_2]},\ I_S = \left\{I_{v_i}\right\}_{i\in[1,m]} \\ &n_2 = n - n_1,\ m = |V|. \end{aligned} \tag{2}$$

The detailed settings and calculations of $\mathcal{F}$ will be presented later. We note that the input/output size of $\mathcal{F}$ is fixed. TA chooses a collision-resistant hash function *Hash*, such as SHA-256 and denotes public parameters $pp = (\mathbb{G}, g_1, g_2, W, Hash)$.

TA runs the KeyGen($pp, \mathcal{F}$) algorithm of the VC framework to generate *CRS*, including (ek, vk). We note that there exists

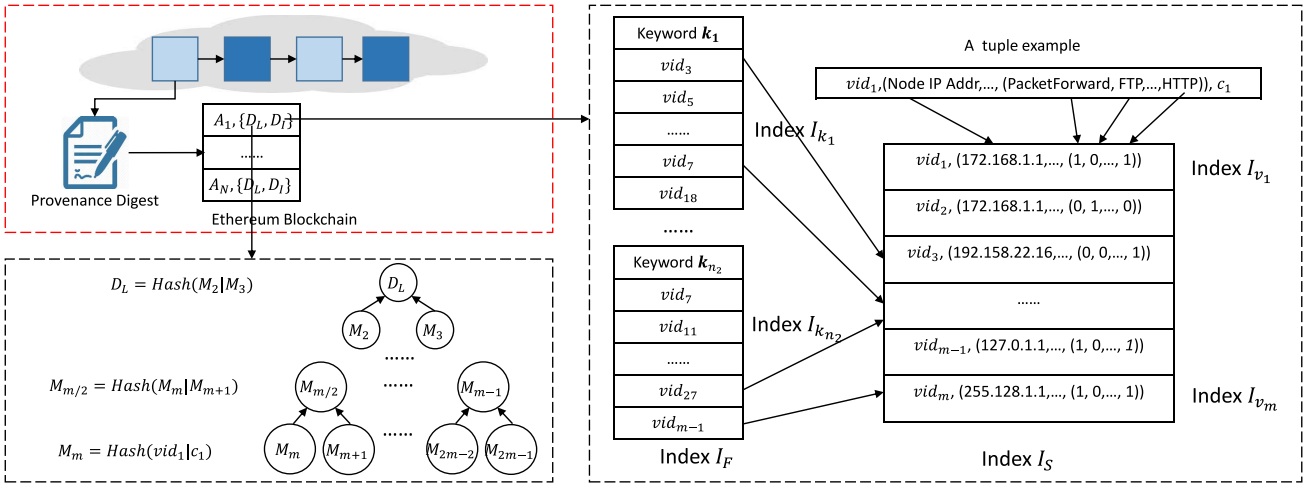


Fig. 3. Example of digest construction.

a set of generators of $\mathbb{G}_1$ in ek

$$S_I = \{\bar{g}_i\}_{i\in[1,m*(n+n_2)]}. \tag{3}$$

Each $\bar{g}_i$ corresponds to one input variable in I. TA randomly chooses $P = \{P_i\}_{i\in[1,m*(n+n_2)]}$ and $X = \{X_i\}_{i\in[1,m*(n+n_2)]}$ from $\mathbb{G}_1$ and $\alpha, \beta, \gamma \in \mathbb{Z}_p$ to compute

$$\hat{E} = g_2^{\alpha}, \quad \hat{F} = g_2^{\beta}, \quad \hat{G} = g_2^{\gamma}, \quad Y = \{Y_i\}_{i\in[1,m*(n+n_2)]}$$
$$Y_i = P_i^{\alpha} X_i^{\beta} \bar{g}_i^{\gamma}, \quad i \in [1, m*(n+n_2)]. \tag{4}$$

TA denotes DK as

$$(DK_E, DK_V) = \left((P, S_I, X, Y), \left(\hat{E}, \hat{F}, \hat{G}\right)\right). \tag{5}$$

Finally, TA publishes public parameters $\{\mathcal{F}, pp, CRS, DK\}$.

Algorithm 1: Index Construction

Input: $\{I_{v_i}\}_{v_i\in V}$
Output: $I_F = \{I_{k_j}\}_{j\in[1,n_2]}$ and $I_S = \{I_{v_i}\}_{i\in[1,m]}$
for $v_i \in V$ **do**
  for $j = 1$ **to** n_2 **do**
    if $k_j \neq 0 \in I_{v_i}$ **then**
      Add vid_i to I_{k_j}
for $j = 1$ **to** n_2 **do**
  Add I_{k_j} to I_F
for $i = 1$ **to** m **do**
  Add I_{v_i} to I_S

B. Provenance Index and Digest Construction

$\mathcal{A}$ first runs the NDlog derivation program over the provenance data to obtain a local provenance graph $G = \{V, E\}$, where $|V| = m$. Based on G, $\mathcal{A}$ constructs the provenance index I as follows.

Mapping to Vector Space: For each $v_i \in V$, $\mathcal{A}$ sets I_{v_i} based on the dictionary W [26] as follows:

$$I_{v_i} = \left(\{r_x\}_{x\in[1,n_1]}, \{k_y\}_{y\in[1,n_2]}\right). \tag{6}$$

r_x is the numeric value of attribute a_x. $k_y = 0$ if keyword a_{n_1+y} does not exist in v_i; otherwise, k_y is set to 1, as shown in Fig. 3.

Multilevel Index Strategy: $\mathcal{A}$ constructs a multilevel index I from event indexes I_{v_i}. I consists of two parts: 1) a keyword-inverted first-level index I_F and 2) a second-level index I_S. We present the bottom-up method to build I in Algorithm 1. For each keyword $k_j \in W$, $\mathcal{A}$ builds an index I_{k_j} that contains all the vertexes v_i, where $k_j \neq 0 \in I_{v_i}$. Then, $\mathcal{A}$ aggregates all the I_{k_j} in the I_F and all the I_{v_i} in the I_S.

K-Hop Ancestor: It is common that a provenance query requires graph traversal operations. However, it is observed that the verifiable graph traversal algorithm is not practical [27] in a general directed provenance graph. To enable efficient and verifiable graph traversal operations, we introduce additional data structures, i.e., K-hop ancestor index I_K. In specific, I_K is indexed by tuple ID vid_i. For each vid_i, I_K stores the vids of its ancestors that are K-hop away from vid_i. I_K is also instantiated in the vector space model and the detailed implementations change with different provenance use cases.

Digest and Upload: Based on the index I, $\mathcal{A}$ computes $\mathcal{D}_I = (\mathcal{D}_F, \mathcal{D}_S)$ using DK_E as follows:

$$\mathcal{D}_F = \prod_{i\in[1,n_2]} \prod_{j\in[1,m]} P_{(i-1)*m+j}^{\mu_{i,j}}$$
$$\mathcal{D}_S = \prod_{x\in[1,m]} \prod_{y\in[1,n]} P_{n_2*m+(x-1)*n+y}^{\omega_{x,y}}. \tag{7}$$

$\mathcal{D}_I$ is a compact multiexponent form that digests each item in the index I. For $\mathcal{D}_F$, $\mu_{i,j} = 0$ if $vid_j \notin I_{k_i}$; $\mu_{i,j}$ is set as a random number of $\mathbb{Z}_p$, if $vid_j \in I_{k_i}$. For $\mathcal{D}_S$, $\omega_{x,y} = r_y \in I_{v_x}$, if $y \in [1, n_1]$; $\omega_{x,y} = k_y \in I_{v_x}$ if $y \in [n_1 + 1, n]$. For each log $c_i \in G$, $\mathcal{A}$ computes a log digest $\mathcal{D}_L$-based Merkle proof technique [28] using Algorithm 2. An example of $\mathcal{D}_L$ construction is shown in Fig. 3. Finally, $\mathcal{A}$ uploads the digest $\mathcal{D}_I$ and $\mathcal{D}_L$ onto the blockchain.

C. Provenance Query

We assume secure and authenticated channels have been setup between IoT network administrators and thus omit the details of message or identity authentication mechanisms.

Algorithm 2: Merkle Digest Generation

Input: Provenance Logs $\{c_i\}_{v_i \in V}$
Output: Log Digest $\mathcal{D}_L$
for $i = m$ **to** $2m - 1$ **do**
  Set $M_i \leftarrow Hash(vid_{i-m+1}|c_{i-m+1})$
for $j = m - 1$ **to** 1 **do**
  Set $M_j \leftarrow Hash(M_{2j}|M_{2j+1})$
Set $\mathcal{D}_L \leftarrow M_1$

1) Query Construction: Any administrator can be an investigating administrator $\mathcal{A}_I$ to construct a query, that consists of range and keyword attributes. Based on the dictionary W, $\mathcal{A}_I$ constructs a query vector Q as follows:

$$Q = \left(\{(r_l, r_r)_i\}_{i \in [1,n_1]}, \left\{\tilde{k}_j\right\}_{j \in [1,n_2]} \right). \tag{8}$$

$(r_l, r_r)_i$ indicates a desired range associated with the numeric attribute $r_i \in W$. $\tilde{k}_j$ is 0 or 1 to indicate if a keyword $a_{n_1+j} \in W$ is included in Q or not. $\mathcal{A}_I$ sends Q to a source administrator $\mathcal{A}_S$.

2) Query Execution: Upon receiving Q, $\mathcal{A}_S$ searches over local provenance I, G to return the relevance score R_I as shown in Algorithm 3. Briefly speaking, the query function first locates a second-level subindex I_{k_*}. Then, for each tuple $vid_i \in I_{k_*}$, the query checks the range values at the first n_1 dimension and computes a relevance score for the last n_2 dimension. Note that, the relevance score is 0 if range value checks fail.

$\mathcal{A}_S$ selects k vid_i with highest relevance scores and adds them in R_I. For every $vid_i \in R_I$, $\mathcal{A}_S$ adds c_i to R_L and sets $R = (R_I, R_L)$. $\mathcal{A}_S$ runs the Eval(ek, Q, I) to get π_I, that proves $\mathcal{F}$ is correctly executed with inputs I, Q. We omit the details of the π_I but note that the multiexponential exponents $m_F, m_S \in \pi_I$, that are digests of the input index I with generators S_I in (3). Then, $\mathcal{A}_S$ proves the input index I is consistent with the on-blockchain index digest $\mathcal{D}_I$ by computing the digest proof $\pi_D = (X_F, Y_F, X_S, Y_S)$ as follows:

$$\begin{aligned} X_F &= \prod_{i \in [1,n_2]} \prod_{j \in [1,m]} X_{(i-1)*m+j}^{\mu_{i,j}} \\ Y_F &= \prod_{i \in [1,n_2]} \prod_{j \in [1,m]} Y_{(i-1)*m+j}^{\mu_{i,j}} \\ X_S &= \prod_{x \in [1,m]} \prod_{y \in [1,n]} X_{n_2*m+(x-1)*n+y}^{\omega_{x,y}} \\ Y_S &= \prod_{x \in [1,m]} \prod_{y \in [1,n]} Y_{n_2*m+(x-1)*n+y}^{\omega_{x,y}}. \end{aligned} \tag{9}$$

$\mathcal{A}_S$ computes the log proof π_L based on the Merkle tree digest in Algorithm 2 to show that each $c_i \in R_L$ is not modified. Finally, $\mathcal{A}_S$ returns the $R = (R_I, R_L)$ and $\pi = (\pi_I, \pi_L, \pi_D)$ to the $\mathcal{A}_I$. π_I ensures $\mathcal{F}$ is correctly executed with inputs I, Q, and π_D ensures the index I is consistent with the on-blockchain index digest D_I and π_L ensures the returned provenance logs are not modified.

Algorithm 3: $\mathcal{F}$ Execution

Input: Query vector Q, index I, graph G
Output: Query result R_I
Find $k_* = 1 \in Q$ with compact subindex $I_{k_*} \in I_F$
for $vid_i \in I_{k_*} \neq 0$ **do**
  for $j \in [1, n_1]$ **do**
    Set $flag = 1$
    if $r_j \in I_{v_i}$ *does not lie in* $(r_l, r_r)_j \in Q$ **then**
      Set $flag = 0$
  Set $score = 0$
  for $j \in [1, n_2]$ **do**
    Compute $score = score + \tilde{k}_j * k_j$
  Set $R_I[i] = score * flag$

D. Provenance Query Verification

Upon receiving R and π, $\mathcal{A}_I$ checks all the proofs are correct. $\mathcal{A}_I$ first retrieves the digest $\mathcal{D} = (\mathcal{D}_I, \mathcal{D}_L)$ from the blockchain, where $\mathcal{D}_I$ is the index digest and $\mathcal{D}_L$ is the log digest. For $m_F, m_S \in \pi_I$, $\mathcal{A}_I$ checks the consistency with the digest $\mathcal{D}_I$ using π_D

$$\begin{aligned} e(Y_F, g_2) &\stackrel{?}{=} e\left(\mathcal{D}_F, \hat{E}\right) e\left(X_F, \hat{F}\right) e\left(m_F, \hat{G}\right) \\ e(Y_S, g_2) &\stackrel{?}{=} e\left(\mathcal{D}_S, \hat{E}\right) e\left(X_S, \hat{F}\right) e\left(m_S, \hat{G}\right). \end{aligned} \tag{10}$$

$\mathcal{A}_I$ checks Verify$(vk, Q, R_I, \pi_I) = 1$. For $vid_i \in R_I$, $\mathcal{A}_I$ checks $c_i \in L$ and is consistent with the Merkle proof π_L, which is a standard Merkle proof check [28]. If all the proof checks pass, $\mathcal{A}_I$ accepts the query result R. Finally, $\mathcal{A}_I$ collects the correct provenance subgraphs from all source administrators and reconstructs the whole provenance graph for the query Q.

E. Discussions

A fixed-size dictionary-based vector space model is adopted for index and query construction in SEDNP, due to the following reasons: 1) a vector space model for integer comparison/addition/multiplication is fully supported by Pinocchio and 2) for different use cases, multiple search optimization techniques, such as B+ tree for range value indexing and prefix encoding for compact-size index/query vector construction, can also be included in the single model.

In SEDNP, we utilize the public blockchain as the trusted and shared public ledger for cross-domain network provenance query services. Network administrators can store and read succinct provenance digests on the public blockchain without participating in the blockchain mining tasks. By adopting the on/off-chain computing model, the on-chain storage and computing overhead are optimized, which makes SEDNP a feasible solution on the public blockchain architecture. Beyond the network provenance, there are many other IoT services, e.g., identity management for IoT devices, that require distributed trust and detailed redesigns of the blockchain architecture [29].

V. Security Analysis

In this section, we first define security notions of the blockchain and the VC framework. Then, we conduct detailed

analysis of the defined security notions and conclude the *Archiving Security* with Theorem 1.

Definition 5: The transaction robustness of the blockchain is defined as persistence and liveness [30].

Persistence: If a miner in the blockchain network claims a transaction to be stable (k-block deep) in his/her ledger, other nodes will either accept the transaction as stable, or will not report any other transaction in the same position as stable.

Liveness: A transaction will be included in the ledger and reported as stable by a certain time (transaction confirmation time) if all honest miners in the blockchain agree on the transaction and answer honestly if queried.

Definition 6: Given λ, Q, I, I', and $\mathcal{F}$, the security of the VC framework with the digest strategy is defined as *completeness, soundness,* and *index digest collision-resistance* [24].

Completeness is achieved if

$$\mathbf{Pr}\left[\begin{array}{l}\mathbf{Verify}(vk, Q, R_I, \pi_I) = 1: \\ CRS \leftarrow \mathbf{Setup}\left(1^\lambda, \mathcal{F}\right) \wedge \\ R_I \leftarrow \mathcal{F}(Q, I) \wedge \pi_I \leftarrow \mathbf{Eval}(ek, Q, I)\end{array}\right] = 1.$$

Soundness is achieved if

$$\mathbf{Pr}\left[\begin{array}{l}\mathbf{Verify}\left(vk, Q, R_I^*, \pi_I^*\right) = 1: \\ (CRS, pp) \leftarrow \mathbf{Setup}\left(1^\lambda, \mathcal{F}\right) \wedge \\ R_I^* \nleftarrow \mathcal{F}(Q, I) \wedge \\ \left(\pi_I^*, R_I^*\right) \leftarrow \mathbf{Adv}(CRS, pp, \mathcal{F}, Q, I)\end{array}\right] = neg(\lambda).$$

Index Digest Collision-Resistance is achieved if

$$\mathbf{Pr}\left[\begin{array}{l}D_I = D_{I'} : I \neq I' \wedge \\ (DK, pp) \leftarrow \mathbf{Setup}\left(1^\lambda, \mathcal{F}\right) \wedge \\ D_I \leftarrow \mathbf{Digest}(DK, pp, I) \wedge \\ D_{I'} \leftarrow \mathbf{Digest}\left(DK, pp, I'\right)\end{array}\right] = neg(\lambda).$$

Adv is an efficient algorithm for an adversary to forge an instance (π_I^*, R_I^*). *Digest* refers to (7).

We utilize Ethereum in SEDNP, which adopts Proof-of-Work (PoW) consensus protocol and will adopt Proof of State (PoS) in the future hard fork. The security notions of the blockchain are achieved if an adversary cannot control the most computational power in the system (e.g., 51% in a PoW blockchain). That is, the on-blockchain storage cannot be modified once a transaction is confirmed by all honest miners.

Completeness is reduced to the correctness of the VC framework. In particular, the query function $\mathcal{F}$ is first translated to a QAP with three sets of polynomials. The evaluation of the $\mathcal{F}$ with input and output variables is equivalent to the divisibility check of the target polynomials [24]. Each input/output/intermediate variable in the QAP is mapped to a group element in $\mathbb{G}$, which further converts the divisibility check to the linear combination check in $\mathbb{G}$ based on the zk-SNARK technique. That is, *completeness* is guaranteed in SEDNP based on the correctness of the QAP and zk-SNARK.

Soundness is reduced to two aspects: 1) security of the trapdoor s when generating *CRS* and 2) impossibility for computational-bounded adversaries to break d-PKE, 2q-SDH, and q-PDH assumptions in a polynomial time, where d is the degree of the QAP and $q \geq 4d+4$. When generating *CRS*, *TA* chooses a trapdoor secret s for each polynomial in QAP. The trapdoor secret s must be destroyed after constructing the *CRS*. Otherwise, an adversary with the secret s can forge a valid instance (R_I^*, π_I^*) with invalid assignments to the input and output variables of the function $\mathcal{F}$. In SEDNP, *TA* is required to destroy the secret s after generation. However, we note that the role of *TA* could be replaced with a multiparty computation protocol among distrustful nodes [25]. For an adversary, without the knowledge of trapdoor secret s, to break the soundness property and forge a valid instance (R_I^*, π_I^*) is equivalent to break d-PKE, q-SDH, and q-PDH [23].

Index digest collision-resistance is reduced to the hardness of the SXDH problem [22]. Based on the *rational* assumption, network administrators honestly compute a multiexponential digest $\mathcal{D}_I$ and upload the digest to the blockchain. Once the transaction is confirmed, the digest $\mathcal{D}_I$ cannot be later modified. Any adversary that can forge the same digest $\mathcal{D}_I$ with a different index I' must solve the SXDH in $\mathbb{G}$. That is, *index digest collision-resistance* is guaranteed.

Following Definition 3, we conduct the proof sketch of *Archiving Security*.

1) With the correct generation and the trustworthiness of the on-blockchain index and log digests, correctness is guaranteed if the *completeness* of the VC framework, the correctness of the Merkle proof scheme, and (10) is achieved. That is, an honest verifier accepts the instance (R, π) if they are correctly generated over the on-blockchain digests.
2) The blockchain robustness ensures the first requirement of integrity. *Soundness* and *index digest collision-resistance* of the VC framework ensure that query result R_I is generated by honestly performing $\mathcal{F}$ with inputs Q, I, G that are authenticated by the digest D. By the design of $\mathcal{F}$, the second and third requirements of integrity are achieved.

The security of the Merkle proof ensures the returned logs R_L is consistent with the original provenance logs. An adversary can break the security of Merkle proof, iff he can break the security of a collision-resistant hash function *Hash* in Algorithm 2. That is, the fourth requirement of the Integrity is achieved.

In summary, we have the following theorem.

Theorem 1: SEDNP is secure in terms of *Archiving Security* if: 1) q-PDH, 2q-SDH, d-PKE, and SXDH assumptions hold for groups $\mathbb{G}$ with security parameter λ and a QAP with degree d for $q \geq 4d + 4$; 2) Ethereum blockchain is robust against computational-constrained adversary; and 3) the collision-resistent hash function *Hash* is secure.

VI. Performance Evaluation

In this section, we implement SEDNP to provide benchmarks for the on/off-blockchain performance and demonstrate the feasibility of SEDNP.

A. Experiment Setup

The implementation consists of two parts: 1) off-blockchain VC framework and 2) a blockchain testing network.

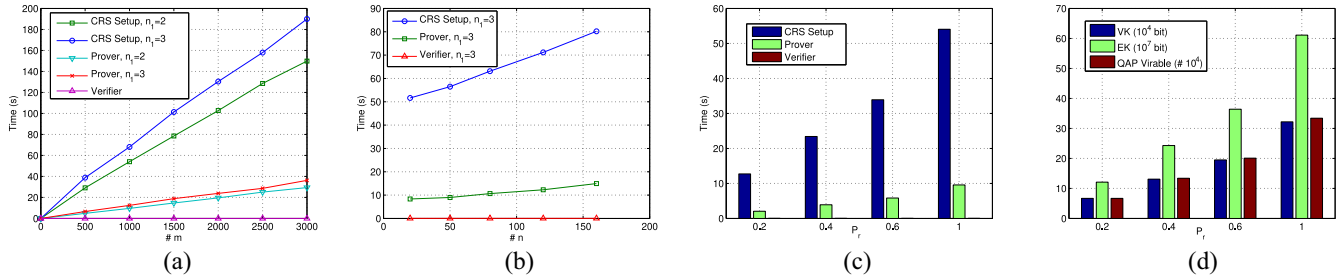


Fig. 4. Computation and storage overhead. (a) Processing time versus m, $n = 100$. (b) Processing time versus n, $m = 1000$. (c) On/off-chain computation tradeoff. (d) On/off-chain storage tradeoff.

1) We implement a VC framework on a laptop with a 2.30-GHz Intel Core processor and 8-GB memory. In particular, we implement the Python interface of the Pinocchio [23] and the *libsnark* interface in [31] with *R1CS* language. Note that, we turn on the "NIZK" for all function inputs. The function $\mathcal{F}$ is written as C code and is converted into a programmable QAP. Based on the obtained QAP, zk-SNARK is implemented over the libff crypto library of *libsnark* [32] with *alt-bn128* curve.
2) We implement a parity [33] Ethereum testing network [12] with the proof-of-authority consensus protocol. The testing network is running on the same laptop and consists of two authority nodes and a few user nodes. The authority nodes are responsible for validating the transactions, while users nodes act as network administrators. We simulate a provenance smart contract using solidity [34] that stores the provenance digests from different network administrators.

For the real-world implementations, the SEDNP is implemented over the Ethereum public blockchain, which is maintained by Ethereum miners. Network administrators can simply generate and publish their blockchain addresses to each other. They collaboratively negotiate the public parameters of the VC scheme and construct a provenance contract for storing and reading cryptographic provenance authenticators.

In the following, we first evaluate the efficiency of $\mathcal{F}$ in Algorithm 3. Then, we give the analysis of the digest scheme. Finally, we demonstrate the optimized on-chain storage overhead in SEDNP and analyze the tradeoffs of the multilevel index strategy.

B. Function $\mathcal{F}$ Evaluation

The index query function $\mathcal{F}$ over the index I consists of keyword-based lookup over I_F and a linear search over I_S, where I_F is the first-level index and I_S is the second-level index. We first provide benchmarks for linear query over I_S. Then, we present the evaluations of the overall query function $\mathcal{F}$ with detailed observations.

For the query over I_S, it takes into a query vector Q and an $m * n$ matrix I_S, where m denotes the number of provenance tuples and n denotes the dimensions of the query and index. $n = n_1 + n_2$, where n_1 is the dimension of the range values and n_2 is the dimension of the keyword values. We set both the last n_2 dimensions of Q and I_{v_i} as "integer" in the experiment, where I_{v_i} is the subindex of v_i. We first compare the range values between Q and each index I_{v_i} in I_S. Then, we compare the last n_2 dimensions of I_{v_i} with that in Q and include the comparison result to R_I as a similarity score. We measure the computation cost as the processing time in terms of the *CRS* setup, proof generation, and verification.

TABLE II
INDEX DIGEST COST VERSUS m, n

Component	DK_E	DK_V	π_D
Size	$4mn'\|\mathbb{G}_1\|$	$3\|\mathbb{G}_2\|$	$4\|\mathbb{G}_1\|$
Operation	Key Generation	Prover	Verifier
Complexity	$3mn'E_1 + 3E_2$	$2mn'E_1$	4 P

In Fig. 4(a) and (b), we can see that the processing time for *CRS* setup and prover cost is linearly increasing with m and n. The verifier cost remains constant (0.027 s) regardless of the input size. Meanwhile, a higher n_1 increases the QAP complexity and the setup/prover/verifier cost, since comparison operations are much more expensive than algebraic operations in a circuit. After one-time *CRS* setup, the prover and verifier cost is reasonable with a few seconds.

In Table III, we summarize the storage cost for the search over I_S. In the experiment, we set Q and I as private input, such that their associated multiexponentiation components are not distinguished in the proof. The representations of group elements in libff are optimized. The number of *QAP* variables indicates the complexity of the generated *QAP*. The same linear increasing property with m, n is observed for the storage cost. The size of *ek* is much larger than that of *vk* (10^3 times), while the size of π_I remains constant.

C. Digest Scheme Evaluation

In Table II, we summarize the storage and computation cost of the index digest. $|\mathbb{G}|$ denotes the storage cost of a group element. E_1/E_2 denotes exponentiation operation in $\mathbb{G}_1/\mathbb{G}_2$. P denotes the paring operation. Note that (10) can be combined into four pairings with the batch verification technique [35]. We can see that the prover storage and computation cost is linearly increasing with m, n' while verifier cost remains constant, where $n' = n + n_2$. However, since exponentiation operations in *alt-bn128* curve only cost a few nanoseconds, the prover computation cost is still practical. Note that, the digest scheme requires additional elements $m_F, m_S \in \pi_I$ with two more checks of their appropriate spans, which slightly increases the proof size and verification cost compared with that in the *libsnark* implementation of $\mathcal{F}$.

TABLE III
STORAGE COST FOR I_S SEARCH VERSUS m, $n = 100$

# m (10^2)	**ek** (10^6 bits)		**vk** (10^3 bits)		# **QAP Variables**		π_I (bits)
	$n_1 = 2$	$n_1 = 3$	$n_1 = 2$	$n_1 = 3$	$n_1 = 2$	$n_1 = 3$	
5	305	386	162		167101	200101	2294
10	610	773	322		334101	400101	2294
15	915	1168	481		501101	600101	2294
20	1221	1547	641		668101	800101	2294

D. On/Off-Chain Tradeoff

Multilevel provenance index is instantiated with an $n_2 * m$-dimensional I_F and an $m*n$-dimensional I_S. I_F is a vector that indicates whether a keyword exists in the subindex I_{v_i}. For a given keyword k^*, we first locate all subindexes I_{v_i} that contain k^* and then perform a linear query over those I_{v_i}. The design is based on the following observations.

1) *CRS* generation phase is expensive compared with the prover and verifier phase. Thus, *CRS* is generated once and instantiated with different Q and I as inputs for provenance query instances.
2) The size of the keyword-inverted subindex I_{k_i} can change with different keywords. However, the input size of the $\mathcal{F}$ has to be fixed, since the subscript for an array needs to be determined for the C implementation of the Pinocchio VC framework. Thus, I_{k_i} is implemented as a fixed n_2 dimension vector.

On-blockchain storage and computation cost is optimized with compact digest $\mathcal{D}_I$ and $\mathcal{D}_L$. However, the optimization is achieved with the increasing off-blockchain computation and storage overhead. This is reasonable since on-blockchain storage and computation cost is much more expensive compared with the off-blockchain cost. An alternative strategy is to store each subindex I_{k_i} as one single digest on the blockchain. This increases the on-chain storage with n_2 times since with the dimension of the index I_F. To give a more clear tradeoff analysis between on/off-blockchain performance, we conduct experiments with another performance indicator *reducing factor* $p_r = |I_{k_*}|/m$. We implement the provenance index with $n = 100, m = 1000$, and $n_1 = 2$, and change p_r to show the performance gap. In Fig. 4(c) and (d), we can see that a smaller p_r indicates lower off-chain storage and computation cost, since the search space is reduced by p_r times. SEDNP can achieve optimized constant on-chain overhead and can still allow system designers to flexibly adjust the on/off-chain tradeoff for specified provenance cases.

VII. RELATED WORK

Zhou *et al.* [11] proposed a query framework with a design of distributed query processing methods. Moreover, a peer-voting strategy was utilized to provide countermeasures against compromised network nodes. In [9], the confidentiality and access control of the network provenance data was investigated and a searchable symmetric encryption (SSE)-based approach was proposed. Hassan *et al.* [36] studied the threat alert fatigue issue in the threat detection software. The authors also utilized a causal dependency graph-based provenance approach to calculate the anomaly scores for neighboring edges. Wu *et al.* [37] introduced temporal provenance to traditional provenance-based network debugging that determines root causes of temporal problems in the network.

Li *et al.* [38] proposed a verifiable energy trading scheme based on a consortium blockchain. The proposed scheme utilized anonymous authentication and fine-grained access control to audit the energy trading process. Hu *et al.* [29] proposed a blockchain-based SSE, by fully exploring the practical challenges of the blockchain technologies with two designs on both public and private blockchains. The identified tradeoff between security and efficiency provided insights for blockchain-based industrial services. Li *et al.* [39] studied the ad dissemination issue in the vehicular networks and proposed a blockchain-based architecture that enhances advertising fairness. Liang *et al.* [40] utilized blockchain to construct a provenance architecture that collects and verifies the provenance data for cloud storage applications. Jiang *et al.* [41] proposed a blockchain-based architecture for the vehicular networking with distributed data storage and transmission. The theocratical and numerical analysis demonstrated the practicality of the proposed architecture.

Different from the existing works, SEDNP investigates the inherit security issues in the distributed network provenance for the IoT, where IoT network administrators from different domains have insufficient mutual trust. SEDNP proposes a blockchain-based architecture with new security notions, practical designs, applicable on-blockchain optimizations, and tradeoff analysis for the distributed network provenance.

In the following, we summarize recent research advances in VCs. Gennaro *et al.* [24] proposed a general VC framework for NP-complete relation families based on quadratic span programs and zk-SNARK. The proposed framework [24] was later improved by Parno *et al.* [23] with the optimizations by using a regular QAP. Fiore *et al.* [22] observed the multiexponential form in the zk-SNARK. Based on the observation, Fiore *et al.* [22] proposed a framework that enables multiple-time VCs over the same authenticated data. Subsequently, Agrawal *et al.* [42] observed that the multiexponential form could collaboratively work with the traditional sigma protocol in the discrete logarithm setting. Agrawal *et al.* [42] extended this observation to propose a framework that flexibly determines the zero-knowledge property of inputs/outputs of the VC framework.

VIII. Conclusion

In this article, we have explored security challenges for operating distributed network provenance for the IoT in different trust domains. By taking the advantages of its decentralization and immutability, we have utilized the blockchain as the essential architecture for storing and querying cross-domain provenance data. We have introduced a unified model for efficiently instantiating provenance query modules in the VC framework. Specifically, the proposed SEDNP has made the blockchain-based provenance architecture for the IoT practical by trading the expensive on-blockchain storage and computation cost with acceptable off-blockchain overheads. By doing so, the on-blockchain storage and computation overhead for network provenance can be optimized to the constant regardless of provenance data size, while the correctness and integrity of query results are guaranteed. The implementation observations and benchmarking results could pave the way for future research on the designs and optimizations of provenance query modules for specified network provenance cases for the IoT.

References

[1] M. S. Mahmud, H. Fang, and H. Wang, "An integrated wearable sensor for unobtrusive continuous measurement of autonomic nervous system," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1104–1113, Feb. 2019.

[2] A. Chen, Y. Wu, A. Haeberlen, W. Zhou, and B. T. Loo, "The good, the bad, and the differences: Better network diagnostics with differential provenance," in *Proc. ACM Conf. Appl. Technol. Archit. Protocols Comput. Commun. (SIGCOMM)*, 2016, pp. 115–128.

[3] T. Pasquier *et al.*, "Runtime analysis of whole-system provenance," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2018, pp. 1601–1616.

[4] Z. Liu and Y. Wu, "An index-based provenance compression scheme for identifying malicious nodes in multi-hop IoT network," *IEEE Internet Things J.*, early access, Dec. 23, 2019, doi: 10.1109/JIOT.2019.2961431.

[5] A. Chen, Y. Wu, A. Haeberlen, B. T. Loo, and W. Zhou, "Data provenance at Internet scale: Architecture, experiences, and the road ahead," in *Proc. Conf. Innov. Data Syst. Res. (CIDR)*, 2017, pp. 1–7.

[6] Z. Li *et al.*, "Multiobjective optimization based sensor selection for TDOA tracking in wireless sensor network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12360–12374, Dec. 2019.

[7] M. N. Aman, M. H. Basheer, and B. Sikdar, "Data provenance for IoT with light weight authentication and privacy preservation," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10441–10457, Dec. 2019.

[8] R. Hu, Z. Yan, W. Ding, and L. T. Yang, "A survey on data provenance in IoT," *World Wide Web*, vol. 23, no. 2, pp. 1441–1463, 2019.

[9] Y. Zhang, A. O'Neill, M. Sherr, and W. Zhou, "Privacy-preserving network provenance," *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1550–1561, 2017.

[10] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020.

[11] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr, "Secure network provenance," in *Proc. ACM Symp. Oper. Syst. Principles*, 2011, pp. 295–310.

[12] G. Wood, *Ethereum: A Secure Decentralised Generalised Transaction Ledger Byzantium Version*, Ethereum, Zug, Switzerland, Jun. 2018.

[13] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2018, pp. 792–800.

[14] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Trans. Depend. Secure Comput.*, early access, Oct. 22, 2018, doi: 10.1109/TDSC.2018.2877332.

[15] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling verifiable Boolean range queries over blockchain databases," in *Proc. SIGMOD*, 2019, pp. 141–158.

[16] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "Gem2-Tree: A gas-efficient structure for authenticated range queries in blockchain," in *Proc. IEEE ICDE*, 2019, pp. 842–853.

[17] J. Eberhardt and S. Tai, "Zokrates-scalable privacy-preserving off-chain computations," in *Proc. IEEE Int. Conf. Blockchain*, 2018, pp. 1084–1091.

[18] S. Steffen, B. Bichsel, M. Gersbach, N. Melchior, P. Tsankov, and M. Vechev, "zkay: Specifying and enforcing data privacy in smart contracts," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2019, pp. 1759–1776.

[19] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing ATOP POS blockchain," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.

[20] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, early access, Apr. 1, 2019, doi: 10.1109/TMC.2019.2908638.

[21] M. Li, L. Zhu, and X. Lin, "Privacy-preserving traffic monitoring with false report filtering via fog-assisted vehicular crowdsensing," *IEEE Trans. Services Comput.*, early access, Mar. 4, 2019, doi: 10.1109/TSC.2019.2903060.

[22] D. Fiore, C. Fournet, E. Ghosh, M. Kohlweiss, O. Ohrimenko, and B. Parno, "Hash first, argue later: Adaptive verifiable computations on outsourced data," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2016, pp. 1304–1316.

[23] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE S&P*, 2013, pp. 238–252.

[24] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *Proc. EUROCRYPT*, 2013, pp. 626–645.

[25] S. Bowe, A. Gabizon, and M. D. Green, "A multi-party protocol for constructing the public parameters of the Pinocchio ZK-SNARK," in *Proc. FC*, 2018, pp. 64–77.

[26] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2010, pp. 1–5.

[27] Y. Zhang, C. Papamanthou, and J. Katz, "ALITHEIA: Towards practical verifiable graph processing," in *Proc. ACM CCS*, 2014, pp. 856–867.

[28] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. CRYPTO*, 1987, pp. 369–378.

[29] S. Hu, C. Cai, Q. Wang, C. Wang, Z. Wang, and D. Ye, "Augmenting encrypted search: A decentralized service realization with enforced execution," *IEEE Trans. Depend. Secure Comput.*, early access, Dec. 2, 2019, doi: 10.1109/TDSC.2019.2957091.

[30] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Proc. EUROCRYPT*, 2018, pp. 66–98.

[31] A. E. Kosba, C. Papamanthou, and E. Shi, "xJsnark: A framework for efficient verifiable computation," in *Proc. IEEE S&P*, 2018, pp. 944–961.

[32] *C++ library for zkSNARKs*. Accessed: Mar. 2020. [Online]. Available: https://github.com/scipr-lab/libsnark

[33] *Proof-of-Authority (PoA) Chains*. Accessed: Mar. 2020. [Online]. Available: https://wiki.parity.io/Proof-of-Authority-Chains.

[34] *Solidity*. Accessed: Mar. 2020. [Online]. Available: https://solidity.readthedocs.io/en/v0.4.25/

[35] C. Zhang, R. Lu, X. Lin, P.-H. Ho, and X. Shen, "An efficient identity-based batch verification scheme for vehicular sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2008, pp. 246–250.

[36] W. U. Hassan *et al.*, "NoDoze: Combatting threat alert fatigue with automated provenance triage," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2019, pp. 1–15.

[37] Y. Wu, A. Chen, and L. T. X. Phan, "ZENO: Diagnosing performance problems with temporal provenance," in *Proc. Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 395–420.

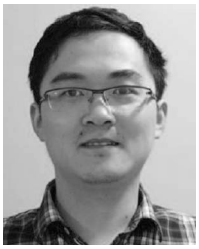
[38] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, early access, Feb. 17, 2020, doi: 10.1109/TII.2020.2974537.

[39] M. Li, J. Weng, A. Yang, J.-N. Liu, and X. Lin, "Towards blockchain-based fair and anonymous ad dissemination in vehicular networks," vol. 68, no. 11, pp. 11248–11259, Nov. 2019, doi: 10.1109/TVT.2019.2940148.

[40] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2017, pp. 468–477.
[41] T. Jiang, H. Fang, and H. Wang, "Blockchain-based Internet of Vehicles: Distributed network architecture and performance analysis," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4640–4649, Jun. 2019.
[42] S. Agrawal, C. Ganesh, and P. Mohassel, "Non-interactive zero-knowledge proofs for composite statements," in *Proc. CRYPTO*, 2018, pp. 643–673.

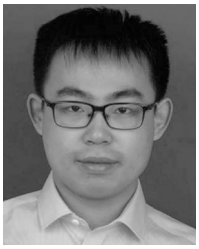
Dongxiao Liu (Student Member, IEEE) received the B.S. and M.S. degrees from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada.

His research interests include applied cryptography and privacy enhancing technologies for blockchain.



Jianbing Ni (Member, IEEE) received the B.E. and M.S. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2018.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada. His current research interests include applied cryptography and network security, with a focus on cloud computing, smart grid, mobile crowdsensing, and Internet of Things.



Cheng Huang (Student Member, IEEE) received the B.Eng. and M.Eng. degrees in information security from Xidian University, Xi'an, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Waterloo, Waterloo, ON, Canada.

He was a Project Officer with the INFINITUS Laboratory, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, till July 2016. His research interests are in the areas of applied cryptography, cyber security, and privacy in the mobile network.

Xiaodong Lin (Fellow, IEEE) received the first Ph.D. degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1998, and the second Ph.D. degree (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2008.

He is currently an Associate Professor with the School of Computer Science, University of Guelph, Guelph, ON, Canada. His research interests include computer and network security, privacy protection, applied cryptography, computer forensics, and software security.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular *ad hoc* and sensor networks.

Dr. Shen received the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) presents in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from Wireless Communications Technical Committee in 2019 and AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, and IEEE Globecom'07, the Symposia Chair for IEEE ICC'10, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the elected IEEE Communications Society Vice President for Technical and Educational Activities, the Vice President for Publications, the Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and the Member of the IEEE Fellow Selection Committee. He was/is the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, *IET Communications*, and *Peer-to-Peer Networking and Applications*. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.