A Distributed Ledger for Non-attributable Cyber Threat Intelligence Exchange

Philip Huff(✉) and Qinghua Li

University of Arkansas, Fayetteville, AR 72701, USA
{phuff,qinghual}@uark.edu

Abstract. Cyber threat intelligence (CTI) sharing provides cybersecurity operations an advantage over adversaries by more quickly characterizing the threat, understanding its tactics, anticipating the objective, and identifying the vulnerability and mitigation. However, organizations struggle with sharing threat intelligence due, in part, to the legal and financial risk of being associated with a potential malware campaign or threat group. An entity wishing to share threat information or obtain information about a specific threat risks being associated as a victim of the threat actors, resulting in costly legal disputes, regulatory investigation, and reputational damage. As a result, the threat intelligence data needed for cybersecurity situational awareness and vulnerability mitigation often lacks volume, quality, and timeliness. We propose a distributed blockchain ledger to facilitate sharing of cybersecurity threat information and provide a mechanism for entities to have non-attributable participation in a threat-sharing community. Learning from Distributed Anonymous Payment (DAP) schemes in cryptocurrency, we use a new token-based authentication scheme for use in a permissioned blockchain. The anonymous token authentication allows a consortium of semi-trusted entities to share the workload of curating CTI for the community's cooperative benefit.

Keywords: Blockchain · Cyber threat intelligence · Zero-knowledge proof

1 Introduction

Adversaries have the upper hand in cyber attacks. They benefit from anonymity, both in person and in purpose. In contrast, targeted entities (e.g., companies) have difficulty distinguishing everyday benign activities from malicious activities. Thus, entities spend prodigious efforts to gain actionable threat intelligence. In a recent survey on Cyber Threat Intelligence (CTI) sharing, security professionals strongly agree that intelligence sharing supports breach detection/recovery and vulnerability identification/mitigation efforts [40]. However, many technical, trust, legal and cultural barriers prohibit more widespread threat information sharing [21].

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 398, pp. 164–184, 2021.
https://doi.org/10.1007/978-3-030-90019-9_9

Many cyber threats target critical infrastructures in the private sector. These target entities have the same trust barriers and even more technical and legal barriers due to the limits of qualified security professionals working at each organization. A recent report on cyber threat sharing indicated only 3% of private sector participants shared any threat indicators in 2018 [25].

Furthermore, the value received from CTI is often lacking due to various technical challenges and missing context. In one study [7] 70% of respondents find shared threat data too voluminous and complex for actionable intelligence. Similarly, [8] finds CTI solutions need to enhance their ability to provide context and flexibility to improve the overall value proposition.

1.1 The Current State of Threat Sharing

Organizations are rapidly developing the competency and appetite to participate more in threat-sharing communities. The global rise in security operations centers, through which most CTI exchange occurs, has an expected market growth of 11.5% through 2025 [9]. However, with current approaches heavily focused on classified data and government intelligence services, actionable data is too little and too late. Likewise, as [22] points out, private sector organizations have little motivation to share their threat data sustainably.

Entities share threat data to gain a better understanding of the risk posed to their mission. An average entity may experience tens of thousands of malicious probes from the Internet per day. However, most probes result from automated scanning and do not represent a motivated and intelligent human adversary. Entities participate in threat sharing to distinguish actual danger from benign in hopes of mitigating the threat before it manifests.

Society has an interest in preventing cyber threats from entities that provide critical services and infrastructure. Military and law enforcement agencies would generally provide protection, but they have limited purview into the interaction between adversaries and private entities. Government agencies, national Computer Emergency Response Teams (CERTs), and non-profit Information Sharing and Analysis Centers (ISACs) offer two-way threat-sharing services to address this gap.

However, private entities have many barriers encumbering CTI sharing. A private entity wishing to share threat information risks attribution of the cyber threat, resulting in costly legal disputes, regulatory investigation, and reputational damage. For example, a mistaken analysis of VPN logs to maintain a failed water pump led to a federal investigation of cyber warfare [39]. In [26], legal compliance and limiting attribution are identified as the primary challenges for organizations wishing to share their own CTI with others.

Additional barriers exist with sharing of classified intelligence to private entities. Programs exist to clear private sector entities, but they come at a high cost. Then, moving classified intelligence to actionable threat and vulnerability mitigation cannot keep pace with adversarial intrusion techniques' dynamic nature. Likewise, attempts for fully bidirectional threat sharing have mostly failed.

1.2 Contributions

This paper provides a solution for entities to share observed CTI without attribution using a permissioned blockchain. We propose a novel approach to a Distributed Anonymous Payment (DAP) scheme [33] for permissioned blockchains to allow for anonymous transactions in CTI sharing. This solution also efficiently maintains anonymous authentication and provides revocation services for entities. It does so by splitting maintenance of the Merkle tree used for anonymous authentication between participating peers, which allows for more regular updates of the Merkle Tree across the distributed ledger.

Anonymous transactions address the legal and regulatory barriers organizations have with cyber threat attribution, increasing CTI sharing on the ledger. We then propose a new chaincode to incentivize CTI creation for the cooperative benefit of participating entities. The chaincode targets the barriers preventing bidirectional threat sharing between private sector entities and government agencies by generating timely and actionable CTI without the need for costly declassification.

The chaincode also seeks to reduce volume and increase value in CTI. Human analysts control the volume of threat data through work evaluation functions. Whereas automated log sharing solutions produce data at the speed of machines, the chaincode produces intelligence at the speed of humans. Furthermore, human analysts should find the intelligence actionable because the chaincode originates directly from private entity queries.

1.3 Organization

Section 2 reviews related work. Section 3 introduces the building blocks for our approach. Section 4–6 presents the proposed approach and its major components. Section 7 discusses evaluation results. Section 8 concludes this paper.

2 Background and Related Work

CTI exchange programs fall into three categories:

1. Classified Threat Sharing - Provides automated classified threat indicators to its members. The DHS Enhanced Cybersecurity Services (ECS) is an example of this type of service [4].
2. Data Lakes - Collects a large volume of logs from its members and centrally analyzes the data. The Department of Energy Cyber Risk Information Sharing Program (CRISP) uses the data lake model [3].
3. Analyst to Analyst - Threat hunting analysts exchange data over a shared platform. The European Union Agency for Cybersecurity recommends the Malware Information Sharing Platform for community threat sharing [8].

This paper targets the third category of CTI in which human analysts directly share threat intelligence and indicators between entities. The most commonly

shared threat data includes low-level indicators such as IP addresses, URIs, DNS names, and file hashes collected automatically or via threat hunting. Our platform supports sharing of other security information as well, e.g., vulnerability mitigation information. Many services provide one-way data sharing to the entity of known malicious threat indicators.

Stillions' Detection Maturity Levels [35] characterizes this type of data as lower-level evidence of an intrusion attempt. In contrast, higher levels of intelligence include data about how the adversary operates and their motivations.

The work of creating CTI involves tying lower-level indicators to adversarial motivation. However, these indicators exist in the networks of private entities and outside of the direct purview of CTI producers. Timely bidirectional CTI exchange means indicators and resulting CTI are shared freely. The producers receive value by better tracking malicious activity, and consumers receive value through an improved understanding of adversarial risk.

Using a distributed ledger, we can commoditize CTI work as described in Sect. 6 while, at the same time, eliminating trust barriers that preclude the sharing of threat indicators.

2.1 Blockchain Technologies

The permissioned ledger fundamentally uses blockchain as a basis for distributed trust. Blockchain has gained popularity with cryptocurrency technologies like bitcoin [30], and ethereum [38] making possible public distributed transactions with no central authority. Several recent works have suggested using blockchain technologies for CTI exchange [23,24,32]. Our work differs by addressing attribution and targeting CTI sharing communities of trust through a permissioned ledger.

The use of a permissioned blockchain presented in [10] has growing acceptance as a general-purpose distributed ledger. While still public, in the sense of being accessible over the Internet, permissioned blockchains take advantage of partial trust relationships in a system. In the Hyperledger Fabric project, network peers first execute transactions and then order and distribute them onto the blockchain. This approach allows for more complex transactions because peers can detect state and denial of service problems before the chaining operation.

We choose a permissionless blockchain over a public blockchain because of privacy considerations. A CTI sharing community is often open only to participating members from a given sector or nation-state. Although peer entities have no problems with attribution among the community, privacy concerns would likely arise in a public blockchain.

2.2 Zero Knowledge Proofs

The public nature of blockchain systems spotlights the need for anonymity and private information retrieval. Common to most solutions to these problems are zero-knowledge proofs (ZKP), which allow authentication without identification.

In [19], Chaum first developed an e-cash system in which a user could present proof of authentication from some certifying entity without revealing the user. Pseudonym systems in [28] have a similar mechanism to allow entities to operate under a pseudonym untraceable to their original authenticated identity and ultimately form a chain of pseudonyms to conduct anonymous transactions in a system.

Direct Anonymous Authentication (DAA) systems extend and implement ZKP and have widely deployed on trusted platform modules (TPM), and blockchain systems [15–18]. Most recently, the anonymizing idemix library has become available as a core service in Hyperledger Fabric.

However, DAA schemes do not have a mechanism for incentives, and they require additional roles in managing access to the ledger. Instead, we look to recent advancements in cryptocurrency. The explosive growth of cryptocurrencies has ushered in a wave of innovation in anonymizing transactions in the past decade. Anonymous spending in cryptocurrency is made possible through zero-knowledge Succinct Non-Interactive ARgument of Knowledge (zk-SNARKS) presented in [20]. Zerocoin [29] is one of the first systems proposed to support anonymous transactions on top of bitcoin. Zerocash [33] and others [27] made use of zk-SNARKS to make this more feasible and extend the system to prevent tracing the history of a coin and improve efficiency.

Although permissioned blockchains do not require a cryptocurrency incentive, we propose an incentive mechanism for the desired outcome of high quantity and quality threat data. The "gas" or currency of cybersecurity exists in human work and actionable CTI.

3 Building Blocks

Before presenting the approach to non-attributable CTI sharing, we introduce the building blocks used by our approach.

3.1 Sparse Merkle Trees

Merkle trees provide an efficient data structure to authenticate information. They are used on the blockchain to verify transactional integrity. Branches of the tree get formed from the combined secure hashes of its children. In this way, anyone can verify the membership of a tree leaf by comparing the calculated Merkle root with some other valid Merkle root.

Sparse Merkle Trees make use of the property that the path to any given leaf is a function of a small number of branches up to the Merkle root. In the example shown in Fig. 1, we store a minted coin, `cm`, as a leaf in the Merkle Tree. The leaf's index is determined by the branch direction down the tree, in which a 0 means the left branch, and a 1 means the right branch. Then, for someone to later validate the inclusion of `cm`, they need only the index and the tree branches along the path indicated by the index, which is necessary to calculate the root.

We represent the tree path as `path`, which contains attributes for the index location in the tree, `path`.*addr*, and the branch siblings, `path`.S necessary to calculate the Merkle root.

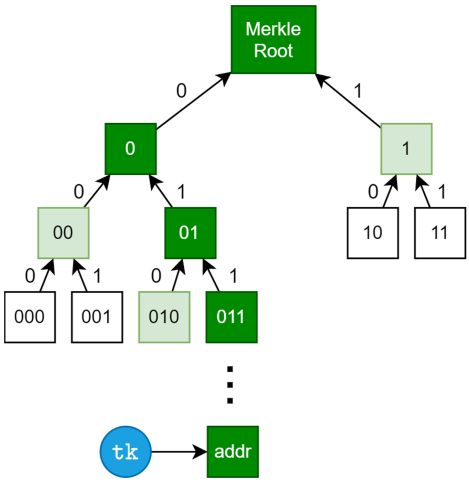

Fig. 1. Sparse Merkle tree.

3.2 Distributed Anonymous Payment

First, distributed anonymous payment (DAP) schemes allow an entity to prove they have an electronically minted coin, `cm`, without actually revealing the coin. The proof also requires the entity to provide knowledge of an associated, yet untraceable, serial number, `sn`, to prevent an entity from double-spending.

DAP schemes have the important property of retaining the minted coin as a valid leaf value in the Merkle Tree. Unlike Bitcoin, they do not have the luxury of maintaining an unspent transaction object (UTXO) inventory. To do so requires identifying spent coins, which DAP schemes do not reveal. Therefore, we must evaluate the Merkle tree size appropriate to support the life of the blockchain.

3.3 zk-SNARKs

The proof of knowledge in [33] uses zero-knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARK) proofs from [13]. zk-SNARKs provide an efficient proof construct and verification mechanism. Our proof demonstrates the knowledge of a `cm` $\in$ `CMList` without revealing `cm`, which equates to an anonymous user proving, "I have a valid token, but to ensure my anonymity, I am not going to tell you which token."

At its core, a zk-SNARK equates to demonstrating knowledge of a well-formed polynomial, $p(x)$, such that $h(x)t(x) = p(x)$, where $t(x)$ is a target polynomial available to the ledger, and $h(x)$ is derived by the prover as $h(x) = p(x)/t(x)$. The prover constructs the polynomial, $p(x)$, through an algebraic circuit available on the ledger which has been translated from code representing the Merkle Tree proof of knowledge. The prover samples some arbitrarily chosen secret s, such that $h(s)t(s) = p(s)$. To ensure the integrity of the target polynomial and sampled value, s, all operations are performed using homomorphic encryption with generator, g, such that $(g^{h(s)})^{t(s)} = g^{p(s)}$.

The process for non-interactive proof and verification consists of the following steps:

1. **Multi-Party Setup** - A multi-party setup protocol occurs to produce the public parameters, $\mathtt{pp}$, which includes the homomorphic encryption of the powers of x in the secret polynomial of dimension, d for secret, s. Thus, the proving key consists of the powers necessary to compute the secret polynomial, the target polynomial, and sampled values to ensure zero knowledge of the secret polynomial. An initial setup requires multiple parties with strong zero-knowledge guarantees [14]. The keys used for proving and verification are referred to as the *common reference string*.
2. **Algebraic Circuit** - A program to construct the zero-knowledge proof converts to an algebraic circuit by flattening the program into a series of expressions in the form $x = y\mathtt{op}z$, which form the so-called circuit wires. Ultimately, these form the basis of the secret polynomial coefficients. In our case, the circuit consists of the Merkle Tree proof of inclusion.
3. **Proof** - An entity constructs a proof of knowledge demonstrating they have a valid token in the Merkle tree using both the public parameters and algebraic circuit. The proof is non-interactive because the prover does not need to exchange keys to produce the proof statement. Zero-knowledge comes through a key sampled by the prover, which conceals the secret polynomial.
4. **Verification** - Verification is performed in the chaincode of the ledger to ensure the construction of the secret polynomial in addition to the public inputs to the circuit is valid.

Besides the original works in zk-SNARKs, the papers [11,31] provide good tutorials on the process.

4 Distributed Ledger for Threat Sharing

Distributed ledgers provide transactional integrity for large and diverse communities. In its most well-known cryptocurrency implementations, distributed ledgers supply a high assurance system for transacting digital goods such as Bitcoin. Our scheme considers human work as the exchanged commodity for cybersecurity threat sharing. The work of threat identification and attribution involves costly human labor to identify artifacts, piece together the adversarial objective, and tie cyber observables to malware campaigns and threat actors.

Entities receive value through more actionable intelligence and an improved understanding of cyber risk.

The use of a distributed ledger for cybersecurity work is not without precedent. [34] proposes the use of economic incentives to incentivize secure data sharing. Also, in many ways, a marketplace for threat information can be compared to software bug bounty programs where companies wishing to fix software vulnerabilities before an adversary exploits them monetize the work of finding vulnerabilities [5]. However, with cyber threats, the work production comes from entities wishing to protect their systems better.

We propose a distributed ledger in which any participating entity submits monetized threat intelligence work in the form of structured work queries as transactions on the ledger. Entities requesting work do so through anonymous credentials using a web application tied to a peer entity on the distributed ledger. Participants use the same web application to search for information about a given threat. The ledger does not record searches as transactions.

4.1 Distributed Ledger Network

This section proposes a permissioned blockchain network architecture to support the exchange of threat intelligence between participating entities. Our implementation for threat sharing uses a permissioned blockchain. These differ from public blockchains by requiring authenticated access and eliminating the need for proof-of-work or proof-of-stake consensus. Chaincode is a set of smart contracts installed by participating entities and serves as the blockchain's central service rather than the currency transaction object. With cryptocurrency, smart contracts are a service of the blockchain, but with permissioned blockchains, the blockchain is a service of the smart contract.

Also, cryptocurrencies overcome almost all trust boundaries, but this is not always desirable, especially with CTI. Instead, we use the permissioned blockchain to overcome trust boundaries existing between organizations.

Figure 2 shows an example blockchain network in where the shaded area represents elements required by the blockchain and users involved in CTI access the network outside of the shaded area. Fundamentally, the blockchain includes a group of entities, referred to as peers, who have consensus on the chaincode execution and maintain a copy of both the blockchain and the current state database of chaincode assets (or objects).

Peers join the network either initially or through peer consensus. The collective peers comprise the distributed system's nodes, and they participate in the validation of new blocks and storage of the data. However, with permissioned blockchains, peers also provide the service of user interaction with the blockchain network.

An organization does not need to be a peer of the blockchain to participate in the service. Instead, peers provide credentialing services through their certificate authority. Users of other organizations are then permitted to execute chaincode transactions through peer applications.

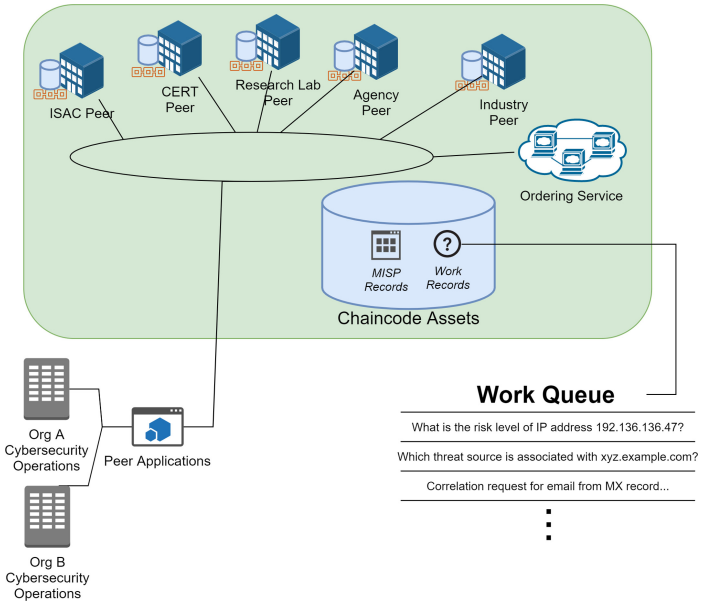

Fig. 2. Threat ledger network.

In the example shown in Fig. 2, the peers include organizations typically involved in threat sharing, such as government agencies, CERTs, ISACs, and research labs. These organizations have the incentive and resources to install and maintain the peer service needed for threat sharing. If a private entity wanted to participate in the network, they would only need to obtain credentials from a peer and use a published web application, thus, significantly lowering the bar of complexity for threat exchange participation.

The network also requires an ordering service. Blocks of transactions get added to the blockchain through the ordering service. Consistent with the execute-order-validate consensus approach described in [10], peers will first simulate the execution of proposed transactions before sending them to the ordering service. The ordering service then packages valid transactions into the next block and sends them to all network peers.

4.2 Chaincode Assets

The network's chaincode centers on CTI reports commonly exchanged between organizations. We choose to use the standard MISP format [37]. Other CTI taxonomies include STIX [12], and the Common Cyber Threat Framework [6], but the MISP format is extensible and concentrates on the threat report instead of the observable artifacts. By aggregating artifacts into event reports, we can more easily form a high-level representation of the CTI report's value.

An sample MISP report with object relationships is shown in Fig. 3. A MISP *Event Report* contains the creating organization (or anonymous), description, and report object, which can range from single threat observation reports to several thousand indicators and sightings of a malware family. There are over 200 open object definitions, and reports can contain multiple objects. Tags describe the report in terms of the information-sharing community. Example tags include the DHS Traffic Light Protocol, malware classifications, IDS rules, and admiralty scale. The tags can be helpful in chaincode for defining access control rules, expiration, and other state transition logic.

Finally, object attributes tie to the reported objects and contain the observable artifacts associated with an event, such as IP addresses, URIs, file hashes, and email addresses. Attributes are the primary search targets for the network. Each network peer stores a document-oriented NoSQL database of existing reports and indexes the attributes for fast searching and correlation.

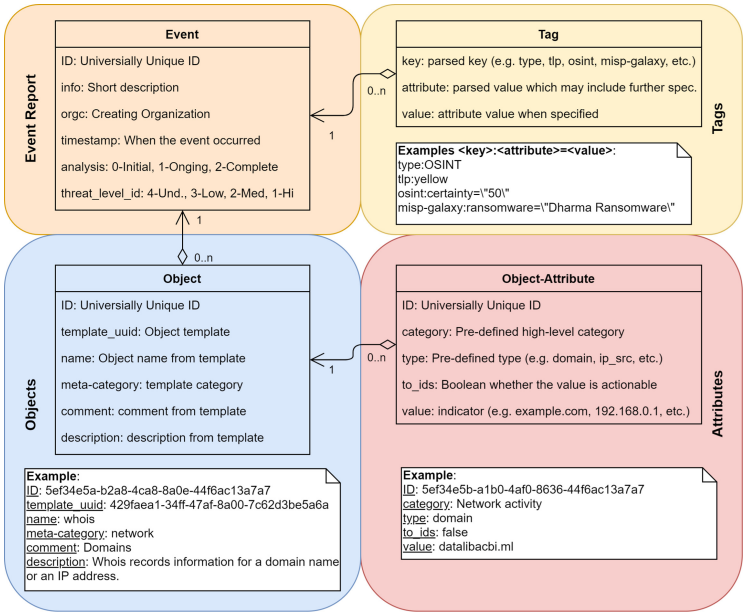

Fig. 3. MISP data object model.

Besides the threat objects, we also define two assets used for managing the quality of threat reports. The *work* asset represents human work and consists of structures for both the problem and the solution. When first submitted, the solution is empty and queued for human analysis. Examples of work may include associating tactics, techniques, and procedures (TTP) to threat artifacts or attribution of a threat report. Other types of work might include validation or annotation of reports to assist in automatic classification, and generation of mitigation actions for vulnerabilities that the adversary tries to exploit.

Finally, a *tree* asset serves to facilitate anonymous authentication and manage the human work by controlling the input, incentivizing the output, and anonymizing the submittal of software artifacts.

5 Non-attributable Token Authentication

For the CTI distributed ledger to function, we must provide its users with anonymization guarantees. We now present the approach for anonymous authentication using a Merkle Tree for zero-knowledge commitment. To start, we present the process of token commitment. Then we show an approach of splitting the tree to support more authentication features such as revocation and value-based spending.

5.1 Anonymous Token Spending

A user receives a token upon the chain code validating some threat intelligence work, or perhaps as part of some bootstrapping process where new users have a limited set of tokens. A user will provide a token to the ledger when performing work for the chaincode to later validate. Then, once the chaincode validates the token, commitment occurs by adding the token as a leaf to the Merkle tree, `tree`.

The user arbitrarily samples a secret key through the security parameter, λ representing the key length and pseudorandom function $\texttt{Gen}(1^\lambda)$. A user may safely use the secret key repeatedly as a witness to multiple tokens. For each new token, a user arbitrarily generates a serial number, `sn`, in the same way. Then, using a collision-resistant hash function, $\texttt{CRH} = (0,1)^* \rightarrow 0, 1^\lambda$, the token is generated as shown in the following functions.

$$
\begin{array}{ll}
1: & \texttt{sk} \leftarrow \texttt{Gen}(1^\lambda) \\
2: & \texttt{sn} \leftarrow \texttt{Gen}(1^\lambda) \\
3: & \texttt{tk} \leftarrow \texttt{CRH}(\texttt{sk} \parallel \texttt{sn})
\end{array}
$$

The sparse Merkle tree then gets calculated with the inclusion of the token as $(\texttt{rt}, \texttt{path})$. The user then has the following public and private data related to the token.

$$
\begin{array}{ll}
1: & \texttt{tk}_{\texttt{pub}} \leftarrow (\texttt{rt}, \texttt{sn}) \\
2: & \texttt{tk}_{\texttt{pri}} \leftarrow (\texttt{sk}, \texttt{path})
\end{array}
$$

Algorithm 1 shows the zk-SNARK circuit for proof and verification. To generate a zk-SNARK proof, a user supplies the public parameters, `pp`, which includes the common reference string for proving and the zk-SNARK circuit. Public input

Algorithm 1 Token Verifier Circuit

Public Parameters: pp
Public Input: rt, sn
Witness: sk, path
Output: π - proof of inclusion

procedure TOKEN_VERIFIER
    tk ← CRH(sk ‖ sn)
    $\texttt{rt}_{\texttt{tk}}$ ← the smt calculation using tk and path
    if rt = $\texttt{rt}_{\texttt{tk}}$ **then**
        return true
    else
        return false
    end if
end procedure

includes both the Merkle root, rt, demonstrating knowledge of a valid token, and the serial number, sn, formed through the witness. The witness includes the secret key, sk, and the path down the tree to the token.

The chaincode on the distributed ledger verifies the proof represented in Algorithm 2. Here, the public parameters, pp, include the portion of the common reference string used for verification in the ledger. The verification includes (i) checking to ensure the zero-knowledge proof is valid, (ii) verifying the Merkle Root is a valid root for the ledger, and (iii) the serial number represents an unspent coin. The first check uses the zk-SNARK for the network. For the second check, the ledger must include a set of valid roots, and we describe this process in Sect. 5.2. The final check on whether sn exists in SNList prevents a double spend.

Algorithm 2 Verify Token Proof

Public Parameters: pp
Input: π, rt, sn
Output: Valid or Invalid

procedure VERIFY_PROOF
    $valid \leftarrow verify(\texttt{pp}, \pi, \texttt{rt}, \texttt{sn})$ ▷ zkSNARK verification
    if $valid \wedge \texttt{rt} \in \texttt{RTList} \wedge \texttt{sn} \notin \texttt{SNList}$ **then**
        return Valid
    else
        return Invalid
    end if
end procedure

5.2 Merkle Tree Structure and Root Updates

In the token spending scheme described above, a root update when inserting a batch of new tokens to the tree would make token spending attribution trivial. An entity would only need to search the ledger for the root associated with a token proof to identify the user.

To prevent this attack, we designate an entity to perform the service of sending out root updates at a time interval, t_{new}. Then validation should only include roots published within some time interval, t_{expiry}. Thus, a user wishing to spend a token must wait within a timespan of t_{new} after receiving the validation. Also, a token proof will be valid within a timespan of t_{expiry} from the proof construction. The expiration prevents token attribution because the prover supplies only recent token roots instead of the root calculated at token insertion.

The problem then becomes regularly distributing the tree paths to the network, which we now address. A Merkle tree in a DAP scheme may have token leaves distributed in any order. The location of the leaf in the tree has no association with the identity of the token owner. However, permissioned blockchains have inherent organizational structures, which the ledger can use for more robust authentication features and storage efficiency.

For a Merkle tree of height, h, the branch levels are split into three levels, h_{net}, h_{org}, and h_{user} as shown in Fig. 4. Thus, the tree supports $2^{h_{net}}$ organizations and each organization may have $2^{h_{org}}$ users.

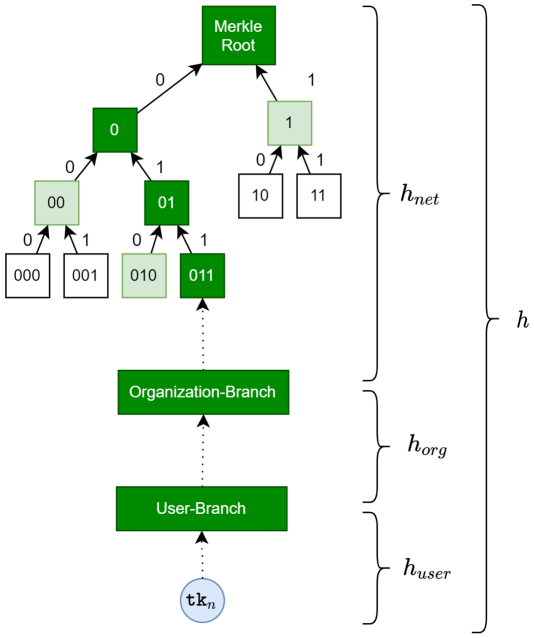

Fig. 4. Merkle tree structure.

By dividing the tree height, we minimize the size of tree updates and storage requirements to only those necessary for the entity's role in the network. As an example, a tree with a height of 32 requires approximately 256 GiB of storage. Also, to keep the siblings, `path`.S, up to date, the network must distribute a similar-sized update. However, using a permissioned blockchain's organizational structure and setting the h_{net} level at 14, the network updates only require 1 MiB while allowing for 2^{14} organizations.

Each organization is responsible for maintaining its similarly sized sub-tree to distribute `path`.S updates to its users.

The organizational tree structure supports several other services, which we now describe.

5.3 Revocation of Anonymous Authentication Tokens

Any network peer entity or organization may wish to revoke tokens as users leave, tokens become compromised, or users abuse the network. Since the tree divides into sub-trees of organizations and users, such revocation becomes trivial. User tokens are revoked by setting the desired token sub-tree to null and recalculating the root. Similarly, the network could revoke entire organizations by setting the organization sub-tree to null.

The revocation scheme works because tokens are not anonymous, and the Merkle tree does not need to hide the token holders' identities. A token spend only reveals the serial number, which cannot associate with the token. The network may safely maintain an identity on the token tree while preserving non-attribution in token spending.

The revocation latency ties to the t_{expiry} time interval associated with root updates. Attempts to authenticate using a revoked token will guarantee to fail after t_{expiry} because the proof of token inclusion no longer works with the new root.

5.4 Adding Value to Tokens

In a cryptocurrency, value is an attribute of the coin itself, and spend operations *pour* an old set of coins into a new set with the same value preserved. However, pouring coin value creates problems in the proposed scheme because the primary purpose of the token is for non-attributable authentication, and supporting a large number of token spends adds unnecessary complexity.

Instead, we propose tokens only have a value of 1, and we increase the user Merkle tree height to support a large number of tokens. Only the user only needs to maintain the path siblings for any levels below h_{org}. Knowing these paths allows the user to construct a valid proof without the network or organization having to maintain a tree height to support a large number of possible tokens.

For example, if the network maintained a tree height of 14 at 1 MiB, and each organization maintained a sub-tree height of 18 at 8 MiB, each user could maintain their sparse sub-tree of 64 levels to support a vast number of tokens far beyond the maximum necessary.

5.5 Authentication Without Spending

Finally, by maintaining tokens with revocation services, they provide a useful means of anonymous authentication. There are several scenarios where users might desire anonymity. A user may wish to perform an anonymous search on the network, e.g., searching for a particular IP address. Performing such a search could infer the organization's attribution as a victim of the malware.

To support anonymous authentication only, we make a minor modification to the token spending circuit and remove the serial number as the public verification parameter. Additionally, we hash the timestamp, ts, with the root to prevent replay attacks.

Algorithm 3 Token Authentication Circuit

Public Input: $\texttt{pp}, \texttt{rt}, ts$
Witness: $\texttt{tk}, \texttt{path}$
Output: Whether the calculated root matches the given root

1: **procedure** TOKEN_AUTH
2: $\texttt{rt}_{\texttt{tk}} \leftarrow$ the `smt` calculation using `tk` and `path`
3: **if** $\texttt{CRH}(\texttt{rt} \parallel ts) = \texttt{CRH}(\texttt{rt}_{\texttt{tk}} \parallel ts)$ **then**
4: **return** true
5: **else**
6: **return** false
7: **end if**
8: **end procedure**

6 Chaincode for CTI Work

This section presents in detail the state program model used for managing work on the network. Several peer-authenticated transactions occur to update threat reports, which this paper does not formalize. The transactional updates to threat reports are essential but straightforward. Instead, we focus on the *Work* asset transactions to facilitate the expansion of threat knowledge and automation beyond existing services. Recall that a *Work* asset consists of problem and solution data structure which maps to an *Event Report* asset.

Work asset transactions focus both on the problem of submitting CTI anonymously and on validating the quality of the CTI. The cybersecurity community has not extensively considered the use of non-attributable CTI, and the chaincode recognizes this by including a set of evaluation states.

Figure 5 shows a state transition diagram of the workflow from the addition of *Work* to the completion of a solution. Each state transition represents a chaincode function made available to the network for processing the ledger. The ledger must maintain state to support asynchronous processing of transactions and high assurance in the logic of the chaincode.

The object variables for the state program include the following chaincode assets:

$$Var = \{\texttt{event\_record}, \texttt{work}, \texttt{token\_tree}\} \tag{1}$$

A `threat_record` asset includes the complex data structure represented in Fig. 3 and described in Sect. 4.2. Assets for `work` have a problem/solution data structure that stores the proposed problem and maintains a set of proposed solutions for evaluation. The tree asset supports the use of tokens described in Sect. 5.

The program graph over Var is defined as

Definition 1. *State Transaction Program Graph*

- S - *Set of states*
- $Effect : Act \times Eval(Var) \rightarrow Eval(Var)$ - *Transition effect function.*
- $R \subseteq S \times Cond(Var) \times Act \times S$ - *Conditional transition relation*
- $S_0 \subseteq S$ - *The set of initial states*
- $g_0 \in Cond(Var)$ - *The initial condition*

The function $Eval$ comprises the set of evaluations over Var, and the function $Cond$ comprises the set of conditional expressions over Var.

Work state is maintained through the smart contract logic. Valid work states include $S =$ `READY_WORK`, `IN_PROGRESS`, `READY_EVAL`, `IN_EVAL`, `ADD_WORK`.

Anyone with a valid token may submit a `work` record to the network accompanied with a token proof. The chaincode first evaluates the token proof as a guard condition for the work queue. In this way, the `work` has no attribution to an entity, but the entity authenticates as a valid user of the network. Also, the ledger preserves the quality of the work queue by requiring an entity to give up something of value in exchange for work performed.

Each `work` asset gets added to a priority queue on the ledger. The priority queue operates based on priority and time to differentiate work value and prevent starvation for lower priority work requests. Workers should also choose work based on their resources and capabilities, but we leave the optimal dequeuing of work to future research.

Finally, the ledger adds an evaluated solution by i) updating the `event_record` with the added context provided through the work solution, ii) inserting the tokens provided with the work solution and evaluation, and iii) publishing a new root to the network based on the updated tokens.

The entity requesting work will likely search for the work solution periodically. Thus, the network supports authentication-only proofs using tokens to preserve the anonymity of the work requester. An entity need not authenticate with an identity, save only to perform work.

Due to the space limitation, an extended version of this paper will include the algorithms for chaincode.

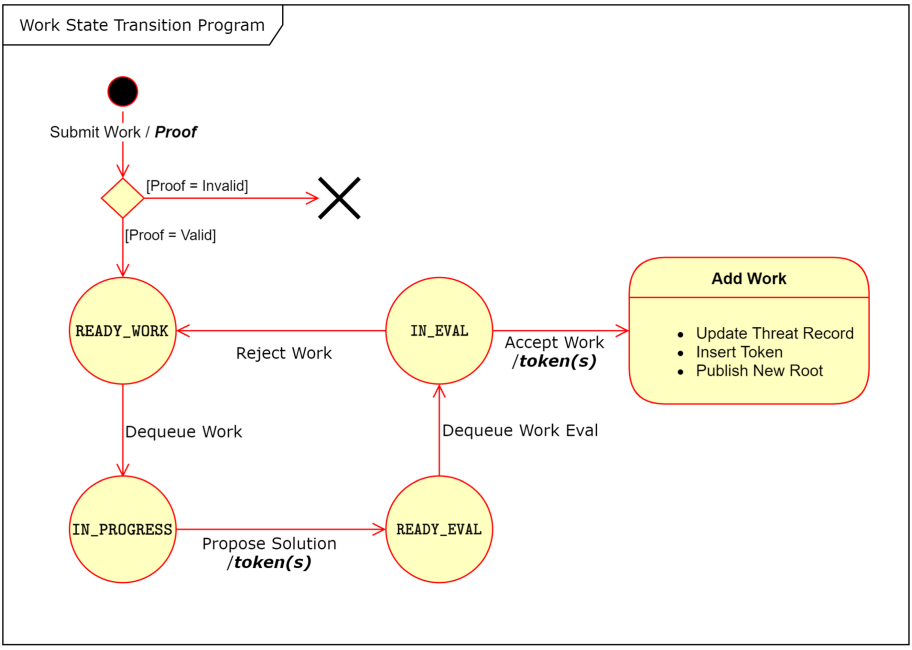

Fig. 5. Work state transition.

7 Implementation

We performed testing to evaluate the Merkle tree maintenance from Sect. 5.2 and token authentication in Sect. 5. Also, we propose implementation guidelines for implementing the blockchain under realistic loading conditions. Our tests of the zk-SNARK proofs use snarkjs and circomlib, and the performance was tested on an Intel Core i5-8356U CPU @1.60 GHz with 16 GB of RAM.

7.1 Token Authentication Performance

The Merkle tree height drives the network performance for token authentication in both storage and time. Authentication allows sparse tree storage at both the organizational, h_{org}, and user levels, h_{user}. However, the network must provide frequent updates at the network level, h_{net}, to support anonymous authentication. Due to the frequency of these updates, we propose setting h_{org} at 15, which for a 256-bit node size, requires 1 MiB of storage.

Users can manage a much deeper portion of the Merkle tree because they only store the sparse tree based on the number of tokens they possess, but the token proof circuit requires a consistent depth. Figure 6 shows the relationship between the depth and proof times. Here, we propose a reasonable tree depth of, at most, 128, which provides ample space for both the foreseeable maximum number of organizational users and the number of tokens allocated for each user.

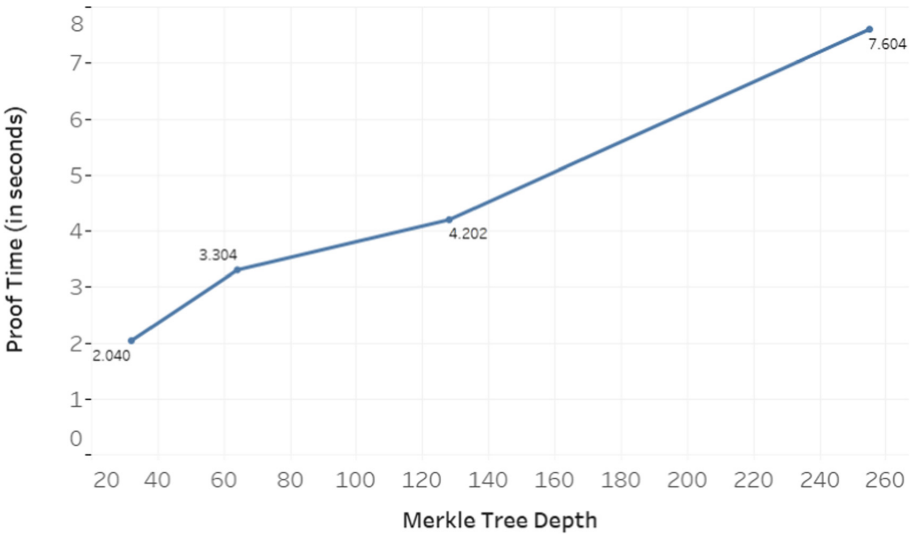

Fig. 6. Proof times relative to Merkle tree height.

The parameters and performance of the algebraic circuit for a tree with this size are shown in Table 1.

7.2 Ledger Operation Guidelines

We developed the chaincode model for use in Hyperledger Fabric, and although a full-scale simulation is in development, we make some observations here about the operation of the network.

There are three types of transactions proposed: i) event reports, ii) work management, and iii) network maintenance activities, including Merkle root updates. To develop a realistic expectation of throughput, we consider the critical infrastructure sectors in the United States. The Department of Homeland Security identifies sixteen critical infrastructure sectors [2]. Using utility data from the Energy Information Administration [1], we find 3,338 individual utility companies in the electric sector. If each organization produced an average of ten transactions per day during a peak of four working hours, we could expect a maximum throughput of 40 transactions per second.

For this level of throughput, Hyperledger Fabric benchmark experiments indicate a latency of approximately 1 s with a block size of 10 transactions [36]. They also indicate that an endorsement policy of up to four network peers for each transaction would have a minimal effect on the overall transaction latency. Overall, the system's theoretical bounds would fall well within the efficient operating conditions of Hyperledger Fabric.

Table 1. Sparse Merkle tree proof circuit parameters and performance

Merkle tree height	128
Number of wires:	32,486
Number of constraints:	32,363
Private inputs:	130
Public inputs:	2
Number of labels:	151,304
Number of outputs:	0
Proof time:	4,200 ms
Verification time:	28.5 ms

8 Conclusion

This paper proposes a new approach for overcoming the trust barriers of inter-organizational threat intelligence sharing using a distributed ledger technology. We have demonstrated a novel use of zk-SNARKs and Sparse Merkle Trees to enable anonymous authentication and anonymous token spending for the ledger's permissioned users. The results pave the way for a new approach to cybersecurity threat intelligence sharing, which commoditizes the work of CTI curation and sharing to produce a greater cooperative value.

Acknowledgement. This work is supported in part by NSF under award number 1751255. This material is also based upon work supported by the Department of Energy under Award Number DE-OE0000779.

References

1. Annual electric power industry report. https://www.eia.gov/electricity/data/eia861/. Accessed 12 Mar 2021
2. Critical infrastructure sectors. https://www.cisa.gov/critical-infrastructure-sectors. Accessed 12 Mar 2021
3. Cyber risk information sharing program (crisp). Tech. rep., Department of Energy, Office of Cybersecurity, Energy Security, and Emergency Response. Accessed 13 Jan 2021
4. Enhanced cybersecurity services (ecs). Tech. rep., Department of Homeland Security. Accessed 13 Jan 2021
5. Hackerone list of bug bounty programs. https://hackerone.com/bug-bounty-programs. Accessed 11 Mar 2021
6. A Common Cyber Threat Framework: A Foundation for Communication (2013)
7. The value of threat intelligence: A study of North American and United Kingdom companies. Tech. rep., Ponemon Institute (July 2016). Accessed 06 Jan 2021
8. Exploring the opportunities and limitations of current threat intelligence platforms. Tech. rep., ENISA (December 2017). Accessed 06 Jan 2021

9. Global security operations center market forecast up to 2025. Business Wire (English) (2019)
10. Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, pp. 1–15 (2018)
11. Banerjee, A., Clear, M., Tewari, H.: Demystifying the role of zk-SNARKs in Zcash. In: 2020 IEEE Conference on Application, Information and Network Security (AINS), pp. 12–19. IEEE (2020)
12. Barnum, S.: Information with the structured threat information expression (STIX) (2013)
13. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_18
14. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. In: Zohar, A., et al. (eds.) FC 2018. LNCS, vol. 10958, pp. 64–77. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_5
15. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 132–145 (2004)
16. Camenisch, J., Chen, L., Drijvers, M., Lehmann, A., Novick, D., Urian, R.: One TPM to bind them all: fixing TPM 2.0 for provably secure anonymous attestation. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 901–920 (2017)
17. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong Diffie Hellman assumption revisited. In: Franz, M., Papadimitratos, P. (eds.) Trust 2016. LNCS, vol. 9824, pp. 1–20. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45572-3_1
18. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_18
19. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
20. Danezis, G., Fournet, C., Kohlweiss, M., Parno, B.: Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In: Proceedings of the First ACM Workshop on Language Support for Privacy-Enhancing Technologies, pp. 27–30 (2013)
21. Daniel, M., Kenway, J.: Repairing the foundation: how cyber threat information sharing can live up to its promise and implications for NATO. Cyber Threats and NATO 2030: Horizon Scanning and Analysis, p. 178 (2020)
22. Douris, C.: Cyber Threat Data Sharing Needs Refinement. Lexington Institute Arlington, Virginia (2017)
23. Gong, S., Lee, C.: Blocis: blockchain-based cyber threat intelligence sharing framework for sybil-resistance. Electronics **9**(3), 521 (2020)
24. He, S., Fu, J., Jiang, W., Cheng, Y., Chen, J., Guo, Z.: Blotisrt: blockchain-based threat intelligence sharing and rating technology. In: Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies, pp. 524–534 (2020)
25. Office of Inspector General: DHS Made Limited Progress to Improve Information Sharing under the Cybersecurity Act in Calendar Years 2017 and 2018 (2020)

26. Johnson, C., Badger, M., Waltermire, D., Snyder, J., Skorupka, C.: Guide to cyber threat information sharing. Tech. rep., National Institute of Standards and Technology (2016)
27. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 839–858. IEEE (2016)
28. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H., Adams, C. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46513-8_14
29. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy, pp. 397–411. IEEE (2013)
30. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Tech. rep., Manubot (2019)
31. Petkus, M.: Why and how zk-SNARK works: definitive explanation
32. Riesco, R., Larriva-Novo, X., Villagra, V.A.: Cybersecurity threat intelligence knowledge exchange based on blockchain. Telecommun. Syst. **73**(2), 259–288 (2019). https://doi.org/10.1007/s11235-019-00613-4
33. Sasson, E.B., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474. IEEE (2014)
34. Shen, M., Duan, J., Zhu, L., Zhang, J., Du, X., Guizani, M.: Blockchain-based incentives for secure and collaborative data sharing in multiple clouds. IEEE J. Sel. Areas Commun. **38**(6), 1229–1241 (2020)
35. Stillions, R.: The DML Model (2014)
36. Thakkar, P., Nathan, S., Viswanathan, B.: Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 264–276. IEEE (2018)
37. Wagner, C., Dulaunoy, A., Wagener, G., Iklody, A.: MISP: the design and implementation of a collaborative threat intelligence sharing platform. In: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security, pp. 49–56 (2016)
38. Wood, G., et al.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Proj. Yellow Pap. **151**(2014), 1–32 (2014)
39. Zetter, K.: Exclusive: comedy of errors led to false 'water-pump hack' report. Wired Threat Level (2011)
40. Zibak, A., Simpson, A.: Cyber threat information sharing: perceived benefits and barriers. In: Proceedings of the 14th International Conference on Availability, Reliability and Security, pp. 1–9 (2019)