Hindawi
Security and Communication Networks
Volume 2020, Article ID 8833765, 15 pages
https://doi.org/10.1155/2020/8833765

Research Article

Distributed Security Framework for Reliable Threat Intelligence Sharing

Davy Preuveneers,[1] **Wouter Joosen**,[1] **Jorge Bernal Bernabe**,[2] **and Antonio Skarmeta**[2]

[1]*imec-DistriNet, KU Leuven, Leuven, Belgium*
[2]*University of Murcia (UMU), Murcia, Spain*

Correspondence should be addressed to Davy Preuveneers; davy.preuveneers@cs.kuleuven.be

Received 21 March 2020; Revised 2 June 2020; Accepted 9 June 2020; Published 1 August 2020

Academic Editor: Neetesh Saxena

Computer security incident response teams typically rely on threat intelligence platforms for information about sightings of cyber threat events and indicators of compromise. Other security building blocks, such as Network Intrusion Detection Systems, can leverage the information to prevent malicious adversaries from spreading malware across critical infrastructures. The effectiveness of threat intelligence platforms heavily depends on the willingness to share among organizations and the responsible use of sensitive information that may potentially harm the reputation of the reporting organization. The challenge that we address is the lack of trust in the source providing the threat intelligence and the information itself. We enhance our security framework TATIS—offering fine-grained protection for threat intelligence platform APIs—with distributed ledger capabilities to enable reliable and trustworthy threat intelligence sharing with the ability to audit the provenance of threat intelligence. We have implemented and evaluated the feasibility of our distributed framework on top of the Malware Information Sharing Platform (MISP) solution, and we evaluate the performance impact using real-world open-source threat intelligence feeds.

1. Introduction

Advances in distributed computing, storage and communication technologies, and the growing amounts of data, as well as the adoption of new business models, have been major drivers behind key computing paradigms, such as cloud and mobile edge computing, the Internet of Things (IoT) and Cyber Physical Systems (CPS), Big Data analytics and artificial intelligence (AI), and 5G. The downside of these emerging trends is that interconnected ICT systems increase the attack surface of critical infrastructures [1], and they have therefore become a valuable target for malicious adversaries and cyber criminals that aim to disrupt services to exfiltrate sensitive data or to abuse the victim machines and networks for performing other malicious activities. Indeed, the accelerating wave of sophisticated attacks and advanced persistent threats is a growing security and privacy concern for both the consumer and businesses. In its 2019 Data Breach Investigations Report [2], Verizon acknowledged that ransomware remains among the most popular and pervasive malware variants.

As the impact of a breach is not always isolated to within the trust boundaries of a single organization or country, it is crucial that organizations not only develop perimeter defenses to keep attackers out but also invest in cyber threat intelligence inside that perimeter to monitor corporate networks so that they (1) can detect how and when a breach occurs, (2) are able to identify compromised systems, (3) can determine how adversaries modified their systems or identify which data was stolen, (4) contain the incident from further contamination, and (5) are able to remediate these incidents and recover from the breach. To prevent the same incident from happening elsewhere, enabling technologies are needed to manage digital evidence, i.e., techniques to collect, examine, analyze, and possibly share digital evidence originating from a variety of digital data sources. Nonetheless, there are limitations with contemporary solutions to adequately address and mitigate cyber threats in a timely manner.

The amount, sophistication, and impact of cyber attacks are increasing and so are the amount of protected end-points and the volumes of data coming from these end-points that need to be scrutinized for anomalous behavior. To enhance detection and prevention of cyber threats, organizations collaborate to define defensive actions against complex attack vectors by sharing information and knowledge about threats, sightings, indicators of compromise (IoC), and mitigation strategies. Threat intelligence platforms (TIP) have therefore become a critical security component within the enterprise to deal with the increasing volume and sophistication of cyber attacks [3]. These software platforms are cloud or on-premise systems that facilitate the aggregation and correlation of threat events from different parties and multiple sources [4], including security monitoring and data analytics tools.

Beyond the technical challenges to realize interoperability, trust remains a key challenge in collective cyber threat intelligence (CTI), especially when the incident data collected and shared in TIPs may harm the reputation of a reporting organization or when the data is used to automatically trigger cyber security solutions. Reporting organizations need to trust the receiving organizations that the information is handled confidentially. At the same time the receiving organizations assume that the information itself is reliable and trustworthy. Obviously, malicious or unreliable input information may compromise the usefulness of such Host Intrusion Detection Systems (HIDS) or Network Intrusion Detection Systems (NIDS) or may even harm an innocent targeted organization.

Despite recent advances [5, 6] to strengthen security and privacy in cyber threat intelligence sharing, current solutions lack proper distributed security and trust mechanisms that can holistically manage access control and confidential data sharing among trusted entities, while ensuring auditability and provenance of the shared CTI data.

To fill this gap, this paper leverages the approach proposed by Preuveneers and Joosen [7] for their system called TATIS: "Trustworthy APIs for Threat Intelligence Sharing with UMA and CP-ABE." TATIS aims to protect access to sensitive information and the way it is shared with other threat intelligence platforms. Namely, this paper leverages and enhances TATIS by making it fully distributed and supporting federated authentication and authorization across different domains. In addition, to make the framework fully reliable and auditable, this paper improves our previous work by integrating distributed ledger technology (DLT) into the system. This way, we achieve verifiability and data provenance of the exchanged CTI events, while enforcing fine-grained access control to the system and protecting the shared data by encrypting it following an approach based on the CP-ABE cryptographic scheme.

Our solution has been fully implemented in a distributed scenario on top of the Malware Information Sharing Platform (MISP) and tested using real open-source CTI data, showing its feasibility and performance to control access to CTI data and share encrypted CTI data across distributed domains as well as accounting and managing CTI data provenance to reinforce trustworthiness in CTI sharing.

The structure of the remainder of the paper is as follows. In Section 2, we present relevant related work on threat intelligence platforms and the use of blockchain technologies. Section 3 describes the high-level architecture design and implementation of our solution. The provenance and verifiability of the CTI sharing process are discussed in Section 4. In Section 5, we evaluate the practical feasibility of our distributed security framework and measure the performance impact using real-world open-source threat intelligence feeds. We conclude the paper in Section 6, summarizing the main insights and identifying opportunities for further research.

2. Related Work

There are multiple recent surveys [3, 8] that review the current state of the art on cyber threat intelligence (CTI) sharing, defining associated benefits and barriers [9]. These works highlight that security, trustworthiness, provenance, and privacy issues are still open research challenges in CTI sharing, in the sense that they have not been holistically addressed in the literature.

Likewise, as surveyed by Kaloudi et al. [10], solutions are emerging for cyber threat intelligence handling which rely on artificial intelligence (AI) techniques and statistical methods. These methods are used to analyze shared cyber threat intelligence (CTI) data in real time and to infer actionable cybersecurity actions while considering the cyber situational context of the shared and trusted CTI data. In this sense, Mittal et al. [11] present an AI-based decision support system for knowledge extraction, representation, and analytics of the cybersecurity informatics domain. Similarly, Liao et al. [12] propose an automatic discovery and analysis of cyber threat intelligence using graph mining techniques. In [13], Khuruna et al. apply a semisupervised learning algorithm to estimate reputation scores to ensure the validity of the information ingested by security analysts. All these contributions are indicative of the growing adoption of machine learning (ML) in the cyber threat intelligence workflow, i.e., the data collection, the security and cyber threat analysis, and the threat intelligence sharing. While the previous works covered the first two phases, our work contributes to enhancing the third phase with a trustworthy sharing phase of the CTI and this is independent of the AI methods used to elicit the threat intelligence information.

Sauerwein et al. [14] analyze the current CTI sharing platforms including MISP [15]. In the analysis, they pointed out several shortcomings in actual platforms, such as lack of trust, issues related to automation in CTI sharing, and interoperability issues. Indicators of compromise (IoCs) might contain personal data, and, therefore, unless proper confidentiality and privacy measures are implemented, the organizations will be reluctant to share CTI data.

Trust mechanisms have been recently proposed to increase the security and reliability in distributed systems. In [16], Yan et al. propose a flexible trust-based access control mechanism for cloud computing which considers user trust evaluations and reputation scores from data centers applying Attribute-Based Encryption and Proxy Reencryption

mechanisms. Meng et al. [17] propose a trust-based mechanism based on Bayesian inference to detect malicious insider attacks in healthcare systems. While none of these works focus on cyber threat intelligence systems, they do reflect the need for trustworthiness as a first-class citizen in any collaborative information sharing solution.

Van De Kamp et al. [6] propose a simple method to encrypt IoCs and share them in a privacy-preserving way. They employ a nonsecret salt, chosen at random for each IoC, to compute the hash for each IoC. Their approach hides the details of an indicator or sighting while still enabling sharing, limiting the possibility of information misuse. However, they do not tackle access control aspects along with the obfuscation of IoCs. Our work goes beyond the state of the art providing not only confidentiality but also access control, reliability, and auditability in CTI sharing, thereby empowering users with fine-grained access control mechanisms that allow selective and privacy-preserving CTI data sharing, while guaranteeing provenance of exchanged data and hereby increasing overall trust.

In [18], the authors analyze CTI sharing and the legal implications under the General Data Protection Regulation (GDPR), when personal data attributes are disclosed. GDPR mandates the confidentiality of personal information in CTI sharing. Our distributed security solution encrypts shared events and associated attributes with the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme, so that only those entities holding attributes that match the attribute policy will be able to decrypt and access the CTI shared data.

As highlighted in [19], the privacy-preservation problem is exacerbated in ledgers due to their immutability nature, and this hampers the right to be forgotten. Thus, to cope with privacy issues in the ledger, personal attribute information should not be added in the clear. Our solution hashes personal attributes so that they are not recorded in plaintext on the ledger.

In [20], the authors evaluate current standards, ontologies, and taxonomies for CTI sharing. Wagner et al. [21] define a trust taxonomy of shared cyber threat intelligence, analyzing 30 threat intelligence platforms and providers according to their trust functionalities. Most solutions do not allow direct external connection or allow this with manual trust establishment among organizations. MISP allows establishing trust but does not allow selective encryption of attributes. To address this limitation, our security solution leverages MISP to return the control over the sharing of threat intelligence back to the users, hereby increasing trust in their CTI sharing. Our security framework has been designed to be agnostic of the CTI platform, the identity management system employed to authenticate users, and the ledger used to audit the exchanged events.

Homan et al. [22] propose a CTI sharing model based on the Hyperledger Fabric blockchain and the STIX 2.0 specification, aimed to overcome the potential trust barriers in this domain. Among others, they use the channel capabilities of Hyperledger Fabric to establish trusted communities to disseminate sensitive data, and they use smart contracts to enforce the Traffic Light Protocol (TLP) based sharing rules within the network.

To address challenges related to CTI automation, various research solutions are being proposed to dynamically manage cyber threat intelligence data. In this sense, Graf and King [23] employ a neural network autoencoder, supported by blockchain technology, for the classification and management of shared CTI information. Blockchain smart contracts are employed for the life-cycle management, supporting the acquisition, usage, and archival disposal of incidents. The autoencoder relies on natural language processing to compress as much information as possible from a threat event into a compact representation, i.e., a vector holding 10 numbers. The authors use this vector to compare incident documents in a fast and scalable way.

Amthor et al. [24] propose a framework that automatically applies security policies to react to shared threat alerts. Likewise, in [25], Kim et al. propose a CTI management framework that supports CTI data aggregation, analysis, and sharing. This solution is able to generate in real time security rules according to the shared CTI data. The security rules are automatically enforced by IDS/IPS to counter any discovered threats. Nonetheless, these proposals do not offer security and trust mechanisms for CTI sharing as proposed herein.

There are previous works that employ CP-ABE to increase the confidentiality in CTI sharing. In [5], Vakilinia et al. extend the CybOX objects for the representation of CP-ABE and to include them in STIX messages. They extend rather than create entirely new CybOX objects. Besides, this work, unlike ours, does not support data provenance of shared threat events. Like in our work, in [26], the authors propose combining both CP-ABE encryption for confidential CTI data sharing and blockchain to address nonrepudiation issues of exchanged data. However, their proposal requires a centralized server to store the encrypted data and a centralized trusted entity in charge of managing all the keys. Our approach is fully distributed as key management is handled in a distributed way by organizations sharing the data in a P2P manner, with federated authentication and access control support. Recently, Badsha et al. [27] propose a privacy-preserving mechanism and protocol that employs homomorphic encryption to share aggregated CTI information (decision trees) across organizations through a central server that performs heavy homomorphic operations. The organizations are then able to learn the decision tree to make predictions or classifications on threat information, without disclosing private information.

3. Distributed Threat Intelligence Architecture

In this section, we will describe the design and implementation of our distributed security framework for collective cyber threat intelligence and how we build upon existing technologies, such as the MISP threat intelligence platform and our previous work TATIS for enhanced API protection.

3.1. Threat Intelligence Sharing with MISP. Wagner et al. [15] presented MISP (https://www.misp-project.org/), an open-source threat intelligence sharing platform, initially focusing

on malware information, but now it is also used for other threat vectors, such as financial indicators for fraud detection and prevention.

MISP, as depicted in Figure 1, operates on events and attributes. It allows creating and receiving IoCs, as events defined through an object and a set of associated attributes. Every event has a distribution scope (e.g., community only), an identifier, a date, and the link to the organization creating and/or sharing the event. In addition to attributes, an event can have different tags, which can be created from collections of taxonomies or be self-created. MISP allows defining and importing galaxies or clusters that can be attached to events or attributes. They express more information than just simple tags. For instance, as shown in Figure 1, the event with Id 28 includes an expansion with an attack intrusion set belonging to the APT (threat group) APT28 (a.k.a. Sofacy), encompassing more than 1520 attributes related to the CTI data.

Events typically encapsulate tags to link events with one another (e.g., the Traffic Light Protocol (https://www.us-cert.gov/tlp) or TLP labels to share with appropriate audiences), objects from other information sharing tools, and attributes with various system or network related indicators. The *category* of an attribute puts it in a certain context, whereas the *type* of an attribute describes the indicator. Given the widespread use of the technology, we use MISP as the base TIP for our experimentation. The enhancements we will present in the following sections do not modify MISP directly but access the platform via its REST APIs. This way, our solution can be easily repurposed for other TIPs.

3.2. Basic Design Principles and Capabilities of TATIS. This work builds upon our previous work, TATIS [7], which aims to offer (1) fine-grained authorization to individual events and attributes via REST APIs and (2) encrypted storage of the information. For enhanced authorization to the REST APIs, TATIS leverages the User Managed Access (UMA) protocol and policy-based access control. The stored threat events and attributes are protected with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to enable fine-grained access to authorized event producers and consumers.

TATIS leverages MISP as the threat intelligence sharing platform for storing and sharing information about threat events and indicators of compromise (IoC) with other organizations. We developed TATIS as a *TIP Reverse Proxy* for MISP, as depicted in Figure 2, such that (1) it acts as an API gateway to the actual TIP instance (i.e., MISP) of the organization and (2) it can be easily adapted to other types of TIPs with a REST API.

UMA 2.0 is used as a standards-compliant authorization protocol built on top of the OAuth 2.0 standard, offering the event producers rather than the MISP platform provider the ability to manage access control. We leverage the implementation provided by RedHat's Keycloak 9.0 open-source identity and access management (IAM) platform (https://www.keycloak.org/). The definition of the authorization policies is beyond the scope of the UMA specification as the policy evaluation is done out-of-band. TATIS uses the Open Policy Agent (OPA) [28] as an attribute-based access policy language to enforce access control rules and conditions under which authorization is granted. As the authorization policy framework of Keycloak does not support this language out-of-the-box, a JavaScript policy from within Keycloak acts as a bridge to an externally deployed and managed OPA 0.17.3 policy engine. A simple OPA policy is depicted in Figure 3.

The second goal of TATIS is to protect the cyber threat events and IoCs stored within MISP's database against software vulnerabilities in the TIP and against *honest-but-curious* MISP platform hosts and infrastructure providers. Database encryption with a single key is typically not feasible for scenarios with multiple authorized event consumers and honest-but-curious hosting and infrastructure providers. Therefore, TATIS individually encrypts events and attributes so that the confidentiality is guaranteed with respect to unauthorized subjects, while allowing the event producer to grant access to event consumers based on his own authorization policies. The *TIP Reverse Proxy* component of TATIS solves this through Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [29], as depicted in Figure 4. First the attributes in the event are encrypted with an AES key resulting in a threat event ciphertext CT_{AES}. The AES key is encrypted with CP-ABE using an attribute policy provided by the event producer, which results in a ciphertext CT_{ABE} that protects the AES key. These attribute policies are a Boolean composition of attribute conditions in disjunctive normal form:

$$\text{role: admin} \vee (\text{subject: bob} \wedge \text{location: office}). \quad (1)$$

It implements the same conditions as those in Figure 3. The attribute policies we use in our experiments are more sophisticated. Their complexity varies from 3 to 5 disjunctions, with each of them having 1 up to 3 attribute conditions.

To decrypt the threat intelligence, an event consumer first obtains a private decryption key that is derived from his user attributes. This private key is first used to decrypt CT_{ABE} to obtain the AES key. Then the AES key is used to decrypt CT_{AES} to retrieve the event attributes in plaintext.

In practice, each TATIS instance has its own CP-ABE master secret key (MSK) and public key, and TATIS will compile the plaintext policy of the event producer into CT_{ABE} using its own MSK and a random AES key. To encrypt the attributes with AES, the event producer needs to retrieve the AES key from CT_{ABE} using its own CP-ABE private decryption key in a similar manner to that of the event consumer. TATIS will construct the CP-ABE private decryption keys for the event producer and consumer using its MSK and the attributes of each subject stored in Keycloak.

We refer interested readers to [7] for more details about the core functionality of the TATIS framework and the implications of using UMA and CP-ABE. In the next section, we will describe how we extended the core functionality with distributed ledger technology (DLT) to assure the provenance and trustworthiness of shared threat intelligence.

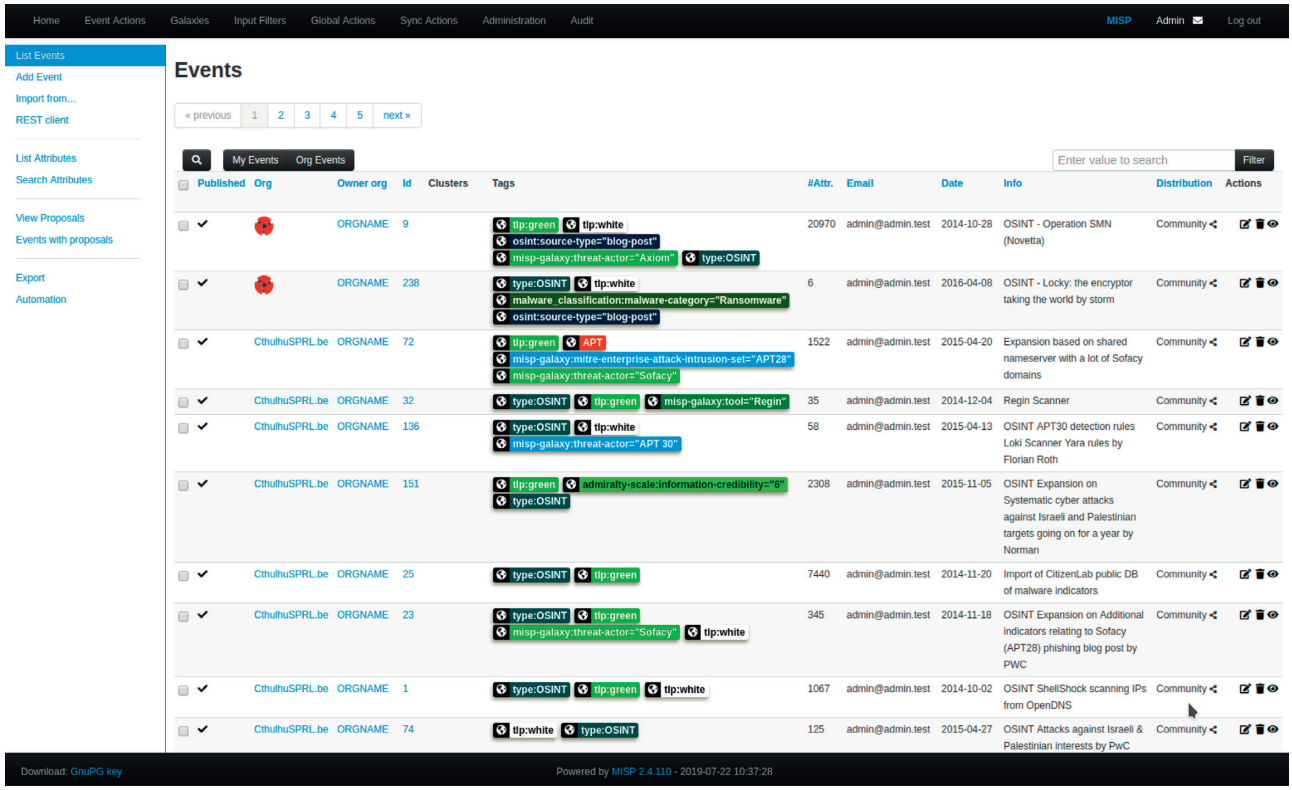

Figure 1: Malware Information Sharing Platform (MISP), an open-source threat intelligence sharing platform.

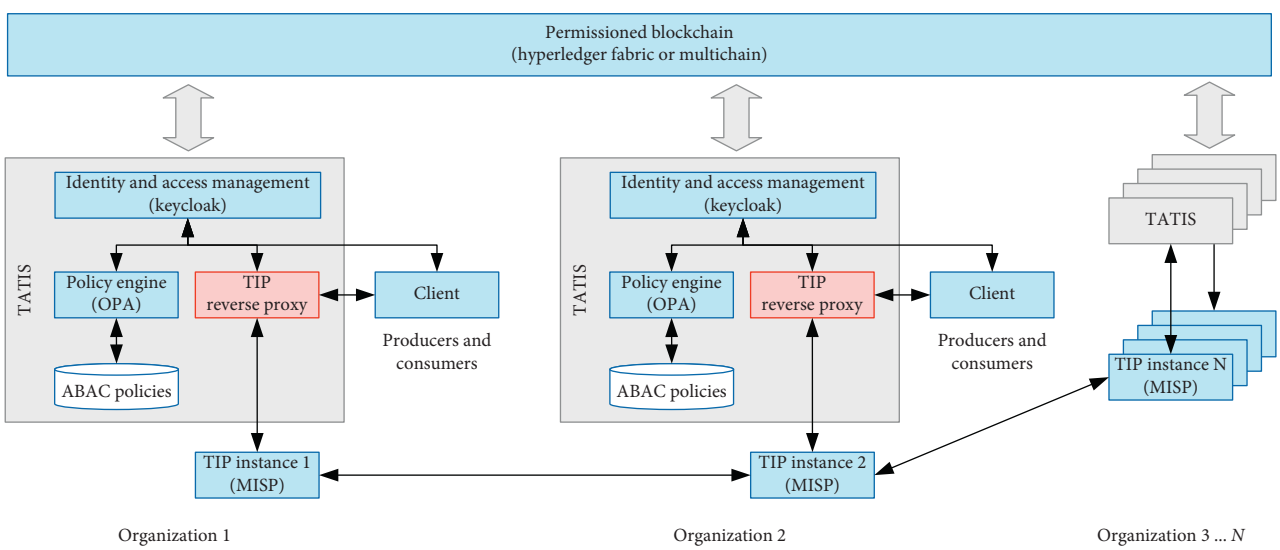

Figure 2: High-level deployment overview of the proposed distributed security framework.

3.3. Distributed Ledger for Reliable and Auditable Threat Intelligence. As the transactions and associated exchanged CTI data are duly recorded in the ledger, our solution provides verifiability, accountability, auditing, and data provenance in CTI sharing.

In case of a cyber security incident, when a producer exchanges the corresponding IoC, it is of paramount importance for third-party event consumers to ensure that shared data is reliable and can be verified at any time. Our DLT-based security CTI-sharing framework supports auditing the exchanged data, fostering confidence between involved parties, as it ensures that entities can get the historical transactions in the chain and check the provenance of exchanged indicators of compromise. In addition, it facilitates the validation of the trustworthiness and reliability of the shared content. Indeed, involved entities, with granted

```
package httpapi.authz.cs4e

default allow = false

allow {
    input.role[_] = "admin"
}

allow {
    input.subject = "bob"
    input.location = "office"
}
```

Figure 3: Example of an OPA policy allowing all admins and Bob to work at the office.

access rights (i.e., having the proper decryption cryptographic material linked to their user attributes), could have access to the historical exchanged data to quantify trustworthiness of both the IoC content and the producer entity.

The immutability properties featured by our framework empower involved entities with mechanisms to deal with liabilities that may be derived because of erroneous or forged shared CTI events, as the framework supports nonrepudiation in the authorship of a given shared CTI event.

Figure 2 depicts how multiple organizations host their own MISP instance and can push and pull threat intelligence from one MISP instance to another. The threat intelligence may be stored in the clear or be CP-ABE-encrypted by TATIS. All REST API requests by event producers and consumers to the MISP instance are assumed to pass through the TATIS Reverse Proxy, during which information about each MISP request is stored on a permissioned blockchain. Reusing our previous work [30], we use the open-source MultiChain 2.0.5 (https://www.multichain.com) blockchain that extends Bitcoin Core with abilities to manage permissions, assets, and streams. The main reason for choosing this enabling technology over other permissioned blockchain solutions, such as Hyperledger Fabric [31], is the lightweight nature and the simplicity with which a MultiChain distributed ledger can be deployed. When more advanced enterprise features are needed, Hyperledger Fabric or the Ethereum-based Quorum blockchain may prove to be more valuable.

3.4. Threat Model. The threat model that we adopt in this work is that of an honest but curious platform provider that hosts the threat intelligence platform (TIP) and malicious threat intelligence consumers.

If the TIP is offered as a service by an untrusted third party, it may have unauthorized access to sensitive information if the CTI is not encrypted. Furthermore, transparent encryption solutions for cloud data storage may not suffice as the key to encrypt and decrypt is available to the third party. As such, the third-party platform provider should not be able to gain access without approval of the threat intelligence provider or the owner of the information.

Even if malicious threat intelligence consumers are not granted access to certain CTI, they may be able to bypass rule-based access control mechanisms, for example, by exploiting vulnerabilities in the software stack of the TIP. This way, they may be able to exfiltrate unencrypted threat intelligence. That is why all data at rest should be encrypted such that no curious or malicious threat intelligence consumer can gain unauthorized access.

4. Reliable and Verifiable CTI Sharing Process

In this section, we will elaborate in more detail with concrete examples how access control to threat intelligence is managed and how the confidentiality of shared data is realized.

We will assume the distributed scenario of an event producer within `Organization 1` sharing threat intelligence that will be leveraged by an event consumer at `Organization 2`. For the sake of brevity, we will use the simplified MISP test event depicted in Figure 5 to illustrate the process. Furthermore, both organizations synchronize their MISP servers via so-called "sync users" so that encrypted threat events from `Organization 1` are replicated and are available in the MISP instance operated by `Organization 2`.

4.1. Confidentiality of Shared Threat Events. We will now elaborate on the different steps involved in sharing a MISP threat event across different organizations and which tactics are put in place to maintain confidentiality for threat events in transit (i.e., communicated over the network via REST requests) or at rest (i.e., stored in a MISP database).

The CP-ABE encryption steps and interactions from an event producer with the MISP, TATIS, and Keycloak systems deployed at `Organization 1` are shown in Figure 6. The key material issuance steps are as follows:

Step 1. `TATIS 1` initializes master secret key (MSK) and public parameters

Steps 2-3. `EventProducer` authenticates at `Keycloak 1` and receives an OAuth 2.0 access token

Step 4. `EventProducer` sends REST request to `TATIS 1` to obtain a CP-ABE private decryption key

Steps 5–7. `TATIS 1` evaluates OPA access control policy to authorize this REST request

Steps 8–9. When successful, `TATIS 1` returns a CP-ABE decryption key based on `EventProducer`'s user attributes asserted by `Keycloak 1`

The confidential data sharing process then continues with the following steps:

Steps 10-11. `EventProducer` authenticates at `Keycloak 1` and receives an OAuth 2.0 access token

Step 12. `EventProducer` sends REST request to `TATIS 1` to obtain an encryption key for a given CP-ABE attribute policy (see equation (1))

Steps 13–15. `TATIS 1` evaluates OPA access control policy to authorize this REST request

Steps 16-17. When successful, `TATIS 1` returns a ciphertext policy CT_{ABE} that protects a random AES encryption key

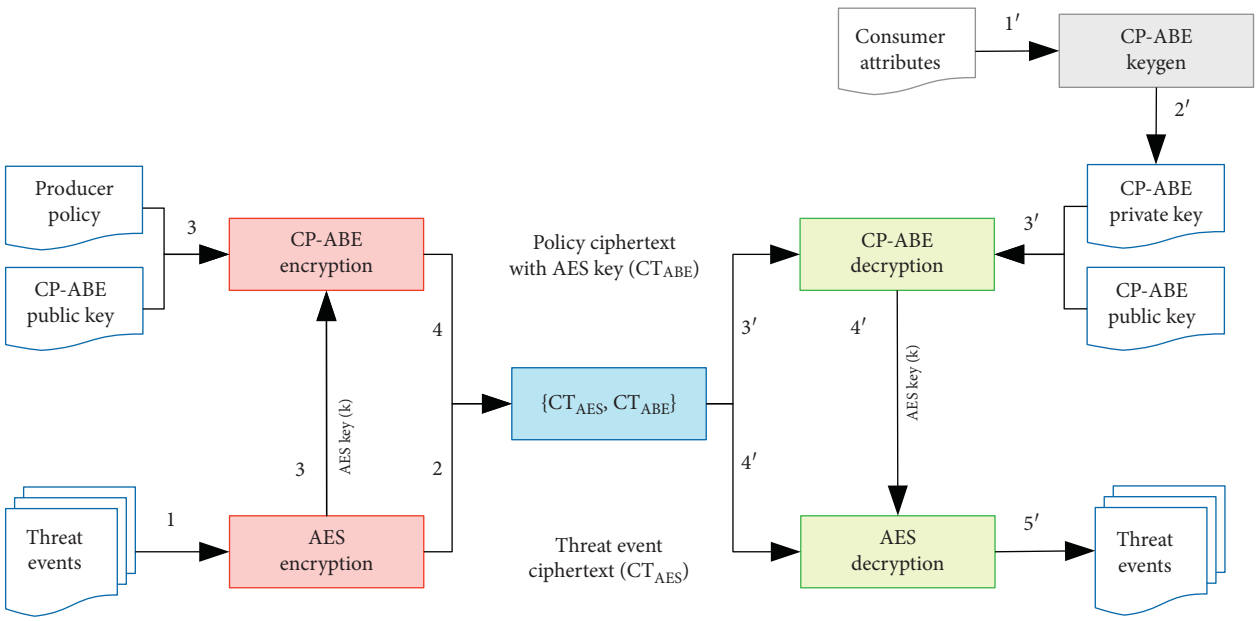

Figure 4: AES based encryption and decryption of event attributes, with the AES key protected by CP-ABE.

```
{
  "Event": {
    "date": "2020-03-12",
    "threat_level_id": "1",
    "info": "testevent",
    "published": false,
    "analysis": "0",
    "distribution": "0",
    "Attribute": [{
        "type": "domain",
        "category": "Network activity",
        "to_ids": false,
        "distribution": "0",
        "comment": "",
        "value": "test.com"
      }
    ]
  }
}
```

Figure 5: Example of a MISP threat event with a single attribute in JSON format.

Step 18. `EventProducer` decrypts the ciphertext policy CT_{ABE} using its CP-ABE decryption key to retrieve the AES encryption key

Step 19. `EventProducer` encrypts MISP event attribute with AES encryption key

Step 20. `EventProducer` sends REST request to upload event with encrypted attributes to `TATIS 1`

Step 21. `TATIS 1` again evaluates OPA access control policy to authorize this REST request

Steps 22–24. When successful, the encrypted event is forwarded to `MISP 1` and provenance is stored in blockchain

Step 25. `MISP 1` synchronizes encrypted threat events with other connected MISP instances

Depending on the sensitivity of each attribute, a different CP-ABE policy can be used to encrypt the attribute. See Figure 7 for a JSON representation of the same threat event as in Figure 5 but now with an encrypted attribute. It is Base64 encoded and stored into the "value" field of the attribute.

To ensure that the event can be stored in line with MISP's field integrity and formatting constraints, its type and category were changed to allow for arbitrary representations. The "comment" field is used to indicate the fact that the attribute is CP-ABE-encrypted. The "value" field in fact consists of 3 parts:

(i) The AES-encrypted event attribute (i.e., CT_{AES})

(ii) The CP-ABE-protected AES key (i.e., CT_{ABE})

(iii) An identifier of the TATIS instance

This way, all synchronized MISP instances have all the information required to be able to decrypt an event attribute. Note that one can also store the CP-ABE-protected AES key as well as a reference to the attribute on the MultiChain ledger and use the latter's authorization scheme to add another layer of access control. However, this may increase the overall latency to decrypt an event with multiple encrypted attributes.

4.2. Enforcing Access Control for Event Consumers. As already indicated in the previous subsection, event producers must be authorized by TATIS for their REST requests to be forwarded to the MISP instance. For event consumers, access control is enforced both at the REST API level and at the data encryption level. This is depicted in Figure 8.

An event consumer at `Organization 2` needs to follow these steps to decrypt events and attributes shared by `Organization 1`:

Steps 1-2. `EventConsumer` authenticates at `Keycloak 2` and receives OAuth 2.0 access token

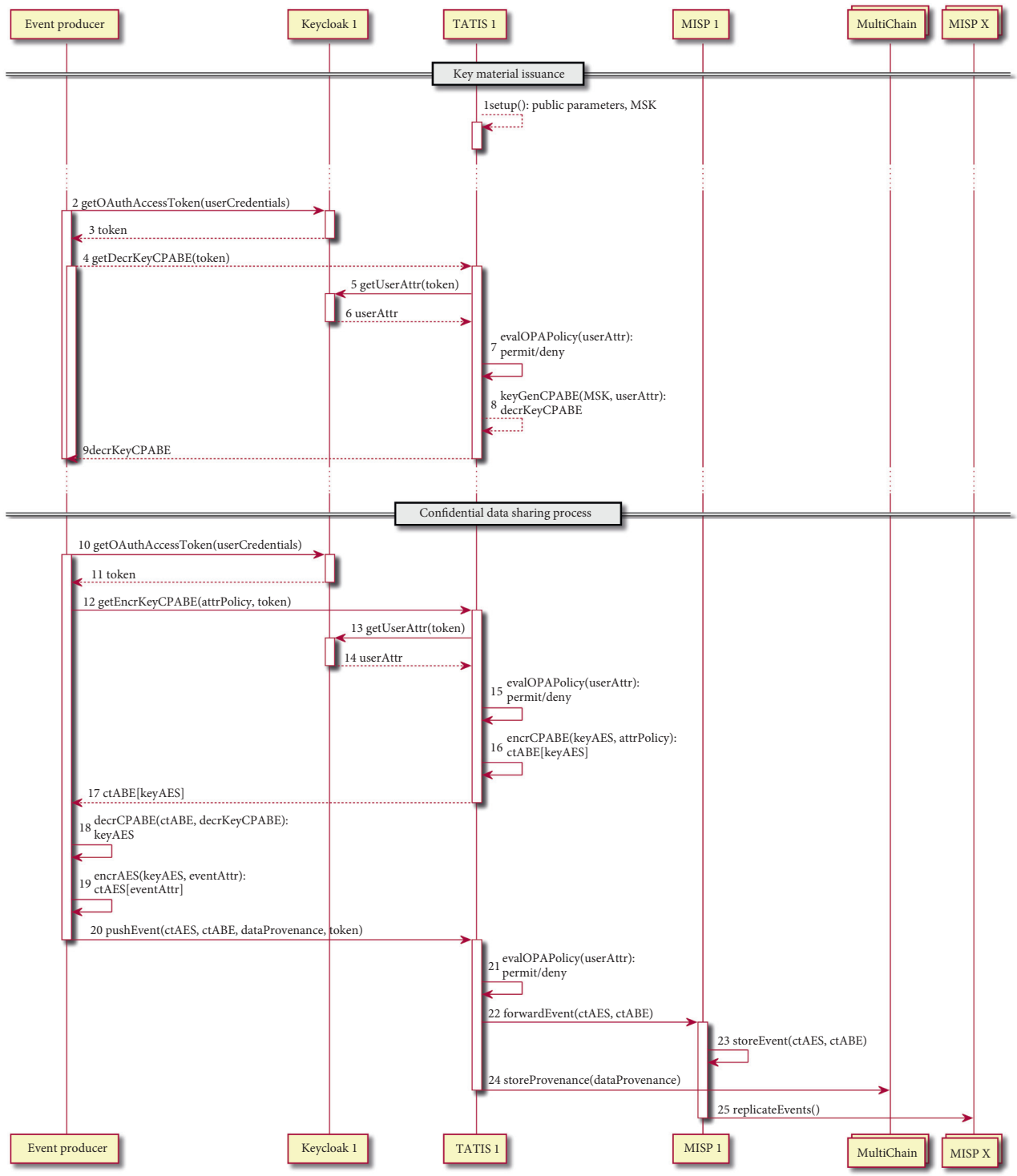

Figure 6: Sequence diagram of event producer.

Step 3. `EventConsumer` sends REST request to retrieve MISP event at `TATIS 2`

Steps 4–6. `TATIS 2` evaluates OPA access control policy to authorize this REST request

Step 7. When successful, `TATIS 2` forwards REST request of consumer to `MISP 2`

Steps 8–11. `EventConsumer` receives threat event with plaintext and encrypted event attributes and provenance data

Note that each organization may host its own MISP platform, resulting in a different CP-ABE public and master secret key for each TATIS instance. As a result, the CP-ABE

```
{
"Event": {
  "id": "2",
  "orgc_id": "1",
  "org_id": "1",
  "date": "2020-03-12",
  "threat_level_id": "1",
  "info": "testevent",
  "published": false,
  "uuid": "5e69f795-09b8-4e75-98aa-02c70a00020f",
  "attribute_count": "1",
  "analysis": "0",
  "timestamp": "1584002965",
  "distribution": "0",
  "proposal_email_lock": false,
  "locked": false,
  "publish_timestamp": "0",
  "sharing_group_id": "0",
  "disable_correlation": false,
  "extends_uuid": "",
  "event_creator_email":
  "eventproducer@organization01.com",
  "Org": {
    "id": "1",
    "name": "ORGNAME",
    "uuid":
    "5e4d973c-58e4-422d-981d-6d906e643788",
    "local": true
  },
  "Orgc": {
    "id": "1",
    "name": "ORGNAME",
    "uuid": "5e4d973c-58e4-422d-981d-6d906e643788",
    "local": true
  },
  "Attribute": [{
    "id": "2",
    "type": "comment",
    "category": "Other",
    "to_ids": false,
    "uuid": "5e69f795-64f4-43c4-addf-02c70a00020f",
    "event_id": "2",
    "distribution": "5",
    "timestamp": "1584002965",
    "comment":
    "CP-ABE-AES Encrypted Attribute Payload",
    "sharing_group_id": "0",
    "deleted": false,
    "disable_correlation": true,
    "object_id": "0",
    "object_relation": null,
    "first_seen": null,
    "last_seen": null,
    "value": "AAAAgGSLhgHviMFKaseGXuAzkoXsi05Cexe
      CIaKPttaYVwJcpGZtWDAmnURw7jmXuQ8vJcsNr ...",
    "Galaxy": [],
    "ShadowAttribute": []
  }],
  "ShadowAttribute": [],
  "RelatedEvent": [],
  "Galaxy": [],
  "Object": []
}}
```


Figure 7: The encrypted MISP threat event.

decryption key of an event consumer at one TATIS instance will not work for cyber threat events encrypted with the public key (and the event producer's CP-ABE attribute policy) at another TATIS instance. The distributed ledger offers the necessary traceability for each event consumer to identify which TATIS instance was responsible for the CP-ABE encryption process of each attribute. These event consumers may then request a decryption key at the appropriate TATIS instance based on the attributes stored within the IAM platform or other claims in the JWT bearer token. When the event consumer has the right attributes that match the CP-ABE policy, he will be able to CP-ABE-decrypt the AES key that was used to encrypt the attribute.

An event may have been shared with multiple encrypted attributes, or an encrypted attribute may have been attached to an existing event by one of the TATIS instances. For each encrypted event attribute, the following steps are carried out:

Step 12. `EventConsumer` identifies TATIS instance at which threat event was shared (e.g., `TATIS 1`)

Step 13–16. `EventConsumer` initiates federated authentication at `Keycloak 1`

Step 17. `EventConsumer` sends REST request to obtain a CP-ABE private decryption key at `TATIS 1`

Step 18–20. `TATIS 1` evaluates OPA access control policy to authorize this REST request

Step 21. `TATIS 1` generates a CP-ABE decryption key based on `EventConsumer`'s user attributes

Step 22. `EventConsumer` obtains and caches the CP-ABE decryption key for `TATIS 1`

Step 23. `EventConsumer` attempts to CP-ABE-decrypt the AES key, which succeeds if `EventConsumer`'s attributes match the CP-ABE attribute policy

Step 24. If successful, `EventConsumer` then decrypts the event attribute with the decrypted AES key

At this stage, the event consumer may verify the provenance data retrieved from the blockchain. This is illustrated in more detail in the next subsection.

4.3. Auditing and Provenance of Shared Information. All requests by event producers and consumers via a TATIS instance are stored on the MultiChain ledger to verify the provenance and trustworthiness of the threat events. The following REST request details are stored in full on the blockchain:

(i) The identity of the event producer

(ii) The domain identifier of the TATIS instance

(iii) The IP address of the client submitting the request

(iv) The HTTP method and URI of the REST request

(v) The timestamp of the REST request

In fact, the above information can be stored for both event producers and event consumers. However, if event consumers decide to share sensitive information out-of-band with unauthorized audiences, the distributed ledger has no way to identify the culprit.

Due to the sensitive nature of the information to be stored in the ledger, for the following properties of the REST request, we only store the SHA256 hash of the raw data:

(i) The JWT bearer token for authorizing the request

(ii) The HTTP body of a POST or PUT request

(iii) The MISP response of the request

A trace of what is stored on the MultiChain ledger is depicted in Figure 9. It depicts a single POST request of a user "EventProducer" for storing an event with encrypted attributes. Only the hash is stored on the ledger, whereas the full event is stored off-chain in the MISP platform. Each transaction on the blockchain results in a transaction ID that can be used to link with the original request and the data in

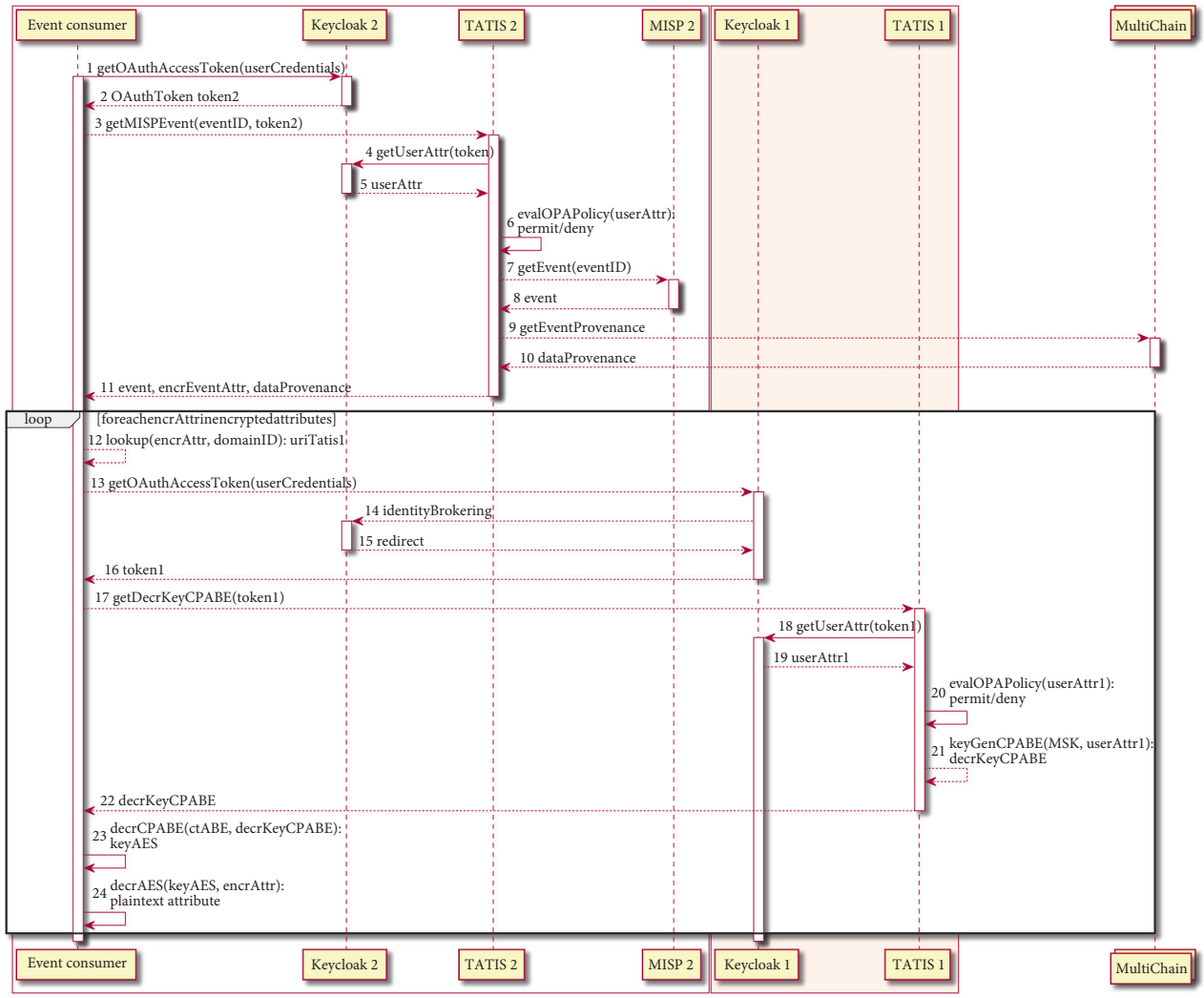

Figure 8: Sequence diagram of event consumer.

the MISP database. This way, the source of a threat event and its attributes can be bound to the reporting organization.

Obviously, all interactions that directly interact with MISP's dashboard or REST APIs inherently bypass TATIS and are therefore not logged in the distributed ledger. In those situations, the source cannot be audited, and then it is up to the event consumers to take the lack of provenance into consideration in their business critical security decisions.

5. Evaluation

For the experimental evaluation, we measure the performance impact imposed by our distributed security framework. We therefore deploy a MISP instance (version 2.4.123), a Keycloak instance (version 9.0) for identity and access management, a MultiChain ledger (version 2.0.5), and our distributed security framework on a Dell Latitude 7480 laptop with an Intel Core i7-7600U CPU running at 2.80 GHz and 32 GB of memory. We will then measure the latency for publishing and consuming threat events, with and without our distributed security framework.

5.1. Baseline Experiment. As a baseline benchmark, we feed our MISP instance with the CIRCL OSINT feed (https://www.circl.lu/doc/misp/feed-osint/). At the time of writing (March 2020), this feed contains 1359 events in JSON format. Figure 10 summarizes into a histogram some key characteristics of this event feed in terms of the file size of each event, as well as the number of attributes in each event. More than 50% of the 1359 events have a file size of 20 kilobyte or less and fewer than 35 attributes. The largest events are as follows: one event with UUID 5d96e112-4894-40de-8e52-4b4d950d210f has 169318 attributes, whereas another event with UUID 5c5be78a-3908-47c4-bf7c-4487950d210f has a file size of 428014 kilobytes.

To set a baseline, we will first upload these 1359 threat events directly to a MISP instance through its REST interface and measure the latency of the HTTP POST requests. The results are depicted in Figure 11, indicating the number of events that impose a certain latency for the request to be handled. Certain events take longer to be processed, and the time required correlates with the number of attributes in the

```
[
  {
    "publishers" : [
      "1RQNmA5nrBxWmG3v7WncfeNkmYDDWu5VB4KMrC"
    ],
    "keys" : [
      "eventproducer"
    ],
    "offchain" : false,
    "available" : true,
    "data" : {
      "json" : {
        "body" : {
            "sha256" : "5df45c6ff4f450682da2742ea9449ffa974afd6019287fb4a70279bfb2fabf49"
        },
        "datetime" : "Wed Mar 18 10:56:06 CET 2020",
        "instance" : "nuage.cs.kuleuven.be",
        "ipaddress" : "134.58.39.58",
        "method" : "POST",
        "response" : {
            "sha256" : "ffc2434a369b9604e6a7602705b1b703d895148eecfa7853c86bd32fa2795834"
        },
        "timestamp" : 1584525366999,
        "token" : {
            "sha256" : "ef64bc45e115cf72ee211681eead99bb9c771c7cfd7016525f03af44a24e6509"
        },
        "uri" : "https://nuage.cs.kuleuven.be/tatis/api/events",
        "username" : "eventproducer"
      }
    },
    "confirmations" : 143,
    "blocktime" : 1584525370,
    "txid" : "2461d5a434fced61dfa781d9d1d4e1b90a52134bc143b7e46693fbd2ae6df448"
  }
]
```


Figure 9: Auditing the threat event requests on the MultiChain ledger.

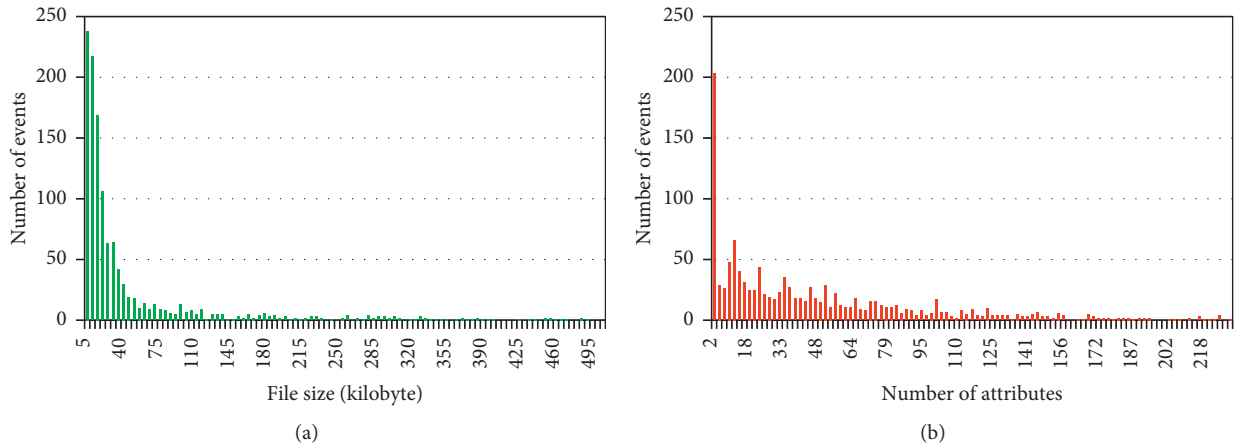

Figure 10: File size and number of attributes for threat events in OSINT feed.

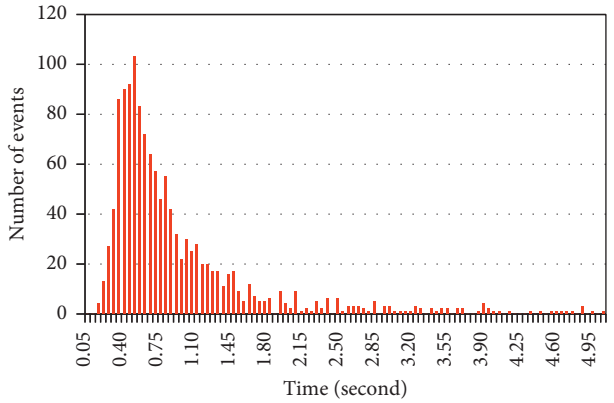

Figure 11: Histogram of HTTP POST requests for plaintext threat events.

event. These outliers are not shown in Figure 11. The average latency is 3.010 seconds with a standard deviation of 36.891.

We will then retrieve the same threat events using HTTP GET requests, again directly interacting with MISP, and use the request latency for comparison with configurations involving our distributed security framework. The results are depicted in Figure 12. Compared with the previous histogram, it is clear that uploading a threat event takes longer than retrieving it. The average latency is 0.087 seconds with a standard deviation of 0.317.

5.2. Performance Impact of Distributed Security Framework. In the following scenario, we will consider the scenario where each attribute in the event is encrypted with an AES key that in turn is protected with a new CP-ABE ciphertext

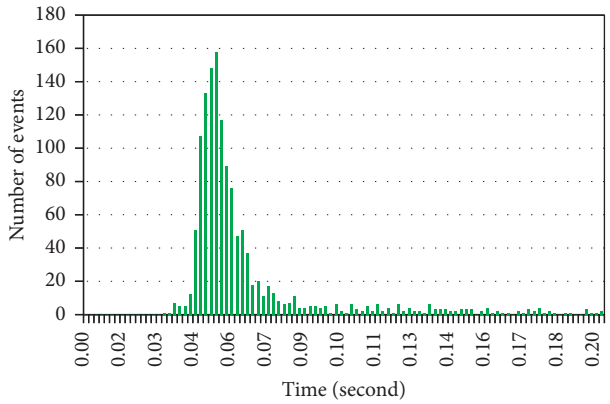

Figure 12: Histogram of HTTP GET requests for plaintext threat events.

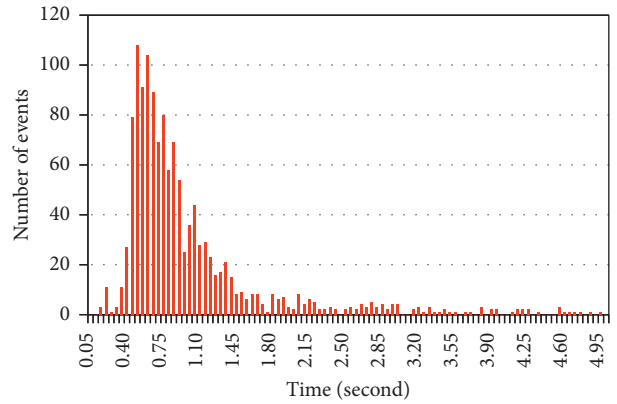

Figure 13: Histogram of HTTP POST requests for encrypted threat events.

policy per attribute. We will consider the full end-to-end latency of each request, from the client encrypting the attributes up to uploading the event to TATIS that will check the access control policy, and finally store the event in MISP as before, as well as leaving a provenance trace of the transaction on the MultiChain ledger.

From Figures 13 and 14 it is immediately clear that GET requests now take longer to be processed, when compared to their plaintext counterparts. For uploading threat events with CP-ABE-protected attributes, the average latency is 2.987 seconds with a standard deviation of 34.904, which is in line with the plaintext variant. However, retrieving and decrypting the CP-ABE-protected events take on average 0.166 seconds with a standard deviation of 1.112. Even though the average is about twice as high, the effect is mainly caused by outlier events with a significantly large number of attributes that all need to be individually decrypted.

Each HTTP POST and HTTP GET request involves multiple steps, including the encryption and decryption of the attributes, the interaction with the MISP platform, and managing the provenance on the MultiChain ledger. Figures 15 and 16 illustrate where most of the time is spent on average for the individual steps. It is clear that MISP itself has the highest impact on the overall latency.

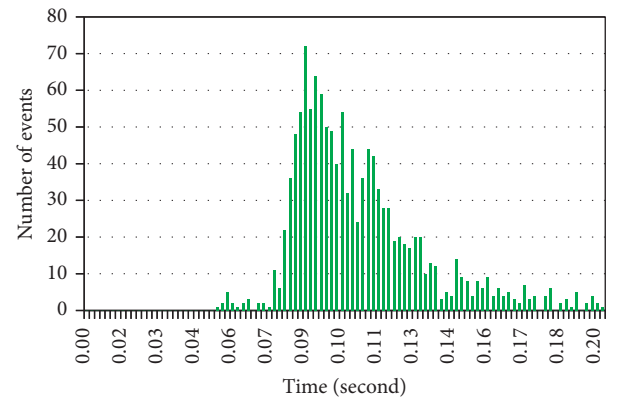

Figure 14: Histogram of HTTP GET requests for encrypted threat events.

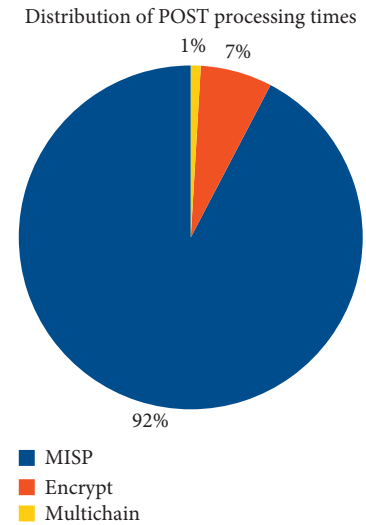

Figure 15: Distribution of processing times for POST requests.

5.3. Computational Complexity of Cryptographic Schemes. Note that the encryption and decryption involve both the CP-ABE and AES encryption schemes. In Table 1, we list the total execution time for encrypting and decrypting 15 example events. The CP-ABE scheme protects an AES key, so that is why the execution times are comparable. The execution time of the AES scheme is accumulated for all attributes in the event, hence the correlation between the number of attributes and the AES execution times.

When averaging these results for all events and per attribute, as depicted in Table 2, producing the CP-ABE ciphertext policy takes on average 0.190 seconds, whereas AES encrypting a single attribute takes 0.013 seconds. The average number of attributes per event in the OSINT feed is 355 (though the median is significantly lower). Decrypting the CP-ABE ciphertext policy takes 0.025 seconds, and AES decrypting one attribute requires 0.009 seconds.

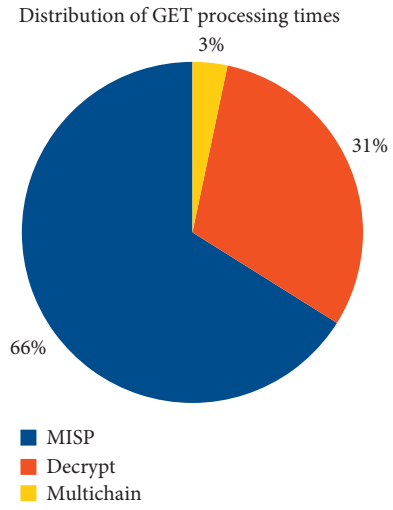

Figure 16: Distribution of processing times for GET requests.

Table 1: Execution time (in milliseconds) of CP-ABE and AES for 15 example events.

No. of attributes	CP-ABE encrypt (ms)	AES encrypt (ms)	CP-ABE decrypt (ms)	AES decrypt (ms)
1067	237	142	63	41
29	258	2	32	1
225	221	10	44	4
65	180	3	51	0
1817	194	41	31	14
31	160	0	27	0
98	159	2	31	0
414	212	4	31	3
6798	166	86	23	54
109	215	6	24	1
25	165	0	24	0
34	177	0	24	0
74	197	1	24	1
39	216	0	24	0
27	193	0	25	0
...				

Table 2: Execution time (in milliseconds) of encryption and decryption schemes.

Measure	Mean (ms)	Standard dev. (ms)
AES encrypt	13	51
CP-ABE encrypt	190	18
AES decrypt	9	39
CP-ABE decrypt	25	5

These results were obtained for CP-ABE ciphertext policies with similar complexity as depicted in equation (1), with 3 to 5 disjunctions having 1 up to 3 attribute conditions.

5.4. Discussion. The impact of the OPA access control policy evaluation is minimal, increasing the overhead with a just a few milliseconds. Even if the latency for GET requests has more than doubled, we believe this value to be acceptable, as any historical analysis on the trustworthiness of an event producer will have a bigger impact. Our solution only retrieves provenance information of a single event from the blockchain and carries out some simple checks but does not attempt, for example, to compute a reputation score of the event producer based on historical entries on the blockchain.

Another aspect not taken into consideration is the impact of the network. In our experiments, all client and backend applications were deployed on the same machine to ensure that variations in the network capacity would not skew the results. As some events are more than 10 megabytes (the largest being 418 MB), the network latency will play a role too, irrespective of whether the event was CP-ABE-encrypted or not.

In addition, the overhead added by our system to interact with the ledger is minimal and represents 1% and 3% of the overall processing time needed to handle the POST and GET requests, respectively.

Last but not least, even if the relative latency impact cannot be ignored, in absolute terms, the amount of time required to process an event must be considered in light of the frequency of events being published. For the OSINT feed, this means about 1359 events shared over multiple years or well below one event per hour.

Reflecting back on our threat model and the assumed attacker capabilities, we achieved the main objectives of ensuring that no untrusted threat intelligence platform provider or malicious threat intelligence consumer is able to gain unauthorized access to the sensitive information. When comparing our work with the approach of Van De Kamp et al. [6] which computes the hash for each IoC to preserve the privacy, our solution achieves the same objective of maintaining confidentiality. The added value of our work is that authorized users can decrypt the IoC and, as such, have the opportunity to gain more detailed insights. Nonetheless, there are some limitations in the current solution which we must be aware of. First of all, the entity that is responsible for the key generation in the CP-ABE scheme uses a master key to bootstrap this process. If an adversary is able to compromise this entity, then he or she can exploit the master key to generate any decryption key at will. In a similar manner, if this entity is acting maliciously and colludes with curious threat intelligence consumers, the same unauthorized access can occur. However, we assume that the identity and access management system, which acts as an attribute authority for its users and is in charge of this master key, is a trusted component. To mitigate this threat, one may enforce that no single attribute authority is allowed to generate a decryption key and must collaborate with at least 1 or 2 other attribute authorities. Such an approach may be realized with Shamir's secret sharing scheme as illustrated in [32]. However, this is beyond the scope of this research.

6. Conclusions

In this paper, we proposed a distributed security framework that extends our previous work TATIS with distributed ledger capabilities to enable reliable and trustworthy threat

intelligence sharing with the ability to audit the provenance of threat intelligence. The goal of our framework is to address the lack of trust in threat event producers and the information itself.

We have implemented and evaluated the impact of our framework on top of the MISP threat intelligence platform using the OSINT threat intelligence feed. In a worst case scenario, where each attribute is individually encrypted with a different AES key that is in turn protected with a different CP-ABE ciphertext policy, the latency impact would be nonsignificant due to the high computational cost to obtain the AES key from the CP-ABE ciphertext. However, when the same ciphertext policy is used to encrypt multiple attributes, the results are much more acceptable, only doubling the overall latency for all steps involved, making the proposed solution practically feasible.

In addition, this paper has shown how our framework is able to audit the CTI events in the ledger to achieve provenance and accountability of shared threat events. In this sense, the overhead imposed by our system to audit the CTI data in the ledger is minimal when compared with the time taken by the MISP platform to handle the requests.

As future work, we will look into advanced analytics mechanisms, e.g., statistical and machine learning methods, for a historic reputation analysis of those stakeholders providing threat intelligence information. Besides, we are investigating AI-based mechanisms for zero-touch security, detecting and countering in real-time threats, and cyber attacks considering the cyber situational context and shared trusted CTI data.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially funded by the Research Fund of KU Leuven. It was also supported by the European H2020-SU-ICT-03-2018 Project no. 830929 CyberSec4Europe (https://cybersec4europe.eu). It has also been partially funded by an AXA Postdoctoral Scholarship awarded by the AXA Research Fund and by the Cyber Security Research Flanders with reference no. VR20192203.

References

[1] C.-W. Ten, G. Manimaran, and C.-C. Liu, "Cybersecurity for critical infrastructures: attack and defense modeling," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 4, pp. 853–865, 2010.

[2] Verizon: 2019 data breach investigations report," *Computer Fraud & Security*, vol. 2019, no. 6, p. 4, 2019.

[3] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Computers & Security*, vol. 72, pp. 212–233, 2018.

[4] S. Qamar, Z. Anwar, M. A. Rahman, E. Al-Shaer, and B.-T. Chu, "Data-driven analytics for cyber-threat intelligence and information sharing," *Computers & Security*, vol. 67, pp. 35–58, 2017.

[5] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Attribute based sharing in cybersecurity information exchange framework," *Simulation Series*, vol. 49, no. 10, pp. 68–73, 2017.

[6] T. Van De Kamp, A. Peter, M. H. Everts, and W. Jonker, "Private sharing of IOCs and sightings," in *Proceedings of the 2016 ACM Workshop on Information Sharing and Collaborative Security*, pp. 35–38, Vienna, Austria, 2016.

[7] D. Preuveneers and W. Joosen, "TATIS: trustworthy APIs for threat intelligence sharing with UMA and CP-ABE," in *Proceedings of the 12th International Symposium on Foundations and Practice of Security*, vol. 12056, Springer, Toulouse, France, November 2019.

[8] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: survey and research directions," *Computers & Security*, vol. 87, Article ID 101589, 2019.

[9] A. Zibak and A. Simpson, "Cyber threat information sharing: perceived benefits and barriers," in *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES'19*, 2019.

[10] N. Kaloudi and J. Li, "The AI-based cyber threat landscape," *ACM Computing Surveys*, vol. 53, no. 1, pp. 1–34, 2020.

[11] S. Mittal, A. Joshi, and T. Finin, "Cyber-all-intel: an ai for security related threat intelligence," 2019, https://arxiv.org/abs/1905.02895.

[12] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the ioc game: toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 755–766, Vienna, Austria, 2016.

[13] N. Khurana, S. Mittal, A. Piplai, and A. Joshi, "Preventing poisoning attacks on ai based threat intelligence systems," in *Proceedings of the 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, Pittsburgh, PA, USA, pp. 1–6, 2019.

[14] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, "Threat intelligence sharing platforms: an exploratory study of software vendors and research perspectives," in *Proceedings of the 13th International Conference on Wirtschaftsinformatik (WI 2017)*, pp. 837–851, St. Gallen, Switzerland, 2017.

[15] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "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, WISCS'16,*, ACM, New York, NY, USA, pp. 49–56, 2016.

[16] Z. Yan, X. Li, M. Wang, and A. V. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 485–498, 2017.

[17] W. Meng, K.-K. R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst, "Towards Bayesian-based trust management for insider attacks in healthcare software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 761–773, 2018.

[18] A. Albakri, E. Boiten, and R. De Lemos, "Sharing cyber threat intelligence under the general data protection regulation," in

Privacy Technologies and Policy, M. Naldi, G. F. Italiano, K. Rannenberg et al., Eds., pp. 28–41, Springer International Publishing, Cham, Switzerland, 2019.

[19] J. Bernal Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. Torres Moreno, and A. Skarmeta, "Privacy-preserving solutions for blockchain: review and challenges," *IEEE Access*, vol. 7, pp. 164908–164940, 2019.

[20] V. Mavroeidis and S. Bromander, "Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence," in *Proceedings of the 2017 European Intelligence and Security Informatics Conference*, pp. 91–98, Athens, Greece, 2017.

[21] T. D. Wagner, E. Palomar, K. Mahbub, and A. E. Abdallah, "A novel trust taxonomy for shared cyber threat intelligence," *Security and Communication Networks*, vol. 2018, Article ID 9634507, 11 pages, 2018.

[22] D. Homan, I. Shiel, and C. Thorpe, "A new network model for cyber threat intelligence sharing using blockchain technology," in *Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–6, Canary Islands, Spain, 2019.

[23] R. Graf and R. King, "Neural network and blockchain based technique for cyber threat intelligence and situational awareness," in *Proceedings of the 2018 10th International Conference on Cyber Conflict (CyCon)*, pp. 409–426, Tallinn, Estonia, 2018.

[24] P. Amthor, D. Fischer, W. E. Kühnhauser, and D. Stelzer, "Automated cyber threat sensing and responding: integrating threat intelligence into security-policy-controlled systems," in *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES'19*, 2019.

[25] E. Kim, K. Kim, D. Shin, B. Jin, and H. Kim, "CyTIME," in *Proceedings of the 13th International Conference on Future Internet Technologies-CFI 2018*, Seoul, Korea, 2018.

[26] S. Badsha, I. Vakilinia, and S. Sengupta, "Blocynfo-share: blockchain based cybersecurity information sharing with fine grained access control," in *Proceedings of the 10th IEEE Annual Computing, and Communication Workshop and Conference (CCWC)*, Nevada, Las, NV, USA, 2020.

[27] S. Badsha, I. Vakilinia, and S. Sengupta, "Privacy preserving cyber threat information sharing and learning for cyber defense," in *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 708–714, Las Vegas, NV, USA, 2019.

[28] T. Sandall, "Open policy agent," 2020, https://www.openpolicyagent.org/.

[29] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP'07)*, pp. 321–334, Oakland, CA, USA, 2007.

[30] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: an intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.

[31] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference, EuroSys'18*, 2018.

[32] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," in *Proceedings of the International Conference on Cryptology in India*, Springer, Kharagpur, India, pp. 426–436, 2008.