BloCyNfo-Share: Blockchain based Cybersecurity Information Sharing with Fine Grained Access Control

Shahriar Badsha
Dept. of Computer Science and Eng.
University of Nevada, Reno, NV, USA
sbadsha@unr.edu

Iman Vakilinia
School of Computing
University of North Florida, FL, USA
i.vakilinia@unf.edu

Shamik Sengupta
Dept. of Computer Science and Eng.
University of Nevada, Reno, NV, USA
ssengupta@unr.edu

***Abstract*—To build a proactive cyber defense system, sharing the cybersecurity information has been very popular by which any organization can get more information about unknown and new threats. Cybersecurity Information Exchange (CYBEX) is one of the important platforms which has been playing an important role in implementing proactive cyber defense system by allowing organizations sharing their cybersecurity information. However, they are centralized and therefore they may suffer from complete failure in case of any damage or accident. Moreover, while sharing private information it lacks the mechanism of providing rights to query organizations i.e., enabling the access control over the shared sensitive information. Finally, non-repudiation of the system does not exist i.e., there is no way to track or keep the record what any organization is sharing and it is necessary to keep the record in case anyone denies after sharing false information. To address these issues, in this paper we propose blockchain based privacy preserving cybersecurity information sharing using proxy re-encryption and attribute-based encryption (BloCyNfo-Share) where the organization can achieve fine-grain access control by delegating which organization can have the access to its cybersecurity information leveraging the benefits of blockchain technology. We conduct privacy and experimental analysis of the proposed system and the findings show that the model is private as well as efficient.**

Keywords- Blockchain, Cybersecurity Information Exchange, Threat, Privacy, Access Control

I. Introduction

The complexity of cyber attacks is increasing, and there is a growing demand for a proactive defense. Due to the increasing rate of cyber threats, to effectively tackle the issues, organizations face major difficulties. Although an organization may build its own cybersecurity solution, it may not help to understand better about the current cybersecurity landscape well.

To address these issues, the CYBersecurity information EXchange (CYBEX) [1]–[4] has been playing a crucial role in implementing this proactive defense by sharing cyber threat information with other parties. With the help of CYBEX, organizations can learn about potential cyber threats beforehand and defend themselves against the attacks.

Also, various methods for sharing timely and actionable cybersecurity information such as vulnerabilities or detection signatures are paramount to enable the cooperative cyber-defence [5]. Another popular approach which aim to facilitate automatic cybersecurity information sharing is TAXII [6].

In traditional data sharing domain, the factors like privacy, security, and interoperability are very important. First, the cybersecurity information often contains highly privacy-sensitive data, therefore the leakage of those data could hurt the reputation as well as finance of the organizations. Second, there exists no decentralized cybersecurity information sharing platform. However, the centralized framework has robustness as well as security vulnerabilities such as: single-point-of-failure as well as Distributed Denial of Service (DDoS) attack.

A. Motivation

One of the main challenges with cybersecurity information sharing is how to share the privacy-sensitive information with other organizations or how to provide the access permissions of privacy sensitive information to other organizations. To address this access control problem, we proposed attribute based sharing mechanism for cybersecurity information [7] where we used CP-ABE (Ciphertext Policy Attribute based Encryption) based mechanism to share the cybersecurity information providing rights and access to certain organizations. We found that there are several areas that our previous scheme can be improved by addressing some limitations. Firstly, although CP-ABE potentially satisfies the requirements to apply fine grain access control with cryptographic methods over data, it is essentially a group encryption, therefore any individual organization can not be differentiated. Moreover in their scheme, if any organization is revoked, its attributes are also revoked which is not feasible in a large scale system.

Another issue with the existing cybersecurity information sharing mechanism is that there is no way to check if the system is non-repudiable. There is a chance that any competitive organization requests data to any owner and the owner organization may change the stored data with false information to mislead the query organization. To address the non-repudiability issue with data sharing services, Blockchain has been useful since it brings transparency and no entities can deny once they make any contribution in the system.

This research is supported by the National Science Foundation (NSF), USA, Award #1739032

978-1-7281-3783-4/20/$31.00 ©2020 IEEE

B. Main Contributions

- We propose a **Blo**ckchain based privacy preserving **cy**bersecurity i**nf**ormation sharing with fine grain access control (BloCyNfo-Share) where the organizations can share their CTI (Cyber Threat Intelligence)[1] data in privacy-preserving manner. Motivated from Attribute based Proxy Re Encryption [8] mechanism, we specifically apply the concept using Blockchain in the area of cyber treat intelligence sharing among the organizations.
- The BloCyNfo-Share protocol improves the existing access control based CTI sharing protocol [7] in several ways: (1) unlike the [7], our scheme is not time bounded. (2) The BloCyNfo-Share provides two stages of fine grain access control where in the first stage, the owner can choose the delegatee organization, for instance, which organization should get the encrypted information or not, no matter if the attributes match with the owners' policy. (3) Unlike the [7], we do not revoke the organizations' identity if they want to leave the system, instead, we revoke only their re-encryption key (which is used to re-encrypt using a proxy server) which is more feasible to scale the system in case the number of organizations increases. (4) Because of using blockchain, the owner organization is unable to change its information which is already submitted to the storage and corresponding hash is recorded into blockchain. In this way, the organizations cannot harm other competitive organization in case they want to change the ciphertext with false information.
- We provide the experimental analysis on computation cost in terms of increasing number of attributes of the organization and implementation results on Ethereum Blockchain platform. The results show that the system is efficient in terms of both time consumption as well as gas cost in Ethereum.

C. Organization

The rest of the paper is organized is as follows. The section II and III describe the related work and preliminary studies required for the proposed work respectively. The overall system model is described in section IV and the proposed methodology is described in section V. Section VI and VII show the privacy and experimental analysis, and finally section VIII concludes the paper with possible future work.

II. Related Work

A. Cybersecurity Information Sharing

To improve the mechanism of cybersecurity information sharing in terms of privacy [**?**], [9]–[12] as well as efficiency. The coalitional approaches for cybersecurity information sharing and the underlying privacy challenges were studied in [13]. The game theoretic aspects of cybersecurity information sharing were presented in [14]. Despite their advantage of providing the platform of sharing cyber threat information, they suffer from some major issues like lack of privacy protection mechanism to protect sensitive cybersecurity information. To address the privacy protection of shared cybersecurity information several works were proposed. The scheme used group signature to share the cybersecurity information [15] with privacy preservation by means of hiding the identities of organizations. Later, the [4] proposed a model as game between attackers and organizations to find who wins the game by finding the best strategy of the organziations as well as the attacker. However the solution does not work to protect underlying privacy sensitive information. Moreover, the authors used the signature scheme [15] and access control mechanism [7] to protect sensitive cybersecurity information of any organization. In [7], the CTI information was shared using only CP-ABE scheme where the central server who manages the cryptographic and privacy preserving operations for both owner and requester is trusted. There is no way to keep a record of what information was shared and moreover, in case of revoking any user or organization, the corresponding attributes need to be revoked which is not an efficient or feasible solution.

[1]The terms "CTI" and "Cybersecurity information" are interchangeable for the rest of the paper

B. Proxy Re-Encryption

The [16] proposed the concept of proxy re encryption, and proposed the a bidirectional proxy re encryption method (the delegation from Bob to Alice allows the re-encryption from Alice to Boc). Since then different version of PRE [17], [18] and specifically the attribute-based PRE [8], [19] were proposed for different applications.

Recently, Manzoor et. al. [20] proposed blockchain based proxy re-encryption which is close to our model in terms of using blockchain with proxy re-encryption for secure IoT (Internet of Things) data sharing. The main difference between our proposed work and the [20] is that we use conditional proxy re-encryption with ABE where [20] used only certificate based proxy re-encryption.

C. ABE Access Control

There are plenty of works that have been done in the area of access control [21], [22]. The CP-ABE [23], allows the data owner to encrypt the data with consideration of access policy. Users are only able to decrypt the ciphertext if and only if their attributes can pass the access policy of the ciphertext. There are single-authority [23] and multi-authority [24] CP-ABE frameworks. In the single authority setting, the management of attributes and their corresponding keys are handled by one authority, while in the multi-authority CP-ABE systems, multiple authorities manage the attributes. We are utilizing a simple multi-authority CP-ABE system and the PRE [17] with the help of Blockchain to authorize access to the sensitive cybersecurity information.

III. Preliminaries

In this section, we discuss the key elements and techniques behind the proposed BloCyNfo-Share protocol.

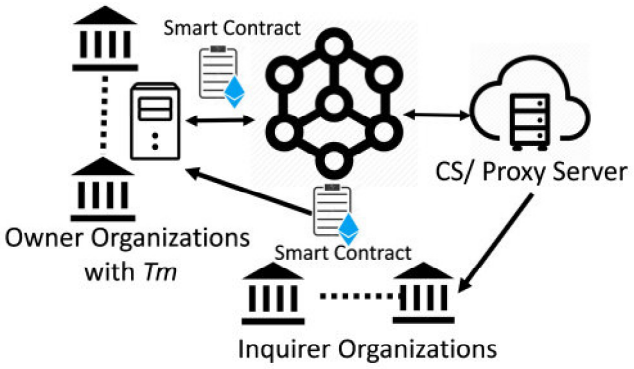

Fig. 1. System Model of BloCyNfo-Share

A. Ciphertext Policy Attribute based PRE

A ciphertext policy based proxy re-encryption scheme [8] consists of following functions.

- Setup(1^k) $\rightarrow params$: On input a security parameter 1^k, the setup algorithm outputs public parameter $params$.
- KeyGen($params$) $\rightarrow (pk, sk)$ On input user identity and paramaeters, the key generation algorithm outputs public and private key pairs pk, sk for user u.
- ReKeyGen($sk_A, \mathbb{A}, pk_B$) $\rightarrow rk_{A\rightarrow B}$ On input sk_A from user A who is delegator, pk_B who is delegatee and attributes $\mathbb{A}$, the re-encryption key generation algorithm generates a re-encryption key $\rightarrow rk_{A\rightarrow B}$.
- Enc1(pk, m) $\rightarrow c$: On input a public key pk, a message m, the first level encryption generates ciphertext c.
- Enc2($pk, m, \mathbb{P}$) $\rightarrow \bar{c}$ On input public key pk, message m and access policy $\mathbb{P}$, the second level encryption outputs another ciphertext $\bar{c}$.
- ReEnc($rk_{A\rightarrow B}, \bar{c_A}$) $\rightarrow c_B$: On input re-encryotion key $rk_{A\rightarrow B}$, a second level ciphertext under public key pk_A and an access policy $\mathbb{P}$, the re-encryption algorithm generates ciphertext c_B under public key pk_B if the attributes $\mathbb{A}$ satisfies access policy $\mathbb{P}$.
- Dec1(sk, c) $\rightarrow m$
- Dec2($sk, \bar{c}$) $\rightarrow m$

B. Blockchain and Ethereum

The blockchain is a peer to peer network that records all the transactions in its distributed database [25]. Every node in a blockchain holds same copy of the updated database. There is no central authority in the blockchain, therefore there is no single point of failure. Ethereum [26] is a open source platform where decentralized application can be developed. It supports the solidity language to develop smart contracts which help to initiate any transactions upon verification of nodes or any given rules in the blockchain. The smart contracts are developed using solidity and can be deployed in Ethereum. Once deployed, it is automatically executed according to the logic provided into it.

IV. System Model of BloCynFo-Share

Our proposed model as shown in Figure 1, consists of mainly three entities: the organizations ($O_i = O_1, O_2, ..., O_N$), a cloud server *CS* which works as a proxy and storage server, and a trusted manager *Tm* who is responsible for key generation and protocol initialization. The organizations share their encrypted CTI into the cloud server, while storing the hash of their data into the blockchain. Inquirer organization $QO_k \in \{QO_1, QO_2, ..., QO_M\}$ sends request to the smart contract for asking CTI information. The owner of the information O_i, generates the re-encryption key accordingly and call the smart contract. Afterward *CS* obtains the key and computes the new cypher text which can be decrypted by the QO_k. Below we describe the entities roles:

- **Organization** (O_i): The O_i holds the cybersecurity information and share the CTIs with other query organizations QO_k. Note that, in our scheme, organizations can record the requests for CTIs. This allows accounting the sharing information and possibly reward and penalize organizations on cybersecurity information sharing platform.
- **Cloud Server/ Proxy Server** (*CS*): The proxy server is any cloud server who stores the encrypted data for the end users, and also performs re-encryption operations to lessen the computation overhead for QO_k. The *CS* provides the hosting service for sharing CTI. We consider *CS* is honest but curious, i.e. *CS* strictly follows the protocol but it is curious about the sensitive data which is shared by O_i.
- **Trusted Manager** (*Tm*): The *Tm* is trusted entity who is responsible for managing the keys in attribute based encryption model, and also managing the proxy re-encryption keys for organizations O_i. Similar to some of the existing work, it is common to use any trusted entity to manage the keys for large number of organizations.

The CTI information of any organization may contain both sensitive and non-sensitive information. The purpose of our protocol is to share only the sensitive CTI information by leveraging proxy re-encryption and CP-ABE scheme with blockchain. Some examples of sensitive information are users' private information, system configuration, zero-day vulnerability information (which can be used by attackers to exploit the vulnerable systems). On the other hand, the information which is not sensitive can be shared publicly.

The proposed system works as follows. First, the organizations register to the blockchain network to be part of the sharing process. The O_i interacts with *CS* via the *Tm* to save and retrieve the encrypted CTI from the cloud database as well as interacts with blockchain to save the hashes of the information exchanged among the organizations. The QO_k shares its public key in the blockchain via the smart contract. The *Tm* also shares the re-encryption key using a smart contract. The proxy server reads re-encryption key for the conversion. The storage of re-encryption key in the smart contract prevents the malicious *CS* to present false information as other entities having access to the blockchain can verify the conversion of ciphertext as well. Upon receiving the ciphertext,

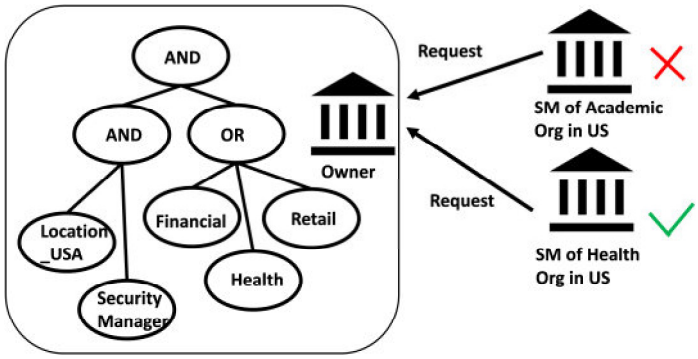

Fig. 2. Example of Access Tree

the QO_k is able to decrypt if and only if the attributes match with the access policy provided by the O_i.

V. The Proposed BloCyNfo-Share Scheme

Below we show how one organization O_i can share its encrypted information with QO_k using proxy re-encryption along with CP-ABE leveraging the blockchain technology. For instance, a data owner organization builds an access tree as shown in Figure 2. The access tree depicts that, any inquirer, who is a security manager from any of the organizations among *Financial, Retail, Health*, located in the US, the protocol of proxy re-encryption will only work converting the encrypted data of owner under the public key of the inquirer. According to the Figure 2, there are two different inquirers *security manager (SM)* of an *Academic organization* and a *SM* of an *Health organization*. Since the *SM* of *Academic Organization* does not match with the access policy, the proxy server is not able to re-encrypt the ciphertext of *Owner* successfully. The detailed protocol is described in the following section.

A. Protocol Description

Below, we show how a requester QO_k can get access to encrypted information of owner organization O_i through proxy re encryption with CP-ABE and blockchain. In this protocol the trusted manager works for O_i by performing all cryptographic operations. To make the system design simpler, the *Tm* has been used, otherwise the owner O_i can perform all these operations by itself.

1) ***Setup****:* The *Tm* runs the setup to establish and publish the public parameters for O_i of the proposed scheme as below.

The *Tm* determines a bilinear map $e : G_0 \times G_0 \rightarrow G_T$, where G_0 and G_T are cyclic groups of k bit prime order p. Then select g which is a generator of G_0. A cryptographic hash function $H : \{0,1\}^* \rightarrow G_0$. Then set $Params = (g, e(g,g), H(.))$. The public parameters is stored using $StoreParams(params)$ and can be retrieved by calling the smart contract $RetreiveParams()$ function. Using blockchain storage guarantees the integrity of the $params$.

2) ***Registration****:* In the registration phase, the organizations are registered in the framework by authorizing themselves. The QO_k receives its credentials from the credential authorities. For instance, a retail store receives the "Retailer" attribute from the retail-center authority. After verification of QO_k's credentials, the corresponding credential authority issues a signature indicating QO_k has a attribute a_l (for example, the "Retailer" $\{$*Security Manager, Located in US, Retailer*$\}$ attributes). When QO_k requests for registration, it needs to submit its own public key along with its credentials to the smart contract by calling $OrgRegistration(ID, pk, creds)$ function. Then the credentials, organization's ID and public key are submitted to the blockchain. The *Tm* authenticates the users identity and assigns appropriate attributes set for the QO_k, then adding the users account address to the set of authorized users in the smart contract.

3) ***KeyGeneration****:* The O_i runs the keygeneration algorithm as below to generate its own key pair pk_i, sk_i. Similarly other organizations (requester and owners) generate their own key pairs.

Choose $x_u, \alpha_u, \beta_u \in Z_p$ and set $(pk_u = (y_u = g^{x_u}, h_u = g_{\beta_u}, w_u = e(g,g)^{\alpha_u}), sk_u = (x_u, \alpha_u, \beta_u))$, where u represents any entity in the system.

4) ***Re-key-generation****:* In this step the O_i generates a re encryption key for query organization QO_k according to its role. Note that, the set of attributes were decided in the registration phase for QO_k. The query organizations signs its own public key and it is uploaded on the blockchain through calling the smart contract $RegPubkey(pk)$ function. The re encryption key is derived using the public key of query organization which is verified in blockchain, then the generated re encryption key is signed by the O_i and stored in the *CS* by calling the smart contract $StoreReKey(rk)$. By this contract, the hash of these keys are also stored for further verification process. To generate re encryption key the O_i performs as follows. Let $sk_{O_i} = (x_{O_i}, \alpha_{O_i}, \beta_{O_i})$ and $pk_{QO_k} = (y_{QO_k} = g^{x_{QO_k}}, h_{QO_k} = g^{\beta_{QO_k}}, w_{QO_k} = e(g,g)^{\alpha_{QO_k}})$.

Pick $r, r_i \in Z_p$ where $i = \mathbb{A}_{\curlywedge}, ..., \mathbb{A}_{\ltimes}$ and set the re encryption key as $rk_{O_i \rightarrow QO_k} = (k = (y_\beta^{\alpha_A} g^r)^{1/\beta_{O_i}})$

For all i in A, $k_{i1} = g^r \cdot H(i)^{r_i}, k_{i2} = g^{r_i}$. Note that, the re encryption key is denoted as the role certificate for query organization. The proxy server *CS* maintains a lookup table where each entry is a query organization. For instance one entry is $(X_{bank}, rk_{y_{retail} \rightarrow X_{bank}})$, which means the re encryption key $rk_{y_{orgA} \rightarrow X_{orgB}}$ is stored corresponding to organization X_{orgB}.

5) ***Encryption****:* Enc1(pk, m): Let, $pk = (y = g^x, h = g^\beta, w = e(g,g)^\alpha)$, now

$$c = (m \cdot e(w', g)^s \cdot e(w', y)^s = e(w', g)^{s \cdot x}) \quad (1)$$

6) ***Data Storing****:* To outsource the sensitive cybersecurity data to cloud, the O_i determines the access policy $\mathbb{P}$ as shown in figure 2, then performs below step followed by sending the ciphertext to *CS*. Moreover, O_i calculates the hash of ciphertext and add it to the blockchain by calling $HashCTI(c)$. This allows the verification of the CTI data in the future. Let $pk = (y = g^x, h = g^\beta, w = e(g,g)^\alpha)$ and $\mathbb{P}$ is access tree which is T'. A polynomial q_n is selected for each of the nodes n in access tree T'. For n, the degree d_n is set,

so that $d_n = t_n - 1$. Here the t_n denotes threshold of a node n. It sets $q_n(0) = s$. Then it selects d_R which is other random points for defining q_R

$$\begin{aligned} &\bar{c} = T', C = m \cdot w^s, C' = h^s, \\ &l = 1 \ to \ L : (C_{l,1} = g^{q_l(0)}, C_{l,2} = H(att(l)^{q_l(0)})) \end{aligned} \tag{2}$$

7) ***Data Access by Query organization****:* When the QO_k requests to retrieve an encrypted data record from the *CS*, the CS first retrieves the corresponding re encryption key for QO_k from the blockchain using smart contract, then verifies if the re encryption key belongs to the requester by calling the smart contract $Verify(pk)$. By this contract the O_i checks whether the hash of requester is matched in the lookup table. if so, then re encryption key corresponding to the requester can be retrieved form the lookup table. performs following step of re encryption of the ciphertext. $\bar{c}$.

Let $c_A = (T', C, C', \{\bar{C}_{l,1}, C_{l,2}\}_{i=1,2...,L})$ and $rk_{A\to B} = (k, k_{i,1}, K_{i,2_{1,2,...,\mathbb{A}}})$. The re encryption algorithm is a recursive process.

$$ReEncNd_n(\bar{c_A}, rk_{A\to B}) = F_n \tag{3}$$

where

$$\begin{aligned} F_n = \frac{e(k_{z1}, C_{n1})}{e(k_{z2}, C_{n2})} &= \frac{e(g^r H(z)^{r_z}, g^{q_n(0)})}{e(g^{r_z}, H(z)^{q_n(0)})} \\ &= e(g,g)^{r.q_n(0)} \end{aligned} \tag{4}$$

When n is a non leaf node, the algorithm $ReEncNd_n(\bar{c_A}, rk_{A\to B})$ works as follows. For each child node v of n, it calls $ReEncNd_n(\bar{c_A}, rk_{A\to B})$ and stores the output as F_v. Let S_n be an arbitrary t_n sized set of child nodes v such that $F_v \neq \perp$. Otherwise, let the lagrangian coefficient $\Delta_{i,s}$ for $i \in Z_p$ and a set of S elements in Z_p be $\Delta_{i,s}(x) = \prod_{j\in S, j\neq i} \frac{x-j}{i-j}$ and we compute

$F_n = \prod_{v\in S_n} F_v^{\Delta_i, S_n'^0}$

where, $i = index(v),\ S_n' = (index(v) : v \in S_n)$

So,

$$\begin{aligned} F_n = \prod_{v\in S_n} (e(g,g)^{r.q_v(0)})^{\delta^{(}_{i,S_n'}0)} &= \prod_{v\in S_n} (e(g,g)^{r.q_n(i)})^{\delta^{(}_{i,S_n'}0)} \\ &= e(g,g)^r.q_n(0) \end{aligned} \tag{5}$$

In this way, $ReEncNd_{rt}(\bar{c_A}, rk_{A\to B})$ for the root node rt can be computed. If $T'(\mathbb{A}) = 1$ then we get $ReEncNd_{rt}(\bar{c_A}, rk_{A\to B}) = e(g,g)^{r.q_{rt}(0)} = e(g,g)^{r.s} = F_{rt}$ Next, the re encryption algorithm computes

$$\begin{aligned} e(k, C')/F_{rt} &= e((y_B^{\alpha_A} g^r)^{1/\beta_A}, h_A^s)/e(g,g)^{r.s} \\ &= e(y_B, g)^{s.\alpha - A} = w - A^{s.x_B} \end{aligned} \tag{6}$$

Finally, set $c_B = (c_1 = C = m.w_A^s.c_2 = w_A^{S.x_B})$

Note that, storing the hash of ciphertext and the re-encryption key in the blockchain allows QO_k to verify the *CS* operations, therefore *CS* can not collude with other entities to provide an invalid CTI information to QO_k.

8) ***Decryption****:* The decryption is performed as follows by the query organization as follows. Let $\bar{c} = (T', C, C'...)$, the decryption of second level ciphertext is done by

$$m = C/e(g, C')^{\alpha/\beta^{-1}} \tag{7}$$

B. Revocation

In [7] we presented that the cybersecurity exchange framework using the CP-ABE is time bounded which is variant of [23]. As stated by [8] on the limitations of time bounded CP-ABE, the decryption key can work for certain time. Therefore, the cybersecurity exchange mechanism of [7] will suffer from real time revocation of decryption capabilities. Moreover, in [7], the organizations has to obtain new access attributes for new program and if any organization is revoked, the attributes are also revoked. Such methods may nit be scalable for organisation revocation in encryption of cloud data since the revoked entity should be in the ciphertext and regarded as a distinct attribute. The problem grows when the number of organization raises. On the other hand, using the proposed mechanism, the re-encryption key can be erased to revoke any organization instead of revoking the organizations' attributes.

VI. Privacy Analysis

According to the protocol, there is no pre-sharing of secret keys. The O_i is able to compute $rk_{O_i\to QO_k}$ using only the sk_{O_i} and pk_{QO_k}; which means the query organizations does not need to share its secret key with O_i or any other party in order to perform proxy re-encryption. Moreover, there is no way for the *CS* to decrypt the ciphertext that is stored in its storage. The re-encryption key $rk_{O_i\to QO_k}$ received from O_i is $y_\beta^\alpha g^r)^{1/\beta}$. From this message, it is computationally hard to perform discrete logarithm to find α and β which are secret keys. Since the *CS* cannot find the secret keys, it is not able to determine the sensitive information encrypted by the O_i. The *CS* is also unable to find the secret key of QO_k. While re-encrypting the ciphertext, the *CS* only uses the re-encryption key received from the O_i and the ciphertext $\bar{C_{O_i}}$ of O_i. We have already discussed earlier that from the re-encryption key the *CS* is unable to find secret keys. Moreover, the ciphertext encrypted under the public key of O_i is semantically secure, i.e., it is also computationally hard to find the message encrypted from the ciphertext. Upon receiving the re-encrypted ciphertext, only the QO_k is able to decrypt it as it was encrypted by the proxy server using the re-encryption key.

VII. Performance Analysis

In our experimental setup we used Ethereum as a blocckahin platform and some of the computations are carried offline (mostly encryotions and key generations). The offline computations are carried out using java (the crypographic algorithms are implemented in Bounty Castle Java Library). The main idea behind using both offline and online is to keep blockchain platform as light as possible and its feasible and practical that users usually perform the encryption locally. The storing and communication of the data is stored in blockchain so that the system becomes tamper proof and verifiable. Using the chain

of hashes of transactions among the entities, later we can verify the transactions. Below, first we discuss the computation overhead of offline computations and then we will discuss the gas cost of our experiment in ethereum.

For the offline computations, we have utilized cpabe toolkit (http://acsc.cs.utexas.edu/cpabe/) for the implementation of CP-ABE in our framework. This toolkit uses PBC library and performs on a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field. In the registration phase, we apply the ECDSA digital signature for its efficiency. We have implemented ECDSA through OpenSSL library (https://www.openssl.org). ECDSAs elliptic curve is over a prime field of $n = 256$ bits. In the registration phase, O_i receives credentials from attribute authority and then present them to *Tm*. Credentials are ECDSA signatures. ECDSA signing operation takes 0.528 (ms) and ECDSA verification takes 0.512 (ms). The setup phase takes 9 ms to generate a public and secret key for the owner organization O_i. Once the QO_k sends the request to O_i it takes 0.1 ms to generate the re-encryption key for the *Tm* which is very efficient and less overhead for the organizations of *Tm* to manage. The important complexity of the system is performing the data storing operation which involves the encryption with the policy and data access by query organization which involves re-encryption by the proxy server. Note that both of these operations depends on the number of attributes or the leaf nodes of the access tree. Precisely the overhead is linear to the number of leaf nodes of the access tree.

To evaluate the performance we built a set of access conditions where the access trees are constructed with a combination of AND and OR gates. Some of the access trees consist of all AND gates so that all the leaf nodes are visited and used to perform the re-encryption by the proxy server. We conduct 10 runs for each experiment and record the average result. As the results depicted in Figure 3, the re-encryption algorithm takes about 2.5 seconds for 100 attributes where the are key generation takes only 0.5 seconds. Therefore it is clear that the time taken by the protocol is practical as it is highly unlikely to have any large number of attributes for an organization which is more than 100.

We have used the most popular Ethereum implementation, Go Ethereum (geth) which is written in Golang. Table 1 shows the smart contract functions and their gas cost. Recall that at the setup phase, the functions $StoreParams()$ stores the public parameters and using $RetrieveParams()$ an user can retrieve the public parameters when needed. These functions also stores the hashes of the parameters so that later one can verify the integrity efficiently. These functions takes only 945 and 854 gas units which are equivalent to 0.0002 USD only. In the registration phase, the $OrgReg()$ function is used to register a new organization using its ID, public key and attributes which takes 952 gas units. In the re encryption generation phase, the query organization sign its own public key and uploads to the blockchain using $RegPubkey()$ function which takes about 961 gas units. Once the re-encryption key is generated it is stored in the blockchain using $StoreReKey()$ function which consumes about 924 gas units. After the first

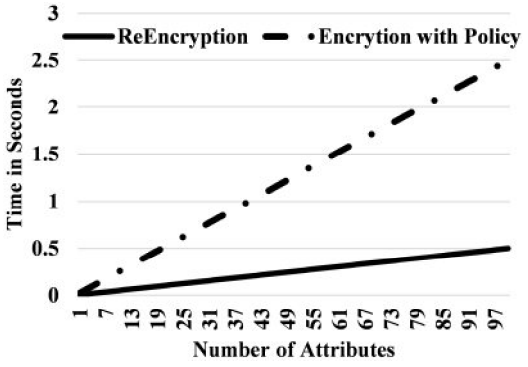

Fig. 3. Performance Result

level encryption, the data owner store the hash of encryption by calling $HashCTI()$ which takes 859 gas units. Once an organization requests for re encryption it needs to send the public key if it claims to have re encryption key stored in blockchain before. In this case, it sends the public key to blockchain and smart contract verifies if the requester's re encryption is already in lookup table by calling $Verify()$ contract which takes about 847 gas units. Our experimental methods can be extended providing more smart contract functions, however, for this work we limit our smart contract functions within above discussed phases. From table 1 it is also clear that our protocol consumes very negligible amount of gas units and total cost of our protocol is not more than 0.001 USD, where 1 ether = 10^9 Gwei.

TABLE I
GAS COST OF BLOCYNFO-SHARE PROTOCOL

Function	Gas Used	Ether Cost	USD
$StoreParams()$	945	9.45^{-7}	0.0002
$RetrieveParams()$	854	8.54×10^{-7}	0.00018
$OrgRegistration()$	952	9.52×10^{-7}	0.0002
$RegPubKey()$	961	9.61×10^{-7}	0.0002
$StroreReKey()$	924	9.24×10^{-7}	0.0002
$HashCTI()$	859	8.59×10^{-7}	0.00018
$Verify()$	847	8.47×10^{-7}	0.00018

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose BloCynFo-Share for cybersecurity information sharing among various organizations in a privacy-preserving manner using blockchain based framework with fine grain access control mechanism. To ensure the tamper-proof and non-repudiation, the blockchain has been used so that the record can be stored as a chained hash in the blockchain and can be audited and trace-back in case any malicious activity happens. More specifically, because of blockchain, any organization cannot change the encrypted information which is stored in the cloud. Also, any requester organization can not deny that it did not receive the information since all the transactions are stored in the blockchain. Also, because of using proxy re-encryption and attribute-based encryption the owner organization can select which organization can get access to the encrypted information shared by it. On the other side, because of using proxy re-encryption even the attributes match with the policy, the owner

organization can limit the access of requester by not generating the proxy key for re-encryption. The privacy analysis and performance results show that there is no information leaked while exchanging the cryptographic parameters and encrypted values, and the proposed system is efficient as well as scalable. In future we will explore homomorphic based cryptography to perform private analytics [27]–[29] besides the access control mechanism to build fast and secure proactive cyber defense.

REFERENCES

[1] D. K. Tosh, M. Molloy, S. Sengupta, C. A. Kamhoua, and K. A. Kwiat, "Cyber-investment and cyber-information exchange decision modeling," in *2015 IEEE 17th International Conference on High Performance Computing and Communications*. IEEE, 2015, pp. 1219–1224.

[2] D. Tosh, S. Sengupta, C. Kamhoua, K. Kwiat, and A. Martin, "An evolutionary game-theoretic framework for cyber-threat information sharing," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 7341–7346.

[3] D. K. Tosh, S. Sengupta, S. Mukhopadhyay, C. A. Kamhoua, and K. A. Kwiat, "Game theoretic modeling to enforce security information sharing among firms," in *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*. IEEE, 2015, pp. 7–12.

[4] I. Vakilinia, D. K. Tosh, and S. Sengupta, "3-way game model for privacy-preserving cybersecurity information exchange framework," in *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 829–834.

[5] D. E. Denning, "Framework and principles for active cyber defense," *Computers & Security*, vol. 40, pp. 108–113, 2014.

[6] R. A. Martin, "Making security measurable and manageable," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*. IEEE, 2008, pp. 1–9.

[7] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Attribute based sharing in cybersecurity information exchange framework," in *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. IEEE, 2017, pp. 1–6.

[8] Y. Yang, H. Zhu, H. Lu, J. Weng, Y. Zhang, and K.-K. R. Choo, "Cloud based data sharing with fine-grained proxy re-encryption," *Pervasive and Mobile Computing*, vol. 28, pp. 122–134, 2016.

[9] A. Kelarev, X. Yi, S. Badsha, X. Yang, L. Rylands, and J. Seberry, "A multistage protocol for aggregated queries in distributed cloud databases with privacy protection," *Future Generation Computer Systems*, vol. 90, pp. 368–380, 2019.

[10] A. Thakkar, S. Badsha, and S. Sengupta, "Game theoretic approach applied in cybersecurity information exchange framework," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 2020.

[11] S. Badsha, X. Yi, I. Khalil, and E. Bertino, "Privacy preserving user-based recommender system," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1074–1083.

[12] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, and K.-Y. Lam, "Privacy preserving user based web service recommendations," *IEEE Access*, vol. 6, pp. 56 647–56 657, 2018.

[13] I. Vakilinia and S. Sengupta, "A coalitional game theory approach for cybersecurity information sharing," in *Military Communications Conference, MILCOM 2017-2017 IEEE*. IEEE, 2017, pp. 237–242.

[14] D. Tosh, S. Sengupta, C. A. Kamhoua, and K. A. Kwiat, "Establishing evolutionary game models for cyber security information exchange (cybex)," *Journal of Computer and System Sciences*, vol. 98, pp. 27–52, 2018.

[15] I. Vakilinia, D. K. Tosh, and S. Sengupta, "Privacy-preserving cybersecurity information exchange mechanism," in *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. IEEE, 2017, pp. 1–7.

[16] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1998, pp. 127–144.

[17] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[18] R. H. Deng, J. Weng, S. Liu, and K. Chen, "Chosen-ciphertext secure proxy re-encryption without pairings," in *International Conference on Cryptology and Network Security*. Springer, 2008, pp. 1–17.

[19] K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, Y. Yu, and A. Yang, "A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing," *Future Generation Computer Systems*, vol. 52, pp. 95–108, 2015.

[20] A. Manzoor, M. Liyanage, A. Braeken, S. S. Kanhere, and M. Ylianttila, "Blockchain based proxy re-encryption scheme for secure iot data sharing," *arXiv preprint arXiv:1811.02276*, 2018.

[21] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[22] S. Chakraborty and I. Ray, "Trustbac: integrating trust relationships into the rbac model for access control in open systems," in *Proceedings of the eleventh ACM symposium on Access control models and technologies*. ACM, 2006, pp. 49–58.

[23] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[24] M. Chase, "Multi-authority attribute based encryption," in *Theory of cryptography conference*. Springer, 2007, pp. 515–534.

[25] I. Vakilinia, S. Vakilinia, S. Badsha, E. Arslan, and S. Sengupta, "Pooling approach for task allocation in the blockchain based decentralized storage network," in *15th International Conference on Network and Service Management. IEEE*, 2019.

[26] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[27] S. Badsha, X. Yi, and I. Khalil, "A practical privacy-preserving recommender system," *Data Science and Engineering*, vol. 1, no. 3, pp. 161–177, 2016.

[28] S. Badsha, I. Vakilinia, and S. Sengupta, "Privacy preserving cyber threat information sharing and learning for cyber defense," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0708–0714.

[29] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, E. Bertino, and K. Lam, "Privacy preserving location-aware personalized web service recommendations," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.