CRYSTAL: A PRIVACY-PRESERVING DISTRIBUTED REPUTATION MANAGEMENT

Ngoc Hong Tran[1], Tri Nguyen[2], Quoc Binh Nguyen[3],
Susanna Pirttikangas[2], M-Tahar Kechadi[4,5]

[1]Vietnamese-German University, Vietnam
ngoc.th@vgu.edu.vn
[2]Center for Ubiquitous Computing, University of Oulu, Finland
tri.nguyen@oulu.fi
susanna.pirttikangas@oulu.fi
[3]Ton Duc Thang University, Vietnam
nguyenquocbinh@tdtu.edu.vn
[4]School of Computer Science, University College Dublin, Ireland
[5]Insight Centre for Data Analytics, University College Dublin
tahar.kechadi@ucd.ie

ABSTRACT

This paper investigates the situation in which exists the unshared Internet in specific areas while users in there need instant advice from others nearby. Hence, a peer-to-peer network is necessary and established by connecting all neighbouring mobile devices so that they can exchange questions and recommendations. However, not all received recommendations are reliable as users may be unknown to each other. Therefore, the trustworthiness of advice is evaluated based on the advisor's reputation score. The reputation score is locally stored in the user's mobile device. It is not completely guaranteed that the reputation score is trustful if its owner uses it for a wrong intention. In addition, another privacy problem is about honestly auditing the reputation score on the advising user by the questioning user. Therefore, this work proposes a security model, namely Crystal, for securely managing distributed reputation scores and for preserving user privacy. Crystal ensures that the reputation score can be verified, computed and audited in a secret way. Another significant point is that the device in the peer-to-peer network has limits in physical resources such as bandwidth, power and memory. For this issue, Crystal applies lightweight Elliptic Curve Cryptographic algorithms so that Crystal consumes less the physical resources of devices. The experimental results prove that our proposed model performance is promising.

KEYWORDS

Reputation, peer to peer, privacy, security, homomorphic encryption, decentralized network.

1. INTRODUCTION

Nowadays the Internet covers almost everywhere, which eases people in communication. However, the Internet is not completely accessible freely through Wi-Fi in a number of places. Whenever someone wants to use the Internet at a certain place for free to ask for some urgent information, (s)he has to be their customer, and the Internet connection owns limits. Moreover, the Internet fee is still not trivial to everyone. The other cases in which people cannot access the Internet are many. For instance, people forget to renew the mobile Internet payment before going out, or just arrive at an airport in a new country without a new sim card, etc. Therefore, there is really a need for a peer-to-peer network so that people in the aforementioned situations can send questions and recommendations, especially for an emergency. In the meanwhile, the swift growth of mobile devices today, in both quality and quantity, requires a specific type of network for exploiting their emergence that is to comfort users by the availability of services.

One of those networks can be listed as Near-me Area Network (NAN) by Wong [1]. In NAN, the connection among nearby wireless devices (i.e., smartphones) is deployed instead of using the Internet to connect different LANs or cellular networks. Nodes in NAN can freely exchange or provide information to each other without the need for the Internet. This type of network properly fits the aforementioned lacking Internet but needing connection cases. Thus, it is necessary to apply this decentralized network architecture to allow people in an area to communicate together without the Internet.

However, security and privacy issues raise when we apply this decentralized network to those users (or, nodes) who are strangers. Nodes concern the reliability of the ones who are communicating to them. Hence, there is a need for a tool measuring the trustworthiness of each node. For that purpose, the reputation system [2] has been a competent candidate for this issue. In particular, each user holds its own reputation value based on previous feedbacks by the others. This reputation value indicates the trust level of each node. However, due to the fear of retaliation [3], the users can avoid giving correct feedback. Thus, privacy including confidentiality of feedback [4] is an important feature of reputation systems. Additionally, the reputation systems [5, 6] considers that each node needs to trust several nodes before joining the network. Meanwhile, Petrlic et al.'s system [7] requires a powerful computation with a use of Paillier cryptosystem [8]. Hence, this idea cannot apply directly to mobile carrier networks as the public key scheme consumes much time and computing performance.

To cope with the drawbacks of the related studies, our work contributes a privacy-aware protocol, namely Crystal, with its services as in Table 1. More specifically, for Crystal, we propose a reputation system basis providing the core activities of storing, computing, evaluating reputations scores of advising users, and propose a secure model of computing and evaluating reputation scores without allowing the both questioning user and advising user to intervene the reputation grading process or to learn anything from that process. Nodes in Crystal model are mobile devices with a limit in bandwidth, power and memory. Therefore, the cryptographic algorithms that are selected for Crystal are the lightweight ones so that they cannot exceed the restricted physical resources of Crystal nodes. More specifically, Crystal model leverages homomorphic Elliptic Curve Cryptography (ECC) algorithm as a lightweight encryption to optimize computing performance in comparison with [7]. Moreover, Crystal's participants do not require the knowledge of others prior to accessing.

Table 1. Services of reputation management system and its privacy-preserving version.

Reputation Management	**Privacy Preserving**
Store reputation score	Computing the encryption - Secure arithmetic operators - Secure comparative operators
Convert reputation score to reputation level	
Compute reputation score	
Evaluate reputation score	Evaluating the encryption
Update the reputation score	Protecting the integrity

The rest of this paper is organized as follows. Section 2 is about current privacy-preserving reputation managements in decentralized networks. A proposed reputation management protocol is described in Section 3. The privacy issues and the privacy-preserving protocol is presented in Sec 4. The experiment is described in Section 5. Eventually, Section 6 concludes the work.

2. RELATED WORKS

A study related to decentralized privacy-preserving reputation was proposed by Hasan et al. [5, 6] through the utilization of set technologies including set-membership, plain-text equality, non-interactive zero-knowledge proof and additive homomorphic cryptosystem. Since the use of a sub-set of each member in the network for communications, the authors mentioned the fewer number of messages transmitted in the network than a previous study by Gudes et al. [9]; however, in [5, 6] each node needs to trust in a small set of participants before proving them about its correct shares as feedback based on zero-knowledge proof. Note that the correct shares from the node are not directly sent to its trusted nodes, they are forwarded by several middle nodes before obtaining the trusted nodes instead. Once receiving the messages, those middle nodes collect and add that information to the knowledge. As a consequence, those reputation systems focus on confidentiality-preserving more than privacy where the list of users joining the rating is not hidden.

Another study in [16] deploys data anonymization techniques to conceal the identity of rating users. Another work in [17], the authors used a pseudonym-based scheme to protect the user's reputation that is simply the number of cryptographic "votes". Although anonymization and pseudonymity are still effective in specific applications, however, in our work, the protocol is more complicated to use those techniques. Moreover, recently some works [18, 19] deploying the blockchain to protect the user identity from their used web services. Some other works in [20-22] create a tamper-proof ledger that is delivered among all participants. This technique can the data recorded in the ledger safe from illegally changing by the other unauthorized users and is applied in a diversity of contexts as in voting system [24], machine-to-machine environment [23], and identifier systems [25]. Another work that is the closest to our work is [7]. Petrlic et al. [7] showed a study on the privacy-preserving reputation management. Both reliability and privacy protection are based on cryptographic algorithms Paillier cryptosystem [8], a public-key encryption scheme, and the zero-knowledge proof [10]. However, this approach allows a reputation provider (RP) who manages providers' reputation value rated by users. In detail, when a user desires to rate any service provider, she/he is to be allowed by the RP. To rate service providers, users utilize a public key scheme to sign and prove with RP that the message is correct exploiting the zero-knowledge proof. Using Paillier scheme leads to a requirement of time and powerful computation. Therefore, in our work, as we address the resource-related issues in Section 1, we apply the lightweight ECC algorithm to protect the data and make the cost low to satisfy the restriction of the node's physical resources.

3. CRYSTAL - REPUTATION MANAGEMENT MODEL

Let us consider a specific scenario that includes many users in a close area. Each user keeps a mobile device installed with *Crystal application* denoted as *app*. It is assumed that each user has a different experience about places surrounding their position. For example, each user knows different money exchange agencies with the best price. As a user sends a request for advice through the app, we call this user *requester*. If the other peers receive the request and know the answer, they will make a recommendation to the requester. The answering user is called *advisor*. Actually, in a popular communication way, Crystal users can communicate with each other using the Internet. However, in the specific case when the area is uncovered with the Internet, each node[1] (i.e., mobile device) needs Bluetooth or direct Wi-Fi. When all mobile devices are connected, they create a peer-to-peer (P2P) Crystal network (see Figure 1). The nodes contact together inside their physical range. For example, Bluetooth V2.1 has a 10 - 100 meter physical range on theory, whereas direct Wi-Fi has a 45-200 meter physical range [12]. When a peer enters the physical range of another peer's, they both can send messages to each

[1]The three terms node, user and mobile device are used interchangeably. This work focuses on the application layer, not on the network layer.

other. For example, in Figure 1, there are three overlapping P2P networks. Network 1 (i.e., green dashed circle) involves $\{u_0, u_1, u_2\}$ while Network 2 (i.e., yellow dashed circle) involves $\{u_1, u_5\}$, and and Network 3 (i.e., purple dashed circle) involves $\{u_3, u_4, u_5\}$. Node u_1 is in the network 1 and 2. Node u_5 is in networks 2 and 3. When a node moves out of one network, it can enter the other networks. Each user has his/her own memory which summarizes the most frequently used data, the remaining data is saved in the cloud storage.

Moreover, not all Crystal users know each other. Although a requester receives advice from their peers, the advice is not trustworthy enough for the requester until it is evaluated. However, the advice itself cannot contain adequate proof of the honesty of its owner. Therefore, the advisor has to give his/her requester the evidence to prove that their recommendation is trustworthy. For that purpose, in a P2P network, a user can ask the neighbours for the reputation score of the advisor. However, there is a case that the neighbours do not have any information of that advisor, or in case they compromise to cheat on the requester.

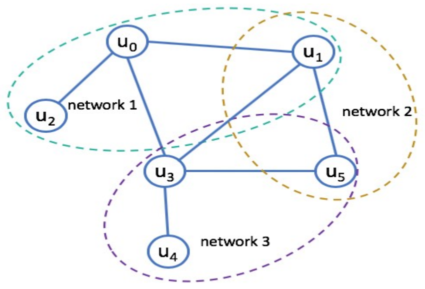

Figure 1: P2P Communication among Nodes in Crystal

In this work, each user keeps his reputation score and provides this score to his requester. This method can violate the reputation score if the owner is not honest. More details of how to protect the reputation score's integrity are discussed in Section 4. Then, the requester can evaluate advisor's reputation level. More specifically, the way to calculate a reputation score is defined as follows. Note that in this paper, we denote the requester as v and the advisor as u.

Definition 1 (Reputation Score). *Let u be an advising user who receives the grades from its requesters. Let N be the number of users attending to grade the reputation of u. Let P_i, with $i = 0, \cdots, (N-1)$ and $P_i \in [0, 100]$, be the mark which u is graded by user i. The reputation score of u denoted as R_u that is computed as follows:*

$$R_u = \frac{\Sigma_{i=0}^{(N-1)} P_i}{N}$$

Example 1. Consider the case of $N = 5$ users grading user u sequentially with {50, 60, 30, 70, 20}, respectively. Therefore, $R_u = \frac{(50+60+30+70+20)}{5} = \frac{230}{5} = 46$.

Commonly, the reputation score that is used for evaluation contains only positive scores made by the others. Hence, it is not fair enough as aforementioned. A reputation evaluation is fairer when each user does not only keep the positive scores (i.e., $R_{V_{3pos} \to u}$) made by his past requesters, but also the negative scores (i.e., $R_{V_{2neg} \to u}$) they made for him. The reputation grades, which the user u made for his past advisors, are also needed for computing the reputation level, i.e., $R_{u \to neg\, V_0}$ and $R_{u \to pos\, V_1}$. Notice that a requester cannot give positive and negative grades at the same time for the same advisor after using his advice. Remarkably, whenever the users connect to the Internet, for user u to get an honesty evaluation for itself from the server, these values $R_{u \to neg\, V_0}$ and $R_{u \to pos\, V_1}$ are requested. They are saved in the local memory when *u* grades his advisor. Therefore, in this work, we propose that a user always keeps a tuple of four reputation scores as in Definition 1.

Definition 2 (Reputation Score Tuple). *Let u be the advisor. Let V_0, V_1 be two sets of u's past advisors. Let V_2, V_3 be two sets of u's past requesters. Let RT_u be a quadruple of reputation scores of user u in Crystal model. More formally,*

$$RT_u = [R_{u \to negV_0} | R_{u \to posV_1} | R_{V_2 neg \to u} | R_{V_3 pos \to u}]$$

where $R_{u \to neg\, V_0}$ is the total negative reputation score which u grades his past advisors v, $R_{u \to pos\, V_1}$ is the final positive reputation score which u reduces the scores of the others v, $R_{V_{2neg} \to u}$ is u's reputation score which u reduces the scores of the others v, $R_{V_{3pos} \to u}$ is u's reputation score which u reduces the scores of the others v. Moreover, each element in RT_u is defined in Definition 1.

Example 2 Let users u_0 be an advisor. Let $V_0 = \{u_1, u_2, u_3\}$ be the set of users whom u_0 negatively evaluates, $V_1 = \{u_3, u_4\}$ be the set of users whom u_0 positively evaluates, $V_2 = \{u_3, u_5\}$ be the set of users negatively evaluating u_0, and $V_3 = \{u_1, u_2, u_4\}$ be the set of users positively evaluating u_0. Assume that we have $R_{u_0 \to neg\, V_0} = 60$, $R_{u_0 \to pos\, V_1} = 40$, $R_{V_{2neg} \to u_0} = 40$, and $R_{V_{3pos} \to u_0} = 60$. Thus, as in Definition 2, we have a reputation score tuple RT_u with RT_u = [60, 40, 40, 60].

When *v* receives the reputation score tuple from *u*, *v* retrieves the reputation level of *u* before following *u*'s advice. The reputation level is defined as follows.

Definition 3 (Reputation Level). *Let u be the advisor. Let R_u be the reputation score of user u as in Definition 1. Let L_u be the reputation level of u. The reputation level of u is classified in different ranges of scores as below.*

$$f_L(R_u) = \begin{cases} Low & if\ L_1 \le R_u < L_2 \\ Average & if\ L_2 \le R_u \le H_1 \\ High & if\ H_1 < R_u \le H_2 \end{cases}$$

where L_1, L_2 are the minimum and maximum values of the level Low, H_1 and H_2 are the minimum and maximum values of the level High.

Figure 2 illustrates Definition 2. L_1, L_2, H_1, H_2 can be customized depending on each specific system.

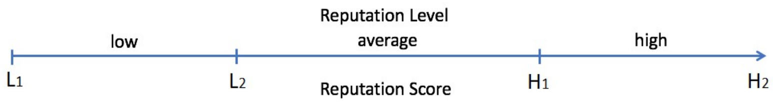

Figure 2: Reputation Level vs Reputation Score.

Evaluating Reputation. To convert a given reputation score tuple to the reputation level, *v* applies Algorithm 1. The input data of Algorithm 1 is the reputation score tuple RT_u as in Definition 2 and the output is the reputation level of *u*, i.e., L_u as in Definition 3. The reputation level is initialized with NULL as it does not contain any reputation level of *u* (line 1). Then, the levels of each reputation score, which the other users grade *u*, are converted based on Definition 2 (lines 2, 3). If the reputation level of the score which the other users positively grade *u* (i.e., f_{L_4}), is high, it means that the reputation level of *u* is high (lines 4, 5, 6). If the reputation level of the score which the other users negatively grade u (i.e., f_{L_3}) is high (lines 8, 9, 10), it means that the reputation level of *u* is low. Moreover, in the other case when the reputation evaluation of u for his past advisors, i.e., $R_{u\to neg\,V_0}$ is extremely high in the range MAX (e.g., MAX = [95%, 100%]) while $R_{u\to pos\,V_1}$ is extremely low in the range MIN (e.g., MIN = [0%, 5%]), the reputation level of *u* is low (lines 12, 13, 14). This case happens to avoid that the requesting user's grading is not correct to degrade the others, as it is not reasonable when most of advisors give bad recommendations to decrease their reputation. The other cases can be covered (line 16) since the positive and the negative evaluation percentages of the same type are complemented, which means their addition is 100%, and one user cannot grade an advisor negatively and positively for the same advice.

```
Algorithm 1 evaluateReputation()
Input: $RT_u = \{R_{u\to negV_0}, R_{u\to posV_1}, R_{V_2neg\to u}, R_{V_3pos\to u}\}$
Output: $L_u$
 1: $L_u = NULL$;
 2: $f_{L_3} = f_L(R_{V_2neg\to u})$;
 3: $f_{L_4} = f_L(R_{V_3pos\to u})$;
 4: if ($f_{L_4} == High$) then
 5:     $L_u = High$;
 6:     return $L_u$;
 7: end if
 8: if ($f_{L_3} == High$) then
 9:     $L_u = Low$;
10:     return $L_u$;
11: end if
12: if ($R_{u\to negV_0} \in MAX$ AND $R_{u\to posV_1} \in MIN$) then
13:     $L_u = Low$;
14:     return $L_u$;
15: end if
16: return $L_u$;
```

Computing Reputation. After evaluating the reputation level of *u*, *v* then decides if *v* follows the recommendation of *u*. Then, *v* can evaluate the advisor by adding or subtracting the reputation scores by a point denoted as M_v, depending on how successful v finds from the advice of *u*. As in Algorithm 2, we have *N* as the number of requesters evaluating *u*. if L_u is low, which means that the reputation of *u* is low (line 1), *v* then increases the negative points for *u* by M_v (line 2) and updates the positive points for *u* as well (line 3). In case L_u is high, *v* updates the negative points for *u* (line 5), and increases the positive points by M_v (line 6). After all, the number of requesters *N* is increased by one (line 8). The results are returned including the updated reputation score tuple RT_u and the number of requester *N*.

```
Algorithm 2 computeReputation()
Input: $L_u, N, M_v$,
    $RT_u = \{R_{u\to negV_0}, R_{u\to posV_1}, R_{V_2neg\to u}, R_{V_3pos\to u}\}$
Output: $RT_u = \{R_{u\to negV_0}, R_{u\to posV_1}, R_{V_2neg\to u}, R_{V_3pos\to u}\}$, $N$.
 1: if ($L_u == Low$) then
 2:     $R_{V_2neg\to u} = (R_{V_2neg\to u} * N + M_v)/(N+1)$;
 3:     $R_{V_3pos\to u} = (R_{V_3pos\to u} * N)/(N+1)$;
 4: else
 5:     $R_{V_2neg\to u} = (R_{V_2neg\to u} * N)/(N+1)$;
 6:     $R_{V_3pos\to u} = (R_{V_3pos\to u} * N + M_v)/(N+1)$;
 7: end if
 8: $N = N + 1$;
 9: return $RT_u, N$;
```

Updating Reputation Score. New scores are then updated directly if *u* and *v* are still in the radio range of each other. In this work, we focus on this case. Otherwise, another case is that the scores are modified when they connect to the Internet. Then, the requester *v* moves to a new position, it may leave their network, and create a new network with a sum of neighbours which may be old or new to *v*. In case, a new network includes a few old nodes with *u*, *v* can send a request to the whole network to ask for the experience with the advice of u. Based on that, *v* can decide how much trust v has for *u*. In this work, we do not focus on this direction. Our work focuses on the case that no node connecting the advisor in the previous transactions, the requester then computes the reputation score based on the data which the advisor gives it. However, a trust question happens as the advisor's reputation scores given to the requester can be forged by the advisor.

4. CRYSTAL - PRIVACY PRESERVING REPUTATION PROTOCOL

4.1. Privacy Requirements

When a requester *v* requests for the information from an advisor *u*, *u* sends *v* its reputation score tuple (see Definition 2) so that *v* can evaluate the reputation of *u* before sending *v* more information of the answer. In this situation, leaking data can happen, so the privacy requirements are needed.

(1) *Non-violated Reputation Scores against Requester*: This issue focuses on the requesters in case they try to learn the reputation scores received from the advisor. The requester *v* can misuse the received reputation score of *u*'s, e.g., impersonating the owner's received reputation score. Hence, there is a need to protect the reputation score computation.

(2) *Reputation Score Integrity against Advisor*: This privacy issue focuses on the advisor *u* in case they counterfeit their reputation scores before sending them to the requester *v*. So *v* needs to have a solution to check if the received reputation score is integrity.

In this work, we focus on proposing a secure solution to avoid the two above issues.

4.2. Homomorphic Elliptic Curve Cryptography (ECC)

ECC is a public schema key system based on the elliptic curve that is defined by an equation of the form $y^3 = x^3 + ax + b$. In ECC, the logical operators, such as Exclusive OR (+), are used to speed up the computation. As a quick view, assume that Bob wants to send message M to Alice. She generates a pair of keys including a public key *k.B* and a private key *k*, where the operator "." implies the scalar point multiplication, and B is a base point. Then, *k.B* is shared with Bob so that he can use it to encrypt *M* into a ciphertext; $Enc_{k.B}(M)$. We have $Enc_{k.B}(M) = (r.B, M + r.k.B)$ where *r* is a random number generated by Bob. Once Alice receives the encryption from Bob, she uses her private key to read the message *M* without requesting *r* from Bob, $Dec(Enc_{k.B}(M)) = M + r.k.B + r.B.k = M$.

4.3. Encryption-driven Crystal Protocol

In this section, the Crystal protocol operates as presented in Section 3 but in a way that the Crystal protocol is protected with the encryption algorithms against the privacy issues addressed in Section 4.1. In this work, we accept a server which is only responsible for storing data of all nodes, providing the keys for security purposes, and evaluating the reputation of all nodes when the nodes connect to it through the Internet. The last purpose aims that the Crystal server bans or reduces the reputation score of the node that is identified to be dishonest or malicious.

a) For the first privacy issue caused by the dishonest requester. Let (PK, SK) be a pair of public and private keys, respectively, of the Crystal server. The reputation scores are encrypted by a public key of the server, i.e., PK. No users can read the plain text encapsulated inside the encryption. Therefore, all computations at the users are performed in a blind way, i.e., in a secret and secure way. For this goal, we apply the ECC algorithm (see Section 4.2) to encrypt the reputation scores to gain the secure Crystal reputation score as in Definition 3.

Definition 4 (Crystal Reputation Score). *Let u be the advisor. Let PK be the public key of Crystal server. Let RT_u be the reputation score tuple as in Definition 2. The Crystal reputation score tuple $Enc_{PK}(RT_u)$ is a set of encryptions of each element in the tuple. More formally,*

$$Enc_{PK}(RT_u) = [Enc_{PK}(R_{u\to negV_0})|Enc_{PK}(R_{u\to posV_1})$$
$$|Enc_{PK}(R_{V_2neg\to u})|Enc_{PK}(R_{V_3pos\to u})]$$

where V_0, V_1, V_2, V_3, $R_{u\to neg\,V_0}$, $R_{u\to pos\,V_1}$, $R_{V_{2neg}\to u}$, $R_{V_{3pos}\to u}$ are defined as in Definition 2.

```
Algorithm 3 secureMultiply()
Input: $Enc_{PK}(x), N$
Output: $Enc_{PK}(nx)$
1: for $(i = 0; i < (N-1); i++)$ do
2:     $Enc_{PK}(nx) = Enc_{PK}(nx) + Enc_{PK}(x)$;
3: end for
4: return $Enc_{PK}(nx)$;
```

There are two types of computing operators in Crystal, that is, arithmetic and comparative operators (see Algorithms 1 and 2).

Secure Arithmetic operators. The arithmetic operators such as + (addition), - (subtraction), * (multiply), / (division) mostly used in Algorithm 2. The reputation scores are encrypted by ECC (see Definition 4) with the public key provided by Crystal server *PK*. Operators + and – are can be enforced easily by ECC as they are basic operators of ECC. For example with operator +, the addition encryption of the two values x_1 and x_2 is performed as follows: let $Enc_{PK}(x_1)$ and $Enc_{PK}(x_2)$ be encryptions of x_1 and x_2 responsively. The encryption of the addition of x_1 and x_2 is $Enc_{PK}(x_1+x_2) = Enc_{PK}(x_1) + Enc_{PK}(x_2) = (r_1B, x_1.r_1.PK.B) + (r_2B, x_2.r_2.PK.B) = (B(r_1 + r_2), (x_1 + x_2 + PK.B.(r_1 + r_2))$ (refer Section 4.2). Then, to execute the operator * that is the multiplication of encryptions, we can exploit the operators +. The operator * of two operands *$Enc_{PK}(x)$* and *N* are executed using the operator + as in Algorithm 3. There is a loop of *$(N - 1)$* times (line 1) of adding *$Enc_{PK}(x)$* into the output data *$Enc_{PK}(nx)$* (line 2). Hence, we have $\Sigma_{i=0}^{(N-1)} Enc_{PK}(x) = Enc_{PK}(x + \cdots + x) = Enc_{PK}(Nx)$. For the operator / of *$Enc_{PK}(x)$* and *N*, we need to calculate the inverse value of the denominator, denoted N^{-1}, by a binary polynomial inversion calculation. The ECC operators are used for enforcing the operators in Algorithm 2.

Moreover, Crystal executes the comparison operations on the encrypted reputation scores (see Definition 4) for evaluating the reputation level as in Definition 3 or for checking conditions to re-compute the reputation scores (see Algorithm 2).

Secure comparative operators. For this type of comparative operator, we adopt the framework supporting the secure comparison in [13]. This protocol leverages the asymmetric encryption to secretly evaluate if an encryption satisfies a condition in the form of *cond = (encrypted variable, operator, threshold)*, where the operators include: $<, >, \leq, \geq, =, \neq$. Specifically, this protocol generates a token T for each condition. This is done by using the *T = GenToken(SK,< cond >)* function defined in [13], where SK is the Crystal server's private key generated according to [13]. Crystal server's SK is used to ensure that only Crystal server can generate tokens containing the secret conditions unknown to the unauthorized nodes. In order to evaluate whether the encryption of value *x* with the corresponding public key PK, i.e., *$Enc_{PK}(x)$*, satisfies *cond*, we make use of the *Query()* function defined in [13]. This takes as input the encryption *$Enc_{PK}(x)$* and the token T generated for cond and returns a predefined message M, if cond is satisfied, ⊥, otherwise. As in Algorithms 2, a condition for if statement is *(L_u, =, "Low")*, and as in Algorithm 1, four conditions for if statement are *(f_{L_4}, =, "High")*, *(f_{L_3}, =, "High")*, *($R_{u\to neg\,V_0}$, =, 100)*, and *($R_{u\to pos\,V_1}$, =, 0)*. In order to use the function *Query()* for the above conditions, the variables in the conditions are encrypted. Therefore, we have *($Enc_{SK}(Lu)$, =, "Low")*, *($Enc_{SK}(f_{L_4})$, =, "High")*, *($Enc_{SK}(f_{L_3})$, =, "High")*, *($Enc_{SK}(R_{u\to neg\,V_0})$, =, 100)*, and

(EncSK($R_{u \rightarrow pos} v_1$), =, 0). The returned result can be of these conditions are "true" or "false". Moreover, we have three other conditions in Definition 3, that is, *($Enc_{SK}(R_u)$, <, 30)* returns "Low" or ⊥, while *($Enc_{SK}(R_u)$, >, 70)* and *($Enc_{SK}(R_u)$, ≤, 100)* return "High" or ⊥.

By this cryptographic method, we can achieve the first privacy requirement (see Section 4.1) in *protecting the reputation score against dishonest requesters.*

b) For the second privacy issue of data integrity against dishonest advisor. We adopt the digital signature mechanism, specifically, that is the Elliptic Curve Digital Signature Algorithm (ECDSA) [14]. Nodes receive or update the public keys of the other peers using the Crystal application periodically from the Crystal server when they can access the Internet. Each node has its own private key called SK_v. This private key is used for generating the digital signature of the secure reputation score tuple (see Definition 4) after the requester v uses u's advice and updates u's reputation scores. However, to make the following requester evaluate the integrity of the reputation scores, v needs to add its ID after the digital signature of the secure reputation score tuple as well, so that the following requester can obtain v's public key from v's ID and verify the digital signature to ensure that it was made by a requester different from u. We denote the data structure, consisting of the secure reputation score tuple, digital signature and v's ID, as reputation score of integrity. More formally,

Definition 5 (Reputation Score of Integrity). *Let u, v be the advisor and the requester, respectively. Let PK, SK_v be the public key of Crystal server and the private key of the requester v, respectively. Let $Enc_{PK}(R_u)$ be the secure reputation score tuple as in Definition 4. Let I_{R_u} be the reputation score of integrity of u. More formally,*

$$I_{R_u} = [Enc_{PK}(R_u) \| Sign_{ECDSA}(Hash(Enc_{PK}(R_u))) \| ID_v]$$

This reputation score of integrity I_{R_u} is used instead of the secure reputation score tuple. Only v has its own private key to make the digital signature, so the other nodes cannot know as well as cannot forge v's private key. Let w be the following requester. When w receives I_{R_u} from u, w extracts ID_v and checks if that is u's ID. If it is true, the data integrity is not ensured. Otherwise, w searches the public key of v, that is, PK_v. Then, w decrypts $Sign_{ECDSA}(Hash(Enc_{PK}(R_u)))$ by PK_v to get $h_1 = Hash(Enc_{PK}(R_u))$. Then, w hashes $h_2 = Enc_{PK}(R_u)$ to compare if $h_1 = h_2$. If they are equal, the data integrity is maintained. Otherwise, it is not. Then, the Crystal protocol is enforced and w updates the reputation scores, then generates the signature with its private key and attaches its ID_w into I_{R_u}.

5. EXPERIMENTATION

We compare our work with the work in [7] as both leverage the cryptographic algorithms although we do not solve the same problem. The difference in our approaches is that in [7] the authors use Paillier algorithm while we focus on the Elliptic Curve cryptographic algorithms.

Parameters. We perform several experiments using the cryptographic algorithms ECC, ECDSA and Paillier as well as their different key sizes. The experiments are performed on one PC of Windows 10 Professional, 8 GB RAM, CPU i7 1.80GHz. Each value in the experiments is an average of 20 execution times. The Koblitz Elliptic Curves, which are one kind of 15 recommended elliptic curves analysed in [15], are used. Their key sizes are variant in {233, 283, 409, 571} bits. SHA-2 key sizes are in {224, 256, 384, 512} bits. ECDSA key sizes are in {384, 521} bits. Paillier key sizes are in {1024, 2048, 3072, 4096} bits. Currently, these key sizes satisfy the NIST's conditions against the analysis attacks[2].

[2]https://www.keylength.com/en/4/

Time cost. We create 8 combinations of cryptographic algorithms' key sizes, that is, SHA key size, ECDSA key size = {(224, 384), (256, 384), (384, 384), (512, 384), (224, 521), (256, 521), (384, 521), (512, 521)}. For each pair of SHA-2 and ECDSA keys, the key sizes of ECC and Paillier are variant. So, we have 8 different values for each pair of SHA-2 and ECDSA keys. We compare the computing time in generating one reputation score of integrity for each case of key sizes of {SHA-2, ECDSA, ECC} and {SHA-2, ECDSA, Paillier} as in Figure 3. In all cases of using ECC, the computing time costs are always much less than using Paillier. In the smallest key sizes, of {SHA-2, ECDSA, ECC} = {224, 384, 233} and {SHA-2, ECDSA, Paillier} = {224, 384, 1024}, the combination of key sizes containing ECC costs 36,1ms while the combination of key sizes containing Paillier costs 359.9ms.In the highest key sizes of {SHA-2, ECDSA, ECC} = {512, 521, 571} and {SHA-2, ECDSA, Paillier} = {512, 521, 4096}, the combination of key sizes containing ECC costs 69,98ms while the combination of key sizes containing Paillier costs 390.5ms.

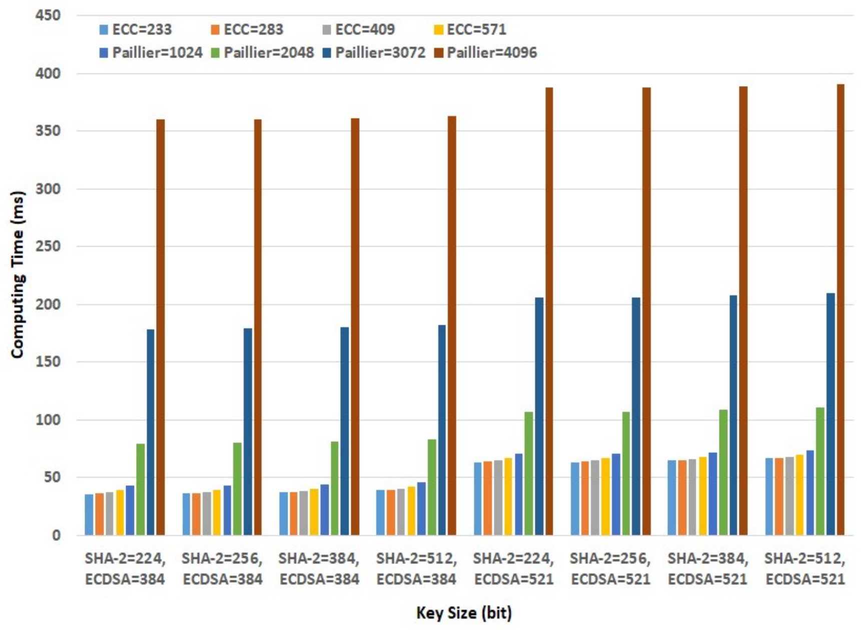

Figure 3: Time cost comparison.

Data Load. We create 8 combinations of key sizes of ECC and ECDSA, that is, (ECC, ECDSA) = {(233, 384), (283, 384), (409, 384), (571,384), (233, 521), (283, 521), (409, 521), (571,521)}, and 8 combinations of key sizes of Paillier and ECDSA, that is, (Pailler, ECDSA) = {(1024, 384), (2048, 384), (3072, 384), (4096,384), (1024, 521), (2048, 521), (3072, 521), (4096, 521)}. We compute and compare the data payloads of the reputation score of integrity with different combinations of key sizes as in Figure 4. In the case of the smallest key sizes of 384-bit ECDSA, 233-bit ECC, 1024-bit Paillier, the payload created from ECC is 446 bytes while the one created from Pailler is 675.2 bytes. In the case of the largest key sizes of 521-bit ECDSA, 571-bit ECC, 4096-bit Paillier, the payload created from ECC is 2248 bytes while the one created from Pailler is 2283 bytes. Therefore, the data payload created by ECC is always smaller than the one created by Paillier.

6. Conclusions

We investigate a scenario of a decentralized close area where users cannot access the Internet, and propose a privacy-preserving protocol supporting the requesting user to evaluate the reputation level of an advising user before following his recommendation. Additionally, the proposal presents the secure process of reputation evaluation and computation against the privacy violation of the requester and the advisor. The future works will cope with the authentication issues for nodes as well as the privacy issues against the Crystal server.

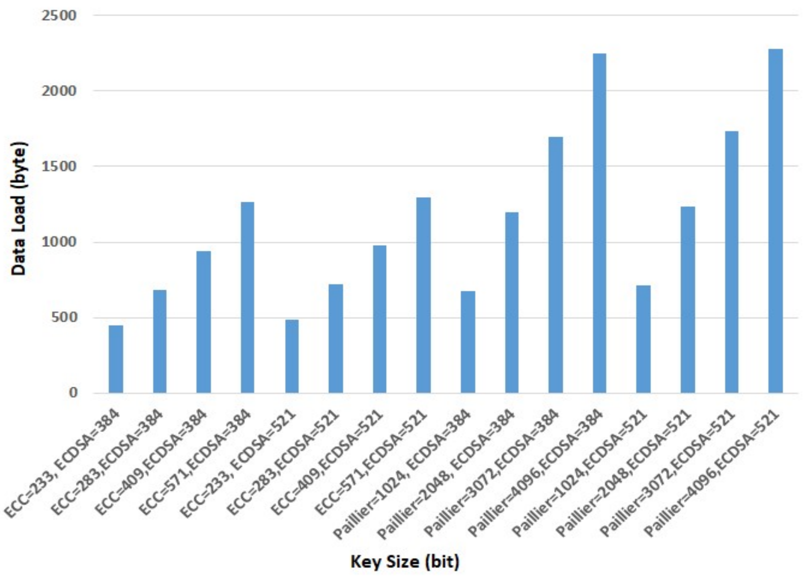

Figure 4: Data load comparison.

Acknowledgements

This work is partially supported by TrustedMaaS project - the Infotech institute of the University of Oulu, Science, and Academy of Finland 6Genesis Flagship (Grant 318927).

References

[1]. A. K. Wong, "The near-me area network," *IEEE internet computing*, vol. 14, no. 2, pp. 74–77, 2010.

[2]. P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, 2000.

[3]. P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," in *The Economics of the Internet and E-commerce*, pp. 127–157, Emerald Group Publishing Limited, 2002.

[4]. O. Hasan, "A Survey of Privacy Preserving Reputation Systems" [Technical Report] LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon. ffhal-01635314f, 2017.

[5]. O. Hasan, L. Brunie, and E. Bertino, "Preserving privacy of feedback providers in decentralized reputation systems," *Computers & Security*, vol. 31, no. 7, pp. 816–826, 2012.

[6]. O. Hasan, L. Brunie, E. Bertino, and N. Shang, "A decentralized privacy preserving reputation protocol for the malicious adversarial model," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 949– 962, 2013.

[7]. R. Petrlic, S. Lutters, and C. Sorge, "Privacy-preserving reputation management," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pp. 1712–1718, ACM, 2014.

[8]. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238, Springer, 1999.

[9]. E. Gudes, N. Gal-Oz, and A. Grubshtein, "Methods for computing trust and reputation while preserving privacy," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 291–298, Springer, 2009.

[10]. O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," in *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pp. 274– 283, ACM, 2001.

[11]. J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach. USA: Addison-Wesley Publishing Company*, 5th ed., 2009.

[12]. Z. Pei, Z. Deng, B. Yang, and X. Cheng, "Application-oriented wireless sensor network communication protocols and hardware platforms: A survey," in *2008 IEEE International Conference on Industrial Technology*, pp. 1–6, IEEE, 2008.

[13]. D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of cryptography*, pp. 535–554, Springer, 2007.

[14]. D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[15]. H. D. and M. A., "Chapter nist elliptic curves, encyclopedia of cryptography and security (2nd ed.)," 2011.

[16]. E. Zhai, D. I. Wolinsky, R. Chen, E. Syta, C. Teng, and B. Ford. "Anonrep: Towards tracking-resistant anonymous reputation, 13th Usenix Conference on Networked Systems Design and Implementation, pp. 583–596, 2016.

[17]. T. Minkus and K. W. Ross. "I know what you're buying: Privacy breaches on ebay", 14th International Symposium Privacy Enhancing Technologies, pp. 164–183, 2014.

[18]. D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, A. Miller, "CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability", IACR Cryptology ePrint Archive, Report 2020/934, 2020.

[19]. F. Zhang, S. Krishna, D. Maram, H. Malvai, S. Goldfeder, A. Juels, "DECO: Liberating web data using decentralized oracles for TLS", ACM Conference on Computer and Communications Security, 2020.

[20]. M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," IEEE Communication Surveys Tuts., vol. 21, no. 2, pp. 1676–1717, 2019.

[21]. S. Brotsis, N. Kolokotronis, K. Limniotis, S. Shiaeles, D. Kavallieros, E. Bellini, C. Pavué, "Blockchain solutions for forensic evidence preservation in IoT environments", IEEE Network Softwarization (NetSoft), pp. 110–114, 2019.

[22]. A. Bellini, E. Bellini, M. Gherardelli, and F. Pirri, ''Enhancing IoT data dependability through a blockchain mirror model", Future Internet, vol. 11, no. 5, p. 117, 2019.

[23]. B. Shala, U. Trick, A. Lehmann, B. Ghita, and S. Shiaeles, ''Blockchain based trust communities for decentralized M2M application services", Advances on P2P, Parallel, Grid, Cloud and Internet Computing, vol. 24, 2018.

[24]. E. Bellini, P. Ceravolo, and E. Damiani, ‘‘Blockchain-based e-vote-as-a-service”, 12th IEEE Conference on Cloud Computing (CLOUD), pp. 484–486, 2019.

[25]. E. Bellini, ‘‘A blockchain based trusted persistent identifier system for big data in science”, Foundation Computing Decision Science, vol. 44, no. 4, pp. 351–377, 2019.

Authors

Ngoc Hong Tran is a lecturer at the Vietnamese German University, Vietnam, since 2016. She obtained a Bachelor Degree in Information Technology, and a Master Degree in Computer Science, from University of Science, Vietnam National University - Ho Chi Minh City. She achieved a PhD Diploma in Computer Science from University of Insubria, Italy (2016). Moreover, she had been a lecturer at the University of Science, VNU-HCM. She also worked in National Institute of Informatics, Tokyo, Japan, in 2009, and was an exchange scholar in Portland State University (PSU), Portland City, Oregon State, USA, in 2010. She worked in LAAS-CNRS, Toulouse, France, in 2012. She was a researcher in Singapore University of Technology and Design, Singapore, from March to April 2017, and was a postdoctoral researcher in University College Dublin, Ireland, 2018-2020. She has been a reviewer of several international conferences. Her research interests are security and privacy in a variety of scenarios, as in healthcare, data mining, block chain, 6G, IoT, distributed collaboration, mobile social network, web composition, network coding.

Tri Nguyen was born in Ho Chi Minh, Vietnam, in 1993. He received the B.Sc. degree in computer science from the University of Information Technology - Vietnam National University, Vietnam in 2015, and the M.Sc. degree in computer science from the University of Pisa, Italy, in 2018. Since 2018, he has been a doctoral student in the Center for Ubiquitous Computing, University of Oulu. His research interests include distributed systems, blockchain technology, and information security.

Quoc-Binh Nguyen is currently working as a lecturer of Faculty of Information Technology of Ton Duc Thang University. He obtained the Bachelor Degree in Information Technology in 2009 and Master Degree in 2012 at University of Science, VNU of Ho Chi Minh City. His research interest is security and privacy.

Susanna Pirttikangas was born in Kemi, Finland in 1973. She received the M.Sc. degree in theoretical mathematics from the University of Oulu, in 1998 and the D.Sc. (tech.) degree in embedded systems in University of Oulu in 2004.

Dr. Pirttikangas works as a research director in the Center for Ubiquitous Computing at the University of Oulu. She made her postdoctoral visits to Waseda University, Japan (2005-2006), Tokyo Denki University, Japan (2008) and Tsinghua University, China (2011). Her research team Interactive Edge develops adaptive, realiable and trusted edge computing. She is an active member of the international research community as a workshop and conference organizer, as well as serving as a reviewer and PC member in top journals and conferences in her field. Dr. Pirttikangas also works as a freelance lead AI scientist in a Finnish company Silo.AI.

M-Tahar Kechadi was awarded PhD and Masters degree - in Computer Science from University of Lille 1, France. He joined the UCD School of Computer Science (CS) in 1999. He is currently Professor of Computer Science at CS, UCD. His research interests span the areas of Data Mining, distributed data mining heterogeneous distributed systems, Grid and Cloud Computing, and digital forensics and cyber-crime investigations. Prof Kechadi has published over 260 research articles in refereed journals and conferences. He serves on the scientific committees for a number of international conferences and he organised and hosted one of the leading conferences in his area. He is currently an editorial board member of the Journal of Future Generation of Computer Systems and of IST Transactions of Applied Mathematics-Modelling and Simulation. He is a member of the communication of the ACM journal and IEEE computer society. He is regularly invited as a keynote speaker in international conferences or to give a seminar series in some Universities worldwide.