An Efficient Anonymous Reputation System

Mayank Raikwar

An Efficient Anonymous Reputation System

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Mayank Raikwar

[Roll No: CS-1427]

under the guidance of

Dr. Sushmita Ruj

Assistant Professor
Cryptology and Security Research Unit

Indian Statistical Institute
Kolkata-700108, India

July 2016

To my family

CERTIFICATE

This is to certify that the dissertation entitled **"An Efficient Anonymous Reputation System"** submitted by **Mayank Raikwar** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by her under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Sushmita Ruj
Assistant Professor,
Cryptology and Security Research Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Sushmita Ruj*, Cryptology and Security Research Unit, Indian Statistical Institute, Kolkata, for her guidance and continuous support and encouragement. She has literally taught me how to do good research, and motivated me with great insights and innovative ideas. Her timely advice, meticulous scrutiny and scholarly advice have helped me a very great extent to accomplish this task.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Mayank Raikwar
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

Reputation systems play a very crucial role in establishing trust on line, especially in e-commerce settings. In a reputation system user provides the feedback in the form of ratings and reviews to other users. Feedback for a user is accumulated, which becomes the reputation of the user. By using the reputation, a user knows about the reliability of another user. In this work, we present an efficient scheme for anonymous reputation system where every user is able to rate a product only once otherwise anyone can detect the user who deviates from this property. Our scheme is based on group signature scheme so a user stay anonymous while rating the product as long as he rates only once. In this work, we present complexity analysis for our scheme and compare our scheme with other schemes as well.

Keywords: *Reputation system, anonymity, exculpability, traceability, linkability, group signature.*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Backgrounds

Reputation systems are a major feature of every modern e-commerce website, helping buyers to carefully choose their service providers and products. Nowadays reputation systems are an increasingly popular tool to give providers and customers valuable information about the previous transactions. Examples of reputation systems include the systems used on eBay, Amazon or Shopzilla etc. These examples are based on the centralized reputation system, which implies that their security relies on the assumption that the underlying server is honest and secure. Some online services like Yelp and Stack Overflow employ reputation systems to evaluate information quality. For example in Yepl, users post their messages and give feedback on other users' post so a user reputation is decreased or increased based on the feedback.

A reputation system should allow the users to rate products that they purchased previously. However, this rating affects the aggregate reputation of a provider. This long-term linkage between a user and rating quickly de-anonymize users to hide their true identities [32, 33, 34, 35]. For example, Minkus et al. [34] revealed eBay users sensitive purchase histories by analyzing only pseudonyms transactions and feedback. Since online users privacy has become a major concern, there is a need for an anonymous reputation system which hides the identity of the rater.

The design of an anonymous reputation system where the ratings are trustworthy, reliable, and honest should also guarantee that customers rate products only once. As long as they do so, they stay anonymous otherwise everybody should be able to detect them deviating from the rate-products-only-once policy and the anonymity of such dishonest users can be revoked by a system manager. To further increase trust in the system, everyone even outsiders should be able to verify the validity of ratings. In a reputation system, a user should be able to check reputation of any candidate to know how much evaluations the candidate has obtained in the history of interactions.

1.2 Our Work

We present a scheme similar to "traditional" online marketplaces as these are the centralized services. Our scheme has two entities: users and a group manager. A user can be either a buyer or a seller. The target application of our reputation system will be e-commerce: where a buyer buys the goods and rates them therefore a buyer is called as *rater* and seller who receive theses ratings called as *ratee*. In our scheme, there is no direct interaction between buyer and seller. Buyer and seller interacts with each other through group manager.

Assumptions:

1. Group manager is an honest entity.

2. Every interaction between a user and GM takes place via secure channel.

We have designed a reputation system in which there can be many sellers who sell the items of same or different item ids. First a buyer and seller has to be registered with GM to take the advantage of scheme. Registration process gives a certificate to the seller. Certificate contains the seller's current reputation and current number of ratings. Then a buyer who wish to buy an item of some item id can look for the sellers' reputation who sell the item of given item id. Then our scheme makes it easier for a buyer to choose his products carefully. Then buyer asks to GM for the signing key corresponding to the item requested by the buyer. GM provides the signing key to buyer after verifying whether the buyer is the new user asking for the signing key or he already possess the signing key for the corresponding item.

After buying the item, buyer gives rating and signature on this rating along with seller id to GM. GM verifies the signature for the rating and then asks for the seller whose id is given by the buyer. Then seller gives his certificate to GM. Then GM updates the seller's reputation and number of rating and then gives it to the seller. The security of our scheme is based on Decision Linear Assumption and the q-SDH Assumption in bilinear groups.

1.3 Our Contribution:

We designed an efficient anonymous reputation system which provides anonymity, public linkability, traceability and strong-exculpability. Anonymity means that signature of honest users are indistinguishable. Traceability means that any valid signature can be traced back to its user. Strong-exculpability means that nobody can produce signatures on behalf of any other honest user. Public linkability means that anyone can tell whether two signatures for the same message is created by the same user or not. computational complexity is less than the computational complexity of existing schemes and our scheme gives a more realistic view of online marketplaces than the

existing schemes. We provide a complexity analysis for each algorithm to demonstrate the efficiency of our scheme and compare our scheme with existing schemes.

1.4 Thesis Outline

This thesis is organized as follows. We define some of the essential preliminaries in Chapter 2. In Chapter 3, we review some of the existing works in the field of anonymous reputation system. Chapter 4 outlines all the details of our proposed scheme which reduces the cost when a user signs for items more than once and reduces the cost if a user signs for some item which has already been signed by other user previously. In Chapter 5, we provide the complexity analysis for our model. Chapter 6 compares our scheme with the existing schemes. In Chapter 7, we define the security notions and security for our reputation system. Finally, in Chapter 8, we conclude the thesis and also mention our future goals with respect to our scheme.

Chapter 2

Preliminaries

This chapter introduces some basic concepts and preliminaries used in our anonymous reputation system. So below are the main building blocks for our reputation system.

2.1 Bilinear Maps

We follow the notations of [26] to review the bilinear maps:

- $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups of prime order p,
- ψ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$,
- g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$, with $g_1 = \psi(g_2)$, and
- e is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ which has following properties:
  Bilinearity: $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p : e(u^a, v^b) = e(u, v)^{ab}$;
  Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.

We say the group $(\mathbb{G}_1, \mathbb{G}_2)$ are bilinear groups iff the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$, the isomorphism ψ and the bilinear map e are efficiently computable.

2.2 Computational Assumptions

This section includes the computational assumptions which are used to prove the security of our reputation system. Below are the assumptions which are used:

Definition 2.1 ***Decision Linear Problem - D-Linear1:***

Let $\mathbb{G}$ be a cyclic group of prime order p. Given arbitrary generators $u, v, w \in \mathbb{G}$ and $u^\alpha, v^\beta, w^\gamma \in \mathbb{G}$ where $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$, the Decision Linear Problem is to decide whether $\gamma = \alpha + \beta$.

Definition 2.2 ***q-strong Diffie-Hellman Problem - q-SDH:***

Let $\mathbb{G}_1, \mathbb{G}_2$ be two multiplicative groups of prime order p (where possibly $\mathbb{G}_1 = \mathbb{G}_2$), let ψ be an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, let $g_2 \in \mathbb{G}_2$ be an arbitrary generator and let $g_1 = \psi(g_2)$. Given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}), ..., g_2^{(\gamma^q)})$, the q-Strong Diffie-Hellman Problem is to output a pair $\left(g_1^{\frac{1}{x+\gamma}}, x\right)$, where $x \in \mathbb{Z}_p$.

2.3 Group Signature

A group signature scheme is a method for allowing a member of a group to anonymously sign a message on behalf of the group. For example, a group signature scheme could be used by an employee of a large company where it is sufficient for a verifier to know a message was signed by an employee, but not which particular employee signed it. Essential to a group signature scheme is a group manager, who is in charge of adding group members and has the ability to reveal the original signer in the event of disputes. A group signature scheme has following basic requirements.

- **Anonymity:** Given a message and its signature, the identity of the individual signer cannot be determined without the group manager's secret key.
- **Traceability:** Given any valid signature, the group manager should be able to trace which user issued the signature.
- **Unlinkability:** Given two messages and their signatures, we can not tell if the signatures were from the same signer or not.
- **Coalition resistance:** A colluding subset of group members cannot generate a valid signature that the group manager cannot link to one of the colluding group members.

2.3.1 Definition:

A group signature scheme is specified by four randomized algorithms : KeyGen, Sign, Verify, Open:

KeyGen: This algorithm runs by group manager and takes a parameter n as input where n is the number of group members. It returns the group public key gpk, private key of group manager $gmsk$ and private key of each group member $gsk[i]$ where i from $\{1, 2, ..., n\}$.

Sign: This algorithm runs by group member. Given a group public key gpk, a group member private key $gsk[i]$ and a message $M \in \{0,1\}^*$, Sign algorithm outputs a

signature σ for message M.

Verify: This algorithm runs by anyone in the group even by outsider also. It takes group public key gpk, message M and signature σ and outputs 1 if σ is a valid signature, otherwise outputs 0.

Open: This algorithm runs by group manager for tracing a signature to the signer. It takes as input group public key gpk, group manager private key $gmsk$, message M and signature σ and outputs the identity i of the user who created the signature σ for message M.

For the detailed construction of group signature refer [2].

Chapter 3

Related Work

In this chapter, we review some of the related work done in the field of anonymous reputation system. Reputation Systems are always a popular research topic in economics and computer science, see for example [4, 27, 28, 38, 39]. Many privacy-preserving reputation systems have already been proposed in the literature. In the domain of reputation system many centralized and decentralized schemes have been proposed. In the centralized scheme, a system manager serves as a central authority or trusted third party which controls over most of the operations while in decentralized setting there is no trusted third party. Our scheme is a centralized scheme. So first we present some of the centralized schemes and then we show some of the decentralized schemes.

3.1 Centralized Schemes

Scheme [7] defines the models for secure and anonymous reputation systems and gives a first construction of such a system based on dynamic group signature scheme by Bellare, Shi and Zhang [11]. The construction provides anonymity, traceability, strong-exculpability, verifier-local revocation, and public linkability. In this scheme there are three entities : users (buyers), group manager and key issuer (provider) and all of them interacts with each other. A user first register himself to group manager and then if he wants to buy any item then he interacts with the key issuer for obtaining the signing key for corresponding item. This signing key is used for generating the signature on the rating given by the user for that item. However, in our scheme key issuer and item provider are not the same party. The task of key issuance is performed by group manager and a buyer and provider interacts through GM in our scheme.

The work presented in scheme [8] proposes an anonymous reputation system where the reputation of sellers are secret for even the reputation manager (RM). In this model both buyers and sellers use some anonymous authentication scheme such as anonymous credential scheme and the anonymous buyer submits his rating for the

seller to the RM. Here seller anonymity is provided by the use of zero knowledge proof and the certificate given by the reputation manager to the seller. This scheme also provides the reputation unforgeability by the use of AHO signature [19, 20], BB signature [21] and accumulators [36]. In this scheme, the accumulated reputation of the users are revealed via the range proof as adopted in [17] so anyone including RM can not grasp the average of ratings.

An anonymous reputation system based on voting was also proposed in the work of [22]. In this model, a rater votes on an item of any ratee and a ratee can anonymously and unlinkably produce the signature of the reputation including the votes on his item. This protocol is very interesting and secure, we can only regret the monotonic aspect of the feedback that allows an attacker to take advantage of his old good reputation without being affected by any new dissatisfaction that his recent activity might cause. There are some centralized schemes which are instances of pseudonym based schemes. This type of work is presented in the schemes of [17] and [23].

3.2 Decentralized Schemes

Many related works are presented for the decentralized model of reputation system. The works of Anceaume et al.[24] and LajoieMazenc et al. [25] are more decentralized, the first one prevents Sybil attacks by charging a fee, and in the second one, accredited signers are required to make resource heavy calculations for each rating of each service provider.

In scheme [10] a reputation system is presented which is decentralized yet sceure and efficient. Their protocol is based on Merkle trees [30], blind signatures [31] and non-interactive zero-knowledge proofs. In this scheme there are three types of node: $ServiceProviders$ $(SPs), Clients$ and $Trackers$ which perform the operations like reputation retrieval, transaction, sending feedback and feedback aggregation. Here the users are able to retrieve the reputation score of a service provider directly from it in a constant time, with assurance regarding the information obtained. In this process, first client obtains the reputation value of a SP from the SP itself, and verifies the trackers signature and then the transaction happens between client and SP and client gets the a blind signature from the SP while paying for his purchase. Then client generates a feedback which also contains a commitment and send it to the SP who includes it into his next block. Then trackers sign the header of the next block, containing the current aggregated reputation value, so that the SPs can distribute it directly to the peers without any trust requirement between peers and SPs.

Some related works which uses blockchain [37] to attain similar objectives are also presented. Scheme [9] and [29] are also decentralized and uses blockchain. Scheme [9] presents a decentralized anonymous marketplace “Beaver” which preserves the anonymity of its customer and resistant against Sybil attacks on seller reputation. It provides every user the same global view of reputation of other users through public

ledger based consensus. Beaver builds on an anonymous payment system, consensus protocol(e.g., Bitcoin "blockchain") and various cryptographic primitives as linkable ring signature etc. to provide a globally consistent view of the network to all of its users As blockchain contains transactions in its blocks so Beaver uses a set of transactions like payment, review transaction etc. It also includes fees for some transactions which is used for incentivizing the miner for mining the block. This design provides payment-review unlinkability and review exculpability.

A practical anonymous reputation system AnonRep [18] is also presented offering the benefits of reputation without enabling long-term tracking. In this system there are a smaller set of servers and a large set of client nodes each of them connected to one of the third party servers. AnonRep users anonymously post messages, which they can verifiably tag with their reputation scores without leaking sensitive information. AnonRep reliably tallies other users feedback (e.g., likes or votes) without revealing the users identity or exact score to anyone, while maintaining security against score tampering or duplicate feedback. AnonRep operates in a series of message-and- feedback rounds. In each round clients post their messages anonymously by using their one-time pseudonym. Then each client may then provide feedback (e.g., votes) on other clients posted messages. Then in each round servers tally the feedback received for each one-time pseudonym and update the reputation score.

Chapter 4

Proposed Scheme

4.1 Introduction

In this chapter, we present a new scheme for anonymous reputation system. Participants of our system are user and group manager (GM). The users are sellers and buyers. In this reputation system, there is no direct interaction between buyer and seller. Every interaction between buyer and seller will be through reputation manager. The idea behind our reputation system is every seller has some rating and number of ratings associated with him, a user who wants to buy some product first checks the rating of the sellers and based on the rating he will buy the product from one of the sellers then the buyer gives the rating to that seller. For accomplishing this task we have designed a set of protocols. We use the terms rating and message synonymously. We give a brief description of the algorithms used in our scheme in terms of their input/output specifications. We use the same notation for the authorities, algorithms and security properties used in [7]. First we define all the tables used in the algorithm which makes easier to understand the algorithms. Finally, we present the detailed construction of the algorithms in our scheme and conclude this chapter.

4.2 List of Tables

In this section we define all the tables used in our construction. The descriptions are as follows:

Reg: This is a registration table which stores the user id i and corresponding public key $upk[i]$ for each registered user. Thus each row in this table corresponds to a tuple $(i, upk[i])$. When a user registers himself then GM makes an entry in this table. So this table is maintained by GM and accessible to only registered users. A user can only retrieve the values from Reg table but can not change any entry in this table.

ItemList: This is an item registration table which stores item id $itemId$ and corresponding item-based public key $ipk[itemId]$. Whenever a seller registers a new item, an entry is made to this table. This table is also maintained by GM and like Reg table a user can only retrieve the values from this table but can not alter any entry in this table.

$\boldsymbol{SL_{itemId}}$: This is a seller list which stores the identities of the sellers who sell the item corresponding to $itemId$. Thus for each $itemId$ a table is maintained so When a seller s registers any item corresponding to $itemId_1$ then an entry is made to the table SL_{itemId_1}.

$\boldsymbol{IL_{itemId}}$: This is an identification list which stores signing key for the specified $itemId$ and for the registered user i. When a user buys some item then GM gives a signing key which is used for giving the rating for that item. Thus for each $itemId$ say $itemId_1$ there can be many users who buy that item so if there are n users $i_1, i_2, ..., i_n$ who buy that item then entries in the table IL_{itemId_1} will be like $(upk[i_1], gsk[i_1, itemId_1]), (upk[i_2], gsk[i_2, itemId_1]), ..., (upk[i_n], gsk[i_n, itemId_1])$. Thus GM maintains each table corresponding to each $itemId$. These tables are private to GM so no user can access these tables.

Rep: This is a reputation table which is maintained by GM and consists of seller id i and corresponding current reputation as current rating $rep[i]$ and current number of ratings $num[i]$. So an entry in this table is of the form $(i, rep[i], num[i])$. This table is maintained by GM in which an entry corresponding to a seller i is updated when update algorithm runs by the seller. This table is not accessible to any user. This table is used to show the seller ratings to a user when a natural join is performed with $itemList$ table by GM.

Aux: This is a hash table where key is hash of $itemId$ and the corresponding value is a set of four bilinear pairing values. This hash table is privately maintained by GM so the entries to this hash table is made only by GM. Any user or outsider can ask for the value corresponding to a key (hash of some $itemId$) and then GM provides the value for the given key. An entry to this hash table is made when a user j runs the sign algorithm for giving the rating then first GM checks whether the user possess the valid signing key $gsk[j, itemId]$ for the $itemId$, if so then GM checks whether an entry has already been made or not for the corresponding $itemId$ in Aux table. So GM checks the value corresponding to hash of $itemId$ as key. If there is some entry then GM gives the value to user j and if there is no entry then user j computes all the bilinear pairing values and gives it to GM then GM stores these values as a set in the value field corresponding to the hash of $itemId$. Here GM asks for the possession of valid signing key so that no one other than the valid user can make entry to Aux.

4.3 Algorithms

A reputation system can be defined as a tuple $\mathcal{RS} = (KeyGen, Register, View, Join, Issue, Sign, Verify, Open, Link, Update)$ of polynomial-time algorithms. In the following algorithms **St** and **M** represents state and message (parameters) used in the algorithm for corresponding entity. These algorithms are described as follows:

$\mathbf{KeyGen_{GM}}(\lambda)$: This algorithm is run by group manager only once in the setup phase and output of the algorithm is public key $gmpk$ and secret key $gmsk$. The secret $gmsk$ contains elements which allow tracing of the group members. Here λ is a security parameter.

$\mathbf{KeyGen_U}(i)$: This algorithm is run by group manager to provide user's public key and secret key pair $(upk[i], usk[i])$. Input for the algorithm is userid i.

KeyGen(i, *itemId*): Input of this algorithm is $itemId$ giene by the seller i to GM. Then GM runs this algorithm to create item-based public key and secret key pair $(ipk[itemId], isk[itemId])$. The tuple $(itemId, ipk[itemId])$ is added to the $ItemList$ table, if already not there. Seller id i is added to the SL_{itemId} table.

$\mathbf{Register_{GM}}(\mathbf{St_{GM}}, \mathbf{M_{GM}})$, $\mathbf{Register_B}(\mathbf{St_B}, \mathbf{M_B})$: This is an interactive protocol run by GM and buyer who wants to become a group member. Buyer gives his id i to GM then GM checks the entry in Reg table corresponding to id i. If there is an entry then abort, otherwise GM runs **$\mathbf{KeyGen_U}(i)$** and add the tuple $(i, upk[i])$ to the registration table Reg. After successful registration, buyer generates some user specific secret parameters $usp[i]$.

$\mathbf{Register_{GM}}(\mathbf{St_{GM}}, \mathbf{M_{GM}})$, $\mathbf{Register_S}(\mathbf{St_S}, \mathbf{M_S})$: This is an interactive protocol run by GM and seller. Seller gives his id i to GM then GM checks the entry in Reg table corresponding to id i. If there is an entry then abort, otherwise GM runs **$\mathbf{KeyGen_U}(i)$** and add the tuple $(i, upk[i])$ to the registration table Reg. After successful registration, GM gives a certificate to seller which includes initial reputation and number of ratings.

View(*itemId*): If a user wants to view the reputation of the sellers corresponding to the $itemId$ then user gives $itemId$ to GM. Then GM performs natural join between Rep and SL_{itemId} tables and gives the list of reputations for the sellers corresponding to the given $itemId$.

$\mathbf{Join}(\mathbf{St_U}, \mathbf{M_U})$: This algorithm is run by buyer asking for the corresponding signing key for a given $itemId$. User gives his public key $upk[i]$, userid i and the item id $itemId$ for which he wants the corresponding signing key.

Issue($\mathbf{St_{GM}}$, $\mathbf{M_{GM}}$): GM runs this algorithm only if Join algorithm has already been run. if Issue accepts, GM sends a personal signing key for the given $itemId$ as $gsk[i, itemId]$ to the user and saves the tuple $(upk[i], gsk[i, itemId])$ in the identification list IL_{itemId} for the specified $itemId$.

Sign(*itemId, gmpk, ipk[itemId], gsk[i,itemId], usk[i], usp[i], M, j*): This randomized algorithm is run by a buyer to create a signature for the specified $itemId$. Given an item identifier as $itemId$, the group manager's public key $gmpk$, an item-based public key $ipk[itemId]$, the signing key for the given $itemId$ of buyer i $gsk[i, itemId]$, the secret key of buyer i $usk[i]$, and a message M, Sign computes and outputs a signature σ on M under the given keys. Then buyer sends (M, σ, j) to GM.

Verify(*itemId, gmpk, ipk[itemId], M, σ*): This deterministic algorithm can be run by any user, even by an outsider, having access to the public $ItemList$, the group manager's public key $gmpk$, a message M and a candidate signature σ for M, to check whether σ is a valid signature of M or not.

Open(*gmpk, gmsk, M, σ*): This deterministic algorithm is run by the group manager to open signatures. Given the group manager's public key $gmpk$, the group manager's secret key $gmsk$, a message M and a signature σ, output the identity of the signer or failure.

Link(*itemId, gmpk, ipk[itemId], (M', σ'), (M'', σ'')*): This deterministic algorithm can be run by any user, even by an outsider, having access to the public $ItemList$, the group manager's public key $gmpk$ and two message-signature pairs $(M', \sigma'), (M'', \sigma'')$. This algorithm checks whether these two message-signature pairs are publicly linkable or not.

Update(*M, σ, j*): This is an interactive protocol runs between GM and seller to accumulate the rating. GM first verifies the signature σ for the given rating M and for his provided $itemId$. If he verifies then he asks for the certificate from the seller j and then accumulate the rating M with his previous accumulated rating and also updates the current number of ratings in the updated certificate. Then GM gives the updated certificate to seller and updates the entry in Rep table.

Figure 4.1 illustrates the interaction of the described parties and the algorithms involved.

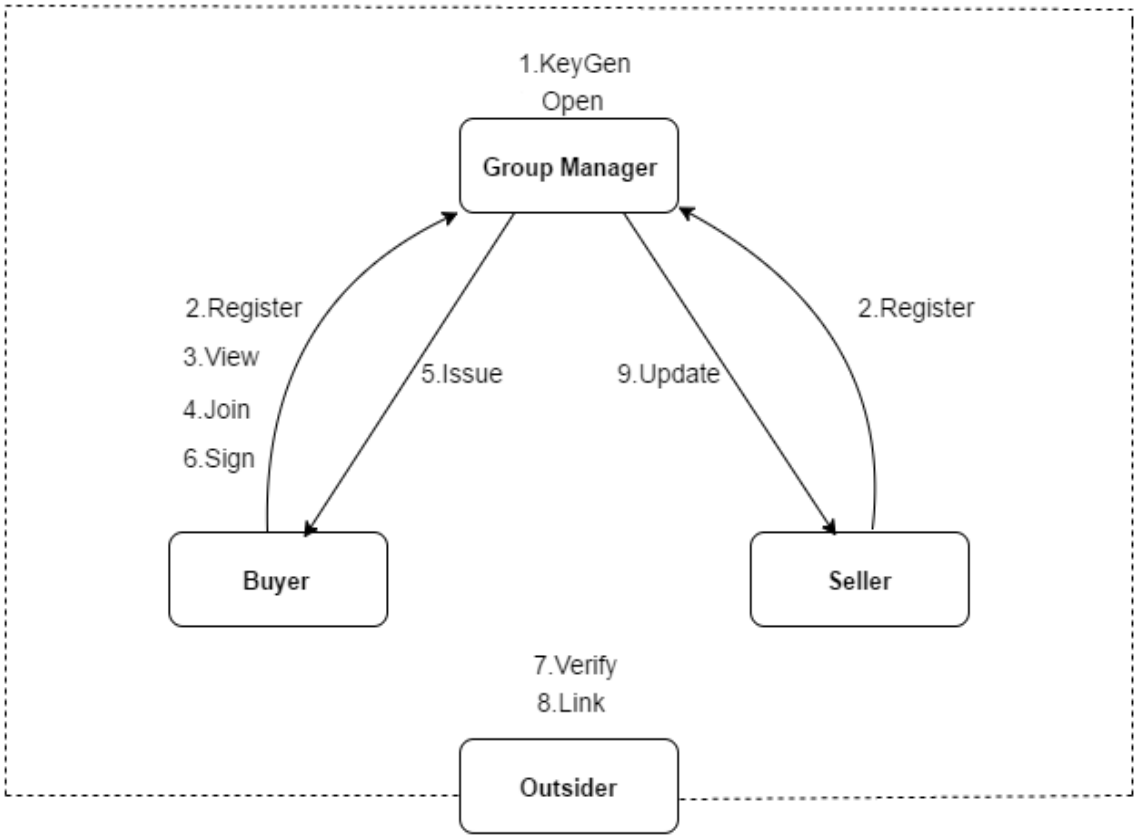

Figure 4.1: Interaction of the parties within our reputation system

4.4 Construction

In our reputation system, the seller publishes item for which the signature is created by the buyer. Every user can create a single signature for every $itemId$ without losing anonymity. We assume the communication between users and the group manager takes place via secure channels.
In the following definitions we consider bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and two hash functions modeled as random oracles: $H : \{0,1\}^* \to \mathbb{Z}_p$ and $H_1 : \{0,1\}^* \to \mathbb{G}_2$.

$\mathbf{KeyGen_{GM}}(\lambda)$:

The group manager's key generation algorithm proceeds as follows:

1. Select $w \xleftarrow{\$} \mathbb{G}_1, \quad \xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$ and compute $u := w^{\frac{1}{\xi_1}}, v := w^{\frac{1}{\xi_2}}$. The values (u, v, w) are the public key of the linear encryption, the values (ξ_1, ξ_2) are the corresponding secret key.

2. Select $\hat{d} \xleftarrow{\$} \mathbb{G}_2, \quad \zeta \xleftarrow{\$} \mathbb{Z}_p$ and compute $d := \psi(\hat{d}), h := d^{\zeta}$ as the basis for public

linkability and revocation.

3. Set $gmpk := (u, v, w, h, d, \hat{d})$ and $gmsk := (\xi_1, \xi_2, \zeta)$ as the group manager's public and secret keys.

KeyGen$_{\mathbf{U}}$(i):

For user's key generation, following steps occur:

1. User(buyer/seller) provides its id i to GM.

2. GM selects $y_i \xleftarrow{\$} \mathbb{Z}_p$, set $upk[i] := h^{y_i}$ and $usk[i] := y_i$ as the user's public and secret keys and gives it to the user.

KeyGen(i, *itemId*):

In this algorithm seller provides its own id i, item id *itemid* to GM and GM proceeds as follows:

1. GM first checks whether item corresponding to this $itemId$ has already been registered from $ItemList$ table. If yes, then GM retrieves corresponding item-based public key and makes an entry in the SL_{itemId} table for the seller i, otherwise

2. GM select $g_{2,item} \xleftarrow{\$} \mathbb{G}_2$ and set $g_{1,item} := \psi(g_{2,item})$.

3. GM select $\gamma_{item} \xleftarrow{\$} \mathbb{Z}_p$ and set $W_{item} := {g_{2,item}}^{\gamma_{item}}$.

4. GM adds the item id $itemId$ and $item$-based public key $ipk[itemId] := (g_{1,item}, g_{2,item}, W_{item})$, to the $ItemList$ and set $isk[itemId] := \gamma_{item}$ as the $item$-based secret key and adds i to the SL_{itemId}

   Here $item$ in the suffix of the keys represents the corresponding $itemId$.

Register$_{\mathbf{GM}}$($\mathbf{St_{GM}}$, $\mathbf{M_{GM}}$), Register$_{\mathbf{B}}$($\mathbf{St_B}$, $\mathbf{M_B}$):

The interactive registration protocol between buyer and GM proceeds as follows:

1. The buyer sends his identity i to the group manager.

2. The group manager checks if there already exists an entry $Reg[i]$ in the registration table. If so, he declares failure and exit. Otherwise, the group manager runs KeyGen$_{\mathrm{U}}(i)$ algorithm to obtain the tuple $(upk[i], usk[i])$, sets $Reg[i] := (i, upk[i])$ and sends $(upk[i], usk[i])$ and a certificate for $upk[i]$ to the user i.

3. After successful registration buyer chooses $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$, and computes $T_1 := u^{\alpha}$, $T_2 := v^{\beta}$, $T_3' := w^{\alpha+\beta}$ and saves these parameters as user secret parameters $usp[i] = (\alpha, \beta, T_1, T_2, T_3')$.

Register$_{\mathbf{GM}}$(St$_{\mathbf{GM}}$, M$_{\mathbf{GM}}$), Register$_{\mathbf{S}}$(St$_{\mathbf{S}}$, M$_{\mathbf{S}}$):

The interactive registration protocol between seller and GM proceeds as follows:

1. The seller sends his identity i to the group manager.
2. The group manager checks if there already exists an entry $Reg[i]$ in the registration table. If so, he declares failure and exit. Otherwise, the group manager runs $\text{KeyGen}_{\text{U}}(i)$ algorithm to obtain the tuple $(upk[i], usk[i])$, sets $Reg[i] := (i, upk[i])$ and sends $(upk[i], usk[i])$ and a certificate for $upk[i]$ to the user i.
3. After successful registration seller ask for an initial certificate $cert_0$ from GM which includes initial reputation of seller as $rep_0[i] = 0$ and number of ratings as $num_0[i] = 0$.

View(*itemId*):

Whenever a user wants to buy any item then first he checks the ratings of the sellers who sell the items corresponding to $itemId$. So user gives $itemId$ to GM and then GM returns the seller list and their reputations by using $SL_{itemList}$ and Rep tables. Now buyer selects a seller based on the reputation scores provided by GM.

Join(St$_{\mathbf{U}}$, M$_{\mathbf{U}}$):

The Join protocol proceeds as follows:
The buyer i looks up the public key corresponding to the used $itemId$ $ipk[itemId] := (g_{1,item}, g_{2,item}, W_{item})$ in the $ItemList$ and sends $(i, upk[i], itemId)$ and the certificate for $upk[i]$ to the group manager.

Issue(St$_{\mathbf{GM}}$, M$_{\mathbf{GM}}$):

After the join protocol, GM runs issue protocol as follows:

1. GM first checks whether any seller j sells item corresponding to $itemId$ from SL_{itemId} table. If yes then proceed to next step, else result $failure$.
2. GM verifies the certificate for $upk[i]$ and checks that buyer i is not in the possession of a signing key for the given $itemId$, i.e. there exists no entry $(upk[i], ...)$ in IL_{itemId}. If the certificate is invalid or there already is a signing key in the list, then GM declares failure and exits. Otherwise proceed to next step.

3. GM selects $x_{i,item} \xleftarrow{\$} \mathbb{Z}_p$, computes $A_{i,item} := (g_{1,item}.upk[i])^{\frac{1}{x_{i,item}+\gamma_{item}}}$, gives $gsk[i, itemId] := (A_{i,item}, x_{i,item})$ to the user i as his signing key for the specified $itemId$, and saves $(upk[i], gsk[i, itemId])$ in the identification list IL_{itemId} for this $itemId$.

Sign(*itemId, gmpk, ipk[itemId], gsk[i,itemId], usk[i], usp[i], M, j*):

The group signing algorithm proceeds as follows and run by the user(buyer):

1. Obtain the value $\hat{f} \in \mathbb{G}_2$ by $\hat{f} := H_1(itemId)$.

2. Choose $\mu \xleftarrow{\$} \mathbb{Z}_p$ and compute $T_3 := A_{i,item}.T_3$, $T_4 := d^{\mu}$, $T_5 := \psi(\hat{f})^{\mu+y_i}$ and the helper values $\delta_1 := \alpha.x_{i,item}$ and $\delta_2 := \beta.x_{i,item}$.

3. Choose$r_\alpha, r_\beta, r_x, r_y, r_\mu, r_{\delta_1}, r_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$R_1 := u^{r_\alpha}$$

$$R_2 := v^{r_\beta}$$

For the computation of R_3, buyer asks to GM for the value corresponding to the $itemId$ in Aux table. Then GM computes the hash of the $itemId$ and for that hash key it looks for the corresponding value in Aux table. If there is some value then GM returns the value to the buyer. The value consists of a set of four values $e(w, W_{item}), e(w, g_{2,item}), e(h, g_{2,item})$ and $e(g_1, g_{2,item})$. If GM finds no entry then GM checks whether the buyer possesses the valid item-based signing key, if so, GM asks to the buyer to compute all these four bilinear pairing values and sent it to him. Then GM stores these value in Aux. Now the computation of R_3 happens as defined below just by putting these values.

$$R_3 := e(T_3, g_{2,item})^{r_x}.e(w, W_{item})^{-r_\alpha - r_\beta}.e(w, g_{2,item})^{-r_{\delta_1} - r_{\delta_2}}.e(h, g_{2,item})^{-r_y}$$

$$R_4 := {T_1}^{r_x}.u^{-r_{\delta_1}}$$

$$R_5 := {T_2}^{r_x}.v^{-r_{\delta_2}}$$

$$R_6 := d^{r_\mu}$$

$$R_7 := \psi(\hat{f})^{r_\mu + r_y}$$

4. Compute a challenge value c using H:
$c := H(M, itemId, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7)$

5. Compute
   $s_\alpha := r_\alpha + c.\alpha \qquad s_\beta := r_\beta + c.\beta \qquad s_x := r_x + c.x_{i,item} \qquad s_y := r_y + c.y_i$
   $s_\mu := r_\mu + c.\mu \qquad s_{\delta_1} := r_{\delta_1} + c.\delta_1 \qquad s_{\delta_2} := r_{\delta_2} + c.\delta_2.$
6. Output the signature $\sigma := (itemId, T_1, T_2, T_3, T_4, T_5, c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2})$.
7. Signer gives (M, σ, j) to GM where j is the seller id which sells the item corresponding to $itemId$.

Verify(*itemId, gmpk, ipk[itemId], M,* σ):

The signature verification algorithm proceeds as follows:

1. Obtain the value $\hat{f} \in \mathbb{G}_2$ by $\hat{f} := H_1(itemId)$.
2. The verifier computes

$$R_1 := u^{s_\alpha}.{T_1}^{-c}$$

$$R_2 := v^{s_\beta}.{T_2}^{-c}$$

For the computation of R_3 verifier fetches the value from the Aux table for the corresponding $itemId$ and then computes the value of R_3 in the following manner. If there is no value for the corresponding $itemId$ then the signature is invalid.

$$R_3 := \underbrace{\frac{e(T_3, g_{2,item})^{s_x}.e(w, W_{item})^{-s_\alpha - s_\beta}.e(w, g_{2,item})^{-s_{\delta_1} - s_{\delta_2}}.e(h, g_{2,item})^{-s_y}}{e(T_3, W_{item})^{c}.e(g_1, g_{2,item})^{-c}}}$$

$$\Downarrow$$

$$e({T_3, {g_{2,item}}^{s_x}.{W_{item}}^{-c}}).e(w, W_{item})^{-s_\alpha - s_\beta}.e(w, g_{2,item})^{-s_{\delta_1} - s_{\delta_2}}.e(h, g_{2,item})^{-s_y}.e(g_1, g_{2,item})^{c}$$

$$R_4 := {T_1}^{s_x}.u^{-s_{\delta_1}}$$

$$R_5 := {T_2}^{s_x}.v^{-s_{\delta_2}}$$

$$R_6 := d^{s_\mu}.{T_4}^{-c}$$

$$R_7 := \psi(\hat{f})^{s_\mu + s_y}.{T_5}^{-c}$$

3. Check the challenge c is correct:
   $c \stackrel{?}{=} H(M, itemId, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7)$.
   If this holds then output 1, otherwise 0.

Link(*itemId, gmpk, ipk[itemId], (M′, σ′), (M″, σ″)*):

The public linking algorithm proceeds as follows:

1. First check that σ', σ'' are valid signatures for the messages M', M'' respectively. If not, output 0.
2. Obtain the value $\hat{f} \in \mathbb{G}_2$ by $\hat{f} := H_1(itemId)$.
3. Output 1, if $e(\frac{T_5'}{T_5''}, \hat{d}) \stackrel{?}{=} e(\frac{T_4'}{T_4''}, \hat{f})$ holds and 0 otherwise.

Open(*gmpk, gmsk, M, σ*):

The opening algorithm proceeds as follows:

1. GM checks that σ is a valid signature of knowledge for message M. If not, output $failure$.
2. GM computes $A_{i,item} := T_3.{T_1}^{-\xi_1}.{T_2}^{-\xi_2}$ using the group manager's secret key.
3. The group manager looks up the user index i from the identification list IL_{itemId}.
4. If there is an entry for $A_{i,item}$ in IL_{itemId} then return i, otherwise return $failure$.

Update(*M, σ, j*):

This algorithm works as follows:

1. GM first verifies σ for the message M. If he verifies correctly then:
2. GM asks for the certificate from seller j which consists of j's current reputation and current number of ratings.
3. Seller j provides his certificate to GM. GM performs the following operation:

$$rep[j] := rep[j] + M$$

$$num[j] := num[j] + 1$$

   Where right hand side $rep[j]$ and $num[j]$ are the current reputation and current number of ratings of seller j in his certificate and left hand side $rep[j]$ and $num[j]$ are the updated values in the updated certificate. Then GM gives the updated certificate to seller j.

4. GM updates the entry corresponding to seller j in Rep table.

Chapter 5

Complexity Analysis

In this chapter we present the computational complexity of each algorithm of our model. We will calculate the computations required by the users (buyer, seller) and GM. First we will define all the notations used in different operations listed in Table 5.1.

Notations	**Meaning**
$\mathbb{E}_1/\mathbb{E}_2/\mathbb{E}_T$	Time required for an exponentiation in the group $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$
$\mathbb{M}_1/\mathbb{M}_2/\mathbb{M}_T$	Time required for a multiplication in the group $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$
$\mathbb{S}_{\mathbb{Z}_p}$	Time required for an addition/subtraction in $\mathbb{Z}_p$
$\mathbb{M}_{\mathbb{Z}_p}$	Time required for a multiplication in $\mathbb{Z}_p$
$\mathbb{T}_\psi$	Time required for group isomorphism
$\mathbb{T}_H/\mathbb{T}_{H_1}$	Time required to hash using H/H_1
$\mathbb{P}$	Time required for one pairing operation
$\mathbb{T}_{Reg}/\mathbb{T}_{Rep}$	Time required for one table lookup from Reg/Rep tables
$\mathbb{T}_{iList}/\mathbb{T}_{IL}$	Time required for one table lookup from $ItemList/IL_{itemId}$
$\mathbb{T}_{SL}$	Time required for one table lookup from SL_{itemId}
$\mathbb{T}_{Aux}$	Time required to access a value from Aux table
$\mathbb{J}(x, y)$	Time required for natural join between two tables where x and y are the number of entries in first and second table
$\|M\|$	Size of message M
$\|\mathbb{G}_1\|/\|\mathbb{G}_2\|/\|\mathbb{G}_T\|$	Size of group $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$
TC(A)	Computational Complexity for algorithm A

Table 5.1: Notations used for complexity analysis

Remark 1 *In the above notations* $\mathbb{T}_{Reg}, \mathbb{T}_{Rep}, \mathbb{T}_{iList}, \mathbb{T}_{IL}$ *depends on the number of entries in the table. Thus these costs are not the same at each time.* $\mathbb{T}_{Aux}$ *includes the cost of sending the itemId and the computation of hash of itemId to generate the key and* $\mathbb{T}_{Aux} \ll \mathbb{P}$ *as Aux is a hash table.*

KeyGen: There are three algorithms for KeyGen. The analysis for these three algorithms is as follows:

1. KeyGen algorithm for group manager runs only once and as the complexity of an algorithm depends on the number of operations performed so now we encounter all the operations occurred in this KeyGen algorithm.

   - There are three exponentiation operations performed in the group $\mathbb{G}_1$ so total cost for exponentiation is $3\mathbb{E}_1$.
   - There is one isomorphism operation to generate some parameter in public key which cost $\mathbb{T}_\psi$.
   - TC($\text{KeyGen}_{\text{GM}}$) $= 3\mathbb{E}_1 + \mathbb{T}_\psi$.

2. KeyGen algorithm for a user includes only one exponentiation operation in $\mathbb{G}_1$ so

   TC(KeyGen_{U}) $= \mathbb{E}_1$.

3. KeyGen algorithm for each item includes the following operations:

   - One exponentiation operation in $\mathbb{G}_2$, one group isomorphism and two table lookups.
   - So TC($\text{KeyGen}_{\text{itemId}}$) $= \mathbb{T}_{iList} + \mathbb{T}_{SL} + \mathbb{E}_2 + \mathbb{T}_\psi$.

Registration: For the complexity analysis of Registration algorithm, first we analyse all the operation and then compute the total cost.

- GM performs one table lookup from *Reg* table.
- In registration process GM also runs $\text{KeyGen}_{\text{U}}(i)$ if user i is not registered yet. So registration process also includes the cost of key generation which is equal to $\mathbb{E}_1$ as described above.
- TC(Registration):
  = total cost for *Reg* table lookup + total cost for KeyGen
  $= \mathbb{T}_{Reg} + \mathbb{E}_1$

View: This algorithm performs a join operation between SL_{itemId} and *Rep* tables. As the cost of join operation depends on number of records in table SL_{itemId} and *Rep*. Thus if there are m and n number of records in SL_{itemId} and *Rep* table at a given time then:
TC(View) = $\mathbb{J}(m, n)$.

Join: In join algorithm user looks up the public key in the *Itemlist* for some *itemId*. Thus TC(Join) = $\mathbb{T}_{iList}$.

Issue: There are many operations in Issue algorithm so first we analyse cost of all the operations and then compute the total cost for this algorithm. The operations used in this algorithm are as follows:

- This algorithm requires two table lookups, one from SL_{itemId} and another from IL_{itemId}.
- Three mathematical operations are involved for the computation of signing key for the given item id *itemId* and for user i. These three operations are one multiplication in $\mathbb{G}_1$, one addition in $\mathbb{Z}_p$ and one exponentiation in $\mathbb{G}_1$ so total cost for these operation = $\mathbb{M}_1 + \mathbb{S}_{\mathbb{Z}_p} + \mathbb{E}_1$.
- TC(Issue):
  = total cost for table lookup + total cost for signing key generation
  = $\mathbb{T}_{SL} + \mathbb{T}_{IL} + \mathbb{M}_1 + \mathbb{S}_{\mathbb{Z}_p} + \mathbb{E}_1$

Sign: Sign algorithm performs various operations so first we encounter all the number of different operations and then conclude the computational complexity of this algorithm. So Table 5.2 below is the description of all operations occurred in this algorithm.

Operation	**Total Complexity**
Hash Evaluation	$\mathbb{T}_H + \mathbb{T}_{H_1}$
Addition/Subtraction	$16\mathbb{S}_{\mathbb{Z}_p}$
Multiplication	$10\mathbb{M}_{\mathbb{Z}_p} + 2\mathbb{M}_1 + 3\mathbb{M}_T$
Exponentiation	$10\mathbb{E}_1 + 4\mathbb{E}_T$
Isomorphism	$2\mathbb{T}_\psi$

Table 5.2: Complexity analysis for sign algorithm

Based on the number of operations involved in the complexity computation, complexity of sign algorithm can be categorised into two cases.

Case 1. When Sign algorithm computes five bilinear pairing values.

TC(Sign) :

$= \mathbb{T}_H + \mathbb{T}_{H_1} + 16\mathbb{S}_{\mathbb{Z}_p} + 10\mathbb{M}_{\mathbb{Z}_p} + 2\mathbb{M}_1 + 3\mathbb{M}_T + 10\mathbb{E}_1 + 4\mathbb{E}_T + 2\mathbb{T}_\psi + 5\mathbb{P}$

Case 2. When Sign algorithm computes one pairing value alongwith one Aux table lookup.

TC(Sign) :

$= \mathbb{T}_H + \mathbb{T}_{H_1} + 16\mathbb{S}_{\mathbb{Z}_p} + 10\mathbb{M}_{\mathbb{Z}_p} + 2\mathbb{M}_1 + 3\mathbb{M}_T + 10\mathbb{E}_1 + 4\mathbb{E}_T + 2\mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux}$

Verify: This algorithm includes many operations as Sign algorithm. So first we focus on all the operations as described in Table 5.3 then compute the total complexity of this algorithm.

For the computation of R_3, instead of 2 pairing computations in $e(T_3, g_{2,item})^{s_x}$ and $e(T_3, W_{item})^c$, verifier computes $e(T_3, g_{2,item}^{s_x} . W_{item}^{-c})$ as $e(T_3, W_{item})^c$ in the denominator part of R_3 as per the construction.

Operation	**Total Complexity**
Hash Evaluation	$\mathbb{T}_H + \mathbb{T}_{H_1}$
Addition/Subtraction	$13\mathbb{S}_{\mathbb{Z}_p}$
Multiplication	$6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T$
Exponentiation	$12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T$
Isomorphism	$\mathbb{T}_\psi$
Pairing	$\mathbb{P}$
Table lookup	$\mathbb{T}_{Aux}$

Table 5.3: Complexity analysis for verification algorithm

TC(Verify) :

$= \mathbb{T}_H + \mathbb{T}_{H_1} + 13\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux}$

Link: Link algorithm runs two times verification algorithm, one hash evaluation and computes two pairing.

TC(Link) :

$= 2\text{TC}(\text{ Verify }) + \mathbb{T}_{H_1} + 2\mathbb{P}$

$= 2(\mathbb{T}_H + \mathbb{T}_{H_1} + 13\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux}) + \mathbb{T}_{H_1} + 2\mathbb{P}$

$= 2\mathbb{T}_H + 3\mathbb{T}_{H_1} + 26\mathbb{S}_{\mathbb{Z}_p} + 12\mathbb{M}_1 + 2\mathbb{M}_2 + 8\mathbb{M}_T + 24\mathbb{E}_1 + 4\mathbb{E}_2 + 8\mathbb{E}_T + 2\mathbb{T}_\psi + 4\mathbb{P} + 2\mathbb{T}_{Aux}$

Open: This algorithm includes the following operations:

- First it runs the verification algorithm so the total cost of open algorithm includes the cost of one verification.
- Computation of $A_{i,itemId}$ requires two multiplication and two exponentiation operations in $\mathbb{G}_1$ which concludes the total cost of $2\mathbb{M}_1 + 2\mathbb{E}_1$.
- In this algorithm GM performs one table lookup from IL_{itemId} table.
- TC(Open) :

  $= \text{TC}(\text{ Verify }) + 2\mathbb{M}_1 + 2\mathbb{E}_1 + \mathbb{T}_{IL}$

  $= \mathbb{T}_H + \mathbb{T}_{H_1} + 13\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux} +$

  $2\mathbb{M}_1 + 2\mathbb{E}_1 + \mathbb{T}_{IL}$

  $= \mathbb{T}_H + \mathbb{T}_{H_1} + 13\mathbb{S}_{\mathbb{Z}_p} + 8\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 14\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux} + \mathbb{T}_{IL}$

Update: The complexity analysis for update algorithm is as follows:

- It includes one verification and two addition operations in $\mathbb{Z}_p$ which costs $2\mathbb{S}_{\mathbb{Z}_p}$
- GM stores the current reputation of the seller who runs this algorithm thus one *Rep* table lookup is required.
- TC(Update) :

  $= \text{TC}(\text{ Verify }) + 2\mathbb{S}_{\mathbb{Z}_p} + \mathbb{T}_{Rep}$

  $= \mathbb{T}_H + \mathbb{T}_{H_1} + 13\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux} +$

  $2\mathbb{S}_{\mathbb{Z}_p} + \mathbb{T}_{Rep}$

  $= \mathbb{T}_H + \mathbb{T}_{H_1} + 15\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux} + \mathbb{T}_{Rep}$

Chapter 6

Comparison

In this chapter we compare our scheme with other schemes which models anonymous reputation system in centralized setting. We compare our scheme with scheme [7] and scheme [8] as follows:

6.1 General Comparison

Property	**Our scheme**	**Scheme [7]**	**Scheme [8]**
Direct Interaction between Buyer and Seller:	No	Yes	No
Seller Anonymity from GM:	No	No	Yes
Reputation Computation:	Yes	No	Yes

Table 6.1: General Comparison

6.2 Cost Comparison

Our scheme has less computational cost than the existing schemes. In our scheme there are less number of pairing computations compared to other schemes. So it reduces our cost significantly as cost of pairing is much more than the cost of other operations occurred during cost computation.

Remark 2 *Here L is the number of messages to be signed and then verified and signature computation and verification cost of scheme [8] is just one time AHO signature generation and verification cost which is used in various protocols of scheme [8].* TC(AHO Sign) *is same as the cost of signature computation defined in the next step.*

Algorithm	Computational Complexity
Registration:	Our Scheme: $\mathbb{T}_{Reg} + \mathbb{E}_1$ [7]: $\mathbb{T}_{Reg} + \mathbb{E}_1$ [8]: $22\mathbb{E}_1 + 12\mathbb{M}_1 + \text{TC}(\text{ AHO Sign })$
Signature Computation:	Our Scheme: $\mathbb{T}_H + \mathbb{T}_{H_1} + 16\mathbb{S}_{\mathbb{Z}_p} + 10\mathbb{M}_{\mathbb{Z}_p} + 2\mathbb{M}_1 + 3\mathbb{M}_T + 10\mathbb{E}_1 + 4\mathbb{E}_T + 2\mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux}$ [7]: $\mathbb{T}_H + \mathbb{T}_{H_1} + 17\mathbb{S}_{\mathbb{Z}_p} + 10\mathbb{M}_{\mathbb{Z}_p} + 2\mathbb{M}_1 + 3\mathbb{M}_T + 13\mathbb{E}_1 + 4\mathbb{E}_T + 2\mathbb{T}_\psi + 4\mathbb{P}$ [8]: $(2L + 7)\ \mathbb{E}_1 + (2L + 4)\ \mathbb{S}_{\mathbb{Z}_p} + 4\mathbb{M}_{\mathbb{Z}_p} + 2L\mathbb{M}_1$
Verification:	Our Scheme: $\mathbb{T}_H + \mathbb{T}_{H_1} + 13\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + \mathbb{M}_2 + 4\mathbb{M}_T + 12\mathbb{E}_1 + 2\mathbb{E}_2 + 4\mathbb{E}_T + \mathbb{T}_\psi + \mathbb{P} + \mathbb{T}_{Aux}$ [7]: $\mathbb{T}_H + \mathbb{T}_{H_1} + 12\mathbb{S}_{\mathbb{Z}_p} + 6\mathbb{M}_1 + 5\mathbb{M}_T + 12\mathbb{E}_1 + 6\mathbb{E}_T + \mathbb{T}_\psi + 6\mathbb{P} + \mathbb{T}_{Aux}$ [8]: $(2L + 6)\ \mathbb{P} + (2L + 4)\ \mathbb{M}_T$

Table 6.2: Cost Comparison

6.2.1 Complexity Comparison for verification:

In the complexity comparison, the most dominant operation towards the total cost is pairing so we will plot the verification cost of each scheme based on pairing operation. Our scheme uses Type-2 pairing framework involving a pairing friendly BN curve and one pairing operation takes approximately 1.53 milliseconds on a 2.5 GHz Core i5-3210M processor. So we will calculate the cost of verification for each scheme by computing the total cost incurred through total number of pairing operations. Figure 6.1 shows the verification cost for each scheme.

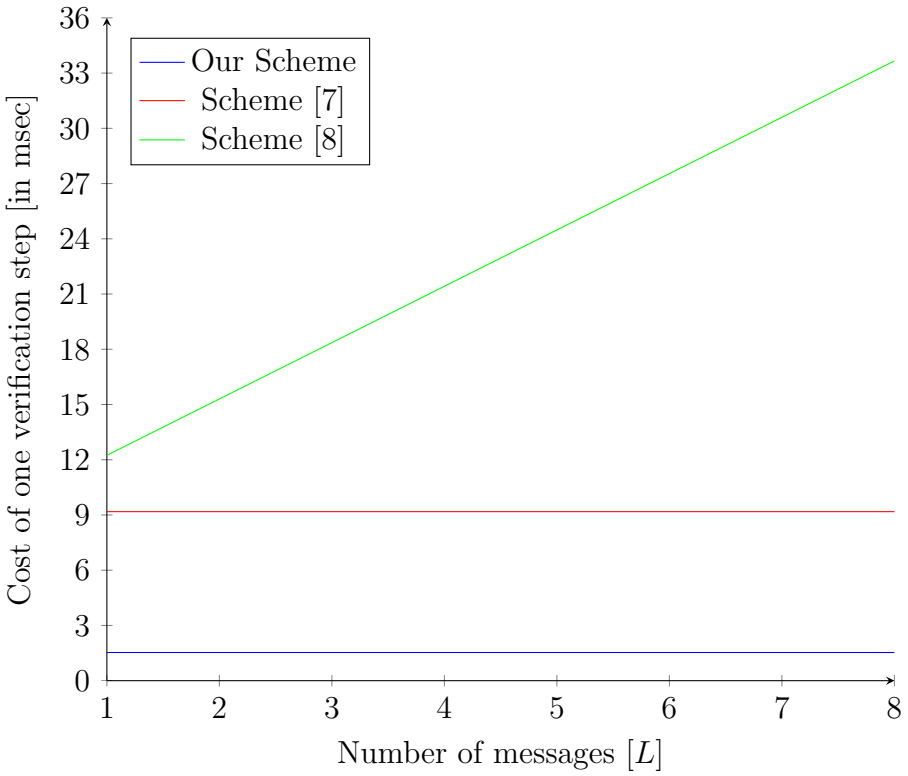

Figure 6.1: Cost Comparison for Verification step

Chapter 7

Security Analysis

This chapter includes all the security notions and adversarial games to prove the security of our scheme. As our scheme possesses properties like anonymity, public linkability, strong-exculpability and traceability. So here we show the adversarial games for each of these properties and give some theorems to prove the security of our scheme. For defining the security for each property, first we define some oracles which will be used for proving the security of these properties. These oracle are defined in Table 7.1 as follows:

Oracle	**Description**
$\mathbf{Hash_H}$	Adversary $\mathbb{A}$ makes hash oracle query to H and gets some $c \xleftarrow{\$} \mathbb{Z}_p$ as a result and the oracle ensures to respond identically to repeated queries.
$\mathbf{Hash_{H_1}}$	Adversary $\mathbb{A}$ call this oracle with an item id $itemId$ as argument and receives $\hat{f} \xleftarrow{\$} \mathbb{G}_2$ as output and the oracle ensures to respond identically to repeated queries.
Add User	To add honest users to the group the adversary $\mathbb{A}$ call this add user oracle with an identity $i \in \mathbb{N}$ as argument. Then oracle runs register protocol for user i and returns $upk[i]$ to the adversary.
Add Item	To add items the adversary $\mathbb{A}$ runs this add item oracle by giving $itemId$ as parameter to this oracle. Then oracle runs KeyGen protocol for this $itemId$ and gives $ipk[itemId]$ to the adversary $\mathbb{A}$.

Table 7.1: Oracle Description

7.1 Anonymity:

For proving the security of anonymity, an adversary is asked to distinguish between two group members who has signed a message for some $itemId$ where $itemId$ and message is chosen by the adversary and these two users must be honest otherwise adversary can link different signatures and find their identities.

Adversarial Game:

This game is played between GM and adversary $\mathcal{A}$. Following is the brief description of the game:

1. **Setup Phase:** GM runs $\text{KeyGen}_{\text{GM}}(\lambda)$ and gives $gmpk$ to adversary $\mathcal{A}$ and keeps $gmsk$ with himself.

2. **Query Phase1:** $\mathcal{A}$ asks queries to GM. These queries are either secret key extraction for users i or signature extraction for $(i, itemId, M)$. Then GM gives secret key or the signature to $\mathcal{A}$ by running KeyGen algorithm for user i and asking for signature from user i respectively.

3. **Challenge Phase:** $\mathcal{A}$ gives two identities i_0, i_1, one message M alongwith item id $itemId$ to GM.

4. **Query Phase 2:** $\mathcal{A}$ again queries to GM which differs from the previous queries.

5. **Response:** GM asks for the signatures from i_0, i_1 for message M and $itemId$. GM runs Issue algorithm for issuing the signing keys for users i_0, i_1 for $itemId$ then user i_0 and i_1 runs Sign algorithm to get the signatures σ_0 and σ_1 and gives σ_0, σ_1 to GM. Then GM chooses $b \xleftarrow{\$} \{0, 1\}$ sends σ_b to adversary $\mathcal{A}$.

6. $\mathcal{A}$ chooses $b' \xleftarrow{\$} \{0, 1\}$. Thus if $b = b'$ then adversary wins the game.

The above game is a CPA-anonymity game. As our scheme is based on short group signature scheme. So anonymity of our scheme follows from the anonymity proof of the group signature scheme. Following theorem given in paper [2] describes the anonymity proof. For the proof refer section 5.1 of [2].

Theorem 7.1.1 *If Linear encryption is (t',ϵ')-semantically secure on $\mathbb{G}_1$ then our scheme is (t, q_H, ϵ)-CPA-fully-anonymous, where $\epsilon = \epsilon'$ and $t = t' - q_H.O(1)$. Here q_H is the number of hash function queries made by the adversary and n is the number of members of the group.*

7.2 Public Linkability

Public linkability implies that if there is one signer who signed for same $itemId$ twice then these two signatures should be linked to the signer. Here we define an adversarial game between GM and adversary.

Adversarial Game:

1. GM runs $\text{KeyGen}_{\text{GM}}(\lambda)$ and gives $gmpk$ to adversary $\mathcal{A}$ and keeps $gmsk$ with himself.
2. Adversary $\mathcal{A}$ outputs $n+1$ tuples of (M_i, σ_i) for the same $itemId$ where i from $\{1, 2, ..., n+1\}$ and n is the number of users and gives it to GM.
3. GM verifies all the message-signature pairs by running verification algorithm for all pairs.
4. If all signatures are verified correctly then GM runs Link algorithm for all possible $(M_i, \sigma_i, M_j, \sigma_j)$ pairs where $i \neq j$. If GM finds any i, j such that Link algorithm returns 1 then GM wins otherwise adversary wins.

Proof for the public linkability of our scheme is followed from the following theorem. For the proof refer lemma 6.2 of paper [7].

Theorem 7.2.1 *If q-SDH (t',ϵ')-holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then our reputation system is (t, ϵ)-public linkable, where $t = t' - Q.O(1)$ and $\epsilon = q_{AI}.\sqrt{c.q_H.(q-1).\epsilon'} + \frac{q_{AI}}{|\mathbb{G}_1|}$. Here q_H, q_{AI} are the number of hash function queries and add item oracle queries by the adversary and c is a constant.*

7.3 Traceability

In this proof an adversary has to output a valid message-signature pair for some $itemId$ which can not be traced back to a user. Thus if adversary wins then GM would not be able to trace back the identity of the user.

Adversarial Game:

1. GM runs $\text{KeyGen}_{\text{GM}}(\lambda)$ and gives $gmpk$ to adversary $\mathcal{A}$ and keeps $gmsk$ with himself.
2. GM has a set T of previously queried signatures where $T = \{\sigma_t\}_{t=0 \text{ to } n}$.
3. Adversary produces one message-signature pair (M, σ) for some $itemId$ and gives $(itemId, M, \sigma)$ to GM.

4. GM runs the Open algorithm which includes the verification of signature σ for message M. Open algorithm returns i.

5. If $(itemId, i, M, \sigma) \in T$ then GM wins otherwise adversary wins.

Security proof for the traceability follows from the following theorem. For proof refer theorem 5.3 of paper [2].

Theorem 7.3.1 *If SDH (q, t', ϵ')-hard on $(\mathbb{G}_1, \mathbb{G}_2)$, then our scheme is $(t, q_H, q_S, n, \epsilon))$-fully-traceable, where $n = q - 1, \epsilon = 4n\sqrt{2\epsilon' q_H} + \frac{n}{|\mathbb{G}_1|}$ and $t = \theta(1).t'$. Here q_H is the number of hash function queries made by the adversary, q_S is the number of signing queries made by the adversary, and n is the number of members of the group.*

7.4 Strong-exculpability

Strong-exculpability means no group member not even the GM can produce the signature on behalf of other users. Thus, no user can be framed for producing a signature he did not produce. A group signature scheme which is secure in the sense of full-traceability also has the exculpability property. For the strong notion even the GM should not be able to produce the signature on behalf of other user.

As in our scheme, for creating a signature, user-specific parameters are required which is only available to user. Thus no user, outsider or GM can produce signatures on behalf of other users which proofs our scheme has strong-exculpability. This property follows from the following theorem. For proof refer lemma 6.4 of paper [7]

Theorem 7.4.1 *If the discrete logarithm problem is (t', ϵ')-hard in $\mathbb{G}_2$, then our system is (t, ϵ)-strong exculpable, where $t = t' - Q.O(1)$ and $\epsilon = q_{AU}.\sqrt{c.q_H.\epsilon'} + \frac{q_{AU}}{|\mathbb{G}_2|}$. Here q_{AU} is the number of oracle queries for adding the new user made by the adversary.*

7.5 Sybil Attack

In general, a Sybil attack is an attack on distributed systems, where many nodes controlled by a few real entities cause the system to misbehave. In reputation systems, this attack concretely means that an adversary controls a large number of nodes and uses them to (1) generate positive reviews for himself to boost his reputation, and (2) leave negative reviews for others (e.g., his competition) to lower their reputation.

Our scheme is not resistant against sybil attack. In our scheme, an attacker can register as a seller and as many buyers and then the attacker generates positive ratings for himself by acting as buyers and boost his reputation. An economically irrational adversary can leave negative reviews for other sellers by buying the products from that seller. Economically irrational adversary can act as multiple buyer and buy cheaper items from a seller and leaves negative reviews for that seller which affects the

aggregate reputation of that seller, as in our scheme we have seller-wise reputation not item-wise reputation. To design an anonymous reputation scheme resistant to sybil attack is left as a future work.

Chapter 8

Future Work and Conclusion

Our objective was to design an efficient centralized anonymous reputation system as most of the online marketplaces use the centralized reputation system. So we designed an anonymous reputation system whose computational complexity is less than the computational complexity of existing schemes. Our scheme is a realistic scheme as in the online marketplaces buyer and seller does not interact with each other and our scheme do so.

There are some open problems which are as follows:

- As our scheme computes only seller aggregate reputation, it will be better if product-wise reputation will also be there in the system. So this will make easier for the customers to choose their products.
- Mitigation of the Sybil attack is also a possible direction of work.
- Implementation of our scheme is also a future work.
- Mapping our scheme from the centralized one to decentralized is also a future work. As paper [9] presents a decentralized anonymous reputation system which uses blockchain. So finding all the transactions and then map it to the scheme like Beaver [9] will give a better understanding of decentralized version of our scheme.

Bibliography

[1] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In EUROCRYPT 2003, pages 614-629. Springer, 2003.

[2] D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In CRYPTO 2004, volume 3152 of LNCS, pages 41-55. Springer, 2004.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. Advances in Cryptology - CRYPTO 2000, LNCS 1880, pages 255-270. Springer, 2000.

[4] S. Clau, S. Schiffner, and F. Kerschbaum. k-anonymous Reputation. In ASIA CCS 2013, pages 359-368. ACM, 2013.

[5] C. Dellarocas. Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior. In EC 2000, pages 150-157. ACM, 2000.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in P2P Anonymity Systems. In Workshop on Economics of Peer-to-Peer Systems, volume 92, 2003.

[7] J. Blomer, J. Juhnke, and C. Kolb. Anonymous and Publicly Linkable Reputation Systems. Financial Crypto, pages 478-488, 2015.

[8] T. Nakanishi and N. Funabiki. An Anonymous Reputation System with Reputation Secrecy for Manager. IEICE Trans, Fundamentals, Vol.E97-A, pages 2325-2335, 2014.

[9] K. Soska, A. Kwon, N. Christin, and S. Devadas. Beaver: A decentralized anonymous marketplace with secure reputation. In cryptology ePrint Archive 464, 2016.

[10] R. Bazin, A. Schaub, O. Hasan, and L. Brunie. A Decentralized Anonymity-Preserving Reputation Systems with Constant-time Score Retrieval. In cryptology ePrint Archive, 416, 2016.

[11] M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In CT-RSA 2005, volume 3376 of LNCS, pages 136-153. Springer, 2005.

[12] A. Jsang and R. Ismail. The Beta Reputation System. In BLED 2002, pages 41-55, 2002.

[13] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In WWW 2003, pages 640-651. ACM, 2003.

[14] J. R. Douceur. The sybil attack, in Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS), 2002. pages 251-260, 2002.

[15] J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. CRYPTO 2004, Pages 56-72. Springer, 2004.

[16] C. Garman, M. Green, and I. Miers. Decentralized Anonymous Credentials. IACR Cryptology ePrint Archive 622, 2013.

[17] E Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin. Reputation System for Anonymous Networks. LNCS, vol. 5134, pages 202218. Springer, 2008.

[18] E. Zhai, D. Issac, R. Chen, E. Syta, and B. Ford. AnonRep: Towards Tracking-Resistant Anonymous Reputation. NSDI 2016.

[19] M. Abe., G. Fuchsbauer, J. Groth, K. Haralambiev, M. Ohkubo. Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pages 209-236. Springer, 2010.

[20] M. Abe, K. Haralambiev, and M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, 133, 2010.

[21] D. Boneh and X. Boyen. Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pages 56-73. Springer, 2004.

[22] J. Bethencourt, E. Shi, and D. Song, Signatures of reputation, in Proceedings of the 14th International Conference on Financial Cryptography and Data Security, ser. FC10. Berlin, Heidelberg: pages 400-407. Springer, 2010.

[23] R. Petrlic, S. Lutters, and C. Sorge. Privacy-preserving reputation management. In Proceedings of the 29th Annual ACM Symposium on Applied Computing, ser. SAC 14. New York, NY, USA: ACM, 2014, pages 1712-1718. ACM, 2014.

[24] E. Anceaume, G. Guette, P. Lajoie Mazenc, N. Prigent, and V. Viet Triem Tong. A Privacy Preserving Distributed Reputation Mechanism. Pages 1951-1956, ICC 2013.

[25] P. Lajoie-Mazenc, E. Anceaume, G. Guette, T. Sirvent, and V. Viet Triem Tong, Efficient Distributed Privacy-Preserving Reputation Mechanism Handling Non-Monotonic Ratings. 2015.

[26] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In ASIACRYPT 2001, volume 2248 of LNCS, pages 514-532. Springer, 2001.

[27] P. Resnick, R. Zeckhauser, J. Swanson and K. Lockwood. The value of reputation on eBay: a controlled experiment. Experimental Economics, 9(2), 2006.

[28] G. Swamynathan, K. C. Almeroth, and B. Y. Zhao. The design of a reliable reputation system. Electronic Commerce Research: pages 239-270, 2010.

[29] A. Schaub, R. Bazin, O. Hasan, and L. Brunie. A trustless privacy preserving reputation system. IFIP SEC: pages 398-411, 2016.

[30] R. Merkle. A certified digital signature. In Advances in Cryptology CRYPTO 89 Proceedings, ser. Lecture Notes in Computer Science, G. Brassard, Ed. Springer New York, 1990, vol. 435, pages 218-238. Springer, 1990.

[31] D. Chaum. Blind signatures for untraceable payments. In Advances in Cryptology, D. Chaum, R. Rivest, and A. Sherman, Eds. Springer US, 1983, pages 199-203. Springer, 1983.

[32] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In 16th International Conference on World Wide Web (WWW), pages 181-190, 2007.

[33] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. A. Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In 22nd Network and Distributed System Security Symposium (NDSS), April 2015.

[34] T. Minkus and K. W. Ross. I know what youre buying: Privacy breaches on eBay. In 14th International Symposium on Privacy Enhancing Technologies (PETS), July 2014.

[35] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In 29th IEEE Symposium on Security and Privacy, pages 111-125, 2008.

[36] A. Sudarsono, T. Nakanishi, N. Funabiki: Efficient proofs of attributes in pairing-based anonymous credential system. In: Fischer-Hbner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pages 246-263. Springer, 2011.

[37] Blockchain, https://blockchain.info/

[38] J. Sanger and G. Pernul. Reusability for trust and reputation systems. Springer Berlin Heidelberg: pages 28-43, 2014.

[39] H. Rahimi and E. Bakkali. A New Trust Reputation System for E-Commerce Applications, volume abs/1405.3199, 2014.