Design and Implementation of an Online Anonymous Feedback System

Saad Inshi

A Thesis

in

The Concordia Institute

for

Information System Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Information Systems Security) at

Concordia University

Montreal, Quebec, Canada

October 2008

Your file Votre référence
ISBN: 978-0-494-45299-8
Our file Notre référence
ISBN: 978-0-494-45299-8

Abstract

Design and Implementation of an Online Anonymous Feedback System

Saad Inshi

Society has long seen anonymity in many forms such as suggestion boxes, unsigned letters and blocked calls. In the online world, the protection of users' anonymity, when performing some online transactions, is an important factor in the acceptance and use of several Internet and web services. Solutions for minimizing release of personal information can be based on many proposed cryptographic techniques for providing anonymity.

In this thesis, we describe the design and implementation of an online feedback system employing an anonymous authentication mechanism based on blind RSA signature scheme. The proposed system prevents malicious evaluators from assuming multiple identities. It also maintains the anonymity of the evaluator even against an abusive authority that has access to the evaluation servers. Based on our design, the authority, responsible for the evaluation process, can be held accountable if it blocks any user's feedback. The system also prevents malicious evaluators from sending multiple evaluations for the same evaluatee. Finally, the developed system is generic enough, user friendly and allows the administrator to change the evaluation form to fit the assessment of essentially any type of function or performance.

Acknowledgments

This is one of the best moments in my Masters' program - to publicly acknowledge those people who have contributed to make my success a part of their own in many different ways. First of all, I would like to express my earnest gratitude to my supervisor, Dr. Amr Youssef, for his perpetual support and guidance with enduring patience.

I'd also like to express my appreciation to all the faculty and people at Concordia Institute for Information Systems Engineering (CIISE) who contributed to my success one way or the other.

I am thankful to my lab-mates, whose help and patience made my work a lot easier.

Finally, I take this opportunity to express my profound gratitude to my beloved parents, my wife, my little daughter and my little son for their moral support and patience during my studies at Concordia University.

Table of Contents

List of Figures

List of Acronyms

DDH	Decision Diffie-Hellman assumption
TTP	Trusted Third Party
OTTP	Online Trusted Third Party
K-TAA	K-Times Anonymous Authentication scheme
IP	Internet Protocol
ASP	Active Server Pages
HTML	HyperText Markup Language
SQL	Structured Query Language
C#	C-sharp Programming Language
XML	Extensible Markup Language
OLAP	On-Line Analytical Processing
CA	Certificate Authority
MMC	Microsoft Management Console
API	Application Programming Interface
CMS	Cryptographic Message Syntax
PKCS	Public Key Cryptographic Standards

Chapter 1

Introduction

Different forms of anonymity, such as suggestion boxes, unsigned letters and blocked phone calls, have been observed by society for many years. Privacy and anonymity are similar entangled concerns; with anonymity often helping individuals better maintain their privacy[1]. For example, a person making an anonymous health query is better assured of her privacy than if she had to rely, for her privacy, on the discretion of those answering her query. In other scenarios, anonymity may even assume a more important role; it is not hard to imagine how whistle-blowers may suffer in many ways, often destroying their career if they lose their anonymity.

Traditional anonymity is prone to technical challenges that might reveal the anonymous source. It is not hard to envision situations where the source of anonymous

---

[1]One can also argue otherwise; anonymity may encourage some people to be careless about revealing other people's private information without the fear of being traced or being held accountable. In this thesis, we focus only on some technical aspects related to anonymity. In other words, we will not concern ourselves with the everlasting debate for or against anonymity.

paper form is revealed using simple fingerprinting techniques. These challenges for anonymity become even greater in the online world where, despite some local increase, there is a dramatic global loss in anonymity.

It is clear that there are a number of aspects to anonymity that must be measured in the anonymous systems. The following types are defined in [23]:

- **Author-anonymity**. In such a system, an author cannot be linked to a document that she has created.
- **Publisher-anonymity**. In such a system, a publisher cannot be linked to a document that she has made available to the network.
- **Reader-anonymity**. In such a system, a user cannot be linked to a request that she has made for a document.
- **Server-anonymity**. In such a system, a server cannot be linked to documents that it is storing.
- **Document-anonymity**. In such a system, a server cannot decide which documents it is storing at this time. This property also implies that there is no information to be gained about stored documents by an attacker that manages to compromise the server. A negative effect of this type of anonymity is that servers are unable to perform advanced matching of stored documents against queries. For example, a free-text search on document titles is impossible as the titles of stored documents are unknown.
- **Query-anonymity**. In such a system, a server cannot determine which document it is serving when satisfying a request.

Because of its multidisciplinary nature, studying different aspects of anonymity has attracted researchers from different fields such as social sciences, law, and engineering. From engineering perspective, cryptographic tools have a lot to offer when it comes to improving privacy and anonymity. As mentioned before, in this thesis, we focus only on the technical aspects of online anonymous feedback systems.

These feedback systems are widely being used in commercial as well as noncommercial contexts. In such systems, anonymity would encourage more participation and if implemented carefully would arguably, also encourage trustworthy behavior, and deter dishonest participants. To further explain this, one should note that in a non-anonymous online feedback system, rater identity is fully known and hence raters are likely to be targets of retaliation in case of negative reporting, especially when feedback information is in the form of qualitative reviews, which contain statements that may be considered as libel. On the other hand, fully anonymous online feedback systems, at the other extreme, make it easier for sources to report without fear of retaliation. However, we have to concede that with this property, we may facilitate irresponsible, manipulative reporting and collusion [30] and hence one may also consider systems where the user anonymity can be revoked (for accountability) when the user violate a pre-specified set of rules.

Many Online feedback (rating) systems are already being used in many online commercial applications, such as eBay (http://www.ebay.com), Amazon (http://www.amazon.com), ePinions (http://www.epinions.com) and Bizrate

(http://www.bitrate.com). The eBay system allows bidders to check the previous history of a particular seller. Following each transaction, eBay encourages both participants to give feedback about their partner, which consists of +1,0,-1 for positive, natural and negative, followed by a text comment. The feedback score is displayed as a part of feedback profile. This feedback can assist other parties in deciding whether or not to transact with that party in the future. Similarly, Amazon allows users to rate others on a 1-to-5 scale with some set of properties such as fairness and product quality. Then the system will attach the result to the user's identity. ePinions' rating system is almost identical to the Amazon system with a text comment. Also users can be rated as "not helpful", "somewhat helpful", "helpful", or "very helpful". Bizrate allows users to rate prices, selections, site navigation, shopping options, and their satisfaction with shopping experiences. The merchant will be granted a customer certification if and only if it gets a sufficient number of positive ratings over a period of time. For the above applications, the use of pseudonyms might be enough to palliate source shyness. However, pseudonyms must be stable over time: if a source constantly changes pseudonyms, it is even worse than anonymity in the sense that it is a potential indication of untruthful reporting.

In this thesis, we are more concerned with applications where the users submitting the feedback belong to a predetermined population of much smaller size compared to the above Internet rating systems. One familiar application is the online course evaluation (questionnaire) at Concordia University which requires students,

registered to a given course, to provide selected response answers to a series of questions in order to anonymously evaluate the instructor teaching effectiveness in this course.

1.1 Traditional vs. Online Feedback Systems

Traditional feedback requires users to fill out at least one paper to rate a user, instructor, course or product. For example, some universities are using printed forms, which are typically filled by hand at the end of every semester. These forms might very tedious to complete during class time; also, results are delivered weeks after the evaluation.

Online feedback systems are capable of accomplishing the same task as traditional evaluations and would be easier and more cost-effective to users. These systems are immediate and flexible. They may also allow users to review real-time results presented in various formats. Digitally stored data could also be readily used for analysis and establishing historical trends. Finally, the users have the opportunity to complete the evaluation on their own time.

These features make online feedback systems more convenient for all users. On the other hand, online users would typically be more concerned about their privacy and anonymity because of the necessary login methods. These concerns will not be alleviated unless these online systems guaranteed that evaluators' identities could not be traced back.

1.2 Motivation of the Research

There are several difficulties and challenges facing online feedback systems. One that often mentioned is the fact that most implementations for online feedback systems do not ensure the user's anonymity. For example, in most of the electronic course evaluation systems presently implemented by several universities, an abusive authority, which has access to the evaluation server log files, can easily link each student with her evaluation. While this is unlikely to happen in the academic environment where ethics and morals are highly valued, one can easily imagine several other scenarios where this may not be the case. This lack of trust in achieving true anonymity and the fear of a retaliatory response may prevent evaluators from casting their true opinions and hence limits the effectives of the overall evaluation process.

On the other hand, allowing uncontrolled anonymous feedback (e.g., RateMyProfessor.com) opens the door to *Sybil* attacks where a malicious user may subvert the system by creating a large number of pseudonymous entities, and use them to gain a disproportionately large influence.

It should also be noted that, in some situations, the authority overseeing the evaluation process may not always be unbiased. For example, a large number of negative ratings for buyers or sellers on a given virtual market website (e.g., eBay) will certainly reduce the total number of transactions and consequently the profit gained by this website. Thus, a dishonest company may try to block some negative feedbacks in order to optimize its profit; at least in the short term.

In this thesis, we present an implementation for a web-based feedback system using RSA blind signature scheme [10]. The proposed system is generic enough, user friendly and allows the administrator to change the evaluation form to fit the assessment of any type of function or performance. In particular, by employing an anonymous authentication mechanism, the developed system ensures that:

(i) The evaluator's anonymity is protected even against an abusive passive authority that has access to the evaluator server (we assume that this authority can read log files but cannot change the software running on this server).

(ii) Every registered user is allowed to cast her vote only once.

(iii) By issuing evaluation receipts, the authority responsible for the evaluation process can be held accountable if it blocks any user's feedback.

1.3 Thesis Overview

The rest of the thesis is organized as follows.

- In Chapter 2, we review some of the available cryptographic primitives for providing anonymous online feedback systems and their security requirements. We also present a comparison of these primitives based on general security properties.
- In Chapter 3, we present essential assumptions and the design of the developed online anonymous feedback system.
- In Chapter 4, we describe the implementation; provide some security analysis and the countermeasures of various attacks on our system.
- In Chapter 5, we give the conclusions as well as some future research directions.

Chapter 2

Cryptographic Primitives for Supporting Anonymity

2.1 Introduction

Cryptographic protocols [14] provide several invaluable primitives that can be used to design systems with stringent anonymity, also known as unlinkability, requirements (e.g., e-voting [2, 9, 11, 40], e-cash [2, 6, 7, 8, 1], and electronic coupons [22]).

In this chapter, we review some of these cryptographic primitives for supporting anonymous online feedback systems and their security requirements. For completeness, we start this chapter with a brief review of some public key preliminaries.

2.2 Preliminaries

2.2.1 Public Key Cryptosystem and Digital Signature

Public-key cryptography, also known as asymmetric cryptography, was first introduced by Diffie and Hellman [6] in 1976. The main idea of this technique is that a user can have a pair of cryptographic keys: a public key, which can be widely distributed, and a private key, which is kept secret. To ensure confidentiality, when the message encrypted by recipient's public key, no one else can decrypt the message without knowing the recipient's private key. A public key cryptosystem can also be used for digital signature [7]. A digital signature scheme can be used to ensure authenticity by establishing that the message must have been signed by the sender's private key. Then it can be verified by anyone who has access to the sender's public key.

Thus a digital signature of a message can be seen as a block of data dependent on some secret known only to the signer, and, additionally, on the content of the message being signed. A digital signature is verifiable; if a dispute arises as to whether Alice signed a message, an unbiased third party should be able to resolve the matter equitably, without requiring access to Alice's secret information. Typically, a digital signature scheme consists of a signature generation algorithm and an associated verification algorithm. A digital signature generation algorithm is a method for a signer to produce a digital signature on a particular message. A digital signature verification algorithm is a

method for verifying that a digital signature is authentic, (i.e., was indeed created by the specified signer).

A digital signature generation algorithm (the signing process) must use some secret information that is only accessible to the signer. A digital signature verification algorithm, in contrast, uses some public information about the signer to verify the signature. Some public-key cryptography algorithms, among other mechanisms, can be deployed as digital signature schemes: the signer signs a message with her private key, while anyone else may verify the signature using the public key of the signer. It should be noted that with some public-key cryptography algorithms such as RSA, signing a message is actually employing the encryption algorithm on the message with the signer's private key, and verifying a signature is actually employing the decryption algorithm on the signature with the signer's public key. But this is not always the case for all public-key systems. The following discussions on digital signature mechanisms are based on public-key cryptography algorithms that are similar to the RSA algorithm.

Suppose Alice wants to send a signed message m to Bob. Generally the first step is applying a one-way hash function $\mathcal{H}$ to the message to create the message digest.

$$h = \mathcal{H}(m)$$

To create a digital signature S, the signing process usually signs the message digest h instead of the message itself. In this way, it saves a considerable amount of time, because a message digest is shorter compared to the message itself.

Next, by encrypting the message digest h with her private key K_{Alice}^{-1}, Alice actually creates a digital signature S on h with a secret of herself:

$$S = Sing(h, K_{Alice}^{-1})$$

Alice sends Bob the signature S together with the message. In order for Bob to verify the signature, he must first apply the same hash function $\mathcal{H}$ as Alice did to the message m she sent him:

$$h_1 = \mathcal{H}(m)$$

Then he verifies Alice's signature S using her public key:

$$Verify(S, K_{Alice}, h_1)$$

The verification process above is actually accomplished in two steps. First, it decrypts S with Alice's public key K_{Alice} to get a value h_2. Second, it compares h_1 and h_2: if they are the same, it means that the signature is successfully verified; otherwise, the verification fails, which may suggest that either someone is trying to impersonate Alice, or the message itself has been altered since Alice signed it, or an error occurred during the transmission.

2.2.2 RSA Algorithm

Rivest, Shamir and Adleman [8] introduced the RSA system, which is considered to be the first algorithm, in the public literature, suitable for both signing as well as encryption. The RSA security is based on the difficulty of factoring large numbers. The RSA algorithm can be described as following.

(i) **Key generation**. To generate the public and the private key pairs, Alice chooses two large prime numbers p and q, and computes their product $n = p.q$. Then,

Alice randomly chooses a number e smaller than n, such that e and $\Phi(n)$ are relatively prime, where $\Phi(n) = (p-1)(q-1)$. After that Alice computes d such that $e.d = 1(mode\ \Phi(n))$. Then (n, e) are announced as public key of Alice and d is kept secret as Alice's private key.

(ii) **Encryption**. Having access to Alice's public key, Bob is now able to send an encrypted message m to Alice. Bob uses Alice's public key to computes $c = m^e \bmod n$, where c is the encrypted message.

(iii) **Decryption**. Alice uses her private key to compute $m = c^d\ mod\ n$, where m is the original message.

2.2.3 Commitment Schemes

In cryptographic protocols making a commitment simply means that a player in a protocol is able to choose a value from some finite set and commit to her choice such that she can no longer change her mind. She does not however, have to reveal her choice, although she may choose to reveal it at some later time [33].

As an informal example, consider the following game between two players, Alice and Bob.

(i) Alice wants to commit to a bit b. To do so, she writes down b on a piece of paper, puts it in a box, and locks it using a padlock.

(ii) Alice gives the box to Bob.

(iii) Later on, if Alice wants to open her commitment, she can do so by giving Bob the key of the padlock.

There are three basic requirements, which are essential to any commitment scheme:

(i) Having given away the box, Alice cannot anymore change what is inside. Hence, when the box is opened, we know that what is revealed was the choice that Alice committed. This is usually called the binding property.

(ii) When Bob receives the box, he cannot tell what is inside before Alice decides to give the key. This is usually called the hiding property.

(iii) If both Alice and Bob follow the protocol, Bob will always recover the committed value. This is usually called the validity.

2.3 Cryptographic Primitives for supporting anonymity

In this section we list several cryptographic primitives that were designed to achieve some anonymity and accountability objectives.

2.3.1 Blind Signature Schemes

A blind signature is the equivalent of signing carbon paper lined envelopes. Writing a signature on the outside of such envelope leaves a carbon copy of the signature on a slip of paper within the envelope. When the envelope is opened, the slip will show the carbon image of the signature. The concept of blind signatures was introduced by Chaum [10] as a method to digitally authenticate and sign a message without knowing the

content of the message. The resulting signature can be publicly verified against the original unblinded message in the same manner of a regular digital signature. Blind signatures are typically employed in privacy-related protocols, such as digital cash and electronic voting, where the signer and message author are different parties.

Blind signature schemes must satisfy the following requirements:

(i) **Anonymity**: also known as unlinkability, which prevents the signer from linking later the blinded message to unblinded version that it may be called upon to verify. In this case, the signer's response is first unblinded prior to verification in such a way that the signature remains valid for the unblinded message. This property makes blind signature useful in schemes where anonymity is required.

(ii) **Verifiability**: The receiver of the signature should be able to verify the stripped signature was formed using signer's private key.

Blind RSA Signature Procedures

Blind signature schemes can be implemented using a number of common public key signing schemes. In what follows, we describe the RSA blind signature scheme. Suppose Bob wants Alice to sign a message m without knowing the content of m, then they have to follow these protocol steps:

(i) Bob randomly selects a random blinding factor k and makes a blinding transformation on m to get the blinded message as:

$$\grave{m} = m.k^e mod\, n$$

where (n, e) are the public signing key of Alice.

(ii) Bob sends $\grave{m}$ to Alice. Note that Alice will not be able to recover m from $\grave{m}$ because she does not know the blinding factor k.

(iii) Alice signs $\grave{m}$ using her private key to produce

$$\grave{S} = \grave{m}^d = (m.k^e)^d = k.m^d\ mod\ n$$

and then returns $\grave{S}$ to Bob.

To obtain Alice's signature on m, Bob computes

$$S = \grave{S}.k^{-1} = m^d\ mod\ n$$

Note that asking Alice to sign a one way function of m (say H(m)) will not satisfy the unlinkability requirement above after m is revealed.

2.3.2 Group Signature Scheme

In 1991, Chaum and van Heyst proposed the concept of a group signature scheme [20], which allows a group member to anonymously sign a message on behalf of the group. The most significant part of this scheme is that the signatures can be verified with a single public key of the group without revealing the identity of the signer. Furthermore, it is not possible to decide whether two signatures have been issued by the same group member. The *Group Manager* (GM) is the only one that can add a group

member to the system while the revocation manager is responsible for revoking the anonymity of signatures.

More precisely, associated to the group there is a single signature-verification key called the group public key, and each group member has its own secret signing key that can produce signature relative to the group public key. The GM has a secret key based in which it can extract the identity of any malicious group member.

Various group signature schemes [16, 20, 24, 32, 34, 35] have been proposed so far. All of them should satisfy these basic requirements:

(i) **Anonymity**: The identity of any group member cannot be revealed without the help of the GM.

(ii) **Soundness and completeness**: The verification of the correct group member's signature must be always accepted. On the other hand, invalid signatures must always fail the verification step.

(iii) **Unlinkability**: It is not possible to decide whether two signatures have been issued by the same group member.

(iv) **Unforgeability**: Only group members are able to make valid signatures.

(v) **Unforgeable Traceability**: GM cannot accuse a group member of creating a signature which she has not created.

(vi) **Colluding Resistance**: Even if all group members, including the GM, collude with each other, they should not be able to forge a signature for a non-participating group member.

The major distinct feature of group signature scheme is that the GM has the ability to reveal and trace any group member even if there is no dispute. In some applications, this feature helps improve accountability, while in other applications it can present a real threat to user's anonymity.

In what follows, we present some details of a typical group signature scheme [24].

Group Signature Procedures

The group signature scheme described in [24] consists of the following five procedures: setup, join, sign, verify and open.

Setup:

Given a security parameter κ, GM generates a secure κ-bit RSA modulus $n = pq$ and $d, e \in \mathbb{Z}_n^*$ such that $de = 1\ (mod\ \varphi(n))$. An Online Trusted Third Party (OTTP) also generates a κ-bit RSA modulus $N = PQ$, where N < n and $D, E \in \mathbb{Z}_N^*$ such that $DE = 1\ (mod\ \varphi(N))$.

The group public keys are (e, n) and (E, N), while d and D are kept secret by the GM and OTTP, respectively. A cryptographic hash function $\mathcal{H} : \{0,1\}^* \rightarrow Z_N$ is defined and made public.

Join:

Suppose that u_i wants to join the group and that the communication among the evaluators, OTTP and GM is secure. The GM chooses a random number r from $\mathbb{Z}_n^*$ and computes $d_i = d - r \ (mod\ \varphi(n))$. Then r is sent to u_i and (d_i,u_i) is sent to OTTP. After this action, u_i becomes a group member and her group membership secret key is r.

Sign:

To generate a group signature on a message m, u_i collaborates with OTTP to do the following:

(i) u_i sends $\mathcal{H}(e||E,m)$ along with her identity to OTTP.

(ii) OTTP first checks that u_i's membership has not been revoked. It then stores $(u_i, \mathcal{H}(e||E,M))$ and computes $S_i = \mathcal{H}(e||E,M)^D (mod\ N)$ and $Sig_i = S_i^{d_i} (mod\ n)$. (S_i, Sig_i) is sent back to u_i.

(iii) After receiving (S_i, Sig_i), u_i checks that $\mathcal{H}(e||E,M) = S_i^E \ (mod\ N)$ holds. She then computes $Sig_i' = S_i^r (mod\ n)$. The group signature on message m is set to be $\sigma = Sig_i.Sig_i' \ (mod\ n)$.

Verify:

Having received the group signature σ on message m, the verifier computes $S = \sigma^e \ (mod\ n)$. She accepts the group signature if

$$\mathcal{H}(e||E,M) = S^E \ (mod\ N).$$

Open:

In the case of a dispute, the GM has to open some group signatures on some messages. Suppose that she wants to open a group signature σ on some message m. She only needs to send an enquiry to OTTP. OTTP consults the storage list and sends the original signer u_i back to GM.

2.3.3 Ring Signature Schemes

In 2001, Rivest, Shamir and Tauman introduced the concept of ring signatures [21], which can be considered as a simplified group signature without group manager. A valid ring signature will convince the verifier that this signature has been generated by one of the ring members without revealing any information that will link to the actual signer.

One of the most significant security properties in ring signature schemes is that it is difficult to determine which of the group members' key was used in producing the signature. Consequently, there is no way to revoke the identity of individual signer, even if there is a dispute. In other words, ring signature can provide full anonymity, but fail to ensure accountability.

This scheme must satisfy the following requirements:

(i) **Anonymity**: A verifier is able to authenticate the validity of the signature, but is unable to identify the signer.

(ii) **Unforgeability** : The signature could neither be forged by the verifier, nor be generated by a non-member.

(iii) **Confidentiality**: Message m is sent in ciphertext form so that it can only be decrypted by those with a secret session key.

In what follows, we describe a typical ring signature scheme [21].

Ring Signature Procedures

The formal description of the ring signature scheme [21] is as follows:

Ring Signature Generation

Let $P_1, P_2, \ldots, P_r$ be the sequence of public keys of all the ring members, where P_i is an RSA public key and $P_i = (n_i, e_i)$ for $i = 1,2, \ldots, r$. Also, each ring member has a secret key d_s , where $s \in \{1,2, \ldots, r\}$.

Given a message m to be signed, the signer computes a ring signature as follows.

(i) Determine the symmetric key: The signer first computes the symmetric key k as the hash of the message m to be signed

$$k = \mathcal{H}(m)$$

(ii) Pick a random glue value: The signer picks an initialization value v uniformly at random from $\{0,1\}^b$ where b is a large number.

(iii) Pick random x_i's: The signer picks random x_i for all the other ring members $1 \leq i \leq r$, where $i \neq s$, and computes $y_i = g_i(x_i)$, where g is a trapdoor permutation function.

(iv) Solve for y_s: Let $C_{k,v}(y_1, y_2, \ldots, y_r)$ be a family of keyed combining functions, which take as input a key k, an initialization value v, and arbitrary values $y_1, y_2, \ldots, y_r$. The signer solves the following ring equation for y_s:

$$C_{k,v}(y_1, y_2, \ldots, y_r) = v$$

By assumption, given arbitrary values for the other inputs, there is a unique value for y_s satisfying the equation, which can be computed efficiently.

(v) Invert the signer's trapdoor permutation: The signer uses her knowledge of her trapdoor in order to invert g_s on , y_s to obtain x_s:

$$x_s = g_s^{-1}(y_s)$$

(vi) Output the ring signature: The signature on the message m is defined to be the $(2r + 1)$-tuple:

$$(P_1, P_2, \ldots, P_r; v; x_1, x_2, \ldots, x_r)$$

Ring Signature Verification

A verifier can verify an alleged signature $(P_1, P_2, \ldots, P_r; v; x_1, x_2, \ldots, x_r)$ on the message m as follows.

(i) Apply the trapdoor permutations: For $i = 1, 2, \ldots, r$ the verifier computes

$$y_i = g_i(x_i)$$

(ii) Obtain k: The verifier hashes the message to compute the symmetric encryption key k:

$$k = \mathcal{H}(m)$$

(iii) Verify the ring equation: The verifier checks that the y_i's satisfy the fundamental equation:

$$C_{k,v}(y_1, y_2, \ldots, y_r) = v$$

If this ring equation is satisfied, the verifier accepts the signature as valid. Otherwise the verifier rejects.

2.3.4 List Signature Schemes

List signature schemes [26] are variants of group signatures which set a limit on the number of signatures each group member may issue. These limits must be enforced without having the group manager (GM) open signatures of honest group members to see whether some group members exceed their limited number of the signatures. However, this scheme has the ability to trace dishonest members without the intervention of the GM.

List signature schemes satisfy the following requirements [26]:

(i) **Anonymity**: Given a set of signatures over different time frames, the adversary cannot determine whether two of them were produced by the same group member.

(ii) **Correctness**: Firstly, an adversary cannot prevent honest group members from producing valid signatures; also, an adversary cannot cause the detection of the signatures of an honest group member.

(iii) **Soundness:** An adversary can produce at most one valid signature per time frame per corrupted player without being detected, or without the identity of a corrupted group member being released. Note that signatures obtained from querying honest signers do not count as produced by the adversary. Also if applicable, an adversary cannot produce a valid signature that does not open to a corrupted player.

In here, we describe the list signature scheme proposed in [26].

List Signature Procedures

Setup

The GM picks a group G_q of prime order q and publishes (G_q, q), together with two randomly chosen generators g and h of G_q. The Decision Diffie-Hellman problem is assumed to be hard in the group G_q. Further, cryptographic hash functions $\mathcal{H} : \{0,1\}^* \rightarrow G_q$ and $\mathcal{H}' : \{0,1\}^* \rightarrow \mathbb{Z}_q$ are defined.

Join

A user u_i joins by picking a random $x \in \mathbb{Z}_q$ and publishing $P = g^x$, together with a non-interactive proof of knowledge of x. Henceforth we assume that u_i is legally bound to its public key P_i and we set $x_i = log_g P_i$.

Sign

Let $(P_1, \ldots, P_N)$ be a group public key and let $i \in \{1, \ldots, N\}$. User u_i wishing to sign a message m in time frame T first computes $t = \mathcal{H}(T)$, $s = \mathcal{H}(T, 1)$ and $X = \mathcal{H}'(m, T)$. The values $T_1 = t^{x_i}$ and $T_2 = s^{x_i} P_i^X$ are published together with a non-interactive proof of knowledge that for some $i \in \{1, \ldots, N\}$ it holds that the user knows an $x \in \mathbb{Z}_q$ such that $P_i = g^x, T_1 = t^x$, and $T2 = (sg^X)^x$ without revealing x nor i.

Verify

Verify that the public key is correct, i.e., it only consists of values resulting from the Join protocol, and verify the non-interactive proof included in the signature.

Match

If two signatures (T_1, T_2) on message m and (T_1', T_2') on message m' based on the same time frame T satisfy $T_1 = T_1'$ compute

$$P = (T_2/T_2')^{1/(X-X')}$$

where $X = \mathcal{H}'(m, T)$ and $X' = \mathcal{H}'(m', T)$. Identify the double-signer as u_i with $P = P_i$.

2.3.5 Electronic Coupons

In this section, we show how electronic coupons can be used to provide user's anonymity. While originally proposed for e-commerce applications, electronic coupons can also be used in the design of anonymous feedback systems.

Similar to group signature schemes, the electronic coupons system described in [22] allows the GM to trace users.

Electronic coupons scheme must satisfy the following requirements:

(i) **Anonymity**: No one except the GM can reveal the identity of the user.

(ii) **Unreusability**: The user can be identified if she submitted same feedback more than once.

(iii) **Unforgeability**: This means that the feedback cannot be forged.

Electronic Coupon Procedures

The formal description of the electronic coupon scheme in [22] is as follows:

Setup

Assume that a ticket is divided into k sub-tickets, each sub-ticket is assigned to the index i ($1 \leq i \leq k$), and all participants agree on the type of ticket, that is, the face values of the ticket and the sub-tickets, and the indices in advance. If different types of tickets are used, the GM and user must execute the following step to make secret and public keys for each type of ticket. Thus, a public key is assigned to a type of ticket.

(i) GM computes the following:

- An RSA modulus n and two public exponents $e_1, e_2 > 1$.
- Two integers $f_1, f_2 > 1$.
- A cyclic group $G = \langle g \rangle$ of order n, where g is a generator of G in which computing discrete logarithms is infeasible.
- An element $h \in G$ whose discrete logarithm to base g is unknown. The public key for GM is $P = (n, e_1, e_2, f_1, f_2, G, g, h)$, and the secret key is the factorization of n.

(ii) Each participant publishes the public key of any digital signature scheme and keeps the corresponding secret key.

Obtaining the Tickets

(i) A user chooses $x \in_R Z_n^*$ to compute $y = x^{e_1} \bmod n$ and $z = g^y$. Then, the user chooses $r_1, r_2 \in_R Z_n^*$ to compute $\tilde{y} = r_1^{e_2}(f_1 y + f_2) \bmod n$, $C_1 = h^{r_2} h^y$, and $C_2 = y_R^{r_2}$. Then the user sends $(\tilde{y}, z, C_1, C_2)$ along with her signatures which will prove the correctness of $(\tilde{y}, z, C_1, C_2)$.

(ii) The GM verifies if the signatures are correct, then she will send the user $\tilde{v} = \tilde{y}^{1/e_2} \bmod n$.

(iii) The user computes $v = \tilde{v}/r_1 \bmod n$ to obtain the ticket (x, v), where

$$v \equiv (f_1 x^{e_1} + f_2)^{1/e_2} \pmod n$$

Performing the Evaluation

(i) When a user spends the i^{th} sub-ticket, she computes $\tilde{g} = g^{\tilde{r}_1}$, $\tilde{z} = \tilde{g}^{y}$, $\tilde{h}_i = h_i^{h^{y}}$, $\tilde{C}_1 = h^{\tilde{r}_2} g^{y}$ and $\tilde{C}_2 = y_R^{\tilde{r}_2}$, where $\tilde{r}_1, \tilde{r}_2 \in_R Z_n^*$. Then she sends $(i, \tilde{g}, \tilde{z}, \tilde{h}_i, \tilde{C}_1, \tilde{C}_2)$ along with her signatures which will prove the correctness of $(i, \tilde{g}, \tilde{z}, \tilde{h}_i, \tilde{C}_1, \tilde{C}_2)$ are assured to the feedback system's server.

(ii) The feedback server verifies that the transcript is correctly formed and saves the feedback in its database.

Tracing

(i) The GM sends the feedback server two transcripts of the feedbacks where the same sub-ticket is used and the feedback server verifies that the transcripts are correctly formed. If they are correctly formed, the feedback system's server sends the GM $\hat{z} = \tilde{C}_1 / \tilde{C}_2^{\,1/\rho}$.

(ii) The GM searches for z identical to $\hat{z}$ to present the user's signature on z, which indicates the multiple submission of the ticket.

2.3.6 K-Times Anonymous Authentication Scheme

K-Times Anonymous Authentication (K-TAA) scheme was first proposed by Teranishi, Furukawa, and Sako [3]. This scheme allows members of a group to be anonymously authenticated by application providers for a bounded number of times. K-TAA provides strong anonymity to a user in the sense that no one, not even an authority,

can trace the identity of the users, as long as they have been authenticated within the allowable K times. K-TAA has applications in e-voting, e-cash, electronic coupons and anonymous trial browsing of content. Nguyen and Safavi-Naini [29] proposed dynamic k-times anonymous authentication scheme from bilinear paring to enhance the privileges of application providers.

K-TAA scheme should satisfy these basic requirements [3]:

(i) **Anonymity**: No one is able to identify the authenticated member, or to decide whether two accepted authentication procedures are performed by the same member if the authenticated member has followed the authentication procedure within the permitted number of times per the GM.

(ii) **Detectability**: A public tracing procedure will be executed if a colluding subset of members has been authenticated beyond the total number of times each member is allowed to be authenticated by the feedback server.

(iii) **Correctness**: An honest member will be accepted by the authentication procedure performed by the feedback server.

(iv) **Unforgeability**: Without the help of GM or members, no colluding group (from non-members) can be authenticated as members.

Compared to blind signature scheme, k-times anonymous authentication scheme can provide traceability for any user who tries to authenticate herself (e.g. to cast her vote) more than the prescribed limit.

K-Times Anonymous Authentication Procedures

The formal description of this scheme [29] is as follows:

Setup

GM randomly chooses 2ν-bit rigid integer n (we call n a rigid integer if natural number n can be factorized into two safe primes of equal length p_1, p_2), where ν is a security parameter.

Let $\mathbb{Z}_n$ denotes the ring of natural numbers and natural numbers from 0 to $n - 1$, and $QR(n)$ be the multiple group of quadratic residues of $\mathbb{Z}_n$; $\mathcal{H}_X$ denotes a full domain hash function onto set X . Then, GM randomly chooses μ-bit string R_{GM}, where μ is a security parameter, and computes:

$$((a', a_0'), b) = \mathcal{H}_{\mathbb{Z}_{n^2} \times G}(R_{GM}) \quad \text{and} \quad (a, a_0) = (a'^2, a_0'^2) mod\, n \in QR(n)^2.$$

The group secret key is (p_1, p_2) and the group public key is (n, R_{GM}, a, a_0, b).

Joining

(i) A user u_i selects $x' \in_U \Lambda$, where Λ as a set of integers that in $(0, 2^\lambda)$ and λ is a security parameter; u_i sends its commitment C to GM with a validity proof to show that C is correctly computed from x'.

(i) The GM verifies the proof, and sends $x'' \in_U \Lambda$ to u_i.

(ii) u_i confirms that $x'' \in_U \Lambda$ is satisfied and computes:

$$x = ((x' + x'')\ mod\ 2^\lambda)$$

and $(\alpha, \beta) = (a^x \bmod n, b^x)$, and adds new data (i, β) to the identification list $LIST$. Then, u_i sends (α, β) to the GM with a validity proof.

(iii) GM verifies (i, β) is an element of the identification list, and the proof is valid. Then, GM generates a prime $e \in_U \Gamma$, where Γ is a set of integers in the range $(2^\gamma, 2^\gamma + 2^\lambda)$, where γ and λ are security parameters; then computes $A = (\alpha\, a_0)^{1/e} \bmod n$, and sends (A, e) to user u_i.

(iv) u_i confirms that equation $a^x\, a_0 = A^e \bmod n$ is satisfied, e is a prime, and e is an element of Γ. The new member M's secret key is x, and her public key is (α, A, e, β).

Bound Announcement

The feedback system's server V publishes (ID_V, k_V). Here, ID_V is her ID. Let:

$$(t_1, \check{t}_1) = \mathcal{H}_{G^2}(ID_V, k_V, 1), \ldots, (t_V, \check{t}_V) = \mathcal{H}_{G^2}(ID_V, k_V, k_V)$$

We call $(t_w, \check{t}_w)$ the w-th tag base of V.

Authentication

(i) A member M increases counter C_{ID_V, k_V}. If value w of counter C_{ID_V, k_V} is greater than k_V, then M sends $\perp$ to V and stops.

(ii) V sends random integer $l \in_U [0, 2^{\mu+\varepsilon}] \cap \mathbb{N}$ to M.

(iii) M computes tag $(\tau, \check{\tau}) = (t_w^x, (b^l\, \check{t}_w)^x)$, using M's secret key x and the w-th tag base $(t_w, \check{t}_w)$, computes proof that $(\tau, \check{\tau})$ is correctly computed, and sends $(\tau, \check{\tau})$ and the validity proof to V.

(iv) If the proof is valid and if τ is different from all search tags in its authentication logs, V adds tuple $(\tau, \check{\tau}, l)$ and the proof to the authentication log LOG of V, and outputs accept.

Public Tracing

(i) From LOG, one finds two data $(\tau, \check{\tau}, l, PROOF)$ and $(\tau', \check{\tau}', l', PROOF')$ that satisfy $\tau = \tau'$ and $l \neq l'$, and that $PROOF$ and $PROOF'$ are valid. If one cannot find such data, then one outputs $NO - ONE$.

(ii) One computes:

$$\beta' = \left(\frac{\check{\tau}}{\check{\tau}'}\right)^{\frac{1}{l-l'}} = \left[\frac{(t^x, (b^l \check{t})^x)}{(t^x, (b^{l'} \check{t}')^x)}\right]^{\frac{1}{l-l'}} = b^x$$

and searches pair (i, β) that satisfies $\beta = \beta'$ from the identification list. Then, one outputs a member's ID i. If there is no such (i, β), then one affirms that GM has deleted some data from the identification list.

Chapter 3

The Proposed System

3.1 Introduction

In this chapter, the main design details of our configurable online anonymous feedback system are provided.

Fundamentally, we can identify three types of entities that participate in our system, namely the *Evaluator (E)*, who performs the evaluation, *Registration Server (RS)*, who signs the evaluation and *Evaluation Server (ES)*, who verifies and saves the evaluation forms in its database.

3.2 Assumptions

In order to make our design useful, it is essential to verify that the following assumptions are satisfied in the implementation setup:

(i) *E*'s identity is unique when using our system.

(ii) We assume that there is a large number of evaluators in the network and some of those evaluators are adversaries, who may submit multiple biased feedbacks against or for the same party.

(iii) We assume the existence of a Mixnet-based anonymous communication system [14, 38, 39] so that noone is able to discover the identity-related information, such as the E's IP address even if they have access to the servers log files.

3.3 System Design

Our proposed system, in addition to providing anonymity, is designed to be user friendly and generic. In other words, it allows the administrator to change the evaluation form to fit the assessment of any type of function or performance. Our system design comes in two flavors:

(i) Untraceable online anonymous feedback system based on the RSA blind signature scheme [10] in which every registered evaluator is allowed to anonymously cast her vote only once and no one will be able to backtrace her;

(ii) Traceable online anonymous feedback system based on k-times anonymous authentication scheme [3] in which, if a malicious evaluator sends multiple feedbacks against the same evaluatee, then she can be identified.

3.3.1 Untraceable Online Anonymous Feedback System

We designed this scheme to maintain the anonymity of the evaluator even against an abusive authority that has access to the evaluation server by employing an anonymous authentication mechanism based on RSA blind signature scheme [10] (see section 2.3.1). Moreover, in this design the evaluator can never be identified by the

adversaries, the *Registration Server* or the *Evaluation Server*, even if all of them collude together. In the following sections we will describe the main procedures of our proposed scheme.

The Procedures

Evaluation

When E wants to anonymously submit her *Feedback (F)* to the *ES*, she does the following (see Figure 3.1):

(iii) E generates a fresh 128 bit random number $EvalNO$.

(iv) E computes:

$$H_{Evaluation} = H(F||EvalNO)$$

where H presents the SHA-1 hash value of the evaluation form concatenated with $EvalNO$.

(v) Generate a random number blinding factor k and perform the blinding transformation on $H_{Evaluation}$ to get the blinded message

$$H'_{Evaluation} = H_{Evaluation}.k^e \ mod\ n$$

where (n, e) is the public key of *RS*. Then E sends $H'_{Evaluation}$ to *RS*.

(vi) *RS* signs the received message as: $Sig' = (H'_{Evaluation})^d \ mod\ n$ and then returns the signature to E, where d key is the private of *RS*

(vii) E unblinds the received signature Sig' by computing

$$Sig = Sig'.k^{-1} = (H'_{Evaluation})^d.k^{-1} = (H_{Evaluation}.k^e)^d.k^{-1}$$

$$= (H_{Evaluation})^d.k.k^{-1} = H_{Evaluation}{}^d \ mod\ n.$$

E sends Sig along with the original F and $EvalNO$ to the ES.

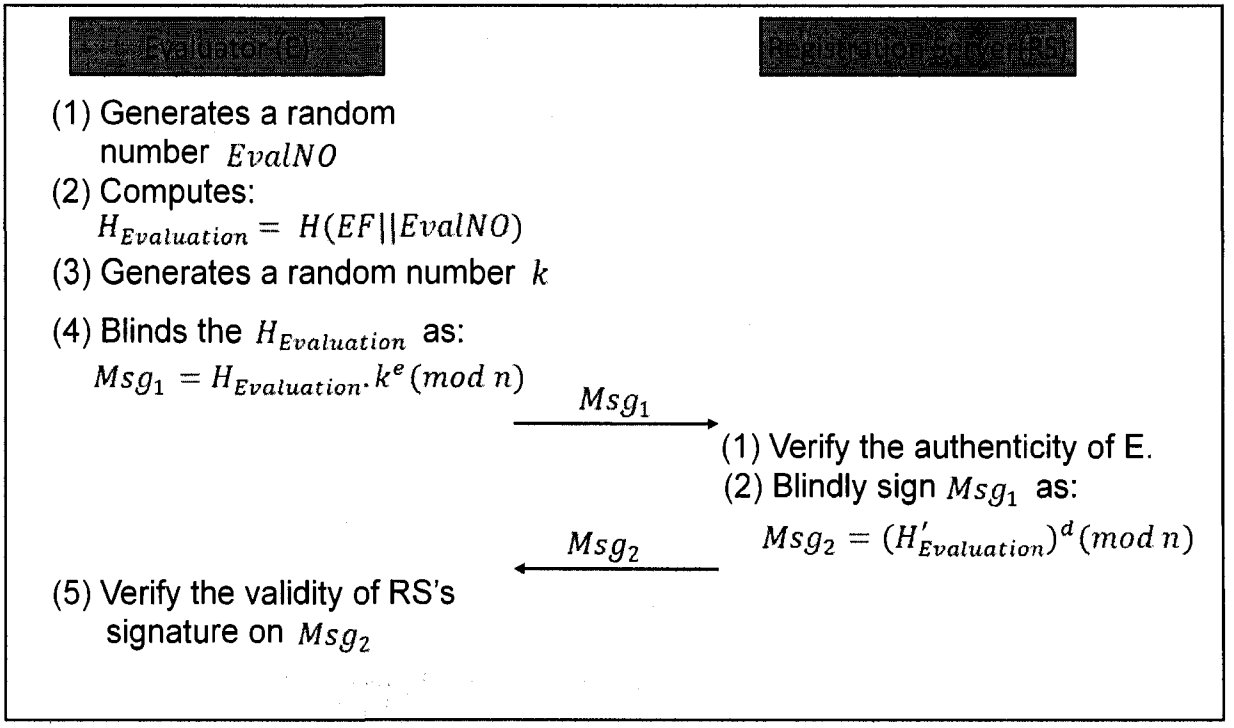

Figure 3.1 Messages Exchange between E and RS.

It is important to note that, the last step in the process above should be performed after arbitrarily random interval (with a pre-specified time limit). This is necessary in order to prevent the ES from revealing the identity of E by simply observing that E was the last entity communicating with RS before it receives Sig.

Verification

As shown in Figure 3.2, when ES receives Sig, F and $EvalNO$ from E, she performs the following verification process:

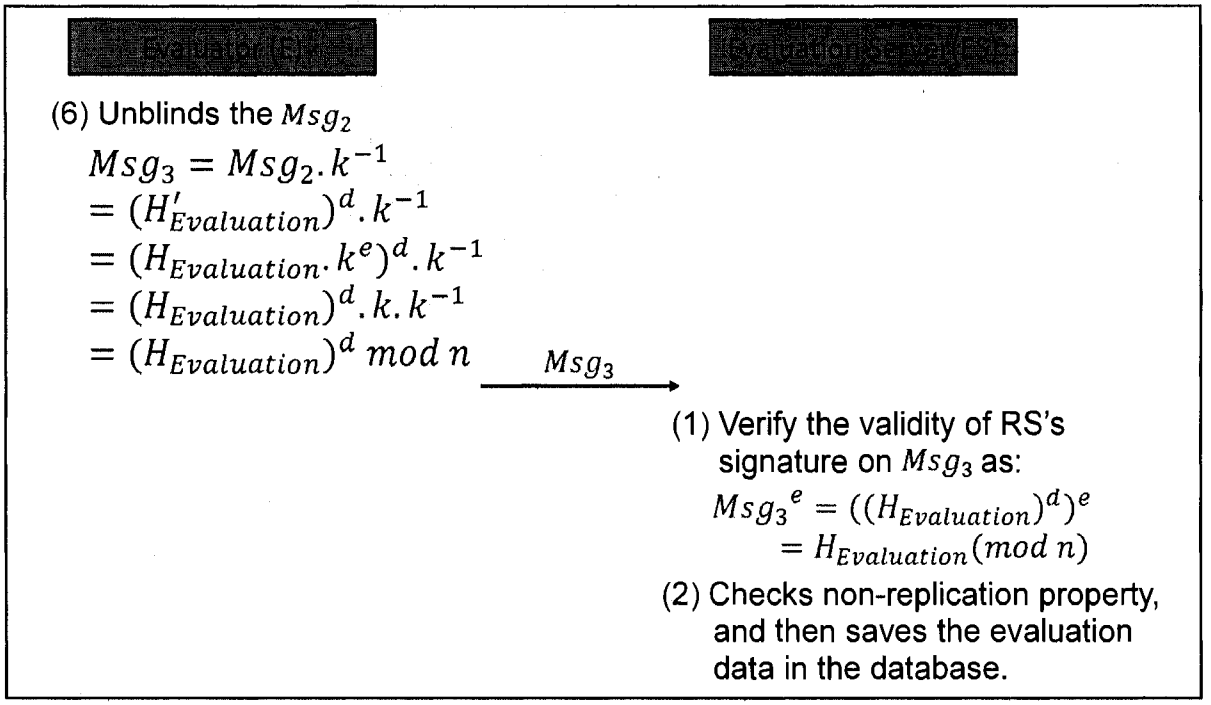

Figure 3.2 Messages Exchange between *E* and *ES*.

(i) *ES* Verifies that

$$Sig^e = (H_{Evaluation}{}^d)^e = H_{Evaluation}\ mod\ n$$

where (n, e) is the public key of *RS*. If this verification fails, the submitted evaluation will be rejected and discarded.

(ii) Check if the evaluation database does not have any previous evaluation records with the same $EvalNO$. If any previously received evaluations has the same $EvalNO$, then the current evaluation will be rejected, otherwise it is saved in the evaluation database with its corresponding $EvalNO$. This step is necessary to prevent any evaluator from submitting the same feedback multiple times. The long length of $EvalNO$ ensures that the probability of having two evaluators generating the same $EvalNO$ is practically zero.

3.3.2 Traceable Online Anonymous Feedback System

While this system ensures the anonymity of honest evaluators, malicious evaluators who misuse their anonymity and attempt to submit multiple feedbacks against the same evaluatee will be identified and their anonymity will be revoked.

In this section we describe the main procedures of our scheme which is based on k-times anonymous authentication scheme [3] (see section 2.3.6).

The Procedures

Registration

Let $\mathbb{N}$ and $\mathbb{Z}_n$ denote the ring of natural numbers from 0 to n – 1, $QR(n)$ be the multiplication group of quadratic residues of $\mathbb{Z}_n$, λ is a security parameter, and Λ, Γ are sets of integers that were in $(0, 2^{\lambda})$ and $(2^{\gamma}, 2^{\gamma} + 2^{\lambda})$ respectively.

As shown in Figure 3.3, E can join the evaluation process as follows.

(i) E selects $x' \in_U \Lambda$, and sends a commitment $C = g^{x'} + h^{s'} \bmod n$ to RS with a validity proof to show that C is correctly computed from x'. Here $(g, h) \in_U QR(n)^2$ and s' is a random natural number.

(ii) RS selects $x'' \in_U \Lambda$ and sends it to E.

(iii) E computes $x = (x' + x'') \bmod 2^{\lambda}$.

(iv) E computes $(\alpha, \beta) = (a^x \bmod n, b^x)$, where a, b are part of the group public key. Next, E adds a new entry (i, β) to the identification list and sends (α, β) to RS with a validity proof to show that C is correctly computed from x'.

(v) RS verifies that the proof is valid, and (i, β) is an element of the identification list. If the verification succeeds, RS generates a prime $e \in_U \Gamma$, computes $A = (\alpha\, a_0)^{1/e} \bmod n$, where a_0 is a part from the group public key, and sends (A, e) to E.

(vi) E confirms that equation $a^x\, a_0 = A^e \bmod n$ is satisfied, e is a prime, and e is an element of Γ. The new evaluator E's secret key is x, and her public key is (α, A, e, β).

Evaluation

As shown in Figure 3.4, E can perform the evaluation process as follows.

(i) ES publishes his identity ID_{ES} and K_{ES} . Here, K_{ES} the maximum number of evaluations that E can perform.

$$\text{Let } (T_j, \check{T}_j) = \mathcal{H}_{\mathbb{G}_T \times \mathbb{G}_T}(ID_{Evaluatee_j}, ID_{ES}, K_{ES})$$

for $j = 1, \ldots, K_{Evaluatee}$, where $ID_{Evaluatee_i}$ is the identity of the *Evaluatee* that the evaluation is accusing, and $K_{Evaluatee}$ is the number of the evaluatees. Here, $(T_j, \check{T}_j)$ called the j^{th} tag base of ES.

(ii) ES sends random integer $l \in_U [0, 2^{\mu+\varepsilon}] \cap \mathbb{N}$ to E, where μ, ε are security parameters.

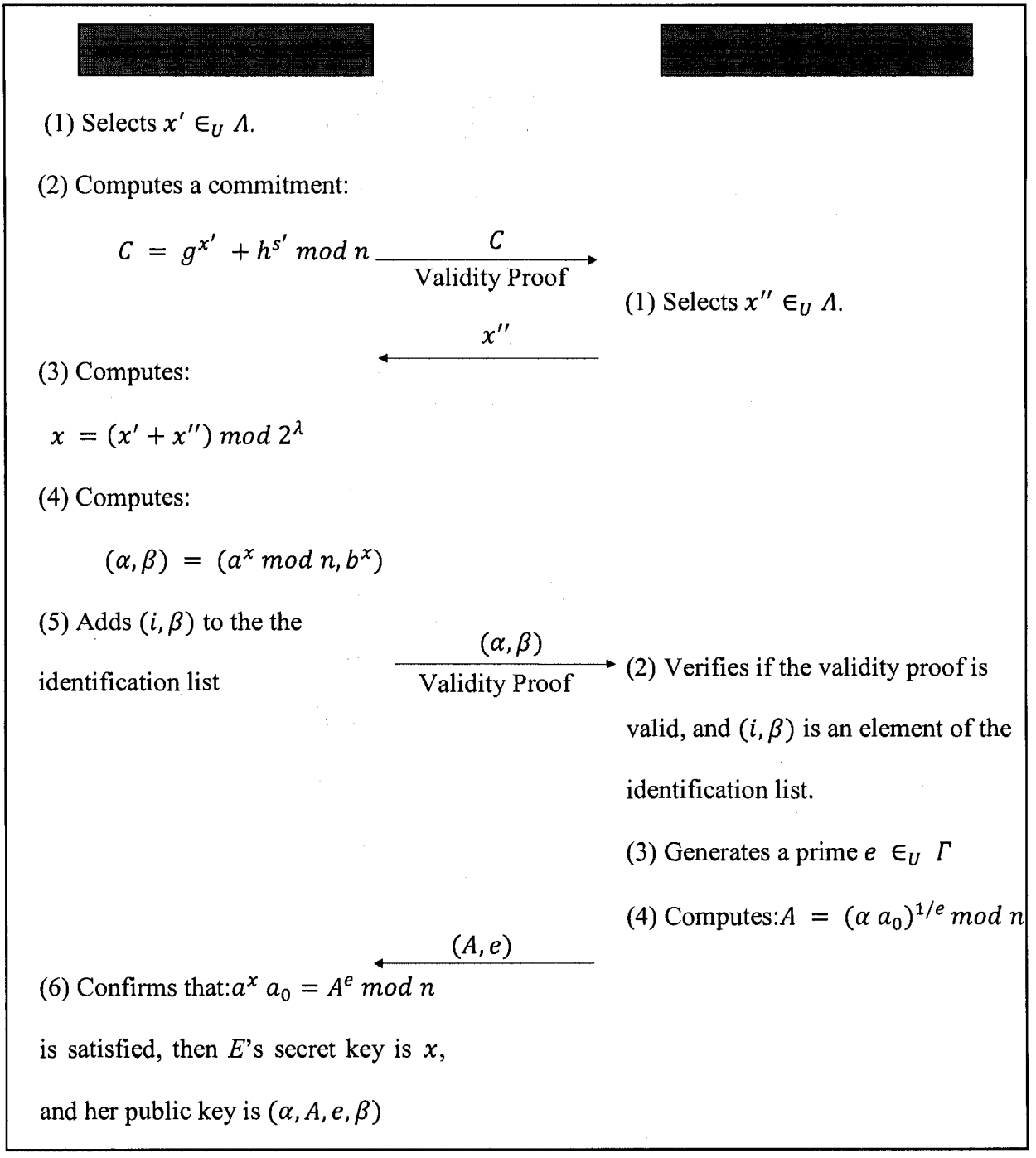

Figure 3.3: Messages Exchange between *E* and *RS*.

(iii) E calculates the tag $(\Gamma, \check{\Gamma}) = (T_j{}^x, (b^l \check{T}_j)^x)$ using the j^{th} tag base $(T_j, \check{T}_j)$.

(iv) E Fills the *Feedback* (F).

(v) Finally, E sends $(\Gamma, \check{\Gamma}, F)$ to ES with validity proof.

(vi) ES verifies If the proof is valid and if Γ is different from all search tags in the database. If the verification succeeds ES adds $(\Gamma, \check{\Gamma}, l, F)$ and the proof to the database.

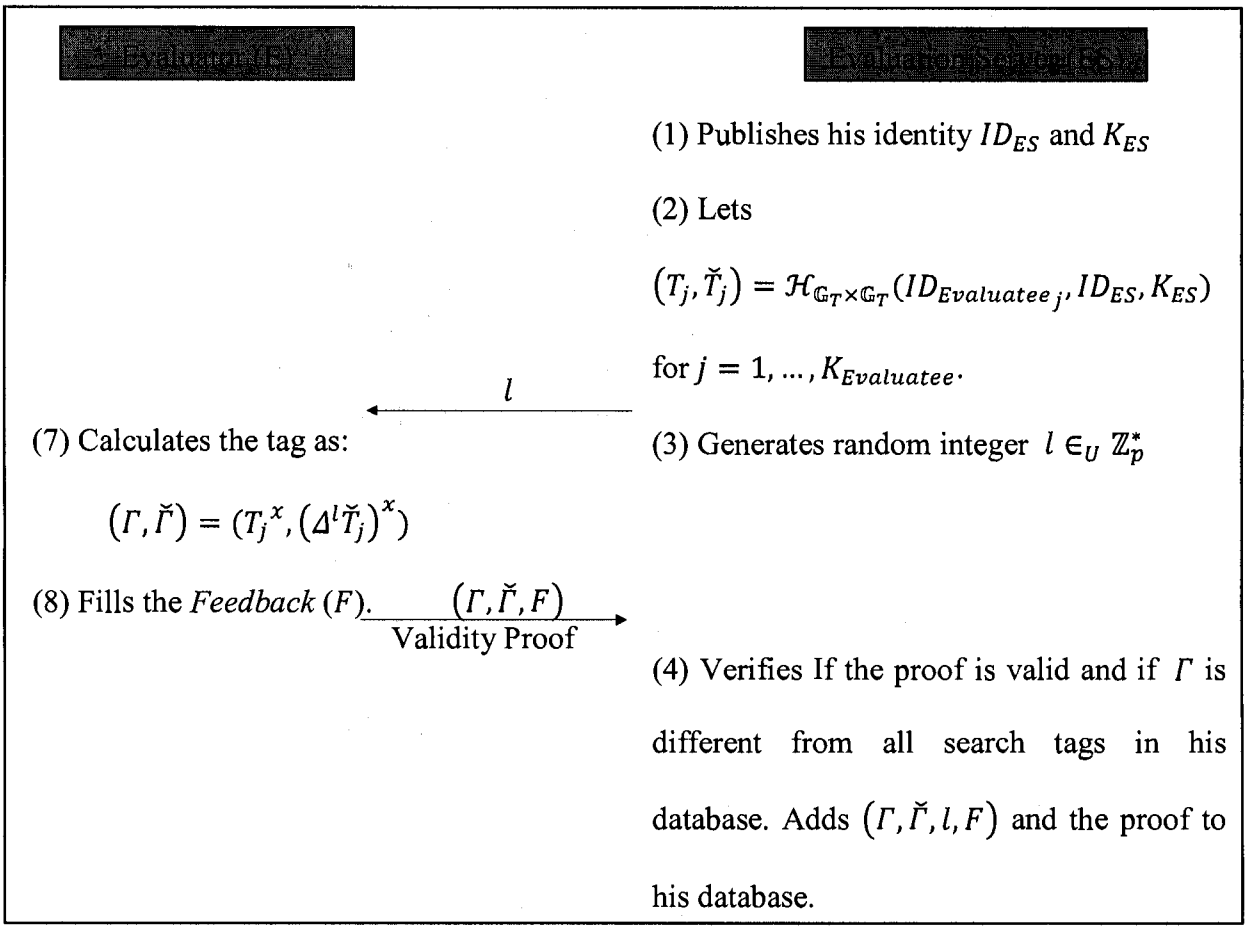

Figure 3.4 Messages Exchange between E and ES.

Tracing

ES can easily identify and trace any malicious evaluator who tries to submit multiple evaluations against the same evaluatee for the same reason as follows.

(i) *ES* finds two entries $(\Gamma, \check{\Gamma}, l, Proof)$ and $(\Gamma', \check{\Gamma}', l', Proof')$ that satisfy $\Gamma = \Gamma'$ and $l \neq l'$, and that both $Proof$ and $Proof'$ are valid, if no such entry can be found, the procedure will not be able to disclose the evaluator identity.

(ii) If such entry is found, ES computes β as

$$\beta = \left(\frac{\check{\Gamma}}{\check{\Gamma}'}\right)^{\frac{1}{l-l'}} = \left[\frac{(T^x, (b^l \check{T})^x)}{(T^x, (b^{l'} \check{T}')^x)}\right]^{\frac{1}{l-l'}} = b^x$$

and searches for a pair (i, β) in the database which discloses the evaluator identity.

Chapter 4

Implementation and Security Analysis

4.1 Introduction

In this chapter, we describe the implementation details of our proposed online anonymous feedback system. We also provide some informal analysis of its security properties.

Figure 4.1 shows the procedure followed by different system participants. The implemented user interface allows the administrator to change the evaluation form to fit the assessment of any type of function or performance and makes it available to the users via a web browser.

4.2 The Architecture Design

In what follows, we describe the general architecture of our system. As described in section 3.3. Our feedback system is composed of the following main components.

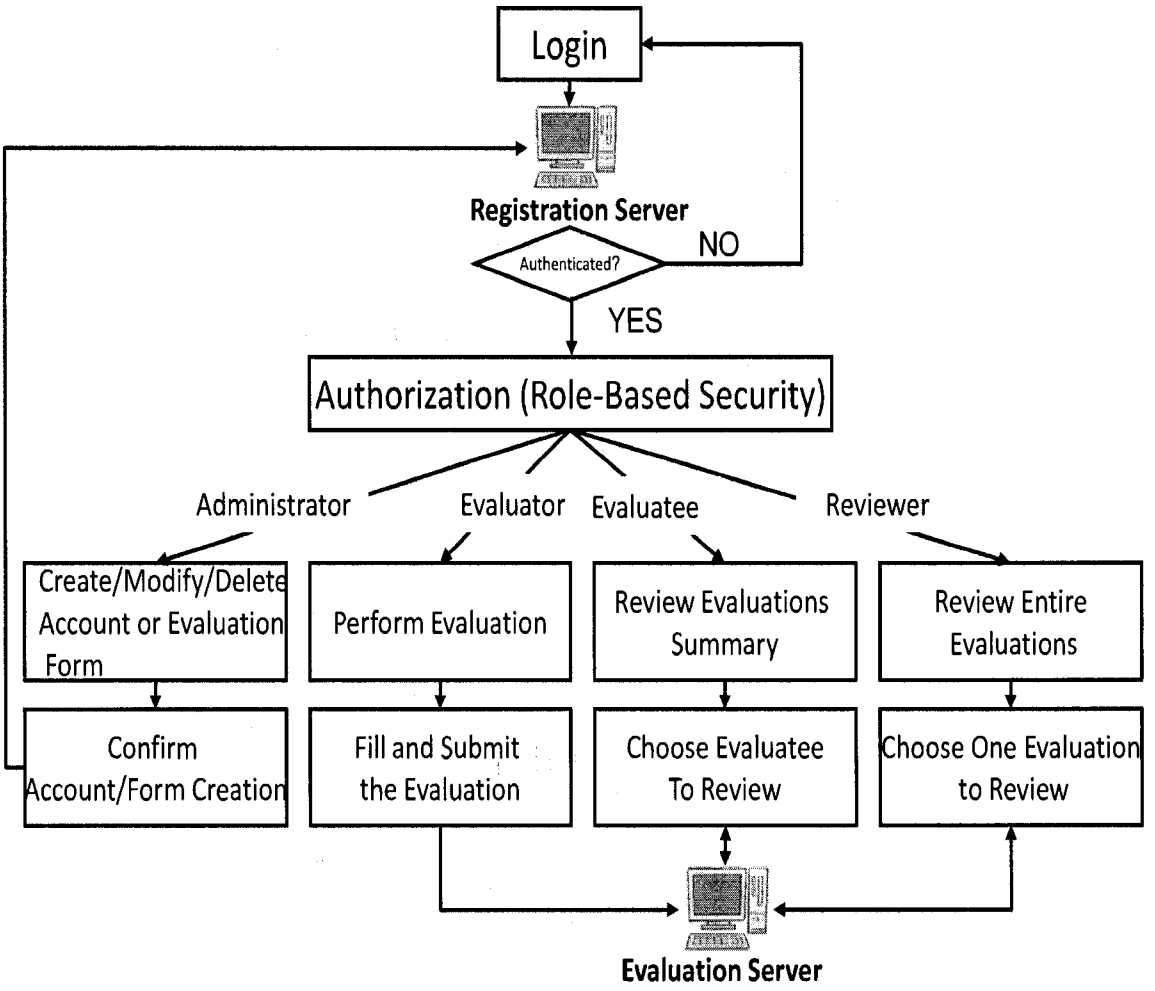

Figure 4.1 The procedure followed by different participants

Registration Server

Before being able to perform any actions, users must first be registered with the registration server. Registered users can then be authenticated by providing valid login information. If a user is authenticated by the registration server, she can perform the authorized actions based on her role. Otherwise she will be redirected to the login page.

Administrator

The administrator is responsible for the creation, modification or deletion of the evaluation forms and the users' accounts. The administrator is also responsible for determining the roles of the system participants.

Evaluator

An evaluator is a registered user that is allowed to perform the evaluation by filling and submitting the evaluation form to the evaluation server. An evaluator may also save a draft of her form before her final submission.

Evaluatee

An evaluatee is the person whose qualifications and performance are being evaluated. An evaluatee is allowed to review the evaluation summary for any one of her evaluations.

Reviewer

A reviewer is allowed to view the entire evaluation form. In other words, a reviewer view is not limited to the summary of the evaluation statistics. For the electronic course evaluation application, this role might be assigned to personals from teaching and learning centers who may need to further investigate the evaluation statistics.

Evaluation Server

The evaluation server is responsible for saving anonymous evaluation forms, and producing the evaluation summery. It should be noted that, although the evaluation server and registration server are shown in Figure 4.1 as two separate entities, they can be implemented on the same server machine.

4.3 The development Environment

Our system has been implemented using Microsoft's new Visual Studio .NET development environment. The server side is an ASP.NET 2.0 dynamic data drive web application written in C#.NET, which is embedded in the HTML pages. The code that executes on the server (to process handlers for events such as button click) is also written in C#.NET. The client-side code has been implemented with HTML and JavaScript. The system utilizes the new membership capabilities of ASP.NET 2.0. In particular, it uses role-based security to control access to the website. We also use Microsoft SQL Server

2005 for database operations, including the store and retrieval of login information, as well as feedback display and submission data.

In what follows, we briefly summarize some features of these development tools.

Microsoft's Visual Studio .NET is a complete suite of tools for building both desktop and team-based enterprise web applications. In addition to building high-performing desktop applications, it also has a powerful component-based development tools and other technologies to simplify team-based design, development, and deployment of enterprise solutions. Microsoft .NET includes the following components:

- .NET Framework: It provides the basic system services that support ASP.NET, as well as Windows Forms development. A part of the purpose of the .NET framework is to allow for "mobile code" to be distributed to users on multiple platforms, security has become a major concern. Mobile code is an application or piece of software that is transmitted from a server to a local system (or other device) to be executed locally.
- .NET Enterprise Servers: It provides much of the functionality needed by most large businesses.
- .NET language and Language tools: It allows developers to work with any .NET-complaint language and take full advantage of advanced language feature.

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful web applications. The first version of

ASP.NET offered several important advantages over previous web development models. ASP.NET 2.0 improves upon that foundation by adding support for several new and exciting features in the areas of developer productivity, administration and management, extensibility, and performance.

Microsoft SQL Server 2005 is Microsoft's enterprise-class database management system. It can deal with small databases such as a personal list of contacts. It can also handle a very large database. SQL Server 2005 offers some very useful features such as input and output of data as XML and integrated OLAP engine.

4.4 Implementation

In this section we describe the implementation of the major components of our online anonymous feedback system.

4.4.1 Authentication

The .NET security model operates in two basic modes: windows authentication and forms authentication. Windows authentication uses users' login information, as provided when they log into the Windows operating system installed on their local machine, to authenticate a user. For web-based system, such as our developed system, the forms authentication method seems to be more practical. Forms authentication allows a user to log in with a username and password they provide on an HTML form. If valid

login information for all users of a system is stored in a SQL Server database, as it is for our work, forms authentication can look up a username and password in the database to authenticate a user attempting to log into the system.

4.4.2 Authorization (Role Based Security)

Authorization can be thought of as folder-based protection. When a user is authenticated with forms authentication, their roles are assigned according to their identity in the form of an Internet cookie that is used for authorization. A web configuration file is used to specify which folders users of specific roles may access. If a user is not authenticated (e.g., user has not provided a valid login) or attempts to access a folder they are not authorized to access, they are redirected to a web page specified in the web configuration file.

When users log into our system, they specify which account they are logging into and provide a username and password. Their username and password are looked up in the specified account database's table to authenticate the user. Furthermore, an authenticated user's ID is looked up in each of the account database's role tables (Administrators, Evaluators, Evaluatees, and Reviewers) to build a list of the user's roles. The user's ID is also looked up in the system database's table to determine if the user is the account's administrator. All of the user's roles, since users may be members of multiple roles, are attached to their identity in the form of a cookie used to authorize them on our feedback system web pages. After successfully logging into the system, users are

taken to an action page that dynamically presents options to them based on their roles. Users who are not authenticated or authorized are redirected to the home page, which is also the login page of our system.

4.4.3 Evaluation

In order to explain the implemented mechanism which is used to perform the online anonymous evaluation, we will describe the main classes and methods (refer to Figure 4.4).

Getting Certificates

We can package public keys as couples of identity and purpose information called a certificate, which give some features such as expiration times, key revocation, and key owner information.

Generally, there are three ways that can be used to request or generate a certificate and the corresponding private key:

- Buy a certificate from companies such as VeriSign and Thawte.
- Use an internal Certificate Authority (CA) if you already have it on your network. This can be done through the Certificates Microsoft Management Console (MMC) snap-in or through the web interface of the CA. Certificate Services are optional components of Microsoft Windows Server 2003.

- For experimental and testing purposes, one can use a self-signed certificate as we did in our implementation. The .NET Framework contains a tool called Makecert.exe. By using this tool, one can generate certificates, which can be used only for test purposes.

The Certificate and Key Store

Windows operating system abstracts the physical storage location of private keys and certificates by using the notion of stores. Our keys could be stored on the hard disk or a smart card or some other storage devices. From the application perspective, we always interact with the store and don't care about the implementation details. In our implementation, we managed the certificate store by using the MMC snap-in (see Figure 4.2).

Windows operating system has two store types: the user store and the machine store. The machine store is for certificates that should be available machine-wide (or for system accounts), whereas the user store contains only user-specific certificates. For services and daemons such as our ASP.NET application, we need to use the machine store. Each store is divided into several folders. The most important ones are Personal, Other People, and Trusted Root Certification Authorities. The folders have the following purposes:

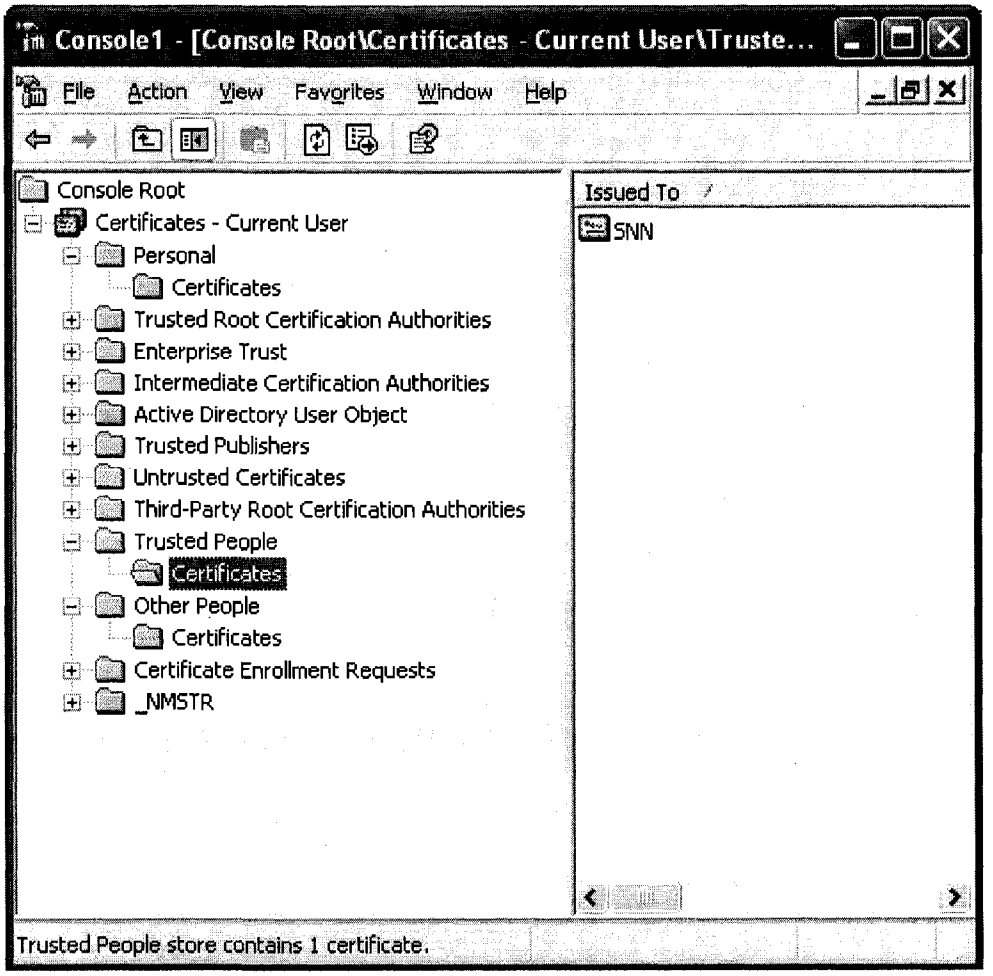

Figure 4.2 MMC snap-in

- ***Personal*** Certificates in the personal store have an associated private key. This key is used for decrypting or digitally signing data.
- ***Other People*** This folder contains the certificates of members to whom we want to send encrypted data or for whom we want to verify their digital signatures.
- ***Trusted Root Certification Authorities*** This list holds all certificates of CAs from which we want to accept certificates. If we buy a certificate from a well-known company such as VeriSign, no special action is necessary because its certificate is

already included by default. In our work we generated certificates by using a Makecert tool; so, we imported the certificate in this folder manually.

Accessing the Certificate Store

The *X509Store* class enables us to do all store-related operations. We have to specify which store (user or machine) and container (Personal, Other People, or Trusted Root Certification Authorities) we want to open. We can also search the certificate store by using one of the several properties of a certificate (for example, name, issuer, or expiration dates) (see Figure 4.3).

Asymmetric Encryption Using Certificates in .NET

.NET 2.0 includes a full-featured Application Programming Interface (API) for asymmetric cryptography and certificates. The functionality is split into the following two namespaces:

- ***System.Security.Cryptography.X509Certificates*** Contains classes to access the certificate store (adding/deleting containers, adding/deleting certificates, and searching) and to show the standard Windows certificate dialog boxes.
- ***System.Security.Cryptography.Pkcs*** Contains classes that implement the Cryptographic Message Syntax (CMS) and the Public Key Cryptographic Standards (PKCS) algorithms for encryption and digital signatures.

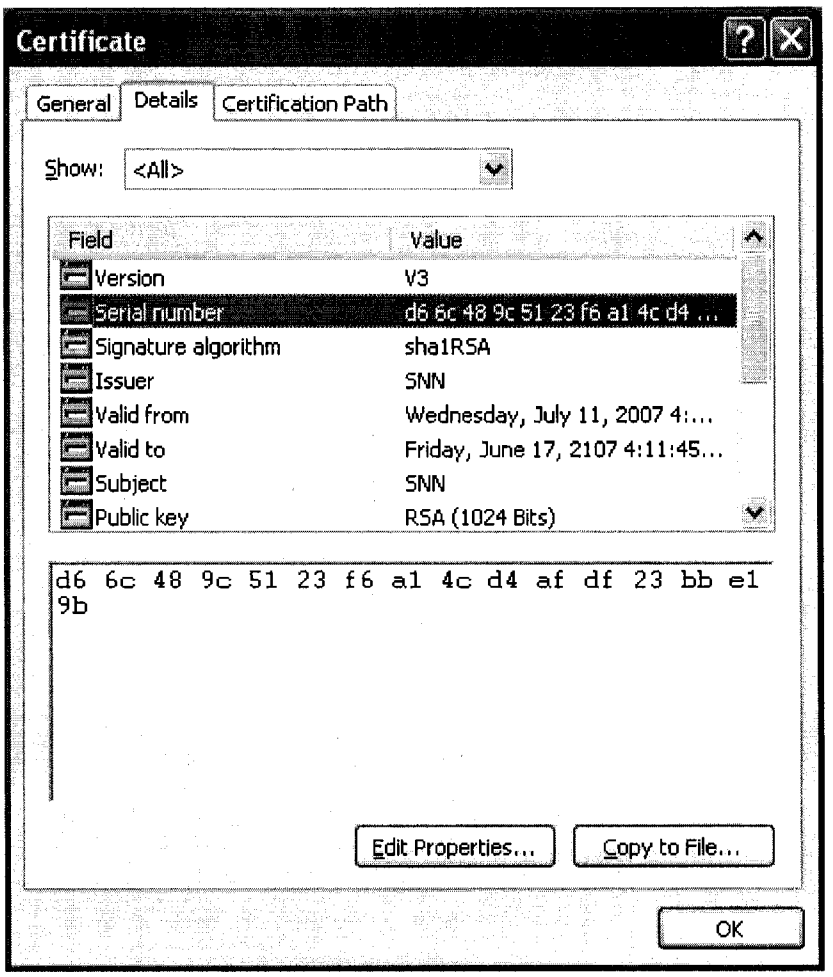

Figure 4.3 Properties of a Certificate

Creating the Evaluation Form

By using ASP.NET programming framework, we created an evaluation form, *PreformEvaluation.aspx*, to gather evaluator's feedbacks. In this form we developed a C# .NET class *PreformEvaluation.cs* to collect some parameters including the evaluatee's name, the questions' IDs and the responses. Another C# .NET class, *RemotePost.cs,* is

responsible for posting the evaluation form's data and all other relative parameters from form to another.

Submit the Evaluation

By clicking the submit button on the evaluation form, after filling the evaluation questions, the method *SubmitBut_Click()* in the class *PreformEvaluation.cs* will be invoked along with a JavaScript files *RSA.js, Hash_sha.js* and *Keygenerator.js* to generate a random numbers and hash the evaluation form data on the client side. Then a C# .NET class *CryptoFunctions.cs* is also invoked to encrypt, decrypt and sign the evaluation data by using some methods developed in it such as *Encrypt(), Decrypt(), SignAndEncrypt(), DecryptAndVerify() and GetCertificate()*. Finally, a C# .NET class *FeedBackSystemDB.cs* will be invoke. This class is responsible for storing the evaluation data on the evaluation server's database.

Evaluation summary Review

ReviewEvaluationSummary.aspx is an ASP.NET form developed to be used by the *Evaluator* and the *Evaluatee* to view the evaluation summary after submitting the evaluation form. In this form, we developed a C# .NET class *ReviewEvaluationSummary.cs* to retrieve some parameters from the evaluation server's database and show it back to the evaluator as a summary. The main method in this class is a *SubmitBut_Click()* which is invoked by clicking the submit button on the evaluation summary form.

ReviewEntitrEvaluations.aspx is an ASP.NET form developed to be used by the *Reviewer* to view the entire evaluations after all evaluators have submitted their evaluation forms. In this form, we developed a C# .NET class *ReviewEntitrEvaluations.cs* to retrieve all parameters from the database and show it back to the Reviewers. The main method in this class is a *SubmitBut_Click()* which is invoked by clicking the submit button on the review entire evaluation forms.

Modifying the Evaluation Form

ModifyEvaluationForm.aspx is an ASP.NET form developed to be used by the *Administrator* to modify the evaluation form to fit the assessment of essentially any type of function or performance. In this form we developed a C# .NET class *ModifyEvaluationForm.cs* to retrieve parameters such as QuestionID and QuestionType. The main method in this class is a *SubmitBut_Click()* which is invoked by clicking the submit button on the modification form after choosing the evaluation style.

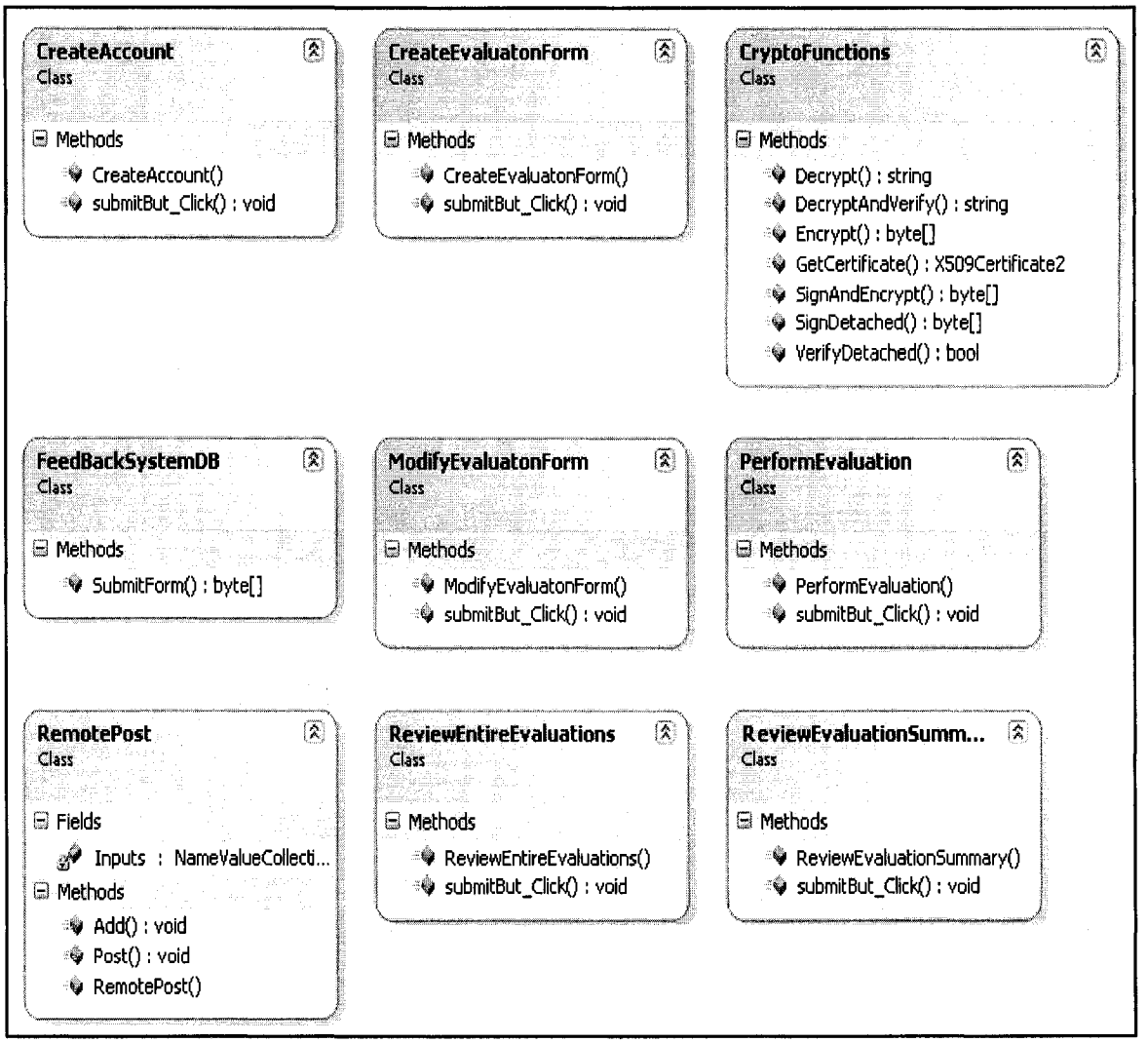

Figure 4.4 Class Diagram of the Online Anonymous Feedback System

4.5 Interaction Scenario

The following interaction scenario, illustrated with some screenshots, give a clear view about how our system works and explain the major functions of the developed system.

Before making any action, a user must be registered in the *Registration Server* by the *Administrator* to become a system member whom we call an *Evaluator*. Now, this *Evaluator* will be allowed to log in the system with her username and password. If the *Evaluator* is authenticated by the system, .NET examines the web.config file in the application's directory for authorization policies. .NET examines each preceding directory up to the virtual root of the Web directory, and appends any additional authorization rules to the list. Finally, rules residing in the machine.config file are appended to the list. .NET determines access to the specified resource by iterating through the rules in the order in which they were read and allows or denies the request based on the first rule that matches the authenticated member. If the member is not authenticated, that means she does not provide a valid login or she attempts to gain access to folders or actions that are not authorized to access. In this case, she will be redirected to the login page. A sample login form is shown below in Figure 4.5.

When a member successfully log into the system, she is taken to a page that dynamically presents the possible actions based on her role. That means she may have one of these different roles: *Administrator, Evaluator, Evaluatee or Reviewer* (refer to section 4.2 for more details).

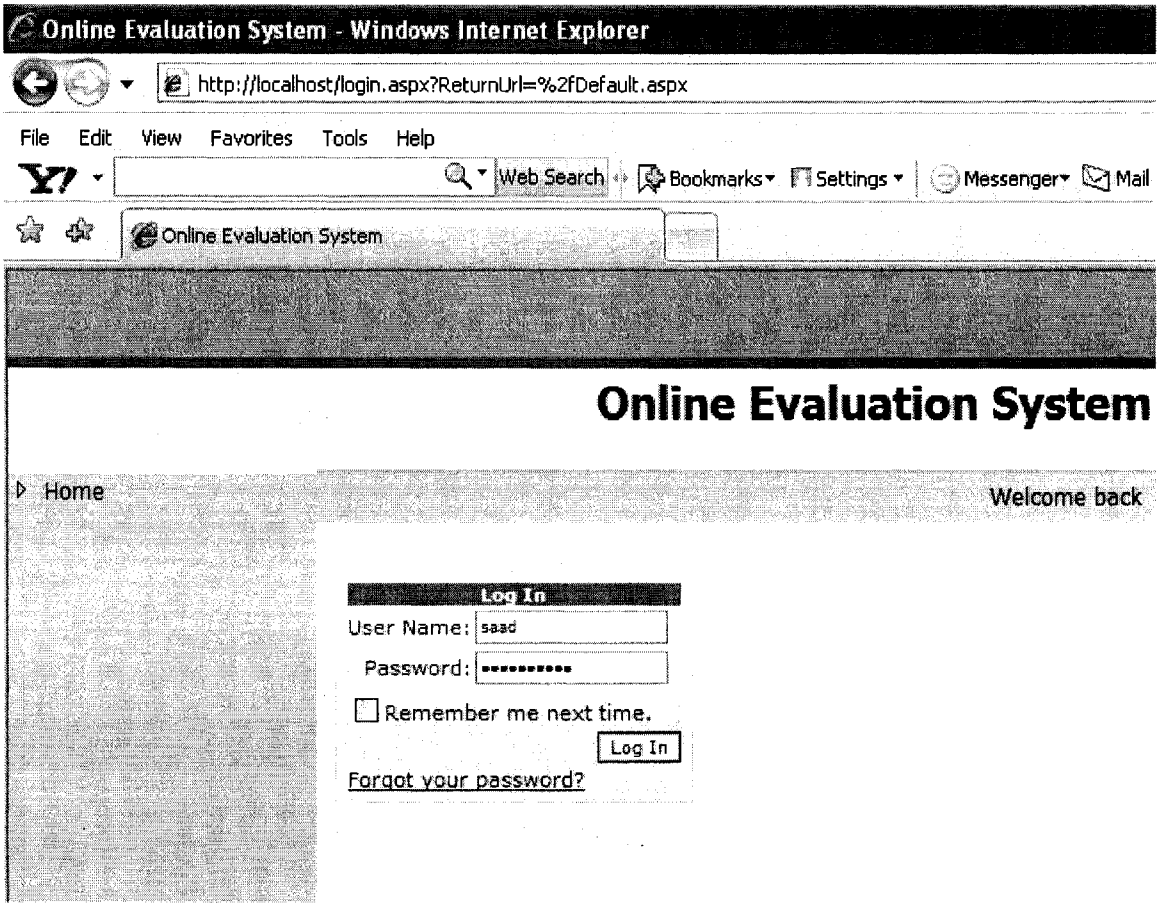

Figure 4.5 The Login Page

Suppose that a member successfully log in as an *Evaluator*. As shown in Figure 4.6, the *Evaluator* will, dynamically, be presented with the list of courses that are available for evaluation and she will be able to select one of the courses and proceed to the next step. The *Administrators* are responsible for specifying which members may evaluate certain courses.

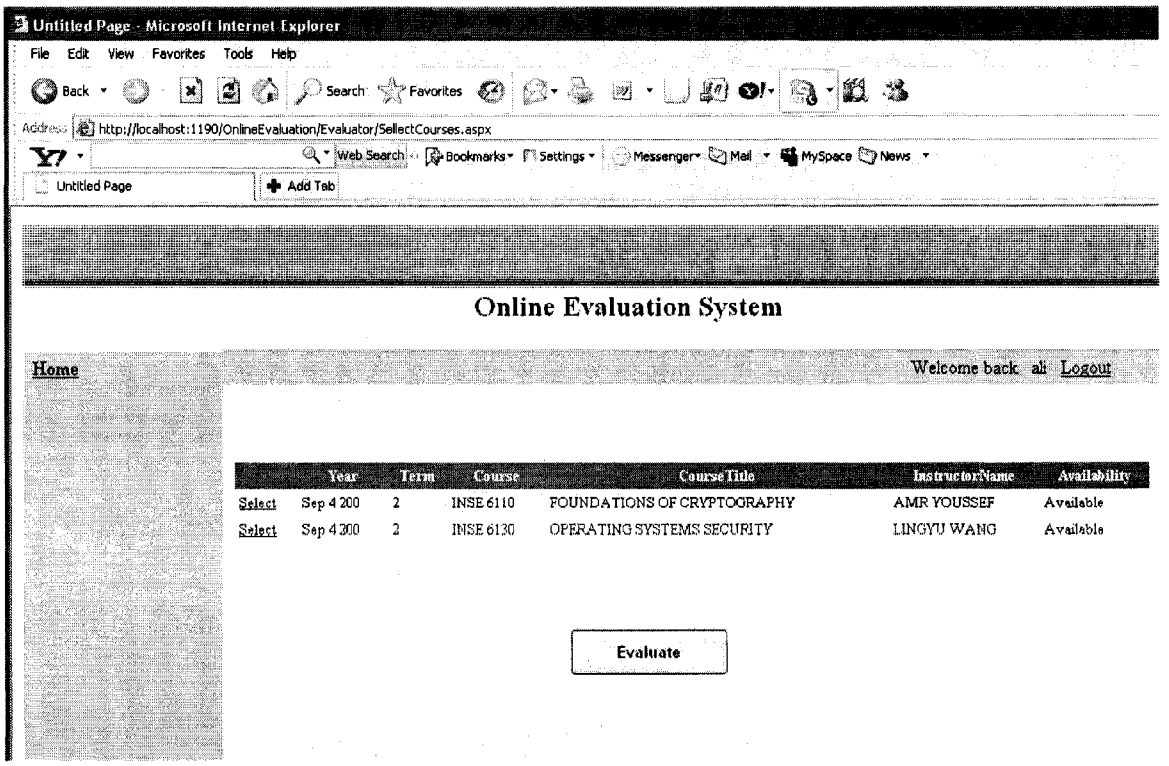

Figure 4.6 Sample of the Course Selection Page

After the *Evaluator* selects one of her available courses, as shown in Figure 4.6, she is be taken to a page that dynamically presents the evaluation form as shown in Figure 4.7. Then the *Evaluator* has to fill up the *Evaluation Form* by answering all the mandatory questions, and submitting it to the *Evaluation Server*. Note that the *Evaluator* will not be able to reevaluate the same course in the future since the *Evaluation Server* can easily identify and reject these multiple submissions.

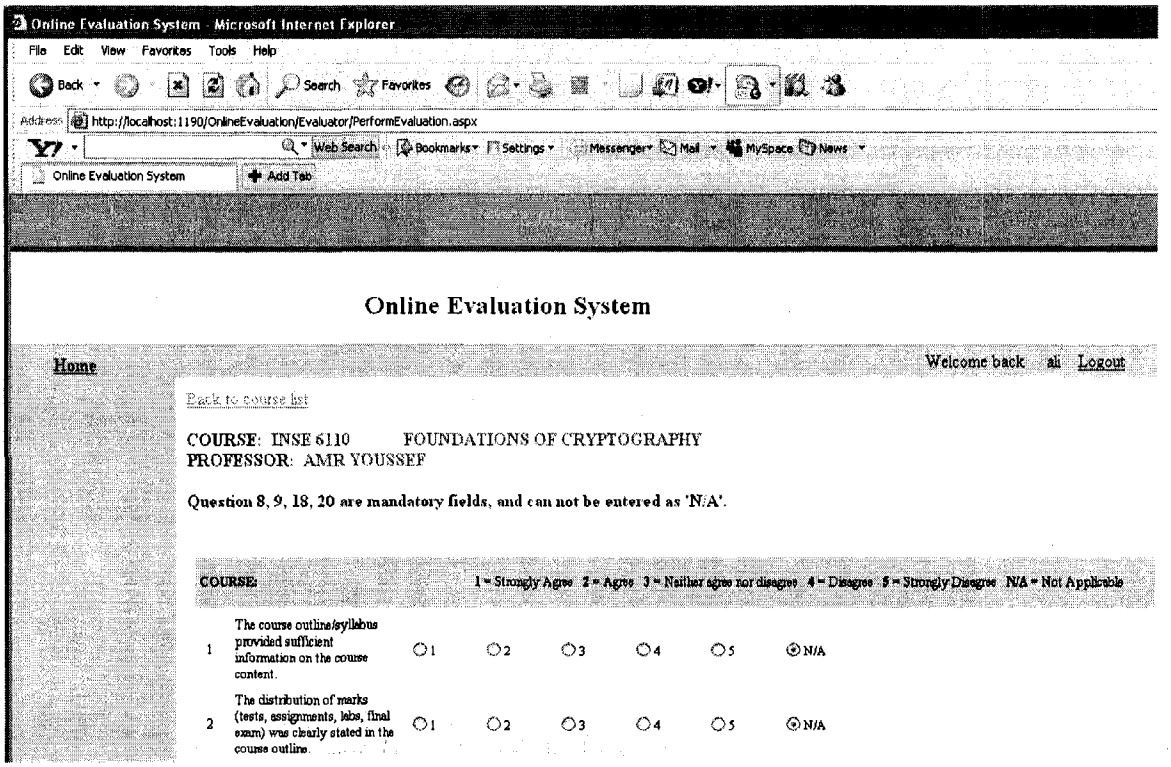

Figure 4.7 Sample of the Evaluation Page

4.6 Security Analysis

In this section, we discuss the security of our system and show that our system satisfies its design goals.

Anonymity

In the registration server side, the submitted hash value of the evaluation form is blinded in this form:

$$H'_{Evaluation} = H_{Evaluation} \cdot k^e \bmod n$$

where (n, e) is the public key of *RS*. Then E sends $H'_{Evaluation}$ to *RS*.

Hence the registration server cannot gain any information that it can use later to link the contents of the evaluation form to the evaluator. Moreover, it cannot link the evaluator to its unblinded evaluation form. Also, only registered users are allowed to obtain the registration server blinded signature on their evaluations.

Verifiability

The evaluation server can verify the submitted evaluations from the evaluator by checking the validity of registration server's signature in the form of

$$Sig^e = (H_{Evaluation}{}^d)^e = H_{Evaluation} \bmod n$$

where (n, e) is the public key of the registration server.

Unforgeability

In our system, only registered evaluators can submit valid feedbacks, as each feedback is signed by the registration server's private key. No one else can forge a valid feedback without registration unless she can break the public key cryptosystem to get the private keys of the registration server.

Non-Replication

In the proposed system, if any evaluator sends multiple evaluations for the same evaluatee, the submitted form will be rejected because each evaluator has a unique evaluator number which is totally random number generated on the client side. Then when we compare the result of the formula $H_{Evaluation} = H(F||EvalNO)$ with the saved hashed forms in evaluation server's database, we will be able to find out whether this evaluation has been submitted before or not. By maintaining a simple record of users that have previously requested the registration server signature, no user will be allowed to perform this step more than once. The evaluation server accepts only evaluations that are signed by the registration server and the unregistered users cannot submit any feedback. Also the last verification step performed by the evaluation server ensures the registered users cannot submit their evaluation multiple times. For example, in a Sybil attack [25], malicious users can assume multiple identities, and thus can control a substantial part of the system. If such an attack is possible, it would undermine the basis of trust schemes for feedback systems, i.e., most *Evaluators* are honest and each evaluator can give her feedback only once when assessing a given *Evaluatee*. According to [25], without a

logically centralized authority, Sybil attacks are always possible except under certain assumptions regarding the resources possessed by the attacker. A straightforward solution for our system is to let the registration server assume the responsibility of the centralized authority. More specifically, during the registration procedure, the registration server is responsible for binding the user registering as a member to a real-world identity, and for limiting the number of the registrations *Evaluator* identity to one.

Accountability

It should be noted that, the registration server signs a blinded copy of the evaluation form and hence even if the authority running this server was biased, it cannot filter out any feedbacks based on its contents. The registration server signature on the evaluation, which in this form:

$$Sig' = (H'_{Evaluation})^d \bmod n$$

where d is the private key of the registration server, can be considered as evidence that can be used to challenge the registration server if the evaluation server refuses to accept/post the feedback of the evaluator.

Traceability

In our system that is based on the K-Times Anonymous Authentication (K-TAA) scheme, if an adversary sends multiple evaluations, she will be identified by the public tracing procedure in this form.

$$\beta' = \left(\frac{\check{t}}{\check{t}'}\right)^{\frac{1}{l-l'}} = \left[\frac{(t^x, (b^l \check{t})^x)}{(t^x, (b^{l'} \check{t}')^x)}\right]^{\frac{1}{l-l'}} = b^x$$

Here β' is representing the identity of the malicious evaluator. Moreover, to successfully disclose an adversary misusing evaluator's anonymity, the evaluation server needs to find a valid record of this adversary, i.e., the identification list item containing the identity and public key of that adversary.

Chapter 5

Conclusions and Future Work

In this chapter, we provide a summary of the main features of our developed system. We also point out some future enhancements that can be applied to our system.

Throughout this work, we designed and implemented an online anonymous feedback system by employing an anonymous authentication mechanism. Our system is characterized with the following features:

- The evaluator's anonymity is ensured even against an abusive authority that has access to both the evaluation and registration servers.
- Only registered evaluators can submit valid anonymous feedbacks.
- Every registered user is allowed to cast her vote only once.
- If an adversary sends multiple claims against a user for the same reason, she will be identified.
- The authority, responsible for the evaluation process, can be held accountable if it blocks any user's feedback.
- The developed system is generic enough, user friendly and allows the administrator to change the evaluation form to fit the assessment of essentially any type of function or performance.

5.1 Future Work

Several extensions are envisaged for this work. Among them are the following:

- IP traceback is the ability to trace IP packets to their origins; for that reason, preventing IP address traceback is a significant step towards anonymous communication systems in which no one is able to discover identity-related information. The system developed in this thesis assumes the existence of a Mixnet-based anonymous communication system as one solution to this problem. Solutions based on the current available network infrastructure require further investigation.
- By correlating information found in the registration and the evaluation servers logs files, an abusive authority might be able to identify the identity of some evaluators since the time of signing a feedback by the registration server and the evaluation form submission time are likely to be close to each other. This problem may partially be solved by adding a random delay between receiving the signed form from the registration server and submitting it to the evaluation server. However, different practical implications need to be considered.
- We only provided a heuristic argument about the security of the proposed system. A more formal analysis of the proposed protocol would give us a better assurance of its security properties.

- Investigating the possible correlation between assuring the users about their anonymity and possible changes in their feedback is an interesting multidisciplinary research problem.

List of References

[1] Y. Hanatani, Y. Komano, K. Ohta, and N. Kunihiro. "Provably Secure Electronic Cash Based on Blind Multisignature Schemes," Financial Cryptography and Data Security (FC 2006), LNCS 4107, pp. 236-250. Springer-Verlag, 2006.

[2] P.P. Tsang and V.K. Wei, "Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation," Information Security Practice and Experience (ISPEC 2005), LNCS 3439, pp. 48-60. Springer-Verlag, 2005.

[3] I. Teranishi, J. Furukawa, and K. Sako, "K-times anonymous authentication," LNCS 3329, pp. 308–322, Springer-Verlag ,2004.

[4] Zhu, B., Setia, S., and Jajodia, S. 2006. "Providing witness anonymity in peer-to-peer systems". In Proceedings of the 13th ACM Conference on Computer and Communications Security (Alexandria, Virginia, USA, October 30 - November 03, 2006). CCS '06. ACM, New York, NY, 6-16.

[5] D. Chaum, "Security without identification: transaction system to make big brother obsolete," Communication of ACM, vol. 28, no. 10, pp.1030-1044, 1985.

[6] M. Abe, and E. Fujisaki, "How to Date Blind Signatures," LNCS 1163, pp. 244-251, Springer-Verlag, 1996.

[7] I. Damgard and M. Jurik, "A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-key system," In Proceedings. of Public Key Cryptography 2001. LNCS 1992, pp. 119-136, Springer-Verlag, 2001.

[8] C. Pavlovski, C. Boyd, and E. Foo, "Detachable Electronic Coins," In Information and Communication Security, Second International Conference, ICICS'99, LNCS 1726, pp. 54-70, Springer-Verlag, 1999.

[9] J. Furukawa, and K. Sako, "An Efficient Scheme for Proving a Shuffle," In CRYPTO 2001, LNCS 2139, pp. 368-387, Springer-Verlag, 2001.

[10] D. Chaum, "Blind signatures for untraceable payments," Proc. Of CRYPTO'82, pp.199-203, Plenum Press, 1983.

[11] C.A. Neff, "A Verifiable Secret Shuffle and its Application to E-Voting," ACMCCS 01, pp. 116-125 2001.

[12] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital structures and public-key cryptosystem," Communication of ACM, vol. 21, no. 2, 1978.

[13] M. Ookubo, F. Miura, M. Abe A. Fujioka, and T. Okamoto. "An improvement of a practical secret voting scheme," In ISW'99, LNCS 1729, pp. 37-46, Springer-Verlag, 1999.

[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," Proc. 13th Usenix Security Symp., Usenix Assoc., 2004, pp.303–319.

[15] Audun Jøsang, Jochen Haller: Dirichlet Reputation Systems. ARES 2007: 112-119

[16] G. Ateniese and B. de Medeiros. "Efficient group signatures without trapdoors," LNCS 2894, pp. 246–268, Springer-Verlag, 2003.

[17] Hitlin & Rainie, Pew Research Center,2005, Pew Internet & American Life Project: The Internet at School http://www.pewinternet.org/PPF/r/163/report_display.asp

[18] L. Rasmusson and S. Janssen, "Simulated Social Control for Secure Internet Commerce," in Proceedings of the 1996 New Security Paradigms Workshop, C. Meadows, Ed. ACM, 1996.

[19] A. Jøsang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," Decision Support Systems, 2007, dOI: http://dx.doi.org/10.1016/j.dss.2005.05.019.

[20] D. Chaum and E. van Heyst. "Group signatures". In D. W. Davies, editor, Advances in Cryptology-Eurocrypt'91, volume 547 of LNCS, pp. 257-265.

[21] Rivest, R. L., Shamir, A., and Tauman, Y. 2001. "How to Leak a Secret". In Proceedings of the 7th international Conference on the theory and Application of Cryptology and information Security: Advances in Cryptology (December 09 - 13, 2001). C. Boyd, Ed. Springer-Verlag, London, 552-565.

[22] Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable Electronic Coupon Protocol with Anonymity Control, In ISW'99, LNCS 1729, pp. 37-46, Springer-Verlag, 1999.

[23] Dingledine, R. 2000. The Free-Haven Project: Design and Deployment of an Anonymous Secure Data Haven. Unpublished Master's thesis, Massachusetts Institute of Technology.

[24] Shi Cui, Choong Wah Chan, Xiangguo Cheng: Practical Group Signatures from RSA. AINA (1) 2006: 111-115.

[25] J. R. Douceur. The sybil attack. In Proceedings of The First International Workshop on Peer-to-Peer Systems (IPTPS 2002), pp. 251–260, 2002.

[26] Canard, S., Traore, J. "List Signature Schemes and Application to Electronic Voting". In International Workshop on Coding and Cryptography 2003. pp.24-28, March 2003.

[27] Ernie Brickell, Jan Camenisch, and Liqun Chen. "Direct Anonymous Attestation" ZISC Information Security Colloquium SS 2004, June 2004.

[28] M. Srivatsa, L. Xiong, and L. Liu. TrustGuard: "Countering vulnerabilities in reputation management for decentralized overlay networks". In Proceedings of the 14th International Conference on World Wide Web, pp. 422–431, 2005.

[29] L. Nguyen and R. Safavi-Naini. "Dynamic k-times anonymous authentication". In Proceedings of The Third International Conference on Applied Cryptography and Network Security (ACNS 2005), pp. 318–333, 2005.

[30] Jean-Claude Usunier and Anthea Monod." The Social Dynamics of Online reputation systems". Working Paper 0606.
Available: http://www.hec.unil.ch/cms_irm/ WP0606.pdf

[31] Holt,D. Rice, M., Smissen, I. & Bowly, J. (2001) Towards Institution-Wide Online Teaching and Learning Systems: Trends, Drivers and Issues. Paper Presented at ASCILITE 2001.

[32] J. Camenisch and M. Stadler. Efficient and generalized group signatures. In: Advances in Eurocrypt'97, LNCS 1233, pp. 465–479, Springer-Verlag, 1997

[33] Hans Delfs, Helmut Knebl: Introduction to Cryptography: Principles and Applications Springer 2002.

[34] L. Chen and T. P. Pedersen. New group signature schemes. In: Advances in Eurocrypt'94, LNCS 950, pp. 171–181, Springer-Verlag, 1994.

[35] X. Ding, G. Tsudik and S. Xu. Leak-free group signatures with immediate revocation. In: Proceedings of 24th International Conference on Distributed Computing Systems (ICDCS 2004), pp. 608–615, IEEE Computer Society, 2004.

[36] Clark, Jeremy, Oorschot, P. C. van and Adams, Carlisle (2007): Usability of anonymous web browsing: an examination of Tor interfaces and deployability. In: Proceedings of the 2007 Symposium on Usable Privacy and Security 2007. pp. 41-51.

[37] D. Boneh. The decision diffie-hellman problem. In Third Algorithmic Number Theory Symposium (ANTS), volume 1423 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, 1998. Invited paper.

[38] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security (TISSEC), 1(1):66–92, 1998.

[39] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In ACM Conference on Computer and Communications Security (CCS 2000), pp. 33–42, 2000.

[40] A. Menezes, P. Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC press, 1997.

[41] Saad Inshi, Amr Youssef, "Design and Implementation of an Online Anonymous Feedback System", proceedings of the 24th Biennial Symposium on Communications, Canada, June 2008.