Anonymity and Trust in the Electronic World

Partha Das Chowdhury

A thesis submitted in partial fulfillment of the requirements of the University of Hertfordshire for the degree of Doctor of Philosophy.

This research was carried out in the Department of Computer Science,

University of Hertfordshire

February 2005

Abstract

Privacy has never been an explicit goal of authorization mechanisms. The traditional approach to authorisation relies on strong authentication of a stable identity using long term credentials. Audit is then linked to authorization via the same identity. Such an approach compels users to enter into a trust relationship with large parts of the system infrastructure, including entities in remote domains. In this dissertation we advance the view that this type of compulsive trust relationship is unnecessary and can have undesirable consequences. We examine in some detail the consequences which such undesirable trust relationships can have on individual privacy, and investigate the extent to which taking a unified approach to trust and anonymity can actually provide useful leverage to address threats to privacy without compromising the principal goals of authentication and audit. We conclude that many applications would benefit from mechanisms which enabled them to make authorization decisions without using long-term credentials. We next propose specific mechanisms to achieve this, introducing a novel notion of a short-lived electronic identity, which we call a surrogate. This approach allows a localisation of trust and entities are not compelled to transitively trust other entities in remote domains. In particular, resolution of stable identities needs only ever to be done locally to the entity named. Our surrogates allow delegation, enable role-based access control policies to be enforced across multiple domains, and permit the use of non-anonymous payment mechanisms, all without compromising the privacy of a user. The localisation of trust resulting from the approach proposed in this dissertation also has the potential to allow clients to control the risks to which they are exposed by bearing the cost of relevant countermeasures themselves, rather than forcing clients to trust the system infrastructure to protect them and to bear an equal share of the cost of all countermeasures whether or not effective for them. This consideration means that our surrogate-based approach and mechanisms are of interest even in Kerberos-like scenarios where anonymity is not a requirement, but the remote authentication mechanism is untrustworthy.

Acknowledgements

I am and will always remain deeply indebted to my supervisor and teacher Professor Bruce Christianson for his advice and support. He is a brilliant researcher and a wonderful person. He gave me the opportunity to conclude with a PhD my career as a student. I hope that this document shows that Bruce's trust in me was justified.

I am thankful to Mr. James Malcolm for his support and help to improve the quality of my English. Apart from being a wonderful teacher James is an extremely kind person. I am grateful to James for his help during all these years. I consider myself lucky for having the opportunity to work with James.

I am also thankful to Mr. Bob Dickerson for all his help and support and the useful discussions we had.

For reading and commenting on earlier papers and drafts of this thesis I am grateful to Dr. Bruno Crispo. He is a brilliant researcher and a fine teacher who has always helped me to learn my subject.

I would like to thank Dr. Mike Roe and Dr. William Harbison for the useful discussions we had about the problem investigated in this dissertation.

Finally I would like to thank my wife Mousomi for constantly encouraging and supporting me.

Declaration

This dissertation is a result of my own work and (except where otherwise explicitly indicated in the text) includes nothing which is the outcome of work done in collaboration. No part of this dissertation is being currently submitted for a degree, diploma or other academic qualifications at any other University.

The pronouns 'we' and 'our' in the text, which have been used for stylistic reasons, should be taken to refer to the singular author.

Dedications

To my beloved grandmother - Indumati Devi: Long gone but never forgotten!

To my parents - Mr and Mrs. Das Chowdhury: Without them this work would not have been possible!

To my aunt - Mrs. Maya Bhattacharya: For making many things possible!

Contents

Chapter 1

Introduction

1.1 Motivation

The purpose of authentication is to verify that a user is who he/she is claiming to be. The goal of authorization is to provide access for certain users to certain resources based on predefined business rules. An audit trail links actions to principals retrospectively. Authentication, authorization and audit trail are three traditional concerns in building a privilege management infrastructure (PMI). Traditionally, authentication is strong (based long term credentials linked to a stable identity) and authorization is linked to audit via the authentication mechanism (explicitly using the same long term credential and identity).

Often authorization decisions are made in electronic services using a stand alone application known as a trust management engine [10]. Trust management engines are used to aid applications in situations where the application is faced with a request for a potentially dangerous action. Long term credentials of requestors are evaluated against policies by the trust management engine before a decision is made about the request. For example in a typical role based access control scenario [7, 35] users present the role server with the user's key certificate (which is long term and linked to a stable identity) and prove that the presenting user is the legitimate owner of the key certificate. The role server can then consult a trust management engine like Keynote [9] or Policymaker [10] before activating the relevant roles.

Traditionally identity management has been a part of the trust management envelope. The repeated use of long term digital credentials (by trust management engines to make authorization decisions) enables an adversary to correlate all the actions of a particular user and then link these actions back to the stable identity to which these credentials correspond. Thus an adversary can build a complete dossier on an individual [19]. Thieves can steal sensitive personal information, terrorists can track their targets using government maintained address records, or servers at the far end can leak sensitive information about us. In many instances in the past people have suffered damage due to the malicious use of sensitive information [4]. One of the problems is that too much information about us is stored at too many places, maybe without our explicit knowledge and consent, and we do not have a clue how personal sensitive information will be used by entities at the other end of the communication channel. Moreover most attacks on electronic databases are by insiders and not outsiders [4], and tighter access controls cannot prevent an attack mounted by a user with legitimate access to the system.

There is a widely held view (which we discuss further in chapter 2) that the rapid erosion of privacy, resulting from the repeated use of long term credentials in the electronic world, is a threat. There have been several previous approaches (which we discuss in chapter 2) advocating anonymous transactions. The problem of using long term digital credentials linked to a stable long term identity has well been understood in the world of commercial transactions, and anonymous payment mechanisms [19, 18] were designed to deal with this. However current anonymous payment mechanisms cannot be used to address the threats to privacy in the domain of access control systems. Digital cash [19] for example advocates complete anonymity and it then becomes difficult for an auditor to link actions to principals retrospectively which is a legitimate requirement.

1.2 Scope of this Dissertation

This dissertation focuses primarily on the area of access control systems, and argues that the exclusive use of long term credentials represents a lost opportunity in this area. We propose that many applications would benefit from having ways of

making authorization decisions without using long term credentials. We propose examples of such mechanisms in this dissertation, using which policies can be enforced in access control systems without compromising the privacy of a user. Our approaches allows a localisation of trust and entities are not compelled to transitively trust [21] other entities which form part of a system.

This dissertation investigates the extent to which taking a unified approach to trust and anonymity can provide useful leverage to address the threat to privacy without compromising the principal goal of access control mechanism, which is to allow access to a resource to authorized persons and to prevent unauthorized access. In our privacy model users reveal their identities to some and conceal their identities from others, which is similar to the privacy model proposed in [1]. We focus on some prominent role based authorization models with emphasis on providing auditable anonymous role activation mechanisms using short lived electronic identities. We propose a new layer of anonymity in the current trust management systems which can coexist with traditional non-anonymous mechanisms. To be precise we concentrate on

1. Establishing the extent to which un-correlatability is an obvious requirement for authorization systems.

2. Examining how we can address classical trust management problems in an anonymous way.

These two requirements are not independent, at present we are prevented from thinking of the former as there is no way of doing the latter. This dissertation provides a happy middle ground between absolute privacy and zero privacy. Our approaches are founded on an ability to control both the availability and linkability of transactions. One of the interesting aspects of transactions is that we have no real control over the actions of the other party, so in order to achieve certainty of privacy we use unlinkability as our weapon.

1.3 Contribution

In this dissertation we define a new notion of surrogates or short lived electronic identities. We construct authorisation mechanisms using these surrogates which

are suitable for use with role based access control systems such as RCBS [7], NIST model [35] and OASIS [6], which currently rely on authentication using long term credentials for activation of roles. We demonstrate how the powerful concept of activation of roles using prerequisites described in [6] can also be achieved by combining our surrogate-based authentication mechanisms with ring signatures [55]. We also propose an authentication mechanism using our surrogates suitable for object based distributed systems like Globe [52].

The designers of various previous anonymous transaction systems [15, 43, 64, 14, 66] emphasize the property of non transferability *i.e.* only the owner of a credential can use the credential and the credential cannot be delegated for use by others. We take issue with such approaches, and investigate ways of supporting auditable delegation using our surrogates in the anonymous world. We show how our surrogates can be used to combine anonymous authentication with delegation, and give some example scenarios where this is beneficial. Our delegation mechanism does not greatly restrict the choice of delegation semantics, although for exposition we adhere to Crispo's [25] delegation model in this dissertation.

We also show that surrogates can be combined with existing electronic payment mechanisms without compromising the anonymity of the user. Our surrogate based authentication mechanism allows an unbiased auditor with appropriate authority to correlate actions to individuals but a malicious auditor cannot forge audit records. Moreover we also show that our surrogates can be combined with delegation in such a way as to allow a two level audit mechanism with different levels of trust assumptions for an external and internal auditor. Our approaches allow a localisation of trust and users are not required to transitively trust [21] external entities. These mechanisms turn out to be useful even when anonymity is not a requirement because they enable remote authentication mechanism to be taken out of the local trust envelope.

Our surrogates are generated by modular exponentiation of the parent public key of the user and the secret value corresponding to a surrogate is generated by modular multiplication of the private key with the exponent used to generate the surrogate. Surrogates differ from other anonymous or blinded credentials [19, 12, 18, 43, 15, 63] in the way that anonymous credentials need to be certified by some authority [15, 63], and can be reused [15], or the user needs to get a new

credential issued after every single transaction [63]. The surrogates we propose are for single use, does not need to be certified and can only be used by their legitimate owner. Surrogates are verified by proving that the user/presenter of the surrogate has knowledge of the secret that was used to generate the surrogate without revealing the secret. The verification authority can verify surrogates but cannot masquerade as a legitimate user of the system.

1.4 Organization

We start in chapter 2 with a brief overview of various trust management systems and anonymous transaction systems. Chapter 3 presents a brief overview of access control and authentication systems. In chapter 4 we define the new notion of surrogates which is the main contribution of this dissertation. In chapter 5 we introduce the basic authorization protocols. We present some mechanisms for auditable anonymous delegation in chapter 6. The implications of our work in some other application areas is described with some examples in chapter 7. We set out our conclusions in chapter 8.

We have deliberately kept the main body of the dissertation short and included additional material in the appendix. The Appendix contains a number of papers and a technical report. The first paper entitled "A Palladium Based Solution for Bi-partite Trust Management" presents a mechanism which was developed at the very early stages of the author's PhD project and which we now regard as a false start. This is followed by "Anonymous Authentication" a paper which was presented at the Cambridge security protocols workshop in 2004 and will be published in the Lecture Notes In Computer Science by Springer Verlag. The paper "Uncorrelatable Electronic Transactions using Ring signatures" where we use ring signatures to authenticate users is included next and was presented at the Wholes workshop organized by the Swedish Institute of Computer Science. The paper titled "Authorization for Ubiquitous Computing", which introduces a protocol which can be used in systems based on distributed shared objects, was presented at the Conference on Distributed Processing and Networking organized by Indian Institute of Technology. The final item in the appendix is a technical report published by the computer science department of the University of Hertfordshire titled "How

to deceive Prying Eyes in the Electronic World".

Chapter 2

Trust and Anonymity

2.1 Introduction

Considerable work has been done both in the area of anonymous transaction systems and of trust management. In this chapter we review some prominent previous approaches in the light of the research question we are addressing in this dissertation. Trust management systems were developed to aid applications in making decisions about requests for potentially dangerous actions, based on local policies and credentials. For our purposes we divide anonymity techniques into two major classes, namely anonymous communication channels and anonymous transaction protocols. The former operates at the network layer, the latter operates on top of the anonymous channels. Examples of the former are Mixnets [17], Mixminion [27] and crowd [54] and examples of the latter are Digital Cash [19], Pseudonym systems [18], [20], [55]etc.

Under various subsections of 2.2 we discuss various trust management approaches like Keynote, Independent Trust Entities, Attribute Vector model, Certificates, trusted computing, which is followed by a discussion, of anonymous communication protocols at the network layer which protects against traffic analysis, in 2.3. Section 2.4 reviews anonymous transaction protocols which operate on top of an anonymous communication network discussed in section 2.3. Since we use ring signatures using RSA keys in some of our protocols, we give a brief description of the RSA cryptosystem and ring signatures in sections 2.5 and 2.6

respectively. This is followed by our conclusions.

2.2 Trust Management Approaches

2.2.1 Trust

Trust is a complex subject and there is a considerable variation in the meaning of trust as used in the literature [38]. Trust is often used to mean reliance *i.e.* A cannot complete A's part of a task unless B completes what is required of B. A can validate B's actions and it is asserted in [21] that in this context A does not need to trust B although A needs to rely upon B. In the context of the trust management system we discuss in section 2.2.2 trust is often used interchangeably with authorization and a trust management system was designed to aid applications to answer the question "Should we allow to carry out this dangerous action". In [38] trust is defined as "the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context". It is noted in [38] that delegation is an example of a transitive trust relationship, but in contrast it is concluded in [21] that trust is not delegation. In this dissertation we adhere to the notion of trust set out in [39] which defines trust as a measure of risk *i.e.* A trusts B means that B has the ability to violate A's security policy. We use the services of a third party whom we refer to as a partially trusted third party (see chapter 4) in the following chapters. Users trust the partially trusted third party to aid them to carry out a transaction but the third party cannot forge transactions or audit records. In other words the partially trusted third party has the potential to violate A's anonymity, but cannot masquerade as A. We attempt where possible to avoid transitivity, as it might have adverse and unexpected results [21] in particular, all trust relationships are local and users are not required to trust unknown entities external to their local domain. In the rest of this section we discuss some trust management systems. Systems like Keynote were proposed to address the issue of authorisation (which we deal with in this dissertation) in access control systems where as independent unbiased trust entities were designed to address the issue of privacy. We keep policy specification languages outside the purview of this discussion as our proposed approach (see chapter 4) is policy neutral and can

be used along with any policy specification language.

2.2.2 Keynote

Keynote [9] is the latest version of a trust management approach [10] that initially came from Blaze and others at AT&T. Policy maker [10] was also developed by the same people and some of them were involved in the development of REFEREE [24] another trust management approach that was developed along with researchers from MIT and W3C. We will focus on Keynote here. Keynote works more or less like a database query engine. It can function as a stand alone application interfacing with other parts of the system and helping them in making decisions. Let's lump these other parts of the system together and call them 'client applications'. Whenever any client application faces the question "Should we carry out this dangerous action " then it refers to Keynote for an opinion and based on that opinion it decides its future course of action. The application presents the Keynote trust management engine with a set of local policies that should be taken into account while taking a decision on this particular request, along with the credentials of the requestor and details about the proposed action. If the proposed action conforms to the local policy then Keynote advises the requestor to proceed, otherwise Keynote advises it not to perform this action as it is against the local policy. Keynote acts as a compliance checker for the client application. The policies are specified in the form of assertions and the actions are specified which are evaluated against these assertions.

Let our policy be that we are only going to allow payment by credit cards to those who are authorized to accept them and have the required credentials from a bank or a building society which is legally empowered to issue such authorizations. A keynote assertion specifying this policy looks like

- Authorizer: "DSA: 1FFG2" # CA's key
- Licensee: "RSA: DEF662" # Buyer's key
- Conditions: "((app_domain == "BUY") "Pay by credit card if seller is authorized to accept credit cards")"

- Signature: "DSA-SHA1:1861234" # Signature of the authorizer.

The sellers need to have digital representation of their authorizations to accept credit cards. Lets the seller use a SPKI certificate which can be roughly as

- Issuer: "RSA 2GG36" # Bank's key
- Subject: "RSA 7YYH5" # Seller's key
- Authorization: "Can accept credit cards"
- Delegation: "No"
- Validity: "10/02 - 10/09"

When there is a request by the buyer then the relevant application fetches the relevant credential of the seller, parses it and presents it to Keynote along with the action requested, the id of the requestor and the id of the policy to be consulted to make a decision. Based upon this information keynote comes out with a decision which is most likely to be positive in the above instance. Keynote perfectly enforced compliance with the above mentioned policy but that is not all that we want to achieve. Here keynote checks whether the remote host is allowed to accept credit cards or not and based on this it gives a decision. But consider a situation where there is a corrupt or disgruntled employee who steals credit card numbers and uses them, or in other cases people getting access to other information like medical records etc. Using Keynote one has to live with the threat of correlation of access requests by an adversary. So we shouldn't use a Keynote affirmation to wrap the entire organization with a trust blanket when we do not have information about the storage and use of records. This static nature of assertions won't be able to support a dynamic real world process, where information is accessed and used by several different users.

2.2.3 Independent Unbiased Trust Entities

Independent unbiased trust entities like TRUSTe, EEF [48] *etc.* issue a seal that is displayed on the websites that do financial transactions online. There are also alternative dispute resolution agencies that intervene whenever there is a dispute

between the consumer and the seller. These seals are more like the trade licenses that we find in most shops or like a safety certificate. For example any boat plying on the Trent River has to have some form of authorization from the British waterways board and display that, but the fact is that just having a seal doesn't guarantee all the employees are trained what to do when there is a man overboard. There are checks while issuing such seals or certificates but someone can always register and get a seal and later on turn dishonest. We are very skeptical about the utility of such practices in the cyberspace as gathering evidence purely in the electronic world is very hard to do [57], and a seal cannot act as a guarantee for somebody's honesty. Seals can have a psychological effect on the customer who doesn't understand the system in much depth. So if we again put the question what happens to our personal information, can these trust entities give us a satisfactory answer? Neither do these seals guarantee proper delivery of goods or save the seller from being defrauded. Merchant websites displaying such seals can argue that since they are displaying the relevant seals they are expected to act properly. The success of these independent unbiased trust entities depends more on self regulation which assumes that everyone will benefit by acting honestly. The organizations issuing seals have little control over the storage, use and access control mechanisms of the target organizations. The seal issuing organizations also lack verifiable proof of use and storage of information by the sellers and so an element of risk or unnecessary trust is involved between these trust entities and the sellers.

2.2.4 Certificates

Digital certificates were first proposed by Kohnfelder in his 1978 MIT bachelor's thesis [42]. A digital certificate was initially devised after public key cryptography was introduced and the need arose to communicate to principals each other's public key. Typically a public key certificate contains information about the issuer which may be a certifying authority, the subject whom it is supposed to represent, the dates the certificate is valid and related information. This kind of certificate could not vouch for the trustworthiness of the subject but is primarily used for binding keys to principals [33]. Digital certificates may also be used to check for

authorization as in SPKI [32]. As we have seen above in section 2.1, a certificate from a bank states whether or not a particular seller is capable of accepting credit cards but that is not enough to ensure that our credit card numbers won't be used maliciously. Certification schemes depend on a global authority [33]. We need detailed assertions about who can be a CA, its authority, revocation services etc in order to build a public key authority of some credibility. In the light of these problems it can be said that current certificates aren't readily able to address the issues which we are interested in like proper use of personal information. More over we are yet to build a global certification authority and our proposed schemes do not need a global certification authority.

2.2.5 Attribute Vector Model

The Attribute Vector Model [60] was developed with the goal of allowing trust decisions in pervasive computing. This model incorporated both the traditional identity based model and the context based model that is of relevance to pervasive computing. In the attribute vector model the degree of trust of an entity S_i on S_j is derived as

$$D(S_i, S_j) = f(A(S_j))$$

where S_i and S_j are separate entities and A is the set of their attributes. S_i's degree of trust D on S_j is evaluated using a function f which takes the attributes A of S_j as an input. Attributes can be traditional credentials or they may be context-based attributes like location. The former can be used for traditional computing purposes and the later can be used for pervasive computing devices. In the light of our question relating to the safety of our personal information now we do not think this model can be of much help when applied in the traditional context as it operates on credentials and we have seen earlier that credentials aren't meant for guaranteeing somebody's trustworthiness.

Apart from the approaches mentioned above there were other systems like PICS [24] which is mostly used for content selection to prevent children from visiting pornographic websites. In PICS we do not have the means to specify

policies and credentials in the metadata format. PICS looks for labels and based on them it approves or disapproves a decision. This kind of system cannot be used to get an answer to the questions we started with. Microsoft Authenticode [24] was another approach that was developed to tackle a particular trust management problem called code signing. Authenticode provides users with the authenticity and assurance for accountability for software downloaded over the internet. These approaches rely on the use of long term credentials which by itself is a problem as discussed above in sections 3.6 and 2.2.2

2.3 Anonymous Communication Channel

In this section we discuss the mechanisms which prevent an adversary from figuring out who is communicating with whom and when, thus preventing traffic analysis. This is important as the protocols we describe later on in chapter 3 of this dissertation assume the existence of an anonymous communication channel between the communicating parties. Here we discuss some approaches to thwart traffic analysis by an adversary who can observe all traffic on the network.

Chaum introduced the idea of using relay servers or remailers or mixes for anonymous communication [17]. The first widespread implementation [26] of an anonymous communication channel were produced by the Cypherpunks mailing list, which was based on the theoretical work on Mixes. Later Mixmaster [45] was developed which added some features missing in Cypherpunk remailers. A mix network which allows the sender to choose the path is known as a *free route* network where as a mix network which allows only fixed routes is known as a *cascade* network. Cascades provide greater anonymity against an adversary who owns many mixes [8] than free routes, but are more vulnerable to blending attacks [59]. Moreover cascade networks have lower anonymity as the anonymity set is limited to the number of messages the weakest node can handle, whereas in free networks larger anonymity sets are possible as no mix acts as a bottle neck and many mixes handles messages in parallel [27].

2.3.1 Chaum's Mix nets

This technique was proposed by David Chaum [17] based on public key cryptography. It allows an electronic mail system to hide who a participant communicates with as well as the content of the communication. The system assumes that:

1. anyone may learn the origin, destination and representation of all messages in the underlying telecommunication system and anyone can inject, modify or remove messages.

2. It is difficult to determine anything about the correspondences between a set of sealed items and the corresponding set of unsealed items.

The users of the computer system include not only the communicating partners but also a computer called a mix which will process each mail before it is delivered. The sender encrypts a message with the recipient's public key, and appends the address of the recipient. The encrypted message and the address of the destination are then encrypted using the public key of the mix. If the public key of the recipient is K_r and that of the mix is K_m then the input to the mix is $K_m[R_1, K_r(R_0, M), A_r]$ where R_0 and R_1 are random numbers used as confounders (the confounder is a random byte string which has been inserted in the message and is intended to make chosen and known plaintext attacks more difficult) and A_r is the address of the recipient and denotes . The mix decrypts the input using its secret key and then forwards the message to the recipient. The mix outputs messages in batches. The goal is to hide the correspondence between the input to, and output of, a mix. However if one item is repeated in the input and is allowed to be repeated in the output then the correspondence is revealed for that item. What is important is that the mix removes redundant copies before the output. In case of a single mix the approach is to maintain a record of items used in previous batches.

2.3.2 Onion Routing

Onion routing [62] is a general purpose communication infrastructure for private communication over a public network. It interfaces with off the shelf software

and systems through specialized proxies making it easy to integrate into existing systems. It operates by building anonymous connections within a network of real time Chaum mixes. Onion Routing's network of core onion routers are distributed and under multiple administrative domains so that no single router can compromise the entire network. Onion routing can be used with both proxy-aware and non proxy-aware applications. Onion routing's anonymous connections are protocol independent and exist in three phases:

1. Connection set up- Set up begins when the initiator creates an onion which defines the path of the connection through the network. An onion is a data structure that specifies the properties of the connection at each point along the route.

2. The next phase is data movement when data travels along the connection and each onion along the route uses its public key to decrypt the onion it receives. As data moves through the anonymous connection each onion router removes one layer of encryption as defined by the control information in the onion defining the route.

3. Connection tear down can be initiated by either end or by the middle.

2.3.3 Mixmaster: Type 2 Remailer protocol

Mixmaster's [45] design philosophy is strongly influenced by Chaum's digital mixes. Messages are sent as one or more packets. All mixmaster packets are of same length and all bits are encrypted with a 3DES [61] key at every hop, so no information about the identity of the message is visible to the observer. Even a compromised remailer can only know the previous and next locations in the chain but it cannot figure out how many preceding hops there were or how many following hops there will be. The header for the last remailer in the chain contains a flag indicating that it is the last hop, and indicates whether the message is part of a multipart message. If it is part of a message then the message ID number is used to identify all the other parts. Only the last hop can see that a group of packets are part of a single message. If not all parts arrive within a time limit then the message is discarded.

2.3.4 Mixminion: Type 3 Remailer protocol

Mixminion [27] a protocol for asynchronous loosely federated remailers addresses some of the flaws of Mixmaster while being as flexible as Mixmaster. Mixmaster does not support replies while Mixminion introduces a new primitive called *single reply use block* (SURB) which makes correlated replies as secure as forward blocks. In Mixminion the servers themselves cannot distinguish reply messages from forward messages. Mixminion uses TLS over TCP for link encryption between remailers and use ephemeral keys to ensure forward anonymity for each message, where as Mixmaster uses SMTP for transport. Most ISPs do not tolerate users who potentially deliver hate mail etc, and this requirement forms a barrier to wide scale remailer deployment. Mixminion allows each node to specify and advertise an exit policy, where as Mixmaster provides no way for the nodes to advertise their capabilities and roles. Replay attack is a serious problem in mix networks: Mixmaster keeps a list of old entries but here an attacker just has to wait till the server has forgets all its old entries and then replay a message. Mixminion addresses this threat using key rotation: a message is sent addressed to a given key and after the key changes no messages to an old key will be accepted. Mixminion uses synchronized redundant directory servers to provide information about the network compared to the ad hoc approach of Mixmaster. There is also a simple dummy policy in Mixminion to improve anonymity.

2.4 Anonymous Transaction Protocols

2.4.1 Private Authentication in Mobile Networks

Protocols for private authentication in mobile networks were proposed in [1] where communicating parties reveal their identities to each other but not to others in a group or third parties. When a mobile principal A wants to communicate to another mobile principal B both in the same location then they both exchange messages encrypted with their public keys in such a manner that an eavesdropper cannot detect the presence of either A or B in the area. The protocols in [1] cannot be used for the scenarios we deal with in this dissertation as in our cases

the communicating partners do not reveal their identities to each other. Moreover in [1] two parties always use long term credentials to communicate with each other which can be used to correlate actions of a particular principal.

2.4.2 On iPrivacy and Lumeria

These systems were developed to protect user's personal information from companies. iPrivacy is a U.S based company whereby the user downloads software from a website [48]. This software encrypts the user's personal details, creates a fictitious identity and a one time credit card number, which is matched by the credit card company with the real credit card number and then the goods are delivered at an address chosen by the customer. With Lumeria [48] all the information is stored with Lumeria and the customer accesses the seller via a proxy server of Lumeria and can then buy goods online. These schemes can be compared with the example that we have mentioned earlier where one trusts his/her doctor to keep their medical records safe. We cannot have any verifiable proof about the integrity of the software downloaded over the net or the integrity of the company storing our personal information. So these kinds of schemes we think are vulnerable to abuse in the same way as the previous ones where we keep our information with the selling websites.

2.4.3 Chaum's Digital Cash

A detail description of digital cash can be found in [19]. The user goes to the bank with a signed request called a note; the bank credits the account of the requestor after checking the signature on the note. In this scheme users generate the note number which the bank cannot see and that's how they ensure that even the bank cannot track the spending habits of the user. The note number is unique for every different note. After a note has been spent the bank can see the note number but cannot figure out to whom the note was issued. Users carry a device called representative supplied by the bank to generate the note numbers. There are also proposals of implementing an observer in the representative of the user which checks against double spending. The anonymity of a user can be compromised if he/she tries to spend a particular note twice. This scheme cannot prevent transfer

of credentials which is not a problem in case of notes but certainly would be for driving licenses, medical prescriptions etc.

2.4.4 Pseudonyms

A scheme using pseudonyms or short lived electronic identities was proposed in [43]. Here the user generates private and public keys and so do the organizations. The user goes to the credential issuing organization and generates an pseudonym (short lived identity) which is a function of the secret and non secret keys of the user and the organization. Credentials contain the nym of the user which the user can then use for the purpose the credential is meant for. The user has different credentials for different organizations and it can use credentials issued by one organization while dealing with another organization. Our proposals differ from pseudonyms: in our scheme the user is globally represented by his/her public key rather than by different pseudonyms, auditable delegation is not possible using pseudonyms.

2.4.5 Idemix

A description of idemix can be found in [15]. In idemix an user first registers with a global pseudonym authority (PA) with which the user registers its pseudonym and PA issues a credential stating that the pseudonym is valid. The user then can use the pseudonym to get a reference for a credit card payment from a different organization. The organization issuing payment tokens trusts the PA. A user can then use the payment tokens with other organizations. The user, PA and other organizations have to be part of the idemix system: idemix issues IP addresses as well as SSL certificates to each of them. Moreover there arises a need for a global pseudonym authority which everyone needs to trust. It is our contention that building a global pseudonym authority is as difficult as building a global public key authority and both in any case are developments yet to come to fruition. Moreover transfer of credentials is not possible in the idemix system. The mechanisms we propose in chapter 5 do not need a global pseudonym authority, as trust relationships are more localised *e.g* students trust their own university net-

work administrator. Moreover our mechanisms can coexist along side with non anonymous mechanisms and do not need a significant change in the infrastructure.

2.4.6 Globally Unique Pseudonym

A scheme for tying attributes to pseudonyms was proposed in [64]. An user contacts a registrar with a proof of his/her identity. The registrar is not a single entity but a group of principals and the user must contact a threshold number of them. The registrar then contacts an issuer who issues a globally unique pseudonym to the user and binds the pseudonym with the public key for signature and encryption which is called a GUP certificate. Issuers like registrars are threshold entities. Having threshold entities prevents disclosure of individual information as well as registration of multiple entities by a single user. GUP certificates can be used either as a attribute certificate or can be used for commercial purposes. The scheme presented in [64] requires a global pseudonym authority and a registration authority which is similar to trusting a third party with personal sensitive information [4]. The schemes we present in this dissertation has a more localised trust relationship rather than a global authority.

2.5 The Rivest Shamir Adleman Cryptosystem

The RSA crypto-system [56] relies on the difficulty of factoring composites of large primes to provide its security. A composite $n = p * q$ is computed, and made public, while the two primes p and q are kept secret. A value e where $1 < e < (p-1) * (q-1)$ is chosen at random, and d such that $d * e = 1 \bmod (p-1) * (q-1)$ is efficiently calculated. The public key is(e, n), while(d, n) is the private key. In order to encrypt a message M for the public key (e,n) one simply performs an exponentiation modulo n. The ciphertext is therefore

$$M^e \bmod n \tag{2.1}$$

To decrypt, the message is simply raised to the power of the decryption key,

$$M^{ed} \bmod n = M \bmod n \tag{2.2}$$

Digital signatures can also be implemented. The public verification key is denoted (e,n) while the signature key is (d,n). The signer raises the message to be signed to the power d as follows

$$M^{d} \bmod n \tag{2.3}$$

and the verifier checks the signature as follows:

$$M^{ed} \bmod n = M \bmod n \tag{2.4}$$

All operations are performed modulo n. Digital signatures provide integrity properties and non-repudiation properties: if the public key of Bob is well known, Alice can prove to a third party that Bob has signed a message. We in this dissertation use RSA keys to generate ring signature (see section 2.6).

One of the major issues in public key cryptosystems is key management. Several techniques have been proposed for the distribution of public keys which are as follows

1. Public anouncement
2. Publicly available directory
3. Public key authority
4. Public key certificates

For a detailed discussion on key management one can refer to [61, 11]. For the protocols we propose in this dissertation we do not assume the existence of a global public key authority, thus eliminating transitive trust relationships between users and global authorities.

2.6 Ring Signatures

Protocols for our example scenarios 5.4, 5.3 and 6.3 make use of a two level authentication mechanism where users first authenticate as members of a group using ring signatures and then use their surrogates to authenticate as discussed in section 4.5. We give a brief description of ring signatures here. Ring signature was designed Ron Rivest, Adi Shamir and Yael Tauman [55]. They call a set of possible signers a ring. One of the members of the ring actually signs using his/her own private key and the public keys of the other ring members. In this signature scheme the verifier doesn't learn who the signer is but can only learn that the signer is a member of a certain group of possible signers. In producing such a signature the signatory doesn't need the co-operation of any other member of the group. The other members do not even have to agree to be in the group. All the signer needs to know is the public keys of all the members of the group. Unlike group signatures ring signatures do not have any set up procedure, have no revocation procedures (but verification depends on ring membership at the time of signing), and any user can choose any set of possible signers. What is important from the point of view of the protocols we present in this dissertation is that ring signature is perfectly signer ambiguous. If there are r members of a ring then it is difficult for an adversary to guess the correct signature with a probability more than $1/r$. Since complete anonymity is not a requirement in our protocols we use ring signature in conjunction with our surrogates to enable an auditor to link actions to individuals retrospectively.

The ring member who actually produces the signature is known as the signer and other ring members are known as non signers. Each member is associated with a public key P_k (via a PKI directory or certificate) and the corresponding private key is known as S_k. It is assumed that the ring members use a trapdoor one way permutation (such as RSA) to generate and verify signatures. Ring signatures have two procedures:

1. $ring-sign(m, P1, P2....Pr, S_k)$: This produces a signature σ on message m, given the public keys $P1, P2....Pr$ of the r ring members and the private key S_k of the signer.

2. $ring-verify(m, \sigma)$: This accepts a message m and the signature σ (which includes public keys of all the members of the ring) and outputs either true or false.

Here we describe a ring signature scheme where individual signers use RSA as their signature scheme.

2.6.1 Ring Sign

Each ring member A_i has an RSA public key $P_i = (n_i, e_i)$ and the trapdoor one way permutation f_i over Z_{n_i} is defined as:

$$f_i(x) = x^{e_i} \bmod n_i$$

By the properties of RSA it is assumed that only A_i can compute the inverse permutation f_i^{-1} efficiently. Since the trapdoor one way permutations of various ring members will have domains of different sizes, so for ring signatures all the trap door permutations are extended to have a common domain $\{0,1\}^b$, where 2^b is some power of two which is larger than all moduli n_is. The trapdoor oneway permutation f_i over Z_{n_i} is being extended to g_i over $\{0,1\}^b$ and a detailed discussion can be found in [55]. The signature is generated as:

1. The signer s first computes a symmetric key as the hash of the message m to be signed:

$$k = h(m)$$

2. Then the signer chooses a random initialization value v uniformly at random from $\{0,1\}^b$.

3. Next the signer for picks random x_i for each other member of the ring, $1 \leq i \leq r$, $i \not= s$, uniformly from $\{0,1\}^b$ and computes y_i for each ring member where $i \not= s$:

$$y_i = g_i(x_i)$$

4. Fourth the signer solves the ring equation for y_s. The ring equation is a combining function which takes as input the key k, and the initialization value v and values $y_1, y_2, ..., yr$ in $\{0,1\}^b$ and produces an output z in $\{0,1\}^b$. The combining function is efficiently solvable for any single input: for each s, $1 \leq i \leq r$, given a b bit value z and values for all the inputs y_i except y_s, it is possible to efficiently find a b bit value for y_s such that $C_{k,v}(y_1, y_2, ..., yr) = z$. In this step the signer solves this equation for y_s where $z = v$.

$$C_{k,v}(y_1, y_2, ..., yr) = v$$

5. Next the signer uses his knowledge of the trapdoor oneway function in order to invert g_s on y_s to obtain x_s.

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

6. The signature σ on the message is defined as:

$$\sigma = (P_1, P_2, ..., P_r; v; x_1, ..., x_r)$$

2.6.2 Ring Verify

A verifier can verify an alleged signature σ on the message m, where

$$\sigma = (P_1, P_2, ..., P_r; v; x_1, ..., x_r)$$

as:

1. First for $i = 1, 2, ..., r$ the verifier computes

$$y_i = g_i(X_i)$$

2. Second the verifier computes the hash of the message to get the key k as:

$$k = h(m)$$

3. Finally the verifier checks whether the y_is satisfy the fundamental equation: If the equation is satisfied then the verifier accepts the signature as valid.

$$C_{k,v}(y_1, y_2, ..., yr) = v$$

We have discussed the RSA cryptosystem based on composites in section 2.5 where we generate inverses of RSA public keys. These inverses are used for doing ring signatures as described here in this section *e.g.* where the private key is used to invert the trapdoor one way permutation as shown in step 5 of the ring sign operation.

2.7 Conclusions

As discussed above trust management systems, independent trust entities or certificates cannot guarantee proper use of personal sensitive information. Anonymity systems provide a solution to the problem of privacy but cannot be used for making authorization decisions using current trust management systems, as trust management systems currently make decisions based on long term credentials like key certificates. Long term credentials, on the other hand, enable an adversary to correlate all the transactions of a particular user, which compromises the privacy of the user. In chapter 5 we shall present some alternative mechanisms where trust decisions can be made using transient identities or surrogates.

Privacy enhancing technologies [19, 12, 18, 43, 15, 63] cannot ignore the fact that delegation is a key organizational practice. For anonymity systems to gain widespread acceptability they must support delegation. In chapter 6 we propose a mechanism using which we can have auditable delegation of credentials in the anonymous world. We illustrate the general approach which, we propose, by presenting a protocol which can be used in a role based access mechanism to authenticate users to roles anonymously.

Chapter 3

Access Control and Authentication Mechanisms

3.1 Introduction

We review some popular access control models relevant to our work in section 3.2, and describe an approach to access control in a system based on distributed shared objects in section 3.3. The delegation model we shall adhere to in this dissertation is discussed in section 3.4 and is followed by a brief description of an open network authentication system called Kerberos in section 3.5. We describe an early version of Microsoft Palladium and our solution based on this Palladium architecture in section 3.6 and subsection 3.6.1 respectively. This is followed by our conclusions in section 3.7.

3.2 Access Control Mechanisms

Access control mechanisms are designed to control access to valuable information and are used by the military, the government and large organizations. Their goal is to permit authorized access as well as to prevent unauthorized access. In the following subsections we give a brief overview of various access control mechanisms.

3.2.1 Mandatory Access Control

Mandatory access control (MAC) supported both by military and civilian governments attaches security labels to objects to be protected and clearance levels to the users. This model was first formalised by Bell and La Padula [5] but later on Sandhu in [58] presented a minimal model called BLP. In these models access is granted on the basis of the security label of the object and the clearance level of the user. MAC was developed primarily for the military environment and did not permit read up or write down: that is an user with a lower clearance level cannot read information with a higher clearance level nor can write objects with a lower clearance level. Thus information flow among entities is restricted. The system administrator is responsible for maintaining and setting both sets of security levels.

3.2.2 Discretionary Access Control

Discretionary access control (DAC) mechanisms unlike MAC allows users to control who has access to their information. Users can delegate their own rights to other users. A matrix can be used to represent access to all objects by all users. This matrix is known as an access matrix and has a row for every user and a column for every object. DAC is more appropriate for static situations where users and the objects remain same for a considerable length of time. In organizations where users as well as objects change frequently DAC is inadequate [6, 7].

3.2.3 Role Based Access Control

Role Based Access Control (RBAC) has been perceived by many [34] as more appropriate than DAC to the secure information processing needs of non-military systems. The RBAC mechanism is built upon the premise that in large corporations individual employees do not own the information they process or have access to, the information is owned by the corporation and RBAC mechanisms should prevent employees from making unauthorized use of information. Roles in RBAC model the roles individuals perform in any organization *e.g.* in a hospital the roles an individual can perform can be doctor, nurse, clinician, pharmacist etc.

Roles are a group of permissions that an individual acting in the role can perform within the context of the organization. The determination of membership of roles and allocation of permissions to roles are in compliance with organization policies and not so much at the discretion of the system administrator. These policies are based on existing practices, laws, or ethics. The users cannot pass access permissions on to other users at their own discretion. Authentication of users to roles is done using long term credentials like key certificates, role certificates etc. This enables an adversary to correlate all the transactions of a particular user. In mechanisms using long term credentials, audit is traditionally linked to access control via authentication using the same identity. We show alternative ways of authentication and audit without using long term credentials in this dissertation. In this dissertation we also deal with activation of roles using prerequisites as described in [6].

NIST model

The NIST model was proposed in [35]. The NIST RBAC model is organised in four step sequence of increasing functional capabilities. The four levels are:

- Flat RBAC
- Hierarchical RBAC
- Constrained RBAC
- Symmetric RBAC

Flat RBAC captures the essential elements of RBAC. Users are assigned to roles and permissions are assigned to roles. Flat RBAC has a requirement for role review whereby the roles assigned to a specific user can be determined as well as the users assigned to specific roles. Flat RBAC also requires that users are able to exercise permissions of multiple roles simultaneously. Hierarchical RBAC is for supporting role hierarchies. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles acquire the permissions of their juniors. Hierarchical RBAC can support an arbitrary partial order or may impose restrictions on the role hierarchy. Constrained RBAC adds a requirement

for enforcing separation of duties. Constrained RBAC allows permission role review like user role review in Flat RBAC. The NIST RBAC model does not directly address the issue of authentication and depends on strong authentication mechanisms based on long term credentials.

RCBS

The Role and Context Based Security (RCBS) model was proposed in [7]. This model introduces permission centric operational Separation of Duty (SoD) constraints which can be built on top of standard RBAC models or can be introduced as an additional feature of RBAC models. RCBS introduces the concept of critical combination which groups the set of transactions of a task and enforces *Per-Role Operational SoD, Static Operational SoD* and *Inheritance in Operational SoD* between roles with reference to the transactions in the critical combination. The application of permission specific constraints as proposed in RCBS allows for finer granularity in mutual exclusion between roles. Authentication in RCBS is done using role certificates which declares the identity of the user and the roles associated with the user. An activation certificate binds the user's identity with the names of the activated roles of the current session. The role manager and the activation manager generate the appropriate certificates. The certificates can also be used for interdomain authentication in the RCBS model.

OASIS

The Open Architecture for Secure Interworking Services [6] (OASIS) is an RBAC model developed at the University of Cambridge Computer Laboratory. The features of OASIS include session support, prerequisite parameters, context awareness, flexible delegation with a fast revocation mechanism and distributed operation and management. Context awareness is achieved in OASIS either by tagging roles with parameters which depend on the environment in which the role was activated or by the use of environmental predicates. These predicates allow information outside the OASIS system to be incorporated in the access control decision process. OASIS supports a form of delegation where the assigner can transfer rights not held by the assigner *e.g.* a nurse can assign a doctor the role of

treating_doctor although the nurse does not have the rights of a doctor. OASIS provides such assignment of rights by mean of appointments. Appointments in OASIS can express traditional delegation as well as transfer of rights not held by the assigner. In OASIS [6] the assignment of users to roles is handled by role activation rules and these rules allow users to enter roles based on valid prerequisites. Such prerequisites can be a role, an appointment or an environmental predicate. Environmental predicates are powerful enough to express constraints such as separation of duties and cardinalities. Every prerequisite can be tagged to make it a membership condition and can be revoked. If a revoked role forms part of a prerequisite to activate other roles, the revocation of the former can trigger further revocation resuting in a cascade. An example scenario [6] is a hospital where the role *on_duty_doctor* can only be activated by a user if the user has already activated the role *employed_paramedic,local_user* and *on_duty*. Another example scenario is a University where the role *research_student* can only be activated by a student if the student has already activated the role *student*.

3.3 Distributed Shared Objects

We live in a world where resources are hardly ever entirely local but are usually distributed across various locations. There is seldom a central authority that manages all access to these resources. Designers of distributed systems generally use long term credentials (key certificates) for authentication purposes. This means that servers can identify their clients, and can correlate their actions.

Globe [52] is a wide area distributed system where objects are physically replicated at several locations. The principal construct in Globe is a Distributed Shared Object (DSO) which consists of several local objects residing in local address spaces. The local objects storing a part of a DSO's state are known as replicas. A replica consists of the following subobjects:

- Semantics subobject, which is the only subobject written by the application developer, and which contains the code that implements the DSO.
- Communication subobject is responsible for communication between local objects residing at local address spaces.

- Replication subobject is responsible for keeping the replica's state consistent with the other replicas.
- Control subobject is responsible for taking care of the invocations from the client process on the host.
- Security subobject is responsible for implementing the security policies of the DSO.

To invoke a method on a DSO, a user has to create a local object in his/her own address space. This object often acts as a proxy routing requests to appropriate replicas. The Globe Location Service facilitates finding of replicas. A user willing to run a replica or a user proxy needs a Globe Object Server in his/her computer, either stand alone or integrated with other application.

The current proposals by the developers [52] of Globe are based on certificates issued by the DSO owner and follow a role based authorization model.

3.3.1 Access Control in Globe

If we model an electronic newspaper as a DSO then the corporate entity running the newspaper is the owner of the DSO and determines the security policies for the DSO by identifying all the meaningful roles for the object. This involves careful examination of the application concerned, which is outside the scope of this dissertation. Once the user role set is identified for an object then with each user role is associated a bit vector indicating the methods the role is allowed to execute. Grouping the bit vectors for all the roles of a particular object forms the access control matrix for the DSO. This matrix is stored in the security subobject and when a replica is created it gets the matrix from the owner of the DSO. When a role wants to invoke a method on an object the security subobject consults this matrix to verify whether or not this role is allowed to invoke this particular method.

The DSO owners sign certificates binding users to their public key, this certificate is then used by the user to authenticate themselves while requesting access. If there are users with the power to delegate then the security subobject verifies the chain of certificates till it reaches the one signed by the DSO owner. The replicas

also authenticate themselves to the user, and the replica and the user can start talking to each other after they have authenticated to each other. Replicas authenticate themselves to the user using replica certificates issued by the DSO owner.

The disadvantage of using long term credentials by the user to authenticate himself/herself arises from the fact that an adversary can correlate all the actions of any particular user. This will leak information about any user's online activities to an adversary which can be used in a way harmful to the user. We present an alternative authentication model in this dissertation whereby an user can authenticate himself/herself to the replica anonymously and an adversary cannot correlate the actions of any user. This we believe is novel because previous approaches to authentication rely on long term credentials. Similarly, at present trust management systems like Keynote [9] make decisions based on long term credentials and policies. Our new approach doesn't require significant changes to the way authentications are done at present. The trust management systems can be queried using the surrogates [28, 25] employed in our approach, and decisions can still be made based on fixed or dynamic policies.

3.4 Controlled Delegation

Delegation [30, 37] is a process by which a principal A authorizes another principal B to act on its behalf by transferring a set of its rights to B possibly for a specific period of time. In many organizations, employees higher up the hierarchy delegate certain responsibilities to their secretaries or subordinates. Such instances are quite common in the real world and so to be useful in the real world anonymity systems should support delegation in certain situations. In an organization a manager can delegate their secretary the power to act on behalf of the manager for a duration chosen by the delegator.

There are several commercial websites whose customer base consists of people who are legally not allowed to have a credit card (children below 18 yrs). In such situations children use their parents' credit card to buy online. Now the problem here is that credit card numbers once learnt can be reused. Things become more complicated when parents have anonymous credentials instead of credit cards: to share anonymous credentials such as those mentioned in [18, 43, 15, 63]

the owner also has to share their secret key. This is a major problem in such situations where delegation is a legitimate requirement.

In this dissertation we adhere to the semantics of the delegation model proposed by Crispo [25] due to the following reasons. The delegation model proposed by Crispo consider threats posed by the delegator thus allowing space for repudiability of actions. The trust assumptions between the delegator and the delegatee are clearly spelt out in the specification of the delegation mechanisms. It is hard for the delegator to frame the delegatee as well for the delegatee to frame the delegator. It is also emphasized that the principle of consent be explicitly implemented; thus the delegator delegates only with the explicit knowledge and consent of the delegatee.

In the mechanism we introduce in section 6.4 we ensure that the delegator is not able to frame the delegatee nor the delegatee is able to frame the delegator. It is also difficult for the delegatee to masquerade as the delegator and an unbiased auditor can uniquely and irrefutably link actions to individuals without being able to forge the audit records. The delegator would no longer be able to use the permissions it has delegated.

3.5 Kerberos

Kerberos [46] is a trusted third party authentication service and users are referred to as the clients of the Kerberos authentication service. Various services (*e.g.* mail services, file servers) trust Kerberos' judgment as to the identity of a user to be accurate. We give here a brief description of the current design of Kerberos which is followed by an overview of our proposals in section 8.5.

Kerberos has a top level authentication service issuing Ticket Granting Tickets (TGT). The TGT contains clients name along with current time, lifetime of the ticket, and the client's IP address and is encrypted in a key known only to the ticket-granting server and the authentication server. The authentication server then sends the ticket, along with another copy of the random session key, and some other information, back to the client. Only Kerberos authentication server and the client, which is actually derived from the user's password, know this response. In order to gain access to a service server, the application needs to build an authen-

ticator that contains the client's name and IP address, and the current time. The authenticator is sent to a Ticket Granting Service (TGS) along with the TGT and TGS verifies the TGT. The TGS, after authenticating the client, constructs a new ticket that contains a new session key, the name of the client, and an expiration time, all encrypted in the service server's key. Users prove their identity for each desired service to the concerned server and server also proves its identity to the user. Servers are allowed to keep track of all past requests with time stamps that are still valid to counter replay attacks. So what we see is strong authentication based on permanent credentials and audit is linked to authorization via the same permanent credential used to authenticate. Such an approach gives birth to some unnecessary trust assumptions between various entities of the system *e.g.* clients are compelled to trust servers not to enable an adversary to correlate their access requests. We discuss the work that can be carried on Kerberos to remove such compulsive trust assumptions in our future work section discussed in section 8.5.

3.6 Microsoft Palladium

Palladium architecture [16] (now known as NGSCB) was proposed in 2002 by Microsoft in association with other companies namely Intel, HP etc. In Palladium the public and secret keys of a user are embedded in a chip soldered with the microprocessor. This chip is popularly known as a Fritz chip. There is a trusted software component called Nexus which communicates between applications on top and the keys embedded in the hardware. Applications running on top can be divided into trusted and untrusted applications. Applications trusted by the user are known as Nexus certified agents or NCAs. Having keys embedded in the hardware would enable recipients of messages to be certain about the sender of the message and keys would be tied to principals by means of certificates which can be retrieved for verification. The problem with such an approach is that it requires a global certification authority. Even if there is a global certification authority revocation would still be a problem. Once keys are compromised it would require users to replace their trusted boxes with a new one. Moreover such an architecture does not enable plausible deniability which is a legitimate requirement in certain situations [57]. Palladium depends on long term credentials which again is not

good for individual privacy.

3.6.1 Trusted Hardware Based Approach

As part of our initial attempt to address the problem of anonymity and trust we designed a solution using Palladium which is explained in appendix A. Here we explain the salient features of this scheme and explain why this approach was not pursued further. Trustworthiness of individuals, in our Palladium based approach, were ascertained using weights assigned to principals by their peers and a decision was made using a simple metric. We assume the Palladium machine will prevent rogue applications from stealing passwords and implement the owner's (*i.e.* owner of the keys stored in the box) security policies. There is also a trusted third party called AAWS which acts as a repository of information about the users. In the example we present in appendix A users get payment tokens form the bank before they buy goods electronically from an online merchant. Users authenticate themselves to the bank using a certificate issued by the AAWS and the bank issues payment tokens to the users. We assume that merchants also use Palladium machines and those machines are certified by the AAWS. There is a two way reputation assessment between the merchant and the seller. Customers use weights assigned to the merchant by other customers and use a simple metric to ascertain the reputation of the merchant. Merchants use a compliance checker and the certificates issued by the AAWS to the sellers to ascertain the trustworthiness of the customers. The users trust the AAWS with personal sensitive information and the AAWS issues pseudonym certificates to users which users then present to the point where they request access or a service.

Although the exercise was useful we later realized that the disadvantages of such an approach using Palladium outweighs the advantages. Our design does not satisfy the requirement that trust should not be transitive [21] and there was a need for a global certificate issuing and revocation authority which we believe is difficult to achieve and has not happened yet. Moreover systems need to trust the authentication mechanism of a third party as in our example merchants need to trust the authentication mechanism of the AAWS and the bank. We discuss how we can use trusted hardware in our conclusions in chapter 8 where users retrieve

and use the keys in the hardware.

3.7 Conclusions

The access control and authentication mechanisms discussed in this chapter rely on long term credentials for making authorization decisions. Services also depend on the authentication mechanism of a trusted third party *e.g.* various services trust Kerberos' judgment about the identity of a requestor and accepts tickets issued by the TGS. Such approaches give rise to a unnecessary and compulsive trust relationship [39] between principals and services they use as well as between services and the authentication mechanism. The approaches we present in chapter 5 eliminate such compulsive trust relationships between principals; thus allowing principals to independently control the threats they are exposed to.

Traditionally, in access control mechanisms allowing delegation, authentication is linked to audit using long term credentials. This enables an auditor to correlate actions with individuals retrospectively. Since anonymity systems do not use long term credentials it is difficult to link actions to individuals using such credentials; thus making it difficult for an auditor to resolve disputes. Some partially anonymous mechanisms allowing audit are presented in chapter 5.

Chapter 4

Surrogates

4.1 Introduction

The generation and use of surrogates is the key contribution of this dissertation. In the protocols discussed in chapters 5, 6, and 7 we replace long term credentials with surrogates or short lived electronic identities; thus allowing identity to be taken out of the trust management envelope. In this chapter we discuss our key and surrogate generation mechanism. We start by describing our key generation mechanism in section 4.2. Since our surrogate generation method depends on the difficulty of computing discrete logarithms, we next introduce the discrete logarithm problem and briefly review some approaches to the solution of this problem in section 4.3. In section 4.4 we give a general overview the features of our surrogates which is followed by a description of the method used to generate surrogates in section 4.5. In section 4.6 we introduce the assumptions on which we build our protocols, and the notations which we use in the following chapters. This is followed by our conclusions in section 4.7.

4.2 Generation of Diffie-Hellman Keys

A detailed description of Diffie-Hellman (DH) key systems can be found in [29, 44], here we give a brief description of generation of DH keys which is as follows: the user selects a generator g mod P (*i.e.* calculating consecutive powers

of g generates all the elements in the multiplicative group modulo P) and selects a secret $s \in Z_P^*$ (*i.e.* s is an element between 1 to $(P-1)$), where Z_P^* is a multiplicative group [51] modulo a large prime P. The user constructs his/her public key X from the secret s as

$$X = g^s \bmod P$$

It is hard for an adversary to calculate s from X and P as this is the discrete logarithm problem (see section 4.3), which is considered to be intractable. The user proves ownership of X by proving knowledge of s without revealing s (see section 4.5.1).

4.3 The Discrete Logarithm problem

The discrete logarithm problem can be described as follows: Let Z_P^* be a multiplicative group. For $g \in Z_P^*$, the *discrete logarithm problem* for the group may be stated as:

Given $g \in Z_P^*$ and $a \in Z_P^*$, find an integer x such that $g^x = a$.

Such an integer x is the discrete logarithm or index of a to the base g and is denoted as $x = ind_g a$.

Diffie and Hellman used the discrete logarithm problem in cryptography as a source of a one way function in [29]. A one way function f can be defined as one for which given a value $y = f(x)$ it is difficult to compute x at least for most values of y [51]. In their paper Diffie and Hellman proposed as a natural candidate a function based on exponentiation modulo a prime. Let P be a large prime and g be a generator in Z_P^*, then $f(x) = g^x \bmod P$, which is relatively easy to calculate using binary expansion [51]. On the other hand inverting the function f clearly requires an algorithm for computing discrete logarithm in Z_P^* which is thought to be an intractable problem. One of the early users of one way functions in cryptography is Roger Needham [67]. Needham used one way functions to store passwords securely in multi-user operating systems. Most operating systems use passwords for authentication and if the passwords are stored in a file then the file needs to be heavily protected by the operating system, but Needham's approach

was to store the values of a one way function applied to the passwords. When a user typed in his/her passwords, a one way function was applied to the password and the resultant value was compared to the stored value.

4.3.1 Complexity of Discrete Logarithm

The security of cryptosystems against computational attacks is dependent upon what computations are feasible. The field of mathematics that deals with this is known as computational complexity theory and a good introduction can be found in a paper by Shafi Goldwasser [51]. Computational complexity theory classifies computational problems according to their difficulty. If one problem can be solved in polynomial time given a solution to another, then we have a polynomial reduction between the two computational problems. Such polynomial reductions can provide information about the intrinsic difficulty of the problem. For example, the order problem for a finite group generated by g mod P is to find the least $n > 0$ such that $g^n = 1 \bmod P$. It is shown in [51] that an algorithm to compute the discrete logarithm can be used to solve the order problem in polynomial time. Hence it can be said that it is at least as difficult to compute discrete logarithms to a base as it is to compute the order of the base in the group [51].

There are algorithms for solving the discrete logarithm problem which are as follows:

1. The algorithm that applies for general groups builds all n powers of g and then looks up the group elements to find the discrete logarithm and this takes n operations to compute the table and $O(n)$ space to store the table [51]. Daniel Shanks improved upon the bounds of this algorithm to the running time of $O(n^{1/2}logn)$ and the space requirement of $O(n^{1/2})$ group elements [41]. An algorithm was proposed by Pollard [50] which has the same running time but very small space requirement.

2. The second class of algorithms applies to groups whose order has no large prime factor. A positive integer is called smooth if it does not have any large prime factors. S. Pohlig and M. Hellman proposed an algorithm to calculate discrete logarithm where the order of g does not have a large prime

factor [49]. The running time for the group operations of the algorithm can be given as

$$O(\Sigma_{i=1}^{k} c_i(logn + p_i))$$

where n is the order of g (n and g has been explained above in this section), c_i is a random integer and p_i is a prime factor. Once this is done the Chinese Remainder theorem can be used to derive the final value of $ind_g a$.

3. The third class of algorithms are probabilistic rather than deterministic and only works for groups endowed with a special structure [51]. The approaches in this category are commonly known as Index Calculation methods.

4.4 Surrogates

Our surrogates are transient numbers generated from a parent number (such as a bank account number, credit card number, or social security number) by a particular mathematical function similar to that used by David Chaum in the generation of group signatures [20]. There are two parts to a surrogate, one is the publicly transmitted part and the corresponding private part from which the public part is generated. The corresponding private part is known only to the user. In this dissertation we denote the public part as surrogate itself and the corresponding private part as the private value. When we use Diffie-Hellman keys, the public part is known as the parent public key while the secret value from which the public key is generated by modular exponentiation is known as the secret or secret key. In this chapter and the rest of this dissertation we use Diffie-Hellman keys unless otherwise explicitly mentioned (see sections 5.3 and 5.4), thus whenever we use the term parent public key we mean Diffie-Hellman key.

Although our mechanism for producing surrogates is novel, the concrete surrogates which it produces have similar properties to those required by the abstract notion of surrogate postulated in [25]. Surrogates differ from other anonymous or blinded credentials [19, 12, 18, 43, 15, 63] in the way that anonymous credentials

need to be certified by some authority [15, 63], and can be reused [15], or the user needs to get a new credential issued after every single transaction [63]. Our surrogates are for single use, do not need to be certified, and can be generated by the user. We use them for a single transaction and then discard them. Other anonymous credentials [15, 43] can be verified by the acceptor if the acceptor knows the public key of the issuer. However, the surrogates defined here are used by proving that the user/presenter of the surrogate has knowledge of the private part that was used to generate the surrogate in a way which does not reveal the private part.

The surrogates in our protocol can be independently generated by the user and by the issuing authority, which is a partially trusted third party in the sense of section 2.2.1. However the issuing authority cannot masquerade as the user. Other parties (verifiers) can verify surrogates but cannot generate them. Note that the verifiers are not in general trusted by the user. The surrogates for a particular user are generated from the parent public key of that user. The private value corresponding to a surrogate is generated from the secret that was used to generate the parent public key. We generate DH keys for different example scenarios and then generate surrogates from the Diffie-Hellman keys.

The requirement for surrogates are:

1. Verifiability – The verifier, who might be the owner of the resource or an access granting service, should be able to unambiguously verify that the user presenting a surrogate is the legitimate owner of the surrogate.

2. Un-correlatability – It should be hard for an untrusted adversary to link actions to individuals retrospectively even if the adversary manages to obtain the surrogate used for the transaction along with the transaction details. For example if the adversary is a legitimate verifier. Someone observing the network should not be able to gain any information about the nature of communication and the communicating parties.

3. Misuse of surrogates – Even if an adversary manages to capture and retain a surrogate used for a particular transaction still it should be hard for him/her to generate and/or use future surrogates.

4. Protection from (partially) trusted third party – No adversary should be able

to pretend to be a legitimate owner of a surrogate. Although a third party generates and issues the information users need to generate and use their surrogates, this third party should not be able to masquerade as the legitimate owner of a surrogate.

5. Audit – An unbiased partially trusted auditor should be able to link actions verifiably and unforgeably to individuals, but only with the explicit knowledge and consent of the legitimate owner of the surrogate.

We allow sharing of surrogates (in particular to surrogate delegation) but learning them won't help the attacker in any way. Keys and surrogates used by principals in the protocols presented in this chapter depend on the difficulty to compute discrete logarithm. We show how our surrogates satisfy these requirements when we describe various example scenarios in the following chapters.

4.5 Generation of Surrogates

Suppose that X is a user's Diffie-Hellman public key and s is the secret key from which X is generated as shown in section 4.2, where P is a large prime and is publicly known.

A surrogate K_i^+ and its corresponding private value K_i^- are generated from a seed $\tau_i \in 1 \ldots (P-1)$ by the following equations:

$$K_i^- = \tau_i * s \bmod (P-1) \tag{4.1}$$

$$K_i^+ = X^{\tau_i} \bmod P \tag{4.2}$$

The important point to note is that, because

$$g^{P-1} \bmod P = 1 \tag{4.3}$$

we have

$$g^{\tau_i * s} = g^{K_i^-} \bmod P \tag{4.4}$$

and hence

$$K_i^+ = X^{\tau_i} = (g^s)^{\tau_i} = g^{K_i^-} \bmod P \tag{4.5}$$

The initial value τ_0 of the seed and the long term values A and L are supplied by a (partially) trusted third party (see section 2.2.1) such as the user's bank, and are secrets shared between the user and the partially trusted third party. The subsequent values (τ_i) of the seed, used to generate the surrogates and their corresponding private values, are generated using the linear congruence equation,

$$\tau_i = (A * \tau_{i-1} + L) \bmod P - 1 \tag{4.6}$$

Then, using τ_i, surrogate K_i^+ and the corresponding private value K_i^- for the i^{th} transaction are generated from equations 4.2 and 4.1 respectively.

To use a surrogate K_i^+ for transaction i a user proves knowledge of the corresponding K_i^-. A potential signing mechanism that can be used to prove knowledge of K_i^- is explained in section 4.5.1. Only the legitimate owner of X and s can generate and use surrogates corresponding to X. An adversary or the partially trusted third party cannot masquerade as the legitimate owner of X as the corresponding s is secret to the owner but a partially trusted third party which knows the seed τ_i can resolve disputes and can correlate transactions conducted with surrogates generated from X using equations 4.6 and 4.2, thus facilitating auditing. However various transactions conducted by the same user cannot be correlated with each other by an adversary at the point of use of the surrogates. Learning one set of K_i^+, K_i^- values does not help the attacker in any way, neither can the attacker generate the current K_i^+, K_i^- values, nor can the attacker generate future surrogates and their corresponding private values. Even if τ_i, A and L are compromised then the adversary might be able to correlate various surrogates belonging to a particular user but cannot masquerade as the user.

4.5.1 Signing using Surrogates

To use a surrogate K_i^+ for transaction i a user proves knowledge of the corresponding K_i^-. For every transaction users sign a transaction description T using

K_i^- and the signature can only be successfully verified using K_i^+. We describe a potential signature mechanism based on El Gamal signature [31] in this section. The user signs a hash of the message T [31] but for simplicity we use T here. The user signs T using K_i^- using El Gamal signature scheme and produces the signature (a, b) of T as:

$$a = g^k \bmod P \tag{4.7}$$

$$b = (T - K_i^- * a) * (k)^{-1} \bmod (P - 1) \tag{4.8}$$

where $k \in Z_{(P-1)}$ is secret and known only to the user. Since the signature on T can only be verified using K_i^+, the verification authority can be sure that the signature has been generated only by the legitimate owner of K_i^+ and no one else. The verification is done as follows:

$$g^T = a^b * K_i^{+a} \bmod P \tag{4.9}$$

The equation 4.9 can only be satisfied using K_i^+ if the signature (a, b) was generated using K_i^-.

4.6 Notation and Assumptions

In this section we outline the assumptions on which we build our new protocols. The assumptions and notation we outline in this section apply to all the protocols we present in chapters 5, 6 and 7 unless otherwise explicitly mentioned.

4.6.1 Assumptions

For our new protocols we assume that principals have a public key generated from a secret key using the conventional Diffie-Hellman mechanism as described in section 4.2. The association between a principal and its public key is known by a partially trusted third party *e.g.* the role server and can be implemented using some offline certification authority. Entities can verify the association between principals and their keys by fetching certificates and this verification is done with

the explicit knowledge and consent of the principal. Ownership of surrogates are verified and established by proving knowledge of the corresponding private value using the signing mechanism discussed in section 4.5.1.

Our protocols depend on certain properties about the underlying communication channels above which they operate. Since we use our protocols for secure web access, and for remote access to distributed objects and as our protocols also can potentially be used for commercial purposes, we choose to use Mixminion (see section 2.3.4), a secure anonymous communication channel. Mixminion uses TLS [61] over TCP [61] for link encryption between remailers and uses ephemeral keys to ensure forward anonymity for each message. Mixminion also supports replies by the form of SURBs and the replies cannot be correlated with the initial message. This enables two way communication between the communicating partners

We also assume the existence of a secure authenticated communication channel such as SSL/TLS. SSL/TLS allows for two way authentication, preserves data integrity and confidentiality and is widely used. SSL/TLS can also be used for the scenarios we describe here in this dissertation for *e.g.* remote web access, secure submission of personal information etc.

Based on the discussion about complexity of the discrete logarithm problem in section 4.3 for our protocols we choose a cyclic group for which computing the order is thought to be very difficult. If $(P-1)$ has small prime factors then computing discrete logarithm is easy [49]. For our protocols P is chosen such that $(P-1)$ has one large prime factor. This can be done by choosing a large prime q and selecting P as the smallest prime congruent to 1 mod q or by choosing P of the form $2q+1$ where P and q [23] are both prime.

4.6.2 Notation

- P denotes as a large prime with the properties mentioned in section 4.6.1. g is a generator modulo P (see section 4.3). P is public and so is g.

We are going to introduce named individuals and examples in the subsequent chapters since it aids the understanding. We use a subscript to the keys to denote the user as illustrated next.

- X_c represents the Diffie-Hellman public key of Carol. s_c represents the corresponding secret key of Carol. X_b represents the Diffie-Hellman public key of Bob. s_b represents the corresponding secret key of Bob.

- K_i^+ represents the surrogate used for the i^{th} transaction. K_i^- represents the private value corresponding to the surrogate K_i^+. Surrogates are always generated from the corresponding user's Diffie-Hellman public key unless otherwise explicitly stated.

- PK_c represents the RSA public key of Carol. SK_c represents the corresponding RSA private key (see section 2.5).

- $SIG(T)$ denotes a signature on message T using the mechanism described in section 4.5.1. Such signatures are always generated with the private value K_i^- corresponding to the surrogate K_i^+ used for the i^{th} transaction. $RSIG(T)$ denotes the ring signature (see section 2.6) on message T and such signatures are always generated by Carol using SK_c.

- τ_i denotes the seed used to generate surrogates and is calculated using the linear congruence equation 4.6.

- A and L are the parameters used in the linear congruence equation (see equation 4.6) and are global and fixed.

- $\longrightarrow_s$ stands for a secure mutually authenticated communication channel with properties similar to SSL/TLS throughout this dissertation. $\longrightarrow_m$ stands for an anonymous communication channel with properties similar to Mixmin-ion throughout this dissertation.

- Items within $\prec, \succ$ separated by commas denote individual elements of a shuffled message being transmitted and the braces denote the begining and end of message respectively. The elements of a shuffled message are actually sent in a random order not in the order in which they appear in the braces.

4.7 Conclusion

Surrogates provide a happy middle ground between absolute privacy and zero privacy. Since one of the interesting aspects of transactions is that we have no real control over the actions of the other party, so in order to achieve certainty of privacy we use unlinkability as our weapon. Surrogates provide the ability to control both the availability and linkability of transactions.

In the following chapters we demonstrate how to use our surrogates to address the problem of trust and privacy in various electronic services. In chapter 5 we integrate our surrogate based authentication mechanism with some prominent role based authorisation models which were discussed in section 3.2.3. In chapter 5 we also show how the powerful concept of activation of roles using prerequisites discussed in section 3.2.3 can be achieved using surrogates. Delegation of surrogates is discussed in chapter 6. We also show how surrogates can be integrated with payment mechanisms in chapter 7.

Chapter 5

Basic Scenarios

5.1 Introduction

We have already discussed some role based access models [6, 7, 35] where long term credentials like role certificates or key certificates are used for making authorization decisions. Applications such as [52] present long term credentials to a trust management engine [9] to check whether or not requested actions conform to local policies.

In this chapter we describe some example scenarios where long term credentials are being used at present for making authorization decisions. The mechanisms described in the following sections show alternative ways of making authorization decisions using the transient identities or surrogates described in section 4.4. Our mechanisms can coexist along with the other non-anonymous authorization mechanisms being used at present. The surrogates can be used to check compliance with local policies by a trust management engine. In our approach an issuing authority, *e.g.* the role server in a role based access mechanism, can communicate policies expressed in terms of roles pertaining to a surrogate or group of surrogates to a trust management engine situated at the point where the surrogates will be used, and then the trust management engine can make authorization decisions on requests from the surrogate owners. Our approach adds an extra layer of anonymity in the trust management engines which can coexist with current trust management architectures.

The first example scenario discussing auditable anonymous role activation in a role based access control system, is described in section 5.2. Section 5.3 gives an alternative mechanism for anonymous activation of roles in the case where auditability is not required. This is followed in section 5.4 by a mechanism for anonymous activation of roles with prerequisites which allow auditing. We conclude in section 5.5.

5.2 Scenario 1: Role Based Access Control with Fixed Roles

The role based authorisation model we adhere to in this example scenario was discussed in section 3.2.3. Suppose Carol is a subscriber of an electronic newspaper. The newspaper uses a role based authorization model where Carol is assigned the role of subscriber. The process is same for every subscriber. The subscription department (SA) prepares and sends the information Carol needs to generate her surrogates. At regular intervals the subscription department also sends to the role server (RS) the information needed to authenticate Carol and other subscribers. (SA) acts as the partially trusted third party (see section 2.2.1). Every time Carol wishes to read a newspaper she first authenticates herself to the local web server (WS) using her surrogate for the current transaction, and upon a successful authentication her role is activated. WS consults a local trust management engine before activating relevant permissions for Carol. SA can link Carol's transactions but WS cannot. For every different transaction Carol uses a different surrogate. The requirements which were identified in section 4.4 are as follows:

1. Verifiability
2. Un-correlatability
3. Misuse of surrogates
4. Protection from (partially) trusted third party (SA here)
5. Audit

The protocol has a preparation phase followed by a transaction phase.

5.2.1 Preparation Phase

Carol sends the subscription department (SA) her public X_c (see section 4.6) key via a secure, confidential and conventionally authenticated channel. Here we use a DH key system modulo a prime as described in section 4.2.

1. $Carol \longrightarrow_s SA : X_c = g^{s_c} \bmod P$

Carol uses her surrogates serially from 1. So that Carol does not need to contact the SA before every single transaction: SA sends Carol the information she needs to prepare subsequent surrogate pairs.

2. $SA \longrightarrow_s Carol : \delta = \prec \tau_0, L, A \succ$

For each subscriber including Carol, SA generates surrogates by repeating equations 4.6 and 4.2 of section 4.5 for successive values of the relevant τ_i (see section 4.6). SA then sends the web server (WS) a batch containing say m surrogates for Carol. Local policies, *e.g.* actions which can be performed by this batch of surrogates, are also sent by the SA to WS.

3. $SA \longrightarrow_s WS : \prec K_1^+, ..., K_m^+,$ surrogates for other users $\succ$

Surrogates of various different customers are sent mixed in a batch. At the end of this phase SA retains just the information (X_c, A, τ_i, L) (see section 4.6) for each subscriber.

5.2.2 Transaction Phase

For transaction i^{th} Carol calculates her i^{th} surrogate K_i^+ (see section 4.6) and its corresponding secret K_i^- using equations 4.6, 4.1 and 4.2 of section 4.5. To activate her role for her current transaction Carol sends the web server via a secure anonymous channel her current surrogate K_i^+ and signs a message (see section 4.5.1) T using the corresponding secret K_i^- (see section 4.6). The signature on T can only be verified using K_i^+; thus WS can be sure that Carol is the legitimate owner of K_i^+. If the web server finds a similar surrogate has been sent to it by

the SA then it grants Carol access. To be specific the WS activates relevant permissions for Carol. To ascertain the validity of Carol's request T the WS consults the local trust management engine.

4. $Carol \longrightarrow_m WS : K_i^+, SIG(T)$

The next time Carol reads the electronic newspaper she uses a different surrogate to authenticate herself. The web server WS can authenticate Carol anonymously without the issuing authority SA being online. The issuing authority SA can periodically send Carol's new surrogates to the web server WS. SA cannot masquerade as Carol to WS.

5.2.3 Analysis

The desired properties of our system set out in section 4.4 are achieved as follows:

1. Verifiability – In step 4, Carol sends a transaction description T signed using the mechanism discussed in section 4.5.1. The signature can only be successfully verified using K_i^+ if T was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user (Carol here) presenting K_i^+ is the legitimate owner of K_i^+.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or WS to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod P \tag{5.1}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving equation 7.11 is still thought to be an intractable problem. If an

adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K_{i+1}^{+} using equation 4.2. Thus it is difficult for an adversary or WS to link surrogates to each other or to the parent public key. Carol communicates with WS via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief.

3. Misuse of surrogates – We have discussed in section 4.5 that to use a surrogate K_i^{+} Carol has to prove that she knows the corresponding secret K_i^{-}.

   An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K_i^{-} without knowing s_c which is secret. So an adversary between Carol and the SA can never prove knowledge of K_i^{-}. Moreover calculating s_c requires the adversary to calculate

$$s_c = \log_g X_c \bmod P \tag{5.2}$$

   which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party – The third party cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to solve equation 7.12 which is widely thought (see section 4.3) to be hard. Moreover even if the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the subscription authority cannot masquerade as Carol to WS.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages to steal the surrogate belonging to a subscriber still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K_i^{-} using equation 4.1 as s_c is secret and calculating s_c requires

the adversary to solve equation 7.12 which is widely thought (see section 4.3) to be an intractable problem.

The subscription authority or SA initially issues the numbers the subscribers use to generate their surrogates in step 2 of the preparation phase and the subscription authority also sends WS the surrogates subscribers will use to activate their accounts during the preparation phase. SA can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but SA cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, SA can resolve disputes retrospectively. SA would divulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order.

5.3 Scenario 2: Activation of Roles Without Auditability

In example scenario 1 we described a mechanism for anonymous activation of roles using surrogates, where an auditor can figure out retrospectively who activated a particular role. In this section we present a mechanism for completely anonymous activation of roles where an auditor cannot figure out who activated a particular role. In our approach we view principals as members of a group which may be a subset of all the members of a particular role. To access a resource the requestor has to prove that he/she is a member of a group that is allowed access to the requested resource.

We have discussed distributed shared objects (DSO) in section 3.3. The protocol we present here can be used to allow users to anonymously authenticate to a globe replica. In this example a DSO models the electronic newspaper. In the original Globe proposal (see section 3.3.1) the DSO owner would create a role of subscriber and issue certificates to paid subscribers. The paid subscribers in turn would authenticate themselves to the replica using their certificate and access the newspaper.

A customer, Carol, registers with the subscription authority (SA) as a sub-

scriber. Like the previous scenario here (SA) acts as the partially trusted third party (see section 2.2.1). This registration requires Carol to prove that she owns the corresponding secret key. SA informs the DSO owner about its new subscriber Carol along with other new subscribers. Because we are not using surrogates in this protocol, the requirements of this protocol are different to the requirements we identified in section 4.4.

The requirements for this protocol are

1. Verifiability – The verifier, who might be the owner of the resource or an access granting service, should be able to unambiguously verify that the user is a member of the group.

2. Un-correlatability – It should be hard for an untrusted adversary to link actions to individuals retrospectively even if the adversary manages to obtain the surrogate used for the transaction along with the transaction details. For example if the adversary is an legitimate verifier. Someone observing the network should not be able to gain any information about the nature of communication and the communicating parties.

3. Unforgeability – It should be hard for an adversary to masquerade as a legitimate user of the system.

The protocol has a preparation phase followed by a transaction phase.

5.3.1 Preparation Phase

Carol sends SA her RSA public key PK_c (see section 4.6) and proves that she is the legitimate owner of the public key.

1. $Carol \longrightarrow_s SA : PK_c$

SA sends WS public keys of various subscribers including Carol's together in a batch along with the local policies for the group.

3. $SA \longrightarrow_s DSOowner : \prec PK_1, ..., PK_m \succ$

5.3.2 Transaction Phase

Whenever Carol wishes to activate her role to read the electronic newspaper server she has to activate according to policy her role of subscriber by authenticating to the security subobject of the DSO. Carol signs a message $[T]$ using ring signature (see section 2.6) and her own RSA public key PK_c and proves to the security subobject that she is a valid subscriber. The security subobject on consulting a local trust management engine activates relevant permissions for Carol.

1. $Carol \longrightarrow_m Securitysubobject : RSIG(T)$

where $RSIG(T)$ (see section 4.6) is the ring signature of Carol on message T. Following successful authentication using the ring signature Carol activates her role as a subscriber.

5.3.3 Analysis

The desired properties of our system set out in 5.3 are achieved as follows:

1. Verifiability – In step 1 Carol signs the transaction description T using ring signatures described in section 2.6. In the ring signature mechanism it is difficult for an outsider to masquerade as a member of a group (see section 2.6, thus the security subobject can be sure that the signer is a member of the group.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. The DSO replica cannot guess the subscriber from the ring signature with a probability more than $1/r$ where r is the total number of the possible signers. Carol communicates with the DSO via a secure anonymous communication channel denoted as $\longrightarrow_m$ (see section 4.6); thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief.

3. Unforgeability – In the ring signature mechanism it is difficult for an adversary to masquerade as Carol. To sign messages using ring signatures the

signer needs to invert a trap door oneway permutation which can only be done using SK_c (see section 4.6) where $(PK_c * SK_c) = 1 \bmod \phi(n)$. Only the legitimate owner of PK_c knows SK_c. Thus an adversary cannot invert the trapdoor oneway permutation corresponding to the public key of a legitimate subscriber. Since ring signature makes it difficult for an adversary to masquerade as a member of a group so the security subobject can be sure that it is talking to a genuine subscriber.

5.4 Scenario 3: Anonymous Activation of Roles with Prerequisites

Here we illustrate the use of a two level authentication mechanism, using both ring signature (see section 2.6) and surrogates. A two level authentication mechanism supports the concept of activate security [6] according to which access control decisions depend on context which is monitored *e.g.* the assignment of users to roles is handled by role activation rules which require users to activate roles on possession of valid prerequisites. If there is a change in the context in which a role was initially activated then the role is revoked. Such an approach helps to organize access control systems into an hierarchical structure.

We have already described OASIS in section 3.2.3. The mechanism we present in this section can be used for anonymous activation of roles using prerequisites in the OASIS role based authorisation system. In our example here the prerequisite for activation of a role is another role. The example we use here emulates access to a University Learning and Information Services (LIS) by students. On registration all students are assigned the role *student* and then they are assigned roles based on their program of study. For research students they are subsequently assigned the role of *research_student*. The role *research_student* can only be activated by a student if the student has already activated the role *student*. The LIS consults a trust management engine like Keynote [9] before granting specific permissions to a user.

Here students first authenticate as a member of the group of students to activate their prerequisite role of student and then use their surrogates to activate specific

roles according to their program of study. LIS is accessed by staff as well as students and the procedure to activate roles for everyone is same. All users are members of a group depending on their affiliation with the university say *student, staff etc.* In order to have a two level authentication we use RSA keys along with DH keys.

The transaction flow can be outlined as:

1. Carol is a research student and registers with the University learning and information services using her public key. She is assigned the role of research student by the role server (RS). (RS) acts as the partially trusted third party (see section 2.2.1). This registration requires Carol to prove that she owns the corresponding secret key. All the students of the University are assigned the role of student when they enroll.

2. Carol first activates her student role before she access the database of print and electronic journals provided by the learning and information services.

3. For activating her student role Carol proves she is a student without revealing her identity. She authenticates as a member of the group *students* using ring signature.

4. She activates her *research_student* role by presenting her surrogates to the LIS. The LIS server consults the local trust management engine to ascertain whether or not Carol is allowed to carry out the actions she requested. For every different transaction she uses a different surrogate pair.

5. In case of a dispute Carol's transactions can be linked by the registration authority. This helps in auditing. But an adversary at the point of use of the surrogates cannot link surrogates belonging to Carol.

The requirements which were identified in section 4.4 are as follows:

1. Verifiability

2. Un-correlatability

3. Misuse of surrogates

4. Protection from (partially) trusted third party (RS here)

5. Audit

The protocol is again described as a preparation phase followed by a transaction phase.

5.4.1 Preparation Phase

The preparation phase is similar to scenario 1 described in section 5.2. Like scenario 1 Carol sends both her DH and RSA public keys to the role server (RS) now instead of the SA and the role server like SA sends the LIS, m surrogates Carol will be using for m transactions. The process is similar for every subscriber. Since we use a two level authentication mechanism here RS also sends the RSA public key of each individual student to LIS. Like SA of scenario 1, RS here only retains (X, A, τ_i, L) for every student at the end of the preparation phase and sends LIS the surrogates of various students mixed together in a batch. $\prec PK_1, ..., PK_n \succ$ denotes the public keys of various students (say n) sent together in a batch.

1. $Carol \longrightarrow_s RS : X_c = g^{s_c} \bmod n, PK_c$

2. $RS \longrightarrow_s Carol : \delta = \prec \tau_0, L, A \succ$

3. $RS \longrightarrow_s LIS : \prec K_1^+, ..., K_m^+,$ surrogates for other students $\succ, \prec PK_1, ..., PK_n \succ$

5.4.2 Transaction Phase

For transaction i Carol calculates her i^{th} surrogate K_i^+ and its corresponding secret K_i^- using equations 4.6, 4.1 and 4.2 of section 4.5. To activate her role of *research_student* for her current transaction Carol first authenticates herself to LIS as a member of the group *students* using ring signature on message T to activate the prerequisite role *student* and then sends the LIS her current surrogate K_i^+ and like scenario 1 signs a message T' (see section 4.5.1) using the private part K_i^-

corresponding to her current surrogate K_i^+. If the LIS finds a similar surrogate sent to it by the role server then it grants Carol access. To be specific the LIS activates relevant permissions for Carol. To ascertain the validity of Carol's request the LIS consults the local trust management engine. LIS cannot correlate an instance of ring signature with the surrogate used by a student.

1. $Carol \longrightarrow_m LIS : T, RSIG(T)$
   where $RSIG(T)$ is Carol's signature on T using ring signature, and the LIS knows the RSA public key of the other members of the group.

2. $Carol \longrightarrow_m LIS : K_i^+, SIG(T')$

5.4.3 Analysis

The desired properties of our system set out in section 4.4 are achieved as follows:

1. Verifiability – In step 1 Carol signs the transaction description T using ring signatures described in section 2.6. In the ring signature mechanism it is difficult for an outsider to masquerade as a member of a group (see section 2.6, thus the LIS can be sure that the signer is a member of the group. In step 2, Carol sends a transaction description T' signed using the mechanism discussed in section 4.5.1. The signature can only be successfully verified using K_i^+ if T was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user (Carol here) presenting K_i^+ is the legitimate owner of K_i^+.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. The LIS cannot guess the subscriber from the ring signature with a probability more than $1/r$ where r is the total number of the possible signers. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or LIS to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod n \tag{5.3}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving equation 7.11 is still thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K^+_{i+1} using equation 4.2. Thus it is difficult for an adversary or LIS to link surrogates to each other or to the parent public key. Carol communicates with LIS via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief.

3. Misuse of surrogates – We have discussed in section 4.5 that to use a surrogate K^+_i Carol has to prove that she knows the corresponding secret K^-_i. An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K^-_i without knowing s_c which is secret. So an adversary between Carol and the RS can never prove knowledge of K^-_i. Moreover calculating s_c requires the adversary to calculate the equation 7.12, which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party – The third party cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to calculate the equation 7.12, which is widely thought (see section 4.3) to be hard. Moreover even if the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the subscription authority cannot masquerade as Carol to the trading server.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages to steal the surrogate belonging to a student still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K^-_i using equation 4.1 as s_c is secret and calculating s_c requires the ad-

versary to calculate the equation 7.12, which is widely thought (see section 4.3) to be an intractable problem.

The role server or RS initially issues the numbers the students use to generate their surrogates in step 2 and the role server also sends LIS the surrogates students will use to activate their accounts. RS can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but RS cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, RS can resolve disputes retrospectively. RS would divulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order.

5.5 Conclusions

RSA keys when used in conjunction with DH keys help us to have two level authentication. In DH key systems modulo a prime it is difficult to keep multiplicative inverses of public keys secret. When both the public key X_c, and the modulus P are public it is easy for anyone to calculate the inverse of X_c modulo P. RSA key systems modulo a composite can have public keys with an inverse and the inverse can be kept secret, because to calculate the inverse of a public key an adversary has to factorize a large number which is thought to be an intractable problem. A two level authentication mechanism helps us to achieve anonymous activation of roles with prerequisites as described in section 5.4 respectively. In the two level authentication mechanism described in this dissertation, users first authenticate as members of a group using ring signatures and then authenticate using surrogates. The two level authentication mechanism in scenario 5.4 allows us to have two level audit level mechanism with different trust assumptions for the internal and external auditor. The external auditors can attribute actions to a group where as only the internal auditor can tie actions to individual DH public keys. We develop this notion further in the context of delegation in the chapter 6.

The mechanisms described above shows the possibility of coexistence of trust management and anonymity systems. Applications can answer the question "Shall

we carry out this potentially dangerous action" without having too much information about the initiator of the said action in question. This leads to the possibility of an extra anonymity layer in the trust management systems *e.g.* Keynote [9], which can coexist with the non anonymous service. The surrogates of a user can be presented along with policies and request for the action by the user, to the trust management engine and thus policies can be enforced using surrogates.

The protocol for example scenario 2 does not make use of surrogates and thus it differs from other protocols. Here users prove membership to groups using ring signatures only. The limitation of such a scheme is that, by preventing an auditor from correlating actions belonging to a particular user, it does not facilitate fair dispute resolution. Protocols like scenarios 1 and 3 allow an auditor to figure out retrospectively who used a particular surrogate pair.

It is worth noting that identity resolution is local in our system. Only the RS can link a surrogate to its corresponding public key and an external auditor cannot. An external auditor can link a surrogate to its corresponding public key only with the help of the RS. We believe this is significant as users do not need to have trust in the honesty and competence of an unknown external auditor; users in our system trust only their local entities (*e.g.* the RS here) whom they know. However the external auditor can prove to a arbitrary third party that a member of the group activated his/role role.

The protocols we have presented here preserve the privacy of the user. For the protocol using two level authentication mechanism in scenario 3 we use ring signature, which is purely signer ambiguous and an adversary cannot determine the signer with a probability more than $(1/r)$. Linking surrogates with a public key, in any of the protocols, requires calculating discrete logarithm, which is an intractable problem. It is difficult for an outsider to masquerade as a legitimate member of a group in scenario 3 as that would involve factorizing large numbers which is thought to be an intractable problem.

Chapter 6

Delegation

6.1 Introduction

Traditionally correlation of transaction records and identity theft has been the primary concern for the designers of anonymity systems [63, 15, 64]. Such systems do not allow principals to share their credentials, which we believe prevent principals from delegating their credentials. However it is often a legitimate real world requirement that users are able to delegate their credentials in an auditable manner. The approach which we take here does not greatly restrict the choice of the delegation semantics which can be adopted, although for exposition we adhere to Crispo's [25] delegation model in this dissertation. We have already reviewed Crispo's delegation model in section 3.4.

Like our mechanisms discussed in chapter 5 we assume that principals have a public key generated from a secret using conventional Diffie-Hellman as discussed in section 4.2. The association between a principal and its public key can be verified by a partially trusted third party. Similar to chapter 5 we also assume the existence of a confidential anonymous communication channel and a secure authenticated communication channel for our protocols. Users in the new protocols presented in this chapter generate surrogates with properties discussed in section 4.4 using the method discussed in section 4.5 Using the approach we describe in this chapter users can delegate their surrogates without the delegatee being able to figure out the secret used to generate the surrogate or reuse the surrogates of the

delegator.

We start in section 6.2 with a mechanism allowing delegation where it is difficult for an auditor to link actions to individuals. In section 6.3 we present a mechanism for delegation where we use ring signatures and a two level authentication mechanism similar to scenario 3 described in section 5.4. The delegation protocol we describe in section 6.4 does not depend on ring signatures and has stronger properties than the one presented in section 6.3. We conclude in section 6.5.

6.2 Scenario 4: Fully Anonymous Delegation - 1

The role based authorisation model we use here has been discussed in section 3.2.3. The transaction flow for this protocol is similar to the one described in section 5.2, the difference being is that here it is hard for an auditor to figure out retrospectively who used a particular surrogate pair. This protocol uses a DH key system as discussed in section 4.2, and the users only use their surrogate to authenticate. Though here it will be hard to figure out whether it was the delegator or the delegatee who used a particular surrogate pair but the auditor can figure out the owner of the surrogate pair. The requirements which were identified in section 4.4 are as follows:

1. Verifiability
2. Un-correlatability
3. Misuse of surrogates
4. Protection from (partially) trusted third party (RS here)
5. Audit

The protocol has a preparation phase followed by a transaction phase.

6.2.1 Preparation Phase

Carol sends the role server (RS) her public key via a secure, confidential and conventionally authenticated channel. Here we use a DH key system modulo a prime as described in section 4.2.

1. $Carol \longrightarrow_s RS : X_c = g^{s_c} \bmod P$

Carol uses her surrogates serially from 1. So that Carol does not need to contact the RS before every single transaction: RS sends Carol the information she needs to prepare subsequent surrogate pairs.

2. $RS \longrightarrow_s Carol : \delta = \prec \tau_0, L, A \succ$

For each subscriber including Carol, RS generates surrogates by repeating equations 4.6 and 4.2 of section 4.5 for successive values of the relevant τ_i. RS then sends the bulletin board (BB) a batch containing say m surrogates for Carol. Local policies, *e.g.* actions which can be performed by this batch of surrogates, are also sent by the RS to BB.

3. $RS \longrightarrow_s BB : \prec K_1^+, ..., K_m^+,$ surrogates for other users $\succ$

Surrogates of various different customers are sent mixed in a batch. At the end of this phase RS retains just the information (X_c, A, τ_i, L) for each subscriber.

6.2.2 Transaction Phase

For transaction i^{th} Carol calculates her i^{th} surrogate K_i^+ and its corresponding secret K_i^- using equations 4.6, 4.1 and 4.2 of section 4.5. To activate her role for her current transaction Carol sends the bulletin board via a secure anonymous channel her current surrogate K_i^+ and signs a message (see section 4.5.1) T using the corresponding secret K_i^-. The signature on T can only be verified using K_i^+; thus BB can be sure that Carol is the legitimate owner of K_i^+. If the bulletin board finds a similar surrogate has been sent to it by the RS then it grants Carol access. To be specific the BB activates relevant permissions for Carol. To ascertain the

validity of Carol's request T the BB consults the local trust management engine. While Carol is away from her office for one day she can send her secretary the surrogate and its corresponding secret next in the series via a secure authenticated channel thus enabling her secretary to access the bulletin board on her behalf. Like Carol now her secretary can prove the knowledge of K_i^- but for the ith transaction only and learning one pair of surrogates wont help her secretary in any way.

4. $Carol \longrightarrow_s Secretary : K_i^-, K_i^+$

5. $Secretary \longrightarrow_m BB : K_i^+$, SIG(T)

6.2.3 Analysis

The desired properties of our system set out in section 4.4 are achieved as follows:

1. Verifiability – In step 5, Carol's secretary sends a transaction description T signed using the mechanism discussed in section 4.5.1. The signature can only be successfully verified using K_i^+ if T was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user presenting K_i^+ is the legitimate owner of K_i^+ or has been authorised by the owner of K_i^+ to use the surrogate.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or BB to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod P \tag{6.1}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving equation 7.11 is still thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K_{i+1}^+ using equation 4.2. Thus it is difficult for an

adversary or BB to link surrogates to each other or to the parent public key. The user communicates with BB via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief.

3. Misuse of surrogates – We have discussed in section 4.5 that to use a surrogate K_i^+ Carol has to prove that she knows the corresponding secret K_i^-.

   An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K_i^- without knowing s_c which is secret. So an adversary between Carol and the RS can never prove knowledge of K_i^-. Moreover calculating s_c requires the adversary to calculate

$$s_c = \log_g X_c \bmod P \tag{6.2}$$

   which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party – The third party cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to solve equation 7.12 which is widely thought (see section 4.3) to be hard. Although the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the subscription authority cannot masquerade as Carol to BB.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages to steal the surrogate belonging to a subscriber still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K_i^- using equation 4.1 as s_c is secret and calculating s_c requires the adversary to solve equation 7.12 which is widely thought (see section 4.3) to be an intractable problem.

The role server or RS initially issues the numbers the subscribers use to generate their surrogates in step 2 of the preparation phase and the role server also sends BB the surrogates subscribers will use to activate their accounts during the preparation phase. RS can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but RS cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, RS can resolve disputes retrospectively. RS would divulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order.

The drawback in this scheme is that although the RS can figure out the owner of a surrogate but it cannot uniquely identify who used a particular surrogate. We address this issue in section 6.4.

6.3 Scenario 5: Auditable Anonymous Delegation - 2

Let us assume that users in an organization are members of various groups, *e.g.* managers belong to a group labeled as *managers* where as secretaries belong to a group labeled as *secretaries*. Carol is a manager in an organization and she uses her surrogates to authenticate, and check or submit announcements in the company bulletin board once every day. The organization uses a role based authorization model where Carol is assigned the role of manager. When she is out of her office she can ask her secretary to login and check or post notices.

We propose a two layered authentication mechanism where an user first authenticates as a member of a group say *managers* or *secretaries* and then uses his/her surrogates to check or post notices in the bulletin board. We have briefly introduced the features of a two level authentication mechanism in section 5.4. We assume that Carol's secretary also calculates her own public private RSA key pair. Users authenticate as a member of a group using ring signatures as described in section 2.6. For example while Carol's secretary is authenticating as a member of the group *secretaries* then he/she does so using the public keys of the other

secretaries and his/her own private and public key pair.

The second layer or the authentication using surrogates helps the auditor to figure out the owner of a particular surrogate while the first layer or group authentication helps the auditor to figure out who used a particular surrogate pair. For example if Carol herself is using the bulletin board she first authenticates as a member of the group *managers* and then uses her surrogate pairs. The auditor, from the group Carol used to authenticate, can figure out that it was a manager who accessed the bulletin board and from the surrogate can figure out that the surrogates belong to Carol. If Carol's secretary is using the bulletin board with surrogates delegated to her by Carol then she first authenticates as a member of the group *secretaries* and then uses the surrogates delegated by Carol. Here the auditor, from the group a secretary uses to authenticate, can figure out that it was a secretary who accessed the bulletin board and from the surrogates can figure out whose secretary accessed the bulletin board. No user can authenticate as a member of a different group and this facilitates fair dispute resolution.

1. Carol is assigned the role of manager by the role server (RS). RS acts as a partially trusted third party and performs the same functions as RS of scenario 3 discussed in section 5.4.

2. RS prepares and sends the information Carol needs to generate her surrogates.

3. Carol generates the required number of surrogate pairs which depends on the number of days she will be out of office. We assume that her secretary will log in to the bulletin board once every day thus requiring one surrogate pair each day.

4. Carol passes the surrogates to her secretary. This transfer of credentials between Carol and her secretary is done via a secure authenticated communication channel thus enabling Carol to prove later on that it was her secretary who used the surrogates in cases of dispute.

5. While using the bulletin board Carol's secretary first authenticates as a member of the group *secretaries* using ring signatures with her RSA key pair.

6. Carol's secretary can only use the surrogates and she learns nothing about Carol's secret key s_c in the process.

7. The bulletin board maintains an audit record containing the surrogates and the name of the group the user belongs to.

8. The next time Carol uses the bulletin board she authenticates as before. Anyone who learns the value of a surrogate cannot do any harm to Carol.

The requirements which were identified in section 4.4 are as follows:

1. Verifiability

2. Un-correlatability

3. Misuse of surrogates

4. Protection from (partially) trusted third party (RS here)

5. Audit

The protocol is again described as a preparation phase followed by a transaction phase.

6.3.1 Preparation Phase

The preparation phase is similar to scenario 1 described in section 5.2. Like scenario 1 Carol sends both her DH and RSA public keys to the role server (RS) now instead of the SA and the role server like SA sends the BB, m surrogates Carol will be using for m transactions. The process is similar for every subscriber. Since we use a two level authentication mechanism here RS also sends the RSA public key of each individual student to BB. Like SA of scenario 1, RS here only retains (X, A, τ_i, L) for every subscriber at the end of the preparation phase and sends BB the surrogates of various subscribers mixed together in a batch. $\prec PK_1, ..., PK_n \succ$ denotes the public keys of various subscribers (say n) sent together in a batch.

1. $Carol \longrightarrow_s RS : X_c = g^{s_c} \bmod n, PK_c$

2. $RS \longrightarrow_s Carol : \delta =\prec \tau_0, L, A \succ$

3. $RS \longrightarrow_s BB :\prec K_1^+, ..., K_m^+,$ surrogates for other subscribers $\succ, \prec PK_1, ..., PK_n \succ$

6.3.2 Transaction Phase

For transaction i Carol calculates her i^{th} surrogate K_i^+ and its corresponding secret K_i^- using equations 4.6, 4.1 and 4.2 of section 4.5. To activate her role for her current transaction Carol first authenticates herself to BB as a member of the group *manager* using ring signature on message T to activate the prerequisite role and then sends the BB her current surrogate K_i^+ and like scenario 1 signs a message T' (see section 4.5.1) using the private part K_i^- corresponding to her current surrogate K_i^+. If the BB finds a similar surrogate sent to it by the role server then it grants Carol access. To be specific the BB activates relevant permissions for Carol. To ascertain the validity of Carol's request the BB consults the local trust management engine. BB cannot correlate an instance of ring signature with the surrogate used by a subscriber. While Carol is away from her office for one day she can send the secretary a surrogate pair next in the series via a secure authenticated channel thus enabling her secretary to access the bulletin board on her behalf. Now her secretary first authenticates as a member of the group *secretaries* using ring signatures with her own public key and public keys of other secretaries and then proves knowledge of K_i^-. Her secretary signs T using K_i^- and the signature can only be verified using K_i^+. Learning one pair of surrogates wont help her secretary in any way. The secretary cannot correlate other transactions of Carol.

4. $Carol \longrightarrow_s Secretary : K_i^-, K_i^+$

5. $Secretary \longrightarrow_m BB : RSIG(T), SIG(T')$

6.3.3 Analysis

The desired properties of our system set out in section 4.4 are achieved as follows:

1. Verifiability – In step 5 Carol's secretary signs the transaction description T using ring signatures described in section 2.6. In the ring signature mechanism it is difficult for an outsider to masquerade as a member of a group (see section 2.6, thus the BB can be sure that the signer is a member of the group. In step 5, the secretary also sends a transaction description T' signed using the mechanism discussed in section 4.5.1. The signature can only be successfully verified using K_i^+ if T was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user presenting K_i^+ is the legitimate owner of K_i^+ or has been authorised by the legitimate owner of K_i^+ to use the surrogate.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. The BB cannot guess the subscriber from the ring signature with a probability more than $1/r$ where r is the total number of the possible signers. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or BB to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod n \tag{6.3}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving equation 7.11 is still thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K_{i+1}^+ using equation 4.2. Thus it is difficult for an adversary or LIS to link surrogates to each other or to the parent public key. Carol communicates with BB via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the

communicating partners beyond its *a priori* belief.

3. Misuse of surrogates – We have discussed in section 4.5 that to use a surrogate K_i^+ Carol has to prove that she knows the corresponding secret K_i^-. An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K_i^- without knowing s_c which is secret. So an adversary between Carol and the RS can never prove knowledge of K_i^-. Moreover calculating s_c requires the adversary to calculate the equation 7.12, which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party – The third party cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to calculate the equation 7.12, which is widely thought (see section 4.3) to be hard. Moreover even if the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the subscription authority cannot masquerade as Carol to the trading server.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages to steal the surrogate belonging to a student still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K_i^- using equation 4.1 as s_c is secret and calculating s_c requires the adversary to calculate the equation 7.12, which is widely thought (see section 4.3) to be an intractable problem.

   The role server or RS initially issues the numbers the students use to generate their surrogates in step 2 and the role server also sends BB the surrogates subscribers will use to activate their accounts. RS can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but RS cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, RS can resolve disputes retrospectively. RS would di-

vulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order. The drawback of this scheme is that the secretary can further delegate Carol's surrogates without the explicit knowledge or consent of Carol. We address this issue in the following protocol.

6.4 Scenario 6: Auditable Anonymous Delegation - 3

There are several commercial websites where users can log in and watch movies by paying a monthly or yearly membership fee. Such a website will require a user to register with their credit/debit card number to become a member. The aim of registration is to produce an identification token that binds one of the user's conventional identities (credit card) to a bit pattern (login name) which uniquely identifies the user in the computer system [22]. So long as a member wishes to continue his/her subscription, his/her login name remains the same. Every time a member wishes to watch a movie they authenticate using the fixed login name and password. So long as a particular member account is valid and active there is no limit to the number of movies a member can watch.

However it is a legitimate real world requirement that adult members allow their children (below 18) to watch a movie using the credentials of their parents. We refer to the principal that delegates as the delegator, and the principal that acts using the credentials of the delegator is referred to as the delegatee. The problem in this scenario is that once a child has learnt the fixed login name and password of his parents he can reuse them at a later date without the explicit knowledge or consent of his parents. Using the following protocol users can share their credentials in such a way that it is difficult for the delegatee to reuse credentials of the delegator. This protocol is for authorization and not for payment. We discuss implications of our protocols in the design of payment mechanisms in chapter 7. The transaction flow is as follows:

1. Carol registers with the bank with her public key.

2. Payment is handled between the bank and the merchant. The bank prepares and sends the information Carol needs to generate her surrogates.

3. For the i^{th} transaction Carol prepares her i^{th} surrogate and sends to Bob in such a way that only Bob can use it.

The requirements which were identified in section 4.4 are as follows:

1. Verifiability

2. Un-correlatability

3. Misuse of surrogates

4. Protection from (partially) trusted third party (RS here)

5. Audit

6.4.1 Message Exchanges

Carol generates her public key X_c using equation ?? from her secret s_c and similarly Bob generates his public key X_b (see section 4.6) as:

$$X_b = g^{s_b} \bmod P \tag{6.4}$$

where Bob's secret key is s_b.

Carol registers with the bank using her public key and the bank sends her the information she needs to generate her surrogates.

1. $Carol \longrightarrow_s bank : X_c$

2. $bank \longrightarrow_s Carol : \prec \tau_0, L, A \succ$

On receipt of Carol's public key the bank selects $\tau_0, L \in Z_P^*$ as described in section 4.5. So that Carol does not need to contact the bank before every single transaction, the bank sends Carol the information she will need to prepare m surrogate pairs using the equations 4.6, 4.1 and 4.2.

Carol uses her surrogates serially from $1...m$.

For each member, the bank sends the merchant m surrogates by repeating equations 4.6 and 4.2 for each of the next m values of i. Surrogates of various different customers are sent together in a batch as,

3. $bank \longrightarrow_s merchant : \prec K^+_{c(i...m)} ... K^+_{f(i...m)} \succ$

where $K^+_{c(i...m)}$ denotes surrogates belonging to Carol and $K^+_{f(i..m)}$ denotes surrogates belonging to some other customer. In the rest of the paper we refer to Carol's i^{th} surrogate as K^+_i and the corresponding secret as K^-_i.

For the i^{th} transaction Carol generates the exponent τ_i, the secret corresponding to the i^{th} surrogate K^-_i and her i^{th} surrogate K^+_i using equations 4.6, 4.1 and 4.2 respectively. Carol selects a random value $r \in Z_P$ and blinds Bob's public key as

$$(X_b)^r = (g^{s_b})^r \bmod P \tag{6.5}$$

Carol signs $M = (X_b)^r$ using K^-_i using the signature scheme discussed in section 4.5.1 and produces the signature $SIG(M)$. Carol sends to Bob via a secure anonymous channel the following:

4. $Carol \longrightarrow_m Bob : r, K^+_i, SIG(M)$

Bob generates M by repeating equation 6.5 with the r he receives from Carol and verifies that M is being signed by someone who knows the secret corresponding to K^+_i using the method discussed in section 4.5.1. Thus Bob can also be sure that Carol has signed his blinded public key and not something else.

Bob generates his secret corresponding to his blinded public key as:

$$s'_b = s_b * r \bmod (P-1) \tag{6.6}$$

and generates the signature a', b' of transaction description T using the method discussed in section 4.5.1. Bob sends the merchant via a secure anonymous communication channel

5. $Bob \longrightarrow_m Merchant$: $M, SIG(M) K^+_i, T, SIG(T)$

where M is Bob's blinded public key.

The merchant has access to the list of surrogates the bank had sent in step 3. This can be thought of as similar to a revocation list proposed in public key infrastructures. Thus it is difficult for an someone who is not a customer of the bank to forge a transaction with random Diffie-Hellman public keys. The merchant verifies that M is being signed someone who knows the secret corresponding to K_i^+ using the verification method discussed in section 4.5.1. Only when Bob proves knowledge of the secret s_b corresponding to M does the merchant accept the fact that Bob has been authorised to use K_i^+ by the legitimate owner of K_i^+.

6.4.2 Analysis

The desired properties of our system set out in section 4.4 are achieved as follows:

1. Verifiability – In step 5, Bob sends a transaction description T signed using the mechanism discussed in section 4.5.1. The signature can only be successfully verified using M if T was signed using s_b'. s_b' can only be successfully generated by Bob using equation 4.1 as s_b is known to only Bob. Thus the verifier can be sure that the user presenting M is the legitimate owner of M or has been authorised by the owner of K_i^+ to use the surrogate.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary to solve the equation

$$\log_{X_c} K_i^+ = \tau_i \bmod P \tag{6.7}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. The values A and L are shared between Carol and the bank. It is difficult for someone other than the bank to calculate τ_i by solving equation 7.11 is thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 and the surrogate using

equation 4.2. Thus it is difficult for an adversary to link surrogates to each other or to the parent public key. An auditor can only link surrogates back to the parent public key with the explicit knowledge and consent of the legitimate owner of the surrogates.

3. Misuse of surrogates – We have discussed in section 4.5 that to use a surrogate K_i^+ Carol has to prove that she knows the corresponding secret K_i^-. An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K_i^- without knowing s_c which is secret. So an adversary between Carol and the bank can never prove knowledge of K_i^-. Moreover calculating s_c requires the adversary to calculate

$$s_c = \log_g X_c \bmod P \tag{6.8}$$

which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) Trusted Third Party – The (partially) trusted third party cannot masquerade as the legitimate owner of X_c as the corresponding s_c is secret. Calculating s_c requires an adversary to solve the equation

$$\log_g X_c = s_c \bmod P \tag{6.9}$$

which is thought to be hard. Moreover even if the value of some exponent τ_i is known, still without the knowledge of s_c it is hard to generate the secret K_i^- corresponding to a surrogate K_i^+ using equation 4.1. Thus the bank cannot masquerade as Carol to Bob.

5. Audit – The bank can generate the surrogates using equations 4.6 and 4.2 and in cases of dispute involving a surrogate the bank can determine the owner of a surrogate; thus an auditor with appropriate authority can uniquely and irrefutably link actions to principals. Although Carol can decide who can use the surrogate, she herself cannot use the surrogate she has delegated and pretend that it was actually used by the delegatee, as s_b is secret.

If K_i^+ is not delegated then Carol will use it by signing T with K_i^-. It is in Carol's interest to delegate by signing the blinded public key of Bob

else Bob can frame Carol or further delegate Carol's surrogate. If a signed blinded public key can be verified using K_i^+ then an auditor can be sure that Carol signed it as K_i^- is only known to Carol. Carol cannot masquerade as Bob since s_b is secret and known only to Bob. Bob also cannot masquerade as Carol as K_i^- is secret and known only to Carol.

Thus it is difficult for Carol to frame Bob or for Bob to frame Carol. Although an auditor can link actions to principals still it cannot forge audit records as it cannot generate Carol's or Bob's signature in steps 4 or 5 respectively.

In the protocol described in this paper, identity resolution is local; neither the bank nor the merchant need to know Bob's identity. There can also be two different auditors, one who links the surrogate back to Carol while the other can prove that it was Bob who used it. This is significant because all trust is now local and Bob does not need to trust the bank or the merchant with personal information in order to use the service of the merchant. This has implications in the design of role based authorisation systems: using mechanisms like ours users can activate roles across domains without revealing personal information and auditors can still link actions back to the original user. An external auditor can link actions back to originating domain but linking the individual user requires the co-operation of an auditor trusted locally. Thus users have control over their personal information without compromising the goals of audit and authorisation.

6.5 Conclusions

Contrary to the previous approaches mentioned under section 2.4 we show here that one can transfer credentials in the anonymous world without revealing the secret from which the credential was generated. Transferability in turn allows delegation. We can have both auditable and non auditable transfer of credentials in the anonymous world using our approaches. In the protocols presented in sections 6.2 and 6.3 although it is possible for an auditor to detect the owner of a surrogate retrospectively it is hard to determine who used a surrogate. Moreover once

the surrogate has been delegated Carol can still use the surrogate and frame her secretary in the protocol discussed in sections 6.2. Moreover Carol cannot control who can use the surrogate and her secretary can well share Carol's surrogate with others.

In the protocol discussed in section 6.4 Carol delegates her surrogate in such a way that only Bob can use it. If K_i^+ is not delegated then Carol will use it by signing T with K_i^-. It is in Carol's interest to delegate by correctly signing the blinded public key of Bob else Bob can frame Carol or further delegate Carol's surrogate. If a signed blinded public key can be verified using K_i^+ then an auditor can be sure that Carol signed it as K_i^- is only known to Carol. Carol cannot masquerade as Bob since s_b' is secret. Bob also cannot masquerade as Carol as K_i^- is secret and known only to Carol. Thus it is difficult for Carol to frame Bob or for Bob to frame Carol. By signing the blinded public key of Bob Carol can specify that only Bob can use the surrogate; thus preventing Bob from further delegating Carol's surrogate. To further delegate Carol's surrogate Bob needs to share his secret key.

For the protocols discussed in this chapter the bank can generate the surrogates using equations 4.6 and 4.2 and in cases dispute involving a surrogate the bank can detect the owner of a surrogate, but cannot masquerade as the legitimate owner of X_c as the corresponding s_c is secret. Calculating s_c requires an adversary to solve the equation

$$\log_g X_c = s_c \bmod P \tag{6.10}$$

which is thought to be hard. Moreover even if the value of the exponent τ_i is known still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the bank cannot masquerade as Carol to Bob or to Carol's secretary.

In the protocol described in section 6.4 identity resolution is local, neither the bank nor the merchant need to know Bob's identity. There can also be different auditors one who links the surrogate back to Carol while the other can prove that it was Bob who used it. This is significant because all trust is now local and Bob does not need to trust the bank or the merchant with personal information although

Bob can use the service of the merchant. This has implications in the design of role based authorization systems. Users can activate roles across domains without revealing personal information and an auditor can still link actions back to the original user. An external auditor can link actions back to the user only with the co-operation of an auditor trusted locally. Thus users have control over their personal information without compromising the goals of audit and authorization.

Chapter 7

Implications And Extensions

7.1 Introduction

Although this dissertation has been primarily concerned with access control our work has implications in other areas such as electronic payment, price discrimination, licensing enforcement. If users access a service by using their surrogates then to pay for the service there should be ways of tying the payment tokens to their corresponding surrogates for auditing and accounting purposes. Similarly for price discrimination if a buyer can prove using his/her surrogate which price band he/she belongs to, and the buyer is the legitimate owner of the surrogate, then price discrimination can be done with online transient identities, thus protecting offline identities. In this chapter we present some approaches where we tie payment tokens to surrogates, and show a way of charging different prices to different customers using transient identities. We assume as usual the existence of a secure anonymous channel, a secure, confidential authenticated communication channel, and prime numbers with the same properties as in the previous chapter, for the examples we present in this chapter.

We give a description of how we generate payment tokens in section 7.2. In section 7.3 we show a way for charging different prices to different users which is followed in section 7.4 by a protocol which ties payment tokens to our surrogates. In section 7.6 we present a mechanism which allows sharing of payment tokens between principals which is followed by conclusions in section 7.7. Prin-

cipals in our protocols generate surrogates as described in section 4.5 and use either conventional Diffie-Hellman keys as described in section 4.3 or generate keys as described in ??. We do not advocate a particular payment mechanism in this dissertation. Other existing payment mechanisms can be combined with our surrogates using the protocols similar to those we present in this chapter

Our protocols depend on certain properties about the underlying communication channel above which they operate. We assume the existence of a secure anonymous communication channel like Mixminion and a secure authenticated communication channel like SSL/TLS, both with the properties discussed in 5.

We start with a description of the method we use to generate payment tokens in section 7.2. The first scenario deals with price discrimination and a protocol which can be used for charging different prices to different customers is described in section 7.3. We show a way of tying the payment tokens to the surrogates in section 7.4 which is followed by a protocol using which one can share payment tokens in section 7.6. We conclude this chapter in section 7.7.

7.2 Pseudonymous Payment Tokens

Here we give a brief overview of how we generate and use payment tokens in the following examples. Our payment tokens build on the Information checking protocol proposed in [53]. The bank in our protocol randomly selects three numbers S_0, B_0 and $Y_0 \in Z_P^*$ (where P is a large prime) and generates the subsequent S_i, B_i and Y_i by exponentiating S_0, B_0 and Y_0 respectively using an exponent σ_i. The initial value of σ, σ_0 is selected by the bank. The subsequent values of the exponent σ used to generate the surrogates and their corresponding secret are generated, by the user, using the linear congruence equation as,

$$\sigma_i = (A * \sigma_{i-1} + L_\sigma) \bmod (P - 1) \tag{7.1}$$

where A and L_σ are selected by the bank and are fixed. Then, using σ_i the subsequent S_i, B_i and Y_i values are generated as:

$$S_i = S_0^{\sigma_i} \bmod P \tag{7.2}$$

$$B_i = B_0^{\sigma_i} \bmod P \tag{7.3}$$

$$Y_i = Y_0^{\sigma_i} \bmod P \tag{7.4}$$

Finally using the following equation we calculate C_i using the S_i, B_i and Y_i values.

$$C_i = S_i + B_i * Y_i \bmod P \tag{7.5}$$

The payment token that the customer presents consists of the B_i and C_i corresponding to a particular S_i and Y_i value. The bank sends the customer B_0, and the information to generate subsequent B_i values. So that the customer does not have to contact the bank before every transaction the bank prepares and sends the customer m, C values prepared using equations 7.1, 7.2, 7.3, 7.4 and 7.5 and sends the customer m C values and B_0 value along with A and L_σ values. S, B, Y and C values are used serially from $1..m$. The bank retains S_0, Y_0, A and L_σ values. For transaction i the customer calculates B_i values using the information sent by the bank and the bank calculates the corresponding S_i and Y_i values. The customer presents the bank the corresponding B_i value it calculates and the C_i value sent by the bank. The bank verifies equation 5 using, the B and C value sent by the customer and the S and Y value it calculates. If equation 7.5 is satisfied then the bank accepts the payment tokens B_i and C_i as valid. It is hard for an adversary who does not posses the correct B_0, A and L_σ values to generate B and C values corresponding to a particular S and Y value, thus making stealing of payment tokens difficult.

7.3 Scenario 7: Price Discrimination using On Line Identities

7.3.1 Motivation

The issue of deployment and use of anonymizing technologies is analogous to a non zero sum game where the players' (merchants and users) interests are not always in direct conflict to each other but there are opportunities for both to gain. Both the merchants and the customers have incentives to support techniques (*e.g.* differential pricing) that facilitate optimal resource allocation. In games like this where the players have common interests it is unlikely that there will be any deployment and use of anonymizing technologies, unless there is an agreement between merchants and the users [2]. A model where everybody (merchants as well as the user) has the incentive to cooperate is more likely to be accepted [2]. The merchants need to cooperate because deployment of fully anonymizing techniques involves a switching cost on the part of the merchant.

The users and the merchants both have incentives in the deployment and use of medium anonymity systems. Medium anonymity systems support techniques (price discrimination) that makes the economy stronger without violating the privacy of the customer. Merchants can also increase their customer base by offering privacy conscious customers the use of anonymizing technologies. It has been argued in [65] that when merchants face privacy conscious customers they will adopt measures to protect the privacy of the customer. Its been reported in [3, 13] that if sellers share information about taste and the buying habits of customers, then "market laws alone might produce pareto-optimal outcomes" [2]. While [3] states that the buyer can benefit from the seller knowing him better which is good for the society, [36] concludes that the use of pseudonyms helps both the society and the individual. So one can say based on these studies [3, 36, 65] that differential pricing based on online identities can be effective. The merchants as well as the users both have incentives to cooperate in the deployment and use of such anonymizing technologies. We talk very briefly about some anonymizing technologies that permit differential pricing in the next section.

Why settling for medium anonymity systems is good for both the merchants

and the sellers can be more explicitly formalized through the principle of sub optimization:

> "Local optimization in general does not lead to global optimization." [40]

We use the term local optimization in this context to mean procedures or designs intended to provide absolute anonymity *i.e.* systems which does not allow a third party to correlate transactions belonging to any particular user, under any condition. By global optimization we mean a globally optimal cooperative arrangement *i.e.* optimal allocation of resources. Since differential pricing facilitates an optimal allocation of resources [47] so widespread use of technologies that do not facilitate price discrimination might create a less efficient economy. Thus the sub optimizing decision (absolute anonymity) is inconsistent with the globally optimizing one (efficient economy).

7.3.2 Transaction Flow

In our protocol a principal (Carol) proves they are entitled to a student discount and then pays for her rail ticket using payment tokens. The amount is eventually charged Carol's credit card. The payment tokens are different for every transaction and various payment tokens cannot be linked either to each other or to the user except by the credit card company. Before the protocol run there is a preparation phase where:

1. Carol registers with a Local Education Authority (LEA) with her public key as a student. This registration requires Carol to provide supporting paper documents the details of which are outside the scope of this paper.

2. The customer Carol requests payment tokens from her bank. The bank prepares and sends the information Carol needs to generate her payment tokens. These payment tokens are not specific to any particular product. These payment tokens can be used for purchasing various products.

During the protocol run:

1. Carol authenticates to the rail company that she is entitled to a students' discount.

2. Carol generates the payment tokens.

3. Carol pays for her tickets with the payment tokens.

4. The payment is authorized similar to the way credit cards are authorized at present. The authorization decision is communicated to the seller.

Every time Carol authenticates to the rail company that she is entitled to a students' discount she doesn't need to give any personal sensitive information to the vendor. The payment tokens cannot be linked to Carol by the vendor. The guard in the train can verify Carol's entitlement to her ticket without Carol disclosing personal sensitive information. The requirements which were identified in section 4.4 are as follows:

1. Verifiability

2. Un-correlatability

3. Misuse of surrogates

4. Protection from (partially) trusted third party (RS here)

5. Audit

7.3.3 Preparation

This phase is not specific to the rail tickets application. Registering with the Local Education Authority (LEA) is required for other situations (for example for a waiver of council tax from the county council) and the payment tokens issued by the bank can be used to buy other goods also.

Registering with the LEA

Carol sends her RSA public key PK_c to the LEA, using a secure, conventionally authenticated channel.

1. $Carol \longrightarrow_s LEA : PK_c$

Once Carol proves that she is the legitimate owner of PK_c, PK_c is added to the list of valid public keys of students. Under normal circumstances Carol is known as PK_c in the public list maintained by the LEA so long as she remains a student. Carol also registers with the bank with her public key PK_c. Carol will have surrogate pairs K_i^+ and K_i^-, generated from X_c as described in section 4.5, to support applications where Carol needs to prove that she has a valid bank account. Two different surrogates of Carol cannot be linked to Carol without the active cooperation of the bank. It is hard for an adversary to deduce X_c from K_i^+.

Registering with the Bank

Carol registers with the bank using her DH public key X_c.

1. $Carol \longrightarrow_s Bank : X_c = g^{s_c} \bmod P$

So that Carol doesn't need to communicate with the bank for every single transaction the bank prepares m payment tokens using equations 7.1, 7.2, 7.3, 7.4 and 7.5 It sends customer only the m C_i values along with the information Carol needs to prepare the corresponding B_i values. Only the bank can generate the S_i and Y_i values. The bank sends Carol:

2. $Bank \longrightarrow_s Carol : \Delta = \prec C_{1..m}, B_0, \sigma_0, L_\sigma, A, P \succ$

The bank retains the B_i value Carol will be using for her first transaction while issuing Δ The bank always calculates the B value the customer will be using for his/her next transaction subsequently after the first transaction. When the seller submits the customer information for authorization to the bank, the bank locates the proper customer information using the B value.

At the end of this phase the bank does not retain all the S, Y and B values but only the initial generators S_0, B_0, Y_0, along with σ_0, the offset L_σ, and the constant A to generate subsequent σ_i values. The bank retains the information Γ

$= \prec K_b^+, S_0, B_0, Y_0, L_\sigma, B_1, A, \sigma_0, X, P \succ$
Carol shares a secret key with the bank S_{cb}.

7.3.4 The Protocol

For transaction i the exponent σ_i is calculated using equation 7.1 and then B_i is calculated using equation σ_i by equation 7.3.

Carol creates a message M containing journey details and indicates that she is a student and wants a discounted price. Carol signs M using ring signature with X and H along with the public keys of the other members of the ring. H is used to invert the trapdoor oneway permutation described in step 5 of section 2.6.1. Carol send the seller

1. $Carol \longrightarrow_m Seller : \prec RSIG(T) \succ$

where P_i are the public keys of the other ring members. The seller verifies the signature as mentioned in section 2.6.2 and if the ring equation is satisfied then the seller is happy to offer a students' discount to Carol. In this process the seller has no idea regarding the identity of the signer. The seller signs a transaction description T and sends that to Carol

2. $Seller \longrightarrow_m Carol : [T']_{K^-_{seller}}$

If Carol is happy with T' and wants to pay she sends the calculated B_i value and the corresponding C_i value to the seller along with T'. T' is being encrypted using the secret key Carol shares with the bank and only the bank can decrypt T'.

3. $Carol \longrightarrow_m Seller : \prec K_i^+, SIG(T'), B_i, C_i \succ$

T' is encrypted to prevent the seller from changing the values of T'. The seller contacts the bank, the bank locates the appropriate customer with the help of the B value. We have mentioned in the preparation phase that the bank calculates in advance the B value a particular customer will be using next. This is done by the bank at the end of each transaction. For the first transaction the bank retains the first B value while issuing the tokens. The bank calculates the corresponding S and Y values from Δ and solves the equation $C_i = S_i + B_i * Y_i$ for transaction

i. If the equation is satisfied by the B and C values presented by the customer the bank agrees to the transaction. This process ensures that the seller receives his payment.

4. $Seller \longrightarrow_s Bank : \prec [T']_{S_{cb}}, B_i, C_i \succ$

5. $Bank \longrightarrow_s Seller : \prec T', AuthorizationDecision \succ$

The seller can be sure that the customer didn't change the value of T' when it gets T' back from the bank in the last step. The bank then calculates the B value the customer will be using for the next transaction using equation 7.1 and 7.3.

7.3.5 Analysis

The desired properties of our protocol set out in section 4.4 are achieved as follows:

1. Verifiability – If equation 7.5 is satisfied then the bank accepts the payment tokens B_i and C_i as valid. It is hard for an adversary who does not posses the correct B_0, A and L_σ values to generate B and C values corresponding to a particular S and Y value.

   The signature can only be successfully verified using K_i^+ if T' was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user (Carol here) presenting K_i^+ is the legitimate owner of K_i^+.

   In the ring signature mechanism it is difficult for an outsider to masquerade as a member of a group (see section 2.6, thus the verifier can be sure that the signer is a member of the group.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or the merchant to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod P \tag{7.6}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving equation 7.11 is still thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K^+_{i+1} using equation 4.2. Thus it is difficult for an adversary or the merchant to link surrogates to each other or to the parent public key. Carol communicates with merchant via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief. The merchant cannot guess the subscriber from the ring signature with a probability more than $1/r$ where r is the total number of the possible signers. Linking the payment tokens B_i and C_i with B_{i+1} and C_{i+1} also requires the adversary to solve the discrete logarithm problem which is thought to be intractable (see section 4.3).

3. Misuse of surrogates We have discussed in section 4.5 that to use a surrogate K^+_i Carol has to prove that she knows the corresponding secret K^-_i.

   An adversary stealing the numbers in step 2 of the registration process with the bank cannot masquerade as Carol as he/she cannot generate K^-_i without knowing s_c which is secret. So an adversary between Carol and the bank can never prove knowledge of K^-_i. Moreover calculating s_c requires the adversary to calculate

$$s_c = \log_g X_c \bmod P \tag{7.7}$$

   which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party (bank here)– The third party cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to solve equation 7.12 which is widely thought (see section 4.3) to be hard. Moreover even if the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the bank

cannot masquerade as Carol to the merchant.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages to steal the surrogate belonging to a subscriber still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K_i^- using equation 4.1 as s_c is secret and calculating s_c requires the adversary to solve equation 7.12 which is widely thought (see section 4.3) to be an intractable problem.

The bank initially issues the numbers the subscribers use to generate their surrogates in step 2 of the preparation phase and the bank also sends the merchant the surrogates subscribers will use to activate their accounts during the preparation phase. The bank can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but it cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, the bank can resolve disputes retrospectively. The bank would divulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order.

7.4 Scenario 8: Tying of Payment Tokens to Surrogates

Now we present a protocol where we tie payment tokens with surrogates. Users use conventional DH keys as described in section 4.3 and generate surrogates as described in section 4.5. The transaction flow is outlined by means of an example as:

1. The customer Carol requests payment tokens from her bank.

2. The bank prepares and sends the information Carol needs to generate her payment tokens and surrogates.

3. Carol goes to a website selling goods she wants to purchase.

4. Carol generates the payment tokens and surrogates.

5. The seller authenticates locally whether or not Carol is the legitimate owner of the surrogate.

6. Carol pays with her payment tokens which the seller validates with the bank.

The next time Carol goes to shop with the same seller she uses different surrogates and payment tokens which can be verified as before but cannot be correlated with a previous surrogate or payment token. Our motivation has been that Carol trusts her bank which is quite a practical thing to do. The requirements which were identified in section 4.4 are as follows:

1. Verifiability

2. Un-correlatability

3. Misuse of surrogates

4. Protection from (partially) trusted third party (RS here)

5. Audit

7.4.1 The Protocol

There are three parties in the protocol, the bank, the customer and the seller. The bank is online. The customer first sends the bank its public key via a secure, conventionally authenticated communication channel.

1. $Customer \longrightarrow_s Bank : X_c = g^{s_c}$

So that the customer doesn't need to communicate with the bank for every single transaction the bank prepares m C values using equations 7.1, 7.2, 7.3, 7.4 and 7.5. The bank sends the information Carol needs to prepare the corresponding B values and the surrogates. Only the bank can generate the S and Y values.

2. $Bank \longrightarrow_s Customer : \Delta = \prec C_{1..m}, B_0, \sigma_0, \tau_0, L_\sigma, L, A, P \succ$

During the preparation phase the bank also generates the surrogate K_1^+, as shown in 4.5, the customer will be using in its first transaction. We will see later that the bank always stores the surrogate the customer will be using next. When the seller submits the customer information for authorization to the bank the bank locates the proper customer information using this surrogate.

At the end of this phase the bank does not retain all the S, Y and B values but only the initial generators τ_0 and σ_1 values, and the offsets and the constant A to generate subsequent σ_i and τ_i values. The bank retains:

$$\Gamma = \prec S_0, B_0, Y_0, L_\sigma, L, A, K_1^+, \sigma_1, \tau_0, X, P \succ$$

The K_1^+, σ_1, and τ_0 values in the bank's record will change after every transaction.

For transaction i the customer calculates its ith surrogate K_i^+ using equations 4.6, 4.1 and 4.2. The corresponding secret K_i^- of the current surrogate is also generated by the customer. The exponent σ_i used to generate the B_i for the current transactions is calculated by the customer using equation 7.1 and the information provided by the bank, and the exponent used for generating B_{i-1} (except for B_1). Then the B_i value is calculated using equation 7.3. The seller generates a transaction description T and sends the customer signed with the seller's private key K_{seller}^-. T is agreed between both the seller and the customer. If the customer wants to pay the customer authenticates to the seller that it is the legitimate owner of the current surrogate. The customer sends the seller the ith payment tokens and the current surrogate.

3. $Seller \longrightarrow_m Customer : [T]_{K_{seller}^-}$

4. $Customer \longrightarrow_m Seller : SIG(T), K_i^+, B_i, C_i$

The seller contacts the bank, the bank locates the appropriate customer with the help of the surrogate, as the bank calculates in advance the surrogate a particular customer will be using next. This is done by the bank at the end of every transaction. For the first transaction the bank retains the value of the first surrogate while

issuing the tokens. The bank calculates the corresponding S and Y value and solves equation 7.5. If the equation is satisfied by the B and C values presented by the customer the bank agrees to the transaction. This process ensures that the seller receives his payment.

5. $Seller \longrightarrow_s Bank : \prec K_i^+, SIG(T), B_i, C_i \succ$

6. $Bank \longrightarrow_s Seller : \prec T, authorizationDecision \succ$

The bank, after every transaction, calculates the surrogate the customer will be using for the next transaction using the method shown in 4.5.

7.5 Analysis

The properties of our protocol set out in section **??** are achieved as follows:

1. Verifiability – If equation 7.5 is satisfied then the bank accepts the payment tokens B_i and C_i as valid. It is hard for an adversary who does not posses the correct B_0, A and L_σ values to generate B and C values corresponding to a particular S and Y value.

   The signature can only be successfully verified using K_i^+ if T was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user (Carol here) presenting K_i^+ is the legitimate owner of K_i^+.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or the merchant to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod P \tag{7.8}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving

equation 7.11 is still thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K_{i+1}^+ using equation 4.2. Thus it is difficult for an adversary or the merchant to link surrogates to each other or to the parent public key. Carol communicates with merchant via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief. Linking the payment tokens B_i and C_i with B_{i+1} and C_{i+1} also requires the adversary to solve the discrete logarithm problem which is thought to be intractable (see section 4.3).

3. Misuse of surrogates We have discussed in section 4.5 that to use a surrogate K_i^+ Carol has to prove that she knows the corresponding secret K_i^-.

   An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K_i^- without knowing s_c which is secret. So an adversary between Carol and the bank can never prove knowledge of K_i^-. Moreover calculating s_c requires the adversary to calculate

$$s_c = \log_g X_c \bmod P \tag{7.9}$$

   which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party (bank here)– The third party cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to solve equation 7.12 which is widely thought (see section 4.3) to be hard. Moreover even if the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the bank cannot masquerade as Carol to the merchant.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages

to steal the surrogate belonging to a subscriber still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K_i^- using equation 4.1 as s_c is secret and calculating s_c requires the adversary to solve equation 7.12 which is widely thought (see section 4.3) to be an intractable problem.

The bank initially issues the numbers the subscribers use to generate their surrogates in step 2 of the preparation phase and the bank also sends the merchant the surrogates subscribers will use to activate their accounts during the preparation phase. The bank can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but it cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, the bank can resolve disputes retrospectively. The bank would divulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order.

7.6 Scenario 9: Delegation of Payment Tokens

There are several commercial websites whose customer base comprises of people who are legally not allowed to have a credit card (boys and girls below 18 yrs). In such situations children use their parents' credit card to buy online. Now as in chapter 6 the problem here is that credit card numbers once learnt can be reused. Things become more complicated when parents have anonymous credentials instead of credit cards: to share anonymous credentials such as those mentioned in [18, 43, 15, 63] the owners also have to share their secret key. This is a major problem in such situations where delegation is a legitimate requirement. In this protocol we show a way of sharing payment tokens and surrogates. In this protocol Carol generates her DH keys modulo a prime as shown in section 4.3. The requirements which were identified in section 4.4 are as follows:

1. Verifiability

2. Un-correlatability

3. Misuse of surrogates

4. Protection from (partially) trusted third party (RS here)

5. Audit

7.6.1 The Protocol

There are four parties in the protocol, the bank, the customer Carol, her son William and the seller. The bank is online and the seller communicates with the bank for authorization decisions. Carol sends the bank her public key X_c using a secure, conventionally authenticated communication channel.

1. $Carol \longrightarrow Bank : X_c = g^{s_c}$

So that the Carol does not need to contact the bank before every single transaction, the bank sends her only the m C values, generated using equations 7.1, 7.2, 7.3, 7.4 and 7.5, along with the information she needs to prepare the corresponding B values and the surrogates. Only the bank can generate the S and Y values.

2. $Bank \longrightarrow Carol : \Delta = \prec C_{1..m}, B_0, \sigma_0, \tau_0, L_\sigma, L, A, P \succ$

During the preparation phase the bank also generates the surrogate K_1^+, as shown in section 4.5, Carol will be using in her first transaction: The bank always stores the surrogate Carol will be using next along with her account information prior to Carol uses it. Customer information are indexed using the surrogates by the bank. When the seller submits any customer information for authorization, the bank locates the corresponding account information using the surrogate. The bank retains for every customer

$$\Gamma = \prec S_0, B_0, Y_0, L_\sigma, L, A, K_1^+, \sigma_1, \tau_0, X, P \succ \tag{7.10}$$

The K_1^+, σ_1, and τ_0 values will change after every transaction.

For transaction i Carol calculates its ith surrogate and its corresponding secret, using the method described in section 4.5. Then, by equations 7.1, exponent σ_i

used to generate the B_i for the current transactions is calculated by Carol from the information provided by the bank and using the exponent used for generating B_{i-1} (except for B_1). Then the B_i value is calculated is calculated using equation 7.3. Carol sends William the value of her current public and private surrogate along with the payment tokens B_i and C_i. The payment tokens are sent to William via a secured authenticated communication channel by Carol. Thus Carol can later on prove that it was William who used it and not Carol. The payment tokens are verified at the point of use by the bank. After verifying the bank also calculates the values of the exponents to be used for calculating the next payment tokens. So any payment tokens that has been previously delegated and used cannot be verified again.

3. $Carol \longrightarrow William : K_i^-, K_i^+, B_i, C_i$

William cannot figure out Carol's secret key s from K_i^- or K_i^+ nor can he generate future B and C values from B_i and C_i. For the next transaction (*i.e.* transaction $(i+1)$) Carol will use B_{i+1} and C_{i+1} as her payment tokens and K_{i+1}^+ or K_{i+1}^- will be her surrogates. The values are different for each successive transactions and are generated serially using the information the Bank sends Carol in Δ.

The seller generates a transaction description T and sends it to William signed with the seller's private key K_{seller}^-. T is agreed between William and the seller. If William wants to pay, he authenticates to the seller that he is the legitimate owner of the current surrogate by signing the transaction description T. Along with the signed transaction description William sends the seller the ith payment tokens and the current surrogate. $\longrightarrow_m$ stands for an anonymous communication channel.

4. $Seller \longrightarrow_m William : [T]_{K_{seller}^-}$

5. $William \longrightarrow_m Seller : [T]_{K_i^-}, K_i^+, B_i, C_i$

The seller contacts the bank, the bank locates the appropriate customer with the help of the surrogate. We have mentioned in the preparation phase that the bank calculates in advance the surrogate a particular customer will be using next. This is done by the bank during the synchronisation phase at the end of every transaction. For the first transaction the bank retains the value of the first surrogate while

issuing the tokens. The bank calculates the corresponding S and Y value and solves equation 7.5 for transaction i. If the equation is satisfied by the B and C values presented by the customer the bank agrees to the transaction. This process ensures that the seller receives his payment.

6. $Seller \longrightarrow Bank : \prec K_i^+, [T]_{K_i^-}, B_i, C_i \succ$

7. $Bank \longrightarrow Seller : \prec T, authorizationDecision \succ$

The bank, after every transaction, calculates the surrogate the customer will be using for the next transaction using equations 4.6 and 4.2.

The surrogates and payment tokens are generated and used serially for synchronisation between the bank and the customer. If Carol wants two of her children to buy something with her credential she can generate surrogates and payment tokens for two of her children separately. In situations where there are two transactions the tokens can be accompanied with serial numbers and the bank can authorise payment tokens in an order as per the serial numbers. The one with a lower serial number will be approved first then one with a higher one. This requires the bank to maintain extra state which is the last serial number used by the customer. We believe that this will not induce any significant delay or processing overhead on the bank or the seller or affect the overall performance of the protocol. Including serial numbers with the payment tokens doesn't in anyway enable the seller to correlate various transactions of any particular customer.

7.6.2 Analysis

The properties of our protocol set out in section 4.4 are achieved as follows:

1. Verifiability - If equation 7.5 is satisfied then the bank accepts the payment tokens B_i and C_i as valid. It is hard for an adversary who does not posses the correct B_0, A and L_σ values to generate B and C values corresponding to a particular S and Y value.

   The signature can only be successfully verified using K_i^+ if T was signed using K_i^-. K_i^- can only be successfully generated by Carol using equation 4.1 as s_c is known to only Carol. Thus the verifier can be sure that the user (Carol here) presenting K_i^+ is the legitimate owner of K_i^+.

2. Un-correlatability – Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction without the explicit consent of the legitimate owner of the surrogate. Linking K_i^+ to K_{i+1}^+ and back to X_c would require an adversary or the merchant to solve the equation:

$$\log_{X_c} K_i^+ = \tau_i \bmod P \tag{7.11}$$

and calculate τ_{i+1} from τ_i using the values A and L, as shown in equation 4.6. Even if the values A and L were compromised, calculating τ_i by solving equation 7.11 is still thought to be an intractable problem. If an adversary cannot calculate τ_i then it cannot calculate τ_{i+1} using equation 4.6 nor the surrogate K_{i+1}^+ using equation 4.2. Thus it is difficult for an adversary or the merchant to link surrogates to each other or to the parent public key. Carol communicates with merchant via a secure anonymous communication channel denoted as $\longrightarrow_m$; thus it is difficult for an adversary observing the communication network to gain any additional information about the communicating partners beyond its *a priori* belief. Linking the payment tokens B_i and C_i with B_{i+1} and C_{i+1} also requires the adversary to solve the discrete logarithm problem which is thought to be intractable (see section 4.3).

3. Misuse of surrogates We have discussed in section 4.5 that to use a surrogate K_i^+ Carol has to prove that she knows the corresponding secret K_i^-.

   An adversary stealing the numbers in step 2 cannot masquerade as Carol as he/she cannot generate K_i^- without knowing s_c which is secret. So an adversary between Carol and the bank can never prove knowledge of K_i^-. Moreover calculating s_c requires the adversary to calculate

$$s_c = \log_g X_c \bmod P \tag{7.12}$$

   which is widely thought (see section 4.3) to be an intractable problem.

4. Protection from (partially) trusted third party (bank here)– The third party

cannot masquerade as the legitimate owner of X_c because the corresponding s_c is secret. Calculating s_c require an adversary to solve equation 7.12 which is widely thought (see section 4.3) to be hard. Moreover even if the value of the seed τ_i is known, still without the knowledge of s_c it is hard to generate the secret corresponding to a surrogate using equation 4.1. Thus the bank cannot masquerade as Carol to the merchant.

5. Audit – It is also difficult for an adversary to activate accounts belonging to other subscribers. To activate accounts belonging to other subscribers the adversary has to prove knowledge of the secret corresponding to the legitimate subscriber's surrogate (by signing T). Even if the adversary manages to steal the surrogate belonging to a subscriber still he/she cannot prove knowledge of the secret corresponding to the surrogate as he/she cannot generate K_i^- using equation 4.1 as s_c is secret and calculating s_c requires the adversary to solve equation 7.12 which is widely thought (see section 4.3) to be an intractable problem.

The bank initially issues the numbers the subscribers use to generate their surrogates in step 2 of the preparation phase and the bank also sends the merchant the surrogates subscribers will use to activate their accounts during the preparation phase. The bank can generate surrogates belonging to subscribers using equations 4.6 and 4.2, but it cannot masquerade as a subscriber as it cannot generate the secret corresponding to a surrogate using equation 4.1 since s_c is secret. Thus, from the surrogate the subscriber uses, the bank can resolve disputes retrospectively. The bank would divulge this link only in circumstances specified under legal authority, such as contract, legislation, search warrant or court order. The problem with this scheme is that an auditor cannot figure out who used a particular surrogate although it can figure out the owner of the surrogate.

7.7 Conclusions

Though we concentrate on access control in this dissertation nevertheless our work has some interesting implications in the areas of payments, differential pricing etc.

Although we have presented some payment protocols in chapter 7 we admit that they are far from perfect and we are happy to use any other payment protocol along with our surrogates in the protocols described above.

In our trust model the users trust the bank which replaces the need for a global pseudonym authority by a more localised trust relationship. Moreover in the real world users have to trust the bank and are legally bound to provide self identifying information to the bank.

The protocol presented in section 7.3 does not make use of surrogates and any other public key cryptosystem based on a trapdoor oneway permutation can be used. What we have shown is that price discrimination using real life identities need not always be a threat to privacy.

The traditional belief that transferability is not desirable in anonymity systems has led to systems which do not support delegation. Our surrogates can be auditably delegated along with the payment tokens, while preserving privacy.

Chapter 8

Conclusions

8.1 Introduction

In this chapter we assess the extent to which our approaches can provide useful leverage to address problems of trust and anonymity in various electronic services. In section 8.2 we revisit the problems of access control, trust management, and delegation in the light of the approaches described in this dissertation. Section 8.3 gives a brief overview of how our approaches can be used in conjunction with trusted hardware. In section 8.4 we describe how our approaches can be exploited so as to reduce trust assumptions necessary between various entities of a system. Future work which could be carried out by extending various approaches presented in this dissertation is discussed in section 8.5. We summarise our work in section 8.6

8.2 Significance

We have shown ways in which authorization decisions can be made without using long term credentials linked to a stable identity. Role based access control models such as [7, 35] currently rely on authentication using long term credentials for activation of roles. But our surrogates can be used in RCBS [7] for activation of roles without compromising individual privacy by means of the protocols described in section 5. The powerful concept of activation of roles using prerequisites de-

scribed under section 3.2.3 can be achieved with surrogates using the protocol described in section 5.4. Our approaches can also be used for activation of roles in applications like Globe [52], discussed in section 3.3, without the proxy being able to correlate actions belonging to a particular user. At the same time, proxies can still get the benefit of consulting a local trust management engine before allowing access to a resource. Compared to the approaches mentioned under section 2.4 our approaches do not need a global pseudonym authority like idemix [15] or depend on some universal authority for registration of pseudonyms like Globally Unique Pseudonyms [64]. Our approaches do not advocate complete anonymity, but allow an auditor with appropriate authority to correlate actions belonging to a particular user retrospectively. None of the protocols we presented in chapter 5 requires communicating partners to share a long term secret, a feature which we believe is significant.

Keynote, discussed in section 2.2.2, currently uses long term credentials to make authorization decisions. This can be a threat to individual privacy. Our approaches can be integrated with systems that currently use trust management systems such as Keynote to make authorization decisions. Surrogates can come with explicit labels and policies which can then be used by the trust management systems to make authorization decisions. This adds an extra layer of anonymity in trust management systems, which can coexist peacefully with traditional non-anonymous mechanisms. We have shown that one can make trust decisions based on transient identities and we believe this ability is useful for certain services.

Users, in the protocols presented in this dissertation, generate and use their own surrogates and it is difficult for an adversary to masquerade as a legitimate user. The anonymity systems described under section 2.4 either require the user to trust a third party with personal sensitive information as in section 2.4.2 or depend on a global authority as in sections 2.4.5 and 2.4.6. Trusting a third party with personal sensitive information is not quite right [4]. A global pseudonym authority as advocated in [15, 64] is difficult to build or at least is as hard as a global public key certification authority which hasn't happened yet. Moreover the trust relationship between users and a global pseudonym authority is unnecessary, compulsive and can have undesirable consequences. In our approaches principals do not share personal sensitive information with everybody and we have more

localised trust relationships. No longer are users required to accept a transitive notion of trust, nor do we presuppose a global pseudonym authority.

Previous anonymous credential management systems do not allow principals to share their credentials, a feature which we believe prevents users from delegating credentials. We have described approaches in chapter 6 where users can delegate their surrogates without the delegatee being able to figure out the secret used to generate the surrogate. The ability to delegate, we feel, is a significant improvement over previous approaches because delegation is useful [25] as resources are hardly ever entirely local and it is often a legitimate real world requirement that users are able to delegate jobs and credentials using which the delegatee can access the resources of the delegatee so as to complete the job.

8.3 Trusted Hardware

Though the palladium based approach we review in section 3.6.1 does not help us to ascertain trust while maintaining individual privacy at the same time, trusted hardware can nevertheless be deployed in conjunction with the protocols we presented in this dissertation.

The two level authentication mechanism we present in sections 5.4 and 6.3 can exploit the availability of trusted hardware. Users could make use of such hardware to authenticate as members of a group before using their surrogates in the protocols described in section 5.4 and 6.3. One could also use trusted hardware to authenticate as a member of a group in the protocol described in section 5.3. The top level authentication mechanism in the protocols described in sections 5.4, 5.3 and 6.3 is done using the public key of the users. Ring signature is perfectly signer ambiguous so it is difficult for an adversary to figure out the actual signer from a group and at the same time it is difficult for an outsider to masquerade as a member of a group.

It is not necessary that keys be permanently stored in particular hardware as the keys can be fetched using a protocol like the one presented in [23]. What is important is that only the legitimate owner of the keys can fetch and use the keys via the trusted hardware. The user can first auditably authenticate to the hardware using surrogates and then the hardware keeps an audit trail. The hardware can

then be used by the user to authenticate as a member of the group.

Unlike our earlier approach (which we present in the appendix and now regarded as a false start) we advocate a more localised trust relationship in the mechanisms presented in chapter 5. In particular, in our protocols there is no need for a global certification and revocation authority. The keys are not stored in the hardware when the hardware is not in use, so even if the hardware is compromised it is still difficult to compromise the keys. Moreover unlike a conventional Palladium based approach, end-systems do not need to trust the authentication mechanism of any third party. Our new approaches, take identity out of the authorization management infrastructure. In the approaches presented in chapter 5 users do not need to reveal any digital credential, such as a public key, linked to a stable identity to authenticate themselves or obtain tokens, but the trusted hardware nevertheless makes it difficult for an adversary to forge transactions.

8.4 Localisation of Trust

Using the protocols proposed in this dissertation it is possible to reduce the need to trust. In the protocols described in this dissertation, when principals request access, the server does not need to trust the authentication mechanism of a third party. For example in the Kerberos authentication service the services must trust Kerberos' judgment as to the identity of a user to be accurate. Thus in the context of access to the LIS described in section 5.4, Kerberos advocates an approach where principals authenticate to an entity other than the LIS and get a ticket which the principals then use to access the LIS. In such an approach the LIS needs to trust the authentication mechanism of the third party whereas in the protocol described in section 5.4 the LIS authenticates users directly, but anonymously, before allowing access. Thus the LIS does not need to trust the authentication mechanism of a third party. The LIS server has the surrogates of users who are allowed access and the use of two different mechanisms makes it very difficult for someone who is not authorized to masquerade as a member of a group.

In our approach principals need only reveal their long term credentials to entities which are legally authorized to verify them. For example, in the protocol described in section 5.2, Carol only reveals her long term credential linked to

her stable identity to the subscription authority while registering, she does not need to reveal her long term credentials every time she reads the newspaper. The subscription authority is legitimately required to know Carol whereas the server where Carol logs in to read the newspaper does not need to know Carol's identity but only needs to know whether or not Carol is authorized to read the newspaper. Since there is now no risk of correlation of Carol's online activities by the server it follows that Carol does not need to unnecessarily trust the server when she logs in to read the newspaper. So the threat that information about us is stored at too many places can be countered using the protocols presented in this dissertation.

In the delegation protocol presented in section 6.4 Carol does not need to trust Bob to be honest. Carol has control over who can use the surrogate as she can specify this. Bob cannot further delegate surrogates without the explicit knowledge and consent of Carol. In the protocol presented in section 6.3 Carol can use the surrogate she delegates and so Bob needs to trust Carol to be honest and not use the surrogate she delegates. But in the protocol discussed in section 6.4 Bob does not need to trust Carol to be honest and Carol cannot use the surrogates Carol delegates to Bob: an auditor can uniquely and irrefutably link all actions to principals. Moreover all identity resolution in the protocol presented in section 6.4 is local; Bob does not need to trust the bank or the merchant with personal sensitive information in order to view movies. An external auditor can link transactions to Bob only with the cooperation of an auditor who is internal to Bob's domain and Bob needs to only trust this local auditor. Local identity resolution of this kind has implications in the design of role based authorization mechanisms where users can activate roles across domains without transitively trusting authorities outside their immediate domain [39].

The problem at present is that users have little control of the risks they are exposed to since they must enter into an unnecessary compulsive trust relationship with the system. This forces them to trust the system to protect them from threats. This dissertation provides the way to allow clients to control the risks to which they are exposed by bearing the cost of relevant countermeasures themselves, rather than clients being forced to trust the system infrastructure (and bear an equal share of the cost of all countermeasures which may or may not be effective for them.) Moreover using our mechanisms systems do not need to trust the

authentication mechanism of a third party. This feature is useful even if privacy is not required, due to the ability which it provides to systematically weaken trust assumptions.

8.5 Future Work

Our work can be extended to design an open network authentication system similar to Kerberos, which was discussed in section 3.5. Kerberos has a strong authentication mechanism based on long term credentials, and audit is linked to authorization via the same long term credential used to authenticate. Such an approach gives birth to some unnecessary trust assumptions between various entities of the system *e.g.* clients are compelled to trust servers not to enable an adversary to correlate their access requests.

We have argued that the trust relationship between users and servers providing various services in this scenario is unnecessary, compulsive and can have undesirable consequences. Services only need to know that the ticket has been issued by the ticket granting service and that the user presenting the ticket is authorized to use the service. Services do not need to know the identity of the user. An important conclusion of this dissertation is that (in this sense) the requirements of trust and anonymity are not in conflict with each other and can coexist peacefully without compromising the requirements of secure authentication and audit. We have demonstrated in this dissertation that even if users authenticate to a service using weak identities an auditor can still link actions to principals, thus enabling the elimination of the present compulsive trust relationship between users and services. The approaches presented in this dissertation could be extended to investigate how they can provide useful leverage to eliminate such unnecessary and compulsive trust relationship between various other parts of the system.

Delegation is difficult in current versions of Kerberos. If tickets issued by a Kerberos authentication service are delegated then it will be hard for an auditor to uniquely and irrefutably link actions to users, and users can frame each other. At present, to delegate anonymous credentials users have to share their secret key, which is not desirable as the delegatee can reuse credentials of the delegator and an auditor cannot figure out whether it was the delegator or the delegatee who used

a particular credential. This dissertation shows a way of introducing auditable anonymous delegation in the electronic world. Open network authentication systems are often used to issue tickets to users, using which users can access a remote resource. The approach for auditable anonymous delegation can be extended to design open network authentication services where users can delegate credentials without compromising the goal of audit.

The goals of the open network authentication system which we propose in this section can be summarized as

1. The service should be sure that the ticket has been issued by a ticket granting service authorized to do so.

2. An adversary should not be able to masquerade by stealing the ticket of a legitimate ticket owner.

3. Only the legitimate owner of the ticket should be able to use a ticket during the duration of the ticket.

4. An adversary who controls a service should not be able to correlate actions of any particular user.

5. The authorization service should allow authorized delegation of credentials.

6. An authorized auditor should be able to uniquely and irrefutably link actions to principals.

An important benefit of an approach with reduced trust assumptions is that risks are reduced *e.g.* clients do not need to trust the server or the remote authentication mechanism to preserve their privacy. Preserving workstation integrity is an open problem in public networks, For example someone might change the software running on a workstation to record the password of a user. Our approach using weak identities or surrogates can be extended to address this problem. Since our passwords change after each use so an adversary does not benefit by recording the password. Only the TGT service and the legitimate owner of the password can use a TGT ticket.

8.6 Summary

We have shown mechanisms which enable us to separate identity management from the trust management envelope, thus eliminating the present compulsive trust relationship between users and various electronic services. We propose a more localized trust relationship rather than having globally trusted entities acting as a repository of personal sensitive information. Even though the protocols described in this dissertation include a partially trusted third party, our third party cannot do any harm to the legitimate owner of a surrogate. In this dissertation we concentrated on access control models, but our work has implications in other application areas some of which we discussed in chapter 7. Chapter 7 shows that the surrogates can be tied to payment tokens and used in conjunction with them. This dissertation also comments implicitly on the desirable level of privacy which does not impede the development of approaches allowing auditability. The approaches we present in chapters 5 and 6 do not guarantee absolute anonymity but make it difficult for an adversary to link actions to individuals. Previous anonymous credential management systems (which we have discussed in chapter 2) do not allow principals to share their credentials, a feature which we believe prevents users being able to delegate credentials. Our delegation mechanism in chapter 6, we feel, is a significant improvement over previous approaches. Delegation is useful [25] because resources are hardly ever entirely local and it is often a legitimate real world requirement that users are able to delegate both jobs and credentials using which the delegatee can access the resources of the delegatee so as to complete the job. Such approaches allow an auditor with appropriate authority to link actions to individuals, thus enabling fair dispute resolution. This dissertation shows that the requirements of trust and anonymity are not in conflict with each other and that the two can coexist in a peaceful manner. Finally, reduced trust assumptions enable users to control the risks to which they are exposed.

Bibliography

[1] Martin Abadi and Cedric Fourmet. Private Authentication. *Journal of Theoretical Computer Science*, 322:427–476, 2004.

[2] Alessandro Acquisti. Privacy In Electronic Commerce and the Economics Of Immediate Gratification. *Proceedings of the* 5th *ACM conference on Electronic Commerce*, pages 21–29, 2004.

[3] Alessandro Acquisti and Hal Varian. Conditioning prices on purchase history. Technical report, University of California Berkeley, 2001.

[4] Ross Anderson. *Security Engineering*. Wiley, Inc, 2001.

[5] David E. Bell and Leonard J. LaPadulla. Secure computer systems: Unified exposition and multics interpretation. Technical report, Mitre Corporation, 1975.

[6] Andras Belokosztolszki. Role based access control policy administration. Technical Report 586, University of Cambridge, 2004.

[7] Yolanta Beresnevichiene. A role and context based security model. Technical Report 558, University of Cambridge, 2003.

[8] Oliver Berthold, Andreas Pfitzman, and Rony Stantke. The Disadvantages Of Free Mix Routes And How To Overcome Them. *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability: Designing Privacy Enhancing Technologies: Lecture Notes in Computer Science Series*, 2009:30–49, 2000.

[9] Matt Blaze, Joan Feigenbaum, J. Ioannides, and Angelos Keromytis. The Keynote Trust Management System. *Request For Comments Series*, (2704), 1999.

[10] Matt Blaze, Joan Feigenbaum, and M. Strauss. Compliance Checking In The Policymaker Trust Management System. *Proceedings of the* 2nd *Conference on Financial Cryptography: Lecture Notes in Computer Science Series*, 1465:251–265, 1998.

[11] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[12] Stefan Brands. Offline cash transfer by smart cards. Technical report, Centrum voor Wiskunde en Informatica, 1994.

[13] Giacomo Calzolari and Alessandro Pavan. Optimal design of privacy policies. Technical report, University of Toulouse, 1992.

[14] J. Camenisch and A. Lysyanskaya. An Efficient System for Nontransferrable Anonymous Credentials with Optional Anonymity Revocation. *Proceedings of Eurocrypt, 2001, Springer Verlag*, 2045:93–118, 2001.

[15] Jan Camenisch and Els Van Herreweghen. Design And Implementation Of The *idemix* Anonymous Credential System. *Proceedings of the* 9th *ACM conference on Computer and Communications Security*, pages 21–30, 2002.

[16] Amy Carrol, Mario Juarez, Julia Polk, and Tony Leininger. Microsoft Palladium: A Business Overview. *Available at http://www.microsoft.com/PressPass/features/2002/jul02/0724palladiumwp.asp*, 2002.

[17] David Chaum. Untraceable Electronic Mail, Return Addresses And Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[18] David Chaum. Security Without Identification: Transaction Systems To Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[19] David Chaum. Achieving Electronic Privacy. *Scientific American*, pages 96–101, August 1992.

[20] David Chaum and Eugene Van Heyst. Group Signatures. *Proceedings of Eurocrypt: Advances in Cryptology: Lecture Notes in Computer Science Series*, 547:257–265, 1991.

[21] Bruce Christianson and William Harbison. Why Isn't Trust Transitive. *Proceedings of the* 3rd *International Workshop on Security Protocols: Lecture Notes in Computer Science Series*, 1189:171–176, 1996.

[22] Bruce Christianson and J. A. Malcolm. Binding Bit Patterns To Real World Entities. *Proceedings of the* 5th *International Workshop on Security Protocols Lecture Notes in Computer Science Series*, 1361:105–113, 1998.

[23] Bruce Christianson, Michael Roe, and David Wheeler. Secure Sessions From Weak Secrets. *Proceedings of the* 11th *International Workshop on Security Protocols Lecture Notes in Computer Science Series*, 3364:190–212, 2003.

[24] Yang Hua Chu. Trust management for the world wide web. Master's thesis, M.I.T., 1997.

[25] B. Crispo. *Delegation of Responsibility*. PhD thesis, University of Cambridge, 1999.

[26] George Danezis. Designing and attacking anonymous communication systems. Technical Report 594, University of Cambridge.

[27] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design Of A type III Anonymous Remailer. *Proceedings of the* 24th *IEEE Symposium on Security and Privacy*, pages 2–15, 2003.

[28] Partha Das Chowdhury, Bruce Christianson, and J.A. Malcolm. Anonymous Authentication. *To Appear in the Proceedings of the* 12th *International Workshop on Security Protocols Lecture Notes in Computer Science Series*, 2004.

[29] W. Diffie and M. Hellman. New Directions In Cryptography. *IEEE Transactions on Information Theory*, 22:472–492, 1976.

[30] Y. Ding, P. Horster, and H. Peterson. A New Approach for Delegation Using Hierarchical Delegation Token. *Proceedings of the* 2nd *Conference on Computer and Communications Security*, 1996.

[31] Taher El-Gamal. A Public Key Cryptosystem And a Signature Scheme Based on Discrete Logarithms. *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, 1984.

[32] Carl Ellison, B. Frantz, Ronald Rivest, B. Thomas, and T Ylonen. Spki Certificate Theory. *Request For Comments Series*, (2693), 1999.

[33] Carl Ellison and Bruce Schneier. Ten Risks of PKI: What You Are Not Being Told About Public Key Infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[34] David Ferraiolo and Richard Kuhn. Role Based Access Control. *Proceedings of the* 15th *National Computer Security Conference*, pages 13–16, 1992.

[35] David Ferraiolo, Ravi Sandhu, Serban Gavrilla, Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard For Role Based Access Control. *ACM Transactions on Information and Systems Security*, 4(3):224–274.

[36] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[37] M. Gasser and E. McDermott. An Architecture for Practical Delegation in a Distributed System. *Proceedings of the IEEE Symposium on Security and Privacy*, 1990.

[38] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Applications. *IEEE communications Surveys Tutorials*, 3(4):2–16, 2000.

[39] William Harbison. Trusting in computer systems. Technical Report 437, University of Cambridge, 1997.

[40] F. Heylighen. Evolution Selfishness and Cooperation. *Journal of Ideas*, 2(4):70–76, 1992.

[41] D. Knuth. *Sorting and Searching: The Art of Computer Programming*, volume 3. Addison Wesley, Inc, 1998.

[42] Loren M. KohnFelder. Towards a practical public key cryptosystem, B.S. thesis, M.I.T., 1978.

[43] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. *Proceedings of the* 6th *Annual International Workshop on Selected Areas in Cryptography: Lecture Notes in Computer Science*, 1758:184–199, 1999.

[44] A. Menezes, P. van Oorschoot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[45] Ulf Moller, Lance Cotrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol Version 2. 2003. Draft.

[46] B. Clifford Neuman and Theodore Tso's. Kerberos: An Authentication Service For Computer Networks. *IEEE Communications*, 32(9):33–38.

[47] Andrew Odylzko. Privacy Economics and Price Discrimination on the Internet. *Proceedings of the* 5th *International Conference on Electronic Commerce*, pages 355–366, 2003.

[48] Mary Ann Patton and Audung Josang. Technologies For Trust In Ecommerce. *Proceedings of the IFIP working conference on Ecommerce*, 2001.

[49] S. Pohlig and M. Hellman. An Improved Algorithm For Computing Logarithms And Its Cryptographic Significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[50] J.M. Pollard. Monte Carlo Method for Index Computation mod p. *Mathematics of Computation*, 32(143):918–924, 1978.

[51] Carl Pomerance. *Cryptology And Computational Number Theory: Proceedings of the Symposia on Applied Mathematics*, volume 42. American Mathematical Society, 1989.

[52] Bogdan Popescu, Martin Van Steen, and Andrew Tanenbaum. A Security Architecture for Object Based Distributed Systems. *Proceedings of the* 18th *Annual Computer Security Applications Conference*, 2002.

[53] Tal Rabin and Michael Ben-Or. Verifiable Secret Sharing and Multiparty Protocols With Honest Majority. *Proceedings of the* 21st *ACM Symposia on Theory of Computing*, pages 73–85, 1989.

[54] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity For Web Transactions. *ACM Transactions on Information and Systems Security*, 1(1):66–92, 1998.

[55] Ronald Rivest, Adi Shamir, and Yael Tauman. How To Leak A Secret. *Proceedings of the* 7th *International Conference on the Theory and Applications of Cryptology and Information Security; Advances in Cryptology: Lecture Notes in Computer Science Series*, 2248:552–565, 2001.

[56] Shamir Adi. Rivest, Ronald and Mark. Adleman. A Method For Obtaining Digital Signatures And Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[57] Michael Roe. *Cryptography and Evidence*. PhD thesis, University of Cambridge, 1997.

[58] Ravi Sandhu. Lattice Based Access Control Models. *IEEE Computers*, 26(2):9–19, 1993.

[59] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From Trickle to a Flood: Active Attacks on Several Mix Types. *Proceedings of the* 5th *International Workshop on Information Hiding: Lecture Notes in Computer Science Series*, 2482:36–52, 2002.

[60] Narendar Shankar and William A. Arbaugh. On Trust for Ubiquitous Computing. *Proceedings of the Workshop on Security in Ubiquitous Computing: available at http://www.teco.edu/ philip/ubicomp2002ws/*, 2002.

[61] William Stallings. *Cryptography and Network Security*. Prentice Hall, 1999.

[62] Paul Syverson, David Goldschlag, and Michael Reed. Anonymous Connections And Onion Routing. *Proceedings of the* 18th *IEEE Symposium on Security and Privacy*, pages 44–54, 1997.

[63] Paul Syverson and David Goldshlag. Unlinkable Serial Transactions: Protocols And Applications. *ACM Transactions on Information and Systems Security*, 2(4):354–389, 2000.

[64] Paul F. Syverson and Stuart Stubblebine. Authentic Attributes With Fine Grained Anonymity Protection. *Proceedings of the* 4th *Annual Conference on Financial Cryptography: Lecture Notes in Computer Science*, 1962:276–294, 2000.

[65] Curtis R. Taylor. Private demands and demands for privacy: Dynamic pricing and market for customer information. Technical report, Department of Economics of Duke University, 2002.

[66] E. R. Verheul. Self Blindable Credential Certificates from the Weil Pairing. *Proceedings of Asiacrypt, 2001, Springer Verlag*, 2248:533–551, 2001.

[67] Maurice V. Wilkes. Time Sharing Computer Systems. *Elsevier Science Inc.*, 1975.

Appendix A

Papers and Technical Reports

A.1 Papers

1. A Palladium Based Solution for Bipartite Trust Management

2. Anonymous Authentication, Presented at the 12th International Security Protocols Workshop, Cambridge, 2004. To appear in the Lecture Notes in Computer Science series.

3. Uncorrelatable Electronic Transactions using Ring Signatures, Appeared in the Proceedings of the Wholes Workshop of Multiple Views of Privacy 2004, Sweden

4. Authentication In An Object Based Distributed System, Presented at the Conference on Distributed Processing and Networking 2004, India

A.2 Technical Reports

1. How to deceive Prying Eyes in the Electronic World, Technical Report no. 392 published by the University of Hertfordshire

A Palladium Based Solution for Bipartite Trust Management

Partha Das Chowdhury

January 31, 2005

1 Introduction

We now live in a world of wide open distributed systems which has unleashed a whole new lot of both commercial and noncommercial opportunities for us. We will concentrate on a system for commercial transaction on the web. The goal is to have a large number of small entrepreneurs who can do business at a low operating cost and we shall argue that this is indeed possible. At the same time we now have to deal with completely new kinds of situations and threats.

The problem very briefly is as follows. Every time we buy something over the phone or over the net the information goes into a database somewhere. All these records can be linked together and people can have access to our medical records, where we buy, whom we communicate with etc. Thieves routinely use stolen credit card numbers to trade on their victim's good payment records or murderers have tracked down their targets by using government maintained address records. It is indeed possible to build a complete dossier on an individual by aggregating school, travel, medical and credit card records [6]. The Total information awareness project of the U.S. government is aggregating information from credit cards, school, travel and medical records to root out potential terrorists but this can well be used by malicious persons to get hold of their targets [1]. Apart from these issues we are not also sure whether we will get the goods we are paying for. There are a whole lot of questions that come to our mind when we do transactions over the net. This implies that we take risks knowingly or unknowingly while we transact over the net. In situations like these we take a leap of trust with the belief that everything will be fine. William Harbison in his PhD thesis argues that this trust results from lack of verifiable knowledge about the entity at the other end of the communication channel [?]. To represent this situation more correctly we should say that here both the buyer and the seller takes a risk. He rightly points out that trust is not a system property but is a property of a state of knowledge that either increases or decreases with the risk involved. There has been studies conducted which concluded that there would have been substantial increase in the number of transactions over the net if the

users knew how their personal information would be used [18]. It is also true that most of the attacks are on the servers that store the information rather on the communication channels as doing the former is easier than doing the later. To be precise the requirement is of anonymity in online transactions. The questions which haunt us while we do transactions online can be summed up as

1. How will our personal information be used?

2. Do we get the goods we pay for?

The question the seller ask is

Will I get the money for the goods I sold?

The issues that we have raised above are not exhaustive but can be regarded as the more important ones. Apart from these we also should take into account the reputation of a seller before shopping with a particular seller. The questions like whether or not our rights as a customer are protected, what is the refund policy etc also crop up when we do transactions over the net. Here the requirement is of ascertaining the reputation of the seller. It is quite evident now that transacting over the web is full of risks. Computer scientists call this a trust management problem. They opine that these problems are in the domain of the trust management systems as according to them this is the entity responsible for answering the question

'Should we carry out this dangerous action or not? [2]

We in this paper examine some of the trust management approaches along with other approaches in the light of an online transaction where we are buying goods online. We also show that these approaches cannot be used satisfactorily to address the issues we have mentioned above. We argue that the trust management approaches we have seen in the past had different goals and interests. They cannot guarantee that there won't be any malicious use of our personal information, or it is not also possible to ascertain the reputation of the entity at the other end of the communication channel. That's what led us to propose a reputation assessment system rather than using conventional trust management approaches. We chose to have complete anonymity in our proposed scheme while buying goods online to guard against malicious use of our personal information and propose a reputation assessment system to ascertain the reputation of the seller. Reputation of a real world entity increases or decreases with time and is based our personal experience. We also refer to friends and other sources to assess the reputation of real world entities before we enter into any sort of transaction with any real world entity. Our key innovation has been a proposal that models this dynamic nature of reputation as we find in the real world. Though the reputation assessment system proposed here forms a part of a compliance checker that implements local policies but the way it implements and makes decisions is different from the trust management systems proposed

by other researchers. The compliance checker dynamically assesses the reputation of the entity at the other end of the communication channel before coming up with a decision on a particular action. So far as anonymity is concerned some past approaches providing anonymity will also be discussed in this paper before we present our approach. A digital equivalent of credit card is proposed in this paper which we use to buy goods online but at the same time preserve anonymity. The interesting part of our scheme is that payment methods are not tied to identity of the payer and like cash the seller is happy as long as it sees the Bank of England stamp on the coin or the stamp of the issuing bank on the card here. When we buy using coins or bank notes the coins or notes doesn't have or we do not need to give our name or other information along with the coins. Similarly the cards we propose do not have our names like the traditional credit cards. The legitimate holder of the cards we propose needs to authenticate him/her every time he/she uses these cards to buy goods online as this protects the cards from theft but this process of authentication doesn't reveal his/her real world identity. This is a significant difference when compared to the traditional credit cards.

The rest of the paper is organized as follows; in section 2 we will have a look at what historically has been done in the area of trust management systems and whether those systems are able to address the questions we have mentioned above. In section 3 we will focus on anonymity and in the following section 4 some approaches to anonymity will be presented. In section 5 we have our proposed approach for anonymous transaction. The approach we will present will also enable two-way verification of credentials of the principals involved in the transactions. A model of assessing reputations and making decisions based on them will also be presented in section 6 followed by a compliance checking mechanism which incorporates the model of section 6 will be presented in section 7. Finally we present our conclusions in section 8.

2 Review of Current Approaches

2.1 Keynote

Keynote [2] is the latest version of a set of trust management approaches that came from Matt Blaze and others of AT&T. Policy maker was also developed by the same people and some of them were involved in the development of REFEREE [7] another trust management approach that was developed along with researchers from MIT and W3C. We shall focus on Keynote here. Keynote works more or less like a database query engine. It can function as a stand alone application interfacing with other parts of the system and helping them in making decisions. Let's lump these other parts of the system together and call them by a common name application. Whenever any application faces the question "Should we carry out this dangerous action" then it refers to Keynote

for an opinion and based on that opinion it decides its future course of action. The application presents the Keynote trust management engine with a set of local policies that should be taken into account while taking a decision on this particular request along with the credentials of the requestor and details about the proposed action. If the proposed action conforms to the local policy then keynote advises the requestor to proceed otherwise Keynote advises it not to perform this action as it is against the local policy. Keynote acts as a compliance checker for the application. The policies are specified in the form of assertions and the actions are specified which are evaluated against these assertions. We shall deal with the structure of keynote assertions and requests later in this section.

We have already mentioned the questions that come to the mind of both the buyer and seller while we do financial transactions over the net. We will have a look whether keynote is able to address these issues. We would like to mention here that may be solving this problem was not the goal of keynote but this is a problem that needs to be addressed somehow and so we wanted to investigate whether the current approaches are already able to address this problem. Let our policy be that we are only going to pay by credit cards those who are authorized to accept them and have the required credentials from a bank or a building society or whoever is legally empowered to issue such authorizations. A Keynote assertion specifying this policy looks like

1. Authorizer: "DSA: 1FFG2" (CA's key)
2. Licensee: "RSA: DEF662" (Buyer's key)
3. Conditions: "((app_domain == "BUY") ("Pay by credit cards if seller is authorized to accept credit cards")
4. Signature: "DSA-SHA1:1861234" (Signature of the authorizer.)

Suppose this is how the keynote assertion looks like when we implement our local policy via key note. The sellers to need to have digital representation of their authorizations to accept credit cards. Lets the seller use a SPKI certificate which can be roughly as

1. Issuer: "RSA 2GG36" (Bank's key)
2. Subject: "RSA 7YYH5" (Seller's key)
3. Authorization: "Can accept credit cards"
4. Delegation: "No"
5. Validity: "1002 1009"

When there is a request by the buyer then the relevant application fetches the relevant credential of the seller, parses it and presents it to keynote along with the id of the requestor and the id of the policy to be consulted to make a decision. Based upon this information keynote comes out with a decision which is most likely to be positive in the above instance. Though keynote was able to act perfectly to implement the above mentioned policy but that is not all that what we want to achieve. We should also be able to answer the questions that what they are going to do with our personal information? Are we going to get the good we are paying for? Will the seller get his money? Here keynote checks whether the remote host is allowed to accept credit cards or not and based on this it gives decision. But consider a situation where there is a corrupt or disgruntled employee who steals in credit card numbers and uses them or in other cases people getting access to other information like medical records etc. So we shouldn't use a keynote affirmation to wrap the entire organization with a trust blanket when we do not have information about the storage and usage of records. The sellers might argue that they have secure access control mechanisms but why should we believe that this is more than advertising? Are we able to test how secure those access control mechanisms are? These verifications also do not guarantee that we will be getting the goods we are paying for. Similar arguments can be given from the seller's point that simply checking that the customer has a valid credit card doesn't guarantee that the seller is going to get the money. The buyer may have used a stolen credit card and the legitimate owner is going to stop all payments before the settlement is done between the seller and the bank. It can be said that the trust management approaches such as Keynote do not attempt to address these issues. So far as the issue of assessing reputation is concerned we will again have a look at the assertions and certificates above. We have talked about certificates in a bit detail below. Now the structure of the keynote assertions doesn't permit us to incorporate our past experiences. Keynote doesn't support this concept at all. This static nature of assertions won't be able to support this dynamic real world process. Keynote only supports authorization based on public keys which doesn't cover the entire trust management problem [10]. It focuses on establishing resource access trust and service access trust [10]. We can even say that the question we are looking at is quite hard for the trust management systems to address.

2.2 Independent Unbiased Trust Entities

There has been considerable interest now a days about independent unbiased trust entities like TRUSTe, EEF [13] etc. These are termed independent unbiased trust entities. They issue a seal that is displayed on the websites that do financial transactions online. There are also alternative dispute resolution agencies that intervene whenever there is a dispute between the consumer and the seller. These seals are more like the trade licenses that we find in most of the shops or like a safety certificates. For example any boat plying on the Trent

River has to have some form of authorization from the British waterways board and display that. But the fact is that just having a seal doesn't prove that you are very honest or that all your employees are honest or are trained what to do when there is a man overboard. There are checks while issuing such seals or certificates but someone can always register and get a seal and later on turn dishonest. We are very sceptical about the utility of such practices as gathering evidences purely in the electronic world is very hard to do [15], and a seal cannot guarantee for somebody's honesty. Seals can have a psychological effect on the customer who doesn't understand the system in much depth. So again if we put the question what happens to our personal information, can these trust entities give us a satisfactory answer? Neither does these seals guarantee proper delivery of goods or save the seller from being defrauded. They can argue that since they are displaying the relevant seals they are expected to act properly. The success of these independent unbiased trust entities depend more on self regulation which assumes that everyone will do things honestly. The organizations issuing seals have little control over the storage, usage and access control of the target organizations. The seal issuing organizations also lack verifiable proof of usage and storage of information by the sellers and so there is an element of risk or unnecessary trust is involved between these trust entities and the sellers. So far as assessing reputation is concerned these trust entities are of little help for reasons similar to as for the other issue of anonymity.

2.3 Certificates

There has been considerable work done in representing principals in a digital world and digital certificates are one way doing that. Digital certificates were first proposed by Kohnfelder in his 1978 MIT bachelor's thesis [11]. A digital certificate was initially devised after public key cryptography was introduced and the need arose to communicate to principals each other's public key. Typically a public key certificate contains information about the issuer which may be a certifying authority, the subject whom it is supposed to represent, the dates the certificate is valid and related information. This kind of certificates cannot vouch for the trustworthiness of the subject but is used for identifying them. They may also be used to check for authorization like we have in SPKI [8]. As we have seen above a certificate from a bank state whether or not a particular seller is capable of accepting credit cards but that is not enough to ensure that our credit card numbers won't be used maliciously. Apart from these certification schemes are also riddled with risks [9]. We need detailed assertions about who can be a CA, its authority, revocation services etc in order to build a public key authority of some credibility. In the light of these problems it can be said that certificates aren't also able to address the issues like proper use of personal information or reputation which we are interested in. They have their own interest and goals and that is identifying principals in the digital world.

2.4 Attribute Vector Model

This was developed with the goal of having trust decisions in pervasive computing. This model incorporated both the traditional identity based model and the context based model that is of relevance to pervasive computing. They derive degree of trust of an entity S_i on S_j as

$$D(Si, Sj) = f(A(Sj))$$

where Si and Sj are separate entities and A is the set of their attributes. Attributes can be traditional credentials or they may be context based like location. The former can be used for traditional computing purposes and the later can be used for pervasive computing devices. In the light of our question relating to the safety of our personal information now we do not think this model can be of much help as when applied in the traditional context as they operate on credentials and we have seen earlier that credentials aren't meant for guaranteeing somebody's trustworthiness.

2.5 Intermediate conclusions

We started with the questions that we need to address in order to improve the credibility of web based transactions. A number of past and present approaches were analyzed in the light of above questions and what we have seen that none of the approaches satisfactorily addresses the issues. They stop with saying whether or not somebody is authorized to do something like accepting credit card numbers but there is more to the problem than this authorization decision. Most of these approaches are based on credentials and are more suited in making better access control decisions but it is beyond doubt that access control decisions just are one part of the entire trust management problem and not the whole. This dependence on credentials makes them very static and renders them unsuitable to emulate the real world dynamic nature of reputation assessment. At this point we would like to explore an alternative approach where nobody can misuse our personal information. In a word what we envisage is a system that provides anonymity as well as helps me to make a decision whether or not a particular action conforms to the local policies taking into account the reputation of the entity at the other end of the communication channel.

3 Anonymity

There has been considerable work done on anonymity [3, 16, 17, 6, 5]. There have been various ways this has been implemented. I think there are at least two kinds of anonymity

3.1 Complete Anonymity

What we mean by this is that everything is under the control of the principal whose identity is to be protected. The principal is responsible for maintaining and implementing the mechanisms necessary for preserving his anonymity. The principal doesn't depend on others to meet this requirement.

3.2 Partial Anonymity

In this scheme the responsibility for implementing the anonymity requirement rests partly on a third party. We can say what our policies are but the information is in the hands of the third party and we trust them to implement our policies. An example of Dr. Ross Anderson is we trust our doctor to keep our medical records and that means that he can manipulate with it. If he is malicious then he can leak information about me and we cannot do anything about it. We do not trust our friends to keep our medical records and this means that even if he wants he cannot do any harm to me. In this scenario a Trusted Third Party is a third party that can break your security policy. In the following sections we will have a look at some current approaches to anonymity.

4 Some Current Approaches To Anonymity

5 TCPA

There have been suggestions from the TCPA [19] about AAWS *i.e.* authenticated anonymity webservice where by the system goes to a trusted third party and then AAWS will assert that the platform is unique but will not reveal anything that can be used to track back to the system. This approach is similar to where we trust somebody to make proper use of our personal information. Below we will discuss an approach that builds on this approach.

5.1 On iPrivacy and Lumeria

These are systems that were developed to protect user's personal information from companies. The first one is iPrivacy a U.S based company where by the user downloads software from the website of iPrivacy. This software encrypts the user's personal details, creates a fictitious identity and a one time credit card number which is matched by the credit card company with the real credit card number and then the goods are delivered at an address chosen by the customer. Another is Lumeria where by all the information is stored with Lumeria and the customer accesses the seller via a proxy server of Lumeria and can then buy goods online. These schemes can be compared with the example that we have mentioned earlier where we trust our doctor to keep our medical records safe.

we have no verifiable proof about the integrity of the software downloaded over the net or the integrity of the company storing our personal information. So these kinds of schemes we think are vulnerable to abuse in the same way as the previous ones where we keep our information with the selling websites.

5.2 Chaum's Digital Cash

The scheme proposed by Chaum speaks of anonymity [6]. The user goes to the bank sends a signed request; the bank credits the account of the requestor after checking the signature. In this scheme the user generates the note number which the bank cannot see and that's how he says that even the bank cannot track the spending habits of the user. There are also proposals of implementing an observer in the representative of the user where by it checks against double spending. As we have mentioned later on that this scheme cannot prevent transfer of credentials which we do not require in case of notes but we certainly need them for driving licenses, medical prescriptions etc.

5.3 Pseudonyms

This was proposed by Ron Rivest and others in [12]. Here the user generates the private and public keys and so do the organizations. The user goes to the organization it wants to communicate generates a nym which is a function of the secret and non secret keys of the user and the organization. After the nym generation the credential is generated and given to the user. The user has different credentials for different organizations and it can use credentials issued by one organization while dealing with another organization. The scheme we present is different from these as ours is based on completely new architecture. In our scheme the user is globally represented by his/her public key rather than by different nymes. Both our scheme and Rivest's scheme overcomes the problem where it was easier to share credentials.

5.4 Mixes and Crowd

Mixes was proposed in [4]. Here the user goes to a mix server gets a ticket and using that communicates through a series of mix servers. That makes it difficult for traffic analysis and spoofing attacks. The communication between the user and the mixes are encrypted using keys that the user obtains from the mix servers. We would like to mention here that it costs n public key encryptions and decryptions when we have n-mixes which is sometimes not desirable. The mix servers scramble the messages, reorder them and thus prevent taffies analysis. It also requires that dummy messages are sent regularly by the proxy on the user's side to counter traffic analysis. We think that the goal of Mixes is different from the issues we are trying to address. We can use mixes below our scheme to guard from traffic analysis but we will discuss that later.

Crowds were proposed in [14]. Here they hide the action of one user within the actions of several users. They put some constraints on the browser by disabling the Active x and Java. Their goal is again to prevent from traffic analysis and unlinkability between the sender and receiver. Each user runs a jondo on his/her machine which contacts the blender server to request an entry into the crowd. All the communications from the user to the web servers is sent via the jondo. The jondo contacts a random jondo among the path and then forwards the request. The randomness in path selection is the key feature of crowd. Similar to mixes we would like to say that Crowds can be used below our proposals to provide sender and receiver unlinkability and guarding against traffic analysis.

6 A Palladium Based Architecture for Online Anonymous Transactions

6.1 Background

We wish to prevent anybody from getting hold of our personal information, like name, address, social security numbers, and credit card numbers etc. One of the instances when we reveal our personal details is when we buy goods online. So our quest has been to find an alternative where we have complete anonymity from the seller and still buy goods from him or avail the services we pay for. We have devised a token-based scheme whereby it is possible to preserve anonymity from anyone we do financial transaction with over the web. There will be only one entity with which we share our personal information. If all other parts of the system work properly and still if there is a leakage then the source of leakage can be concluded with reasonable certainty. Here we will discuss the protocol and taking a particular example of selling of goods over the net we show three stages. After outlining the protocol we present proposals for delivery of goods and informing the customer when the customer doesn't have a physical address with the seller or the post office. Then we present the generation of tokens and propose a token format. The interesting part of this scheme is that we cannot transfer credentials and the credentials can only be used only by those to whom it is issued. We have seen in Chaum's protocol that though it was tough to double spend but we can always transfer credentials or digital cash [6]. The bank issues a note without maintaining a record as the note number is blinded and then we can always share the note with friends. But this is not desirable where we have credentials like driving licenses etc. Ron Rivest proposed a scheme where it is not possible to transfer credential [12]. The scheme we will propose also makes it difficult to transfer credential but it is different as it requires that all the credentials are digitally signed by the requestor. Rivest proposes a scheme where a user is identified by a nym generated between the user and the credential- issuing organization where as in our scheme the nym is

the public key of the user. We propose a digital equivalent of a credit card here. In our scheme the payment method is not tied or dependent on the identity of the payer al though the payer needs to authenticate him/her before he/she uses the cards, but that doesn't reveal their identity to the seller. This is different from the traditional credit cards.

6.2 Assumptions

The approach presented here uses a Palladium machine. Palladium is a new trusted computing venture by Microsoft, Intel and several other companies. The goal is to have a trusted computing environment which is implemented by a combination of hardware and software. Thus the user can be sure that the banking application is not being driven by a virus or the passwords are not heard by a rogue application. The user trusts that a palladium will implement his/her security policies. The hardware is called the Fritz chip which is a dongle soldered near the motherboard contains an AES key along with a pair of RSA keys which also acts as unique identifiers for the system. The trusted software element is called nexus. The hardware provides crypto services to nexus and recursively nexus provides these crypto-services to the applications above. The applications are known as NCAs or Nexus Certified Agents. The proposals also speak of a secure input and output so that the user is sure that the banking application is free from the influence of any virus, or a rogue application viewing the buffer while he/she is typing the password. The nexus communicates with the application we use to sign statements, ensures that the legitimate user is signing the credential or a statement before signing statements. Every program running in a palladium machine has a cryptographic identity which may be its SHA-1 hash value and can be checked for integrity. The applications are free from the influence of malicious applications running in the system and Palladium also claims of performing that by implementing curtained windows. The Palladium system is capable of maintaining logs of transactions, performing encryption and decryption and generating random numbers. Our scheme relies heavily on a third party called AAWS [17] which asserts that the platform is unique and the request is legitimate without revealing any other details that can be used to trace back to the user.

6.3 AAWS

AAWS acts as the third party which authenticates two communicating parties to each other and guarantees that they are not talking to an imposter. We use AAWS to authenticate two communicating parties to each other and yet preserve anonymity. In our proposal we wish to prevent anybody to get information on the buyer or on a person to be general. Here the key embedded in the Fritz chip acts as the pseudonym. It should not be possible to trace back to the owner of the key provided the bank is not malicious or the AAWS is not malicious.

We have to have something where there is some stake involved on the part of the bank and the AAWS to keep our information secret. So far as the bank is concerned the credibility of the bank is involved which directly affects its market share and it is unlikely that the bank is ready to risk its market share. So far as the AAWS is concerned we would propose it to be the manufacturer's certificate which asserts that the platform is unique but will not disclose anything that can be used to trace back to the user. AAWS can also assert that platform can be trusted for certain purposes. The manufacturer's certificate is hardwired in the secret box and can only be activated by the secure path which is only under the control of the legitimate user. The manufacturer's certificate cannot be forged. We can have a hash of the manufacturer's certificate encrypted under the public key of the receiver which is done automatically every time it is sent. If someone tries to tamper with the certificate then that can be easily detected from the hash value. The AAWS contains the legitimate public key assigned to the machine and then if the communications that accompany that particular AAWS can de decrypted using the public key mentioned in the AAWS then that proves that the request originated from the legitimate person and didn't originate from an imposter. In the protocol mentioned below all the communications accompany the AAWS. How they are used is mentioned when we discuss token issuing, or credential verification etc.

6.4 The Protocol

Here we will outline the various steps of the protocol and state the format of the messages at each step. There are three principal entities involved in the transaction namely the bank, the seller and the customer. All communications between these parties take place across a network channel.

1. B represents the Bank
2. S represents the Seller
3. C represents the Customer

We represent a signed message M as M^{-}_{EB} where M is signed by the private key of the bank. When we write AAWS we mean the manufacturer's certificate by that. The manufacturer's certificate from the bank looks like AAWSBank. Where we write $h(x)$ we mean a SHA1 hash of x. The various steps of the protocol are as follows: Step 1: The buyer goes to the bank and asks for a token that it can use to buy goods online. The buyer's request R is signed by its public key.

1. $C \longrightarrow B : \prec R^{-}_{EC}, AAWSBuyer, h(AAWSBuyer) \succ$

The request R contains the SDSI identity of the nexus which generated the request which is globally unique and signed by the secret key of the buyer. This

ensures that credentials can be kept secret and used by the legitimate user. This is achieved in a palladium machine with the help of a Seal/Unseal function. The process of credential issue is discussed later on in this paper. It also contains a date and time to guard against replay attacks. R also contains a statement about the nature of the request. The buyer can be figured out from the AAWS. The buyer is known to the bank by the key mentioned in the AAWS which is unique by virtue of the algorithms used to generate key pairs. The bank after going through its normal procedure issues a token to the buyer.

2. $B \longrightarrow C : \prec T^{-}_{EB}, AAWSBank, h(AAWSBank) \succ$

T contains the pseudonym of the subject, to whom this credential is being issued without revealing any other information. The bank also specifies the credit limit of the buyer and assures the seller that the bank will pay the amount spent by the legitimate holder of the token. People may argue that the bank can have track of the spending habits which it can use for many purposes. What we would reply is that when we open an account with the bank we enter into an agreement with the bank where it is agreed that our personal information won't be shared without our consent. Now since the bank is the only place where our personal details are stored any leak of information is easy to detect. In the past we had information about us at all sorts of places so it would have been difficult to detect from where the leak has been. The threat we are guarding against is malicious use of our personal information. In the light of this threat we trust the bank and believe the bank is trustworthy as any leaking would affect its credibility and market share. We doubt that any bank is ready to risk that.

Step 2: The buyer goes to an online seller and asks for the credentials of the seller before it proceeds with any transaction. The buyer sends a signed request to the seller.

3. $C \longrightarrow S : \prec R1^{-}_{EC}, AAWSBuyer, h(AAWSBuyer) \succ$

The request R1 contains a request for the credentials of the seller which may be whether or not the seller is authorized to accept credit cards and other information that can be used for decision making by the compliance checker. The seller sends a signed statement back to the customer.

4. $S \longrightarrow C : \prec M^{-}_{ES}, AAWSSeller, h(AAWSSeller) \succ$

The message M contains the credentials of the seller as well as other information asked by the buyer. R1 and M can differ with different situations.

Step 3: At this point the buyer gets back to the compliance checker before proceeding further with the seller. We have described a reputation assessment scheme and the compliance checker below. We have identified at the beginning the need to have a reputation assessment scheme and how it works is mentioned

in section 7. The buyer at this point can also refer to other sources for information which is done by the compliance checker before the compliance checker arrives at any decision.

Step 4: If the compliance checker is happy with the seller and gives a positive decision then the buyer and seller at this point agree to a transaction. They generate a transaction description which is unique and let's call it T. The seller sends a signed T to the customer.

5. $S \longrightarrow C : \prec TD^{-}_{ES}, AAWSSeller, h(AAWSSeller) \succ$

TD contains the date, time, description of the goods, the value of the transaction and the identity of the agreeing parties to the transaction. T helps us to guard against a situation where the buyer pays for X and the seller generates a receipt for Y. This raises a dispute and then it might be hard for the buyer to convince to a third party that he intended to buy something else.

Step 5: The buyer presents the token it received from the bank to the seller.

6. $C \longrightarrow S : \prec T^{-}_{EB}, AAWSBank, h(AAWSBank)ECES, AAWSBuyer, h(AAWSBuyer) \succ$

The seller commits the transaction and generates a digitally signed money receipt and sends it to the customer with the same details it had in TD along with a receipt number.

7. $S \longrightarrow C : \prec Receipt^{-}_{ES}, h(Receipt), AAWSSeller, h(AAWSSeller) \succ$

The customer checks the receipt it gets against the TD it received earlier and thus ensures that the receipt is for the goods it selected and not for something else. Without having a valid receipt the customer instructs its bank to ignore a particular transaction. The buyer doesn't gain too much in issuing a false denial that he/she didn't receive any receipt from the seller. The seller gains where as the buyer intends to buy something else but the seller issues a receipt for something else. To guard against this situation we have the step where the seller signs the transaction details to the buyer. Coming on to another related threat which the users have while buying online that do we get the goods we pay for. Now if we retain the receipts we receive which is digitally signed by the seller then that can be used to prove that we indeed paid for the goods which we haven't received though that is not sufficient. We may hide the goods somewhere and then claim, but this can at least start an investigation which will later on involve the post office or courier companies. The mechanisms required to support on the part of the courier companies are proposed below. The seller also maintains a log of the transaction so that he can claim payments. If he doesn't then he doesn't get any money. So eventually if the seller is at fault then it can be traced from the seller's log that the buyer has paid for the goods he/she didn't receive. This way we can preserve anonymity as well as present proof when ever there is a dispute. Here both the parties have to maintain a log for their own interest which can go against them if they try to act maliciously.

Moreover if the seller wants to tamper with his evidence with after he receives the payment then the bank has the proof that the seller asked for the payment which the seller does by digitally signing the transaction from his Palladium machine which unambiguously states that the seller and nobody else, asked for the payment. Similar is the case with receipts where the receipt is signed by the secret key of the seller which can only be done by the seller and the seller cannot deny that later on. The fact that the seller uses a particular machine can be figured out easily as in cases of Palladium we need a third party to prove that this Palladium box belongs to the one who is claiming to own it. This very concept raises another debate, is whoever is in possession of a system is he/she responsible for any and every malicious use of the system. To address this controversy we can have some secret sharing between the application we use sign statements and me or at least between the trusted part of the machine and the intended user. Only the legitimate user can have access to the trusted part of the machine and nobody else can access the trusted part of the system.

6.5 Delivery of Goods

A very important question is if we keep our address secure then how do we get the goods delivered to our address? There have been alternatives proposed and implemented in this regard. A company in U.S which protects customer's privacy does so by getting the customer to choose an address of delivery. The customers can have arrangements with the local post office and have post box numbers. There can be new business opportunities where by the post office enters an agreement with the clients to maintain a post box. There can be various ways of implementing this and we are not going to comment on that here. But we would mention here that there should be mechanisms of logging the details of the goods coming for a particular client in his/her post box and when those goods are collected by the customers. The seller keeps an acknowledgement similar to proof of delivery that comes from the last post office in the chain for its own records and another one from the first one in the chain, to prove that the customer has in fact collected the goods and that the seller posted the goods to be sent to the customer respectively. The post offices should also keep a similar log of the goods as they enter and when they leave. These systems are already in place so can easily be implemented or extended. We can have a system where every post box holder has an account with the post office and when he/she comes to collect the goods they do so after entering their details and the secret they share with the server at the post office. There should also be a unique identifier for the customer and thereby the customer doesn't reveal any other details about his/her whereabouts. It may be that the post office recognizes the customer by his/her public key. The secret should be used to open the door of the locker of the customer and as soon as he/she opens the door the date and time of collection is recorded and a proof of receipt is generated as mentioned above which is forwarded upwards in the delivery chain to

the seller and also stored with the post office. The proof of receipt can only be generated by the customer's key the reason we will see in the next few lines. There is something very important here. The employee who is at the post office has also access to the locker. He can always open the door and run away with the goods. This can lead to a dispute for which neither the customer nor the seller is liable. To counter this threat what we propose is a system where the customer and the employee have different keys both of which will open the door. Based upon the key used the door records who opened the door and at what time. Like the one we have at networks where we have user passwords and administrator privileges. Since the proof of receipt can never be generated by the employee's key and in absence of that in the log of the post office the customer can always prove that it didn't collect the goods. We also ensure that a corrupt employee can never defraud the customer by a scheme of display in the next section. We are also able to address another situation where by the customer forgets or doesn't generate the proof of receipt in other applications. Here when ever the customer comes to collect the goods and opens the lock a proof of receipt is automatically generated whether the customer wants or not. There is another interesting issue here. We check our pigeon hole everyday irrespective of whether there is any mail or not. Here whenever the customer opens her lock receipts will be generated. It might be that such receipts are generated when there are no goods waiting for the customer. How will the customer know whether the deliveries that are scheduled to arrive has arrived or not. We will address this in the next section.

6.6 Informing the customers

Here we will propose a scheme where by we can inform the customer about the delivery of the goods. We are assuming that the customers only come to look at their locks when they have ordered for goods online and only come after the period which is reasonable for the goods to arrive. We can have some display on the door of the lock where by the customer comes to know about the goods. The display can be manually set by the employee of the post office or it can be done electronically when ever a good is recorded in the system against the particular customer. The display can only be set off by the key of the customer. This also ensures that the employee can never set the display off, though he can set the display on. If the display is set on manually locally then there should be local arrangements that the display is set off locally. Even if the display is set on by an entry into the central system of the post office then also there is no harm if the arrangement is to set the display of locally when the key of the customer is entered. The display can also act as a physical verifier to the local authority that the customer hasn't collected the goods. It can also help us in unusual circumstances when there is an unusual delay on the part of the customer then arrangements can be made to physically inform the customer. There is a problem though. The customer is known by its public key

and nothing else. It is only the bank who has information about the customer. Arrangements can be made to deal with such unusual circumstances. There can be a criticism that the users have to go for collecting their goods. We would argue that this is small price for privacy. There can certainly be ways of addressing this problem may be using agents, but we can deal with that problem separately.

6.7 Generation and Delivery of tokens

We would like to have a SPKI certificate format based token with certain changes that can be used by the customer for paying for goods bought online. A very important part of the whole is auditing which is necessary in the event of any dispute between the bank and the seller. Auditing surely helps in resolving those disputes. Both the bank and customer are identified by their public keys although the bank also has a physical presence. The customer sends a digitally signed request to the bank along with the AAWS. The AAWS states that the request originated from the legitimate owner as it gives the public key of the owner and that can prove to the bank that the requestor is the legitimate owner of the corresponding secret key. The proposed token will contain the following fields

1. Issuer The public key of the bank can be the AES/RSA key of its chip.
2. Subject The public key of the customer can be the AES/RSA key of his chip.
3. Authorization This will state the upper limit fixed by the bank up to which a customer can spend.
4. Validity Here we can have a scheme as proposed by Rivest where he specifies three dates whereby; the credential is perfectly valid between dates1 and 2 should be accepted with caution between dates2 and 3 and should not be accepted without verification after date3.

We have programs with cryptographic identities. We would like to propose a scheme whereby a program is identified by a SDSI name where the AES/RSA key serves as the root name and the SHA-1 hash of the program is the local name and both of them makes the program uniquely identifiable. The request from the customer to the bank will contain the SDSI name of the nexus that requested it. The credential is sent by the Seal/Unseal function to the target nexus and the binding between the customer and the particular machine is done by the AAWS. Only the intended nexus is able to retrieve the credentials and no other nexus will be able to get the credentials. The customer's Palladium machine uses the Unseal function to retrieve the credentials and that is presented by the NCA responsible for such transactions to the seller online after digitally signing the credential with the buyer's key. The bank also has a Palladium machine and it has its keys which it uses to sign the customer's credentials.

6.8 Credential Verification

Here we will dwell on another important aspect of credential verification and propose a scheme for the above situation when we are buying and selling over the net. We have already mentioned the questions that come to our mind when we do transactions online. The user would like to know whether the seller is authorized to do business over the net or not, is it authorized to accept the tokens or not etc. We can have a similar model to what we have while verifying the buyer. The bank can issue a similar SPKI certificate which is signed by the bank and again the seller can be identified by the AES/RSA key of its Palladium machine with the help of an AAWS. The seller can present its certificate to the buyer by digitally signing it which can be verified by the buyer. With such verifications we eliminate the need to have online verifications which are very costly in low valued transaction. The format of a certificate possessed by the seller can be as

1. Issuer The public key of the bank can be the AES/RSA key of its chip.
2. Subject The public key of the seller can be the AES/RSA key of his chip.
3. Authorization This will state that the particular trader is allowed to accept tokens issued by banks.
4. Validity Here we can have a scheme as proposed by Rivest where he specifies three dates whereby; the credential is perfectly valid between dates1 and 2, should be accepted with caution between dates2 and 3 and should not be accepted without verification after date3.

The seller sends digitally signed credentials along with the AAWS. If the public key that is mentioned in the AAWS can be used successfully to decrypt the request then that proves that the seller is the legitimate owner of the credentials. We assume that the keys are not compromised and digital signatures uniquely verify the principal with whose public key we decrypt the signature. No one else can generate the signature. Similarly the customer sends a digitally signed request along with the credentials to the seller along with the AAWS which states unambiguously that the request originated from the legitimate owner of the corresponding public key and credentials. After the credentials are authenticated we take a decision based on our policy whether we can carry on with a particular transaction. A mechanism for compliance checking is mentioned below.

6.9 Key Management

Perhaps the most important question that surrounds Palladium today is of key generation and who is going to generate the keys. A reasonable way to do is to let the user generate the keys and then put them in the box and seal this

only in the presence of the user. The box can be sealed once and can never be reopened. The box contains the manufacturer's certificate along with the keys. What we mean by the box is Fritz chip. Another important issue is revocation. This is similar to answering the question, how long are we willing to let the world believe something that is not true. A problem is that since the box can be sealed once we also need to throw the box away but the box contains the manufacturer's certificate. We would like to make some assumptions here. The keys are reasonably secure against cryptographic attacks within their cryptographic lifetime. Another is that the users won't share their keys as that would lead to a situation where the user suffers substantial losses. We have mentioned that in the credentials we can three dates as proposed by Rivest. We will always discard the keys after their cryptographic life time irrespective of whether they are compromised or not. If the keys are compromised then we have to generate new key pairs and reprogram the Fritz chip. We have proposed in our scheme that all requests should accompany the manufacturer's certificate so even if the keys are stolen out of a Palladium machine the victim cannot be impersonated by the attacker as the manufacturer's certificate in the attacker's machine cannot be changed without tampering with the chip. We are not saying that key compromises should no longer be reasons for worrying but for this example it will be hard for the attacker to impersonate as the legitimate user.

7 Reputation Assessment

The system we have proposed above helps us to maintain anonymity, we can check the credentials of the party we are transacting with and above all now we have a means for uniquely identifying a machine and the owner of the machine and what is he using it for with the help of AAWS. But this is not all we wanted to address in the sphere of E-commerce. There remains a very important issue that needs to be addressed. As a customer we worry not only about our credit card numbers but also other issues related to the level of service we get. Do the customers enjoy the same right online as they do while dealing with the shop down the road, Can they get a refund and a whole lot of other issues. We have already mentioned independent trust entities and shown that these do not serve our purpose as somebody can always get a certificate and later on turn dishonest. We would like to have up-to-date information about the dealings of a particular seller. Normally we get information from friends, relatives some of whom we trust about information and some of them we do not trust what they are saying. Gossip is a very powerful way to build impressions about somebody and we will try to emulate that in the electronic world. Each source will have its own weight and those will be calculated while taking a decision. We can always move sources up and down in the list *i.e.* increase and decrease their weights. This makes the decision making process very dynamic. The scheme we are proposing here also can be used for new entrants in the fray in building their

credibility. We would like to mention here that a particular principal qualifies as a source for another principal when both of them have the same interest so far as the outcome of the transaction is concerned. Let's take an example where a insurance company who is going to pay its customer for any losses in the event of any dispute with the seller, and a customer of the insurance company both desire that the goods arrive properly maybe for different reasons. Let's think of a situation where Alice believes what Bob says about x, Alice's belief about x is dependent on three things

1. a. Bob is saying exactly what he believes about x and not misleading Alice

2. b. Bob's knowledge on x can change or decrease over time

3. c. Bob and Alice have the same interest so far as the outcome of the transaction is concerned.

We are assuming that Bob is not misleading Alice and is providing her with correct information. We will express the degrees of reputation on a scale of 100. Suppose Alice assigns the weight 20 to Bob. The weight Alice assigns to her sources of information depends on several factors like how long she knows them, their performance in the past. We can have a separate ratings scheme for that. Bob's impression of a particular seller is calculated on the basis of his experience which can be calculated as, suppose Bob has used the website 10 times and out of that he was satisfied by their service 5 times, and the rest of the time he had some problems like delays or goods received in an improper condition etc. Suppose that our source is our insurance company then it will have different parameters for making decisions like the total number of claims received etc. Bob also takes into account the time it last used the service. If it's too old then that affects the overall weight assigned by Bob, if it's not very old it affects the overall weight but less than the previous case, but if it's fairly new then we do not let this factor affect the overall weighting. The reason being that the services of that particular seller might have deteriorated since Bob used it last and so if Bob's impression is pretty old then Bob should take into account this fact while assigning weights or this seller. Again if this is our insurance company it will have a completely different way of giving its opinion. If the last claim date is pretty old then it is going to ignore the fact that this particular seller had a bad past but if a claim date is fairly new then that is going to affect the decision of the insurance company about a particular seller. This flexibility helps the system to accommodate decisions from various sources. Going back to our example here we are assuming that the information Bob has is fairly new. Now going by the straight logic we could have given it 50 on a scale of 100, but here we take into account those negative experiences of Bob and assign 40. When we look to Bob for his opinion we get 40 from that we derive our decision whether to visit the website by dividing 40 by 20 and the result is 2.

If we have more than one source or several sources we calculate our opinion as above and take the mean of those and get a value. When we have it we compare it with a threshold and if it's greater than the threshold then we do carry on with the transaction else abandon it. If we carry on with the transaction and get good results we increase the weight we assign to Bob else we reduce it in the event of a negative effect. To play it very safe we can have linear increase but multiplicative decrease, but that can be customized with the situation at hand. Thus we can have a very dynamic system that models the real world fact that reputation changes with time. More over as we have already mentioned above we are having Palladium based architecture and here everybody can be identified by the AAWS. This makes sure that we are getting the information from the sources we want and not from a spoofed server. Having said all this we would like to say that we are making a very big assumption that all the sources will pool in proper information and not misguide me. So we have a proposed linear increase and multiplicative decrease. If we find a source acting maliciously we can always decrease the rating and move away to another source of information. We have also mentioned at the beginning that a principal is qualified to be a source for another one if both have same interest so far as the outcome of the transaction is concerned. The actual implementation and the figures will depend on the implementation. We should have a policy of not using any source with a weight below a certain threshold value for decision making process. This threshold value can increase and decrease according to the threat model and the situation. The scheme we have proposed here we think can be used to model the relevant aspects of the real world perception of reputation. We can also extend this scheme and allow for new sellers to come in and pool in trust information which can then be authorized by third parties who can act as our source of information. The buyers can refer to third parties who can help them to make a decision. A similar scheme was proposed by Jean Bacon and others of the University of Cambridge where they have Guarantors who helps the new entrants to build their credibility over time [22]. The sources of information can be the insurance companies, friends or other trusted sources. As was proposed in their scheme we can have a system where the guarantors are liable for wrong information. On the one hand we have guarantors who only benefits if there is a positive outcome as their interest is same as the buyers interests and one the other hand they suffer loss twice when they provide incorrect information as they lose business and also their weight gets decreased. This affects their credibility as well. To conclude we think that the scheme we have presented above helps us to generate an opinion from various sources when we do not have information and at the same time guards from the threats where one or more of the sources may be malicious. This also helps new entrants to build their customer base by pooling in information about them which can be verified by the buyers. We have mentioned the outline the need to have a reputation assessment system at the beginning of this paper and here we propose a model which emulates the way reputation is built in the real world.

8 Compliance Checking

After the credentials are verified we would like to refer to the local policy database for a decision on whether to carry on with the transaction or abort it. The compliance checker is a database of policies which are queried by different applications via different interfaces. The application we use to buy goods online is different from the one we use for emails and each of them has to act according to local policy so each of them queries the local compliance checker. The compliance checker can only be programmed by the local administrator and it has different modules for different activities. We can have one for content verification, one for doing online financial transaction and another for emails. The application calls the corresponding module and then presents it with the credentials and acts according to the reply. The onus of parsing the credentials and interpreting them rests on the application as we have seen in keynote and policymaker. The interesting part here is when we incorporate the reputation assessment model that we have mentioned above into the compliance checker. Whenever the compliance checker doesn't have information about any particular request it can always request the corresponding application to get opinions from others. So far as the sources for each activity are concerned that will be decided by the local administrator along with the weights. The compliance checker has full knowledge about the weights assigned to each of them and then calculates the decision as mentioned above. The threshold value for each activity is also mentioned in each module of the compliance checker. After the application has finished with the transaction it should communicate the appropriate module about the outcome of the transaction so that the weights can be updated accordingly by the compliance checker. The compliance checker operates as a stand alone system apart from the NCAs in the trusted part and all the other applications query the compliance checker for an opinion.

9 Conclusion

When we were analyzing web based transactions we found that anonymity as well as the reputation of the seller is a matter of serious concern for the buyer. The mechanism presented here can be used for anonymous transactions over the net. We have dealt here with a particular transaction but we can generalize this system for other applications too. We can also make decisions before carrying out any actions based on our local policy and it is being ensured that the compliance checker also is able to give decisions when it doesn't have required information. A mechanism which emulates the real world process of building reputations is incorporated into the compliance checker and this makes the compliance checker proposed here from the compliance checkers found in the trust management systems. The mechanism presented above also makes it difficult to impersonate due to AAWS and transfer credentials. It is clear now that we are

yet to have means that guarantees that our information will be properly used and it is better not to disclose sensitive information at all over the net. We use a compliance verification system to implement local policy. We have proposed a digital equivalent of a credit card but unlike the traditional credit cards they do not have our identity. These are issued to our pseudonyms. The legitimate holder of the card needs to verify every time they use the card but that doesn't reveal their real identity. This independence of the payment method on the identity of the payer is useful in many transactions as we have shown here. The mechanism for anonymous transaction coupled with this compliance verification makes it different from all the current and past approaches we have seen. To be precise the system proposed here is a model for anonymous transaction with a compliance checker incorporating a reputation assessment system.

References

[1] Ross Anderson. *Security Engineering.* Wiley, Inc, 2001.

[2] Matt Blaze, Joan Feigenbaum, J. Ioannides, and Angelos Keromytis. The keynote trust management system. *Request For Comments Series*, (2704), 1999.

[3] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. *Proceedings of the* 9th *ACM conference on Computer and Communications Security*, pages 21–30, 2002.

[4] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[5] David Chaum. Security without identification: Transaction systems to make the big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[6] David Chaum. Achieving electronic privacy. *Scientific American*, pages 96–101, 1992.

[7] Yang Hua Chu. Trust management for the world wide web. Master's thesis, M.I.T., 1997.

[8] Carl Ellison, B. Frantz, Ronald Rivest, B. Thomas, and T Ylonen. Spki certificate theory. *Request For Comments Series*, (2693), 1999.

[9] Carl Ellison and Bruce Schneir. Ten risks of pki: What you are not being told about public ky infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[10] Tyrone Grandison and Moris Sloman. A survey of trust in internet applications. *IEEE communications Surveys Tutorials*, 3(4):2–16.

[11] Loren M. KohnFelder. Towards a practical public key cryptosystem, 1978.

[12] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. *Proceedings of the* 6th *Annual International Workshop on Selected Areas in Cryptography: Lecture Notes in Computer Science*, 1758:184–199, 1999.

[13] Mary Ann Patton and Audung Josang. Technologies for trust in ecommerce. *Proceedings of the IFIP working conference on Ecommerce*, 2001.

[14] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and Systems Security*, 1(1):66–92, 1998.

[15] Michael Roe. *Cryptography and Evidence.* PhD thesis, University of Cambridge, 1997.

[16] Paul Syverson and David Goldshlag. Unlinkable serial transactions: Protocols and applications. *ACM Transactions on Information and Systems Security*, 2(4):354–389, 2000.

[17] Paul F. Syverson and Stuart Stubblebine. authentic attributes with fine grained anonymity protection. *Proceedings of the* 4th *Annual Conference on Financial Cryptography: Lecture Notes in Computer Science*, 1962:276–294, 2000.

[18] Curtis J. Taylor. Private demands and demands for privacy: Dynamic pricing and market for customer information. Technical report, Department of Economics of Duke University, 2002.

[19] TCPA Whitepaper. Trusted platform modules strengthen user and platform authenticity. *Available at http://news.bbc.co.uk/1/hi/business/2662491.stm*, 2003.

Anonymous Authentication

Partha Das Chowdhury, Bruce Christianson, James Malcolm
Computer Science Department
University of Hertfordshire
England
{P.Das-Chowdhury, B.Christianson, J.A.Malcolm}@herts.ac.uk

January 31, 2005

Abstract. The contribution of this paper is a mechanism which links authentication to audit using weak identities and takes identity out of the trust management envelope. Although our protocol supports weaker versions of anonymity still it is useful even if anonymity is not required due to the ability to reduce trust assumptions. We illustrate the protocol with an example of authorization in a role based access mechanism.

1 Introduction

Authentication, authorisation and audit are three traditional concerns in building a privilege management infrastructure (PMI); the purpose of authentication is to identify a particular user and verify that a user is who he/she is claiming to be; the goal of authorisation is to provide access for certain users to certain resources based on predefined business rules; and an audit trail links actions to principals retrospectively. Traditionally, authentication is based on permanent credentials linked to a fixed long-term identity and authorisation is linked to audit via the authentication mechanism explicitly using the same permanent credential and identity. Privacy is not an explicit goal of traditional authentication/authorisation mechanisms [10, 9, 1]. This we believe is not quite right even if privacy is not a requirement; such approaches compel users to enter into unneccesary trust relationship with parts of the system infrastructure. For example services trust Kerberos' judgement about the identity of the requestor.

In this paper we present an authorisation mechanism which takes identity out of the trust management envelope. Although our protocol supports weaker versions of anonymity, but it is useful even if anonymity is not required at all, because of the ability to weaken trust assumptions; the approach we present here allows users to control the risks they are exposed to rather than forcing them to enter into an unnecessary compulsive trust relationship with the system infrastructure. Similarly, services also do not need to trust the authentication mechanism of a third party.

Role based access control (RBAC) is one mechanism for building a PMI. The main idea behind RBAC is that the permissions are associated with roles instead of directly with users. The usual approach for role activation is authentication by role members using fixed credentials like a key certificate or a role certificate. It is well known that repeated use of fixed credentials enables an adversary to correlate the transactions of any particular user of a system. The problems of using fixed credentials have been well understood in the world of commercial transactions, and this has led to the advent of several anonymous payment mechanisms. We focus on privilege management infrastructures where key certificates are extensively used for making authorisation decisions. The certificates when used repeatedly also enable an adversary to correlate all the access requests of its victims. Our intention is to show some alternative ways to link authentication, Role Based Access Control, Trust Management using transient identities (surrogates) in an auditable but unlinkable way. What we are suggesting here is creating some conceptually neat layers in trust management approaches using explicit labels and policies. Our approach can coexist with other traditional non-anonymous approaches.

This paper is organized as follows: Section 2 gives an outline of what we mean by transient identities. In section 3 we present a brief overview of trust management systems which is followed by our design goals and assumptions in section 4. This is followed in section 5 by the protocol for anonymous authentication which is the main contribution of this paper, which is followed in section 6 by conclusions.

2 Surrogates

Surrogates [6] are transient numbers generated from a parent number (such as a bank account number, credit card number, or social security number) by some mathematical function. They are used to preserve the anonymity of the legitimate owner of the parent number. The party who accepts the surrogates in exchange of goods or services bears the risk of fake surrogates [4]. There is a difference between blinded credentials and the way we generate surrogates here. Blinded credentials need to be certified by some authority [5, 13], and can be reused [5], or the user needs to get a new credential issued after every single transaction [13]. Surrogates are for single use, doesn't need to be certified and we use and throw them away after every single transaction. Blinded credentials can be verified by the acceptor if the acceptor knows the public key of the issuer. Surrogates are verified by proving knowledge of a secret that was used to generate the surrogate without revealing the secret.

The surrogates in our protocol are generated by the user and although the issuer can verify them but the issuer cannot masquerade as the user. The surrogates for a particular user are generated from the parent public key of that user. The surrogates have corresponding secret keys like the parent public key. These secret values are generated from the secret key corresponding to the parent public key and together the public surrogate and its corresponding secret

value forms the surrogate pair. The surrogate pair for a public private key pair can only be generated and used by the legitimate owner of the corresponding parent public key. The requirements for surrogates can be summarised as:

- Un-correlatability - Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction.
- Unforgeability - It should be hard for an adversary to generate and use credentials belonging to another user.
- Verifiability - The verifier who might be the owner of the resource or an access granting service should be able to able to unambiguously verify that the requestor is who he/she is claiming to be.

2.1 Generation and Use of Surrogates

Our key generation method is similar to the traditional one described by Diffie and Hellman in [8]. Cathy selects a secret $s_c \in 1 \ldots (P-1)$ and generates her public key as

$$X = g^{s_c} \bmod P \tag{1}$$

The surrogates are generated by modular exponentiation of X using an exponent τ where $\tau \in 1 \ldots (P-1)$. The secret value corresponding to a surrogate is generated by modular multiplication of the exponent τ with the secret s that was used to generate X. The initial value (τ_0) of the exponent τ is supplied by a (partially) trusted third party such as the user's bank. The subsequent values (τ_i) of the exponent τ, used to generate the surrogates and their corresponding secret, are generated by the user using the linear congruence equation,

$$\tau_i = (A * \tau_{i-1} + O) \bmod P \tag{2}$$

where A and O are selected by a third party and are fixed. Then, using τ_i, surrogate K_i^+ and its corresponding secret K_i^- for the i^{th} transaction are generated as:

$$K_i^- = \tau_i * s_c \bmod (P-1) \tag{3}$$

$$K_i^+ = g^{K_i^-} \bmod P = X^{\tau} \bmod P \tag{4}$$

To generate surrogates corresponding to a user, first the exponent is calculated by equation 2 and then the secret is calculated by equation 3, finally the surrogate is generated from the secret by equation 4. The third party sends the user A, O, and the initial value τ_0 of the exponent so that the user can generate his/her surrogates.

To use a surrogate K_i^+ for transaction i a user proves knowledge of the corresponding K_i^-. Only the legitimate owner of X and s_c can generate and use surrogates corresponding to X. The third party cannot masquerade as the

legitimate owner of X as the corresponding s is secret but can resolve disputes and can correlate transactions conducted with surrogates generated from X, thus facilitating auditing. However various transactions conducted by the same user cannot be correlated with each other by an adversary at the point of use of the surrogates. Learning one set of K_i^+, K_i^+ values doesnot help the attacker in any way, neither can the attacker generate future K_i^+, K_i^- values nor can the attacker get the secret value s_c.

3 Trust Management Systems

Trust management systems helps applications answer the question "Can I trust this public key for this purpose"? Keynote [2] is the latest version of a set of trust management approaches that came from Matt Blaze, Joan Feigenbaum and John Ioannidis. Keynote works more or less like a database query engine. It can function as a stand alone application interfacing with other parts of the system and helping them in making decisions. Let's lump these other parts of the system together and call them by a common name application. Whenever any application faces the question "Should we carry out this dangerous action "then it refers to Keynote for an opinion and based on that opinion it decides its future course of action. The application presents the Keynote trust management engine with a set of local policies that should be taken into account while taking a decision on this particular request along with the credentials of the requestor and details about the proposed action. If the proposed action conforms to the local policy then keynote advises the requestor to proceed otherwise Keynote advises it not to perform this action as it is against the local policy. Keynote acts as a compliance checker for the application. The policies are specified in the form of assertions and the actions are specified which are evaluated against these assertions.

4 Transaction Flow

1. Cathy is a subscriber of an electronic newspaper. The newspaper uses a role based authorisation model where Cathy is assigned the role of subscriber. The process is same for every subscriber.

2. The subscription department (SA) prepares and sends the information Cathy needs to generate her surrogates. At regular intervals the subscription department also sends the web server (WS) the information needed to authenticate Cathy and other subscribers.

3. Every time Cathy reads the newspaper she first authenticates herself to the web server (WS) using her current surrogate and upon a successful authentication her role is activated.

4. Cathy generates her own surrogate for the current session.

5. The web server authenticates locally whether or not Cathy is a valid subscriber. The role server consults the local trust management engine to ascertain whether or not Cathy is allowed to carry out the actions she requested.

6. The next time Cathy reads this newspaper she uses a different surrogate.

7. The SA cannot masquerade as Cathy.

4.1 Cryptographic and Infrastructural Assumptions

We assume the existence of a secure authenticated communication channel between Cathy and the subscription department of the electronic newspaper. This can be implemented by digital signatures where every communication between Cathy and SA are digitally signed. This provides authentication. The communication link can be secured by for example SSL/TLS. All communications between Cathy and the SA are denoted by $\longrightarrow$ in the protocol.

An anonymous communication channel between Cathy and WS is also assumed for our purposes. This can be implemented by Chaum's mix nets [?] or Mixminion [7]. The mix network makes it harder for an adversary observing the network to gain any additional information about the communicating partners beyond its a priori belief. Communications over the mix nets are denoted by $\longrightarrow_m$ in our protocol.

Our protocol depends on the difficulty of computing discrete logarithms in the multiplicative group Z_P^* where P is a large prime. P should be chosen such that (P-1) has at least one large prime factor, since if (P-1) has only small prime factors then computing discrete logarithm is easy [11]. We also assume that principals are capable of keeping their secrets secret.

5 The Protocol

The protocol can be described under two phases namely

1. Preparation phase

2. Transaction phase

5.1 Preparation Phase

Cathy sends the subscription department (SA) her public key.

1. $Cathy \longrightarrow SA : X = g^{s_c} \pmod P$

where g and P are public and g is the generator modulo P. SA sends Cathy the information she needs to generate her surrogates which is :

2. $SA \longrightarrow Cathy : \delta =\prec \tau_0, O_\tau, A \succ$

SA selects $\tau_0, O_\sigma, O_\tau \in Z_P^*$. We use the Linear Congruential Method to generate exponent τ_i where the offset O_τ and the modulus P are co-prime to each other and A is the constant used in the linear congruence method. τ_i is the exponent used for generating the surrogate. For each subscriber SA sends the web server (WS) m surrogates by repeating the following for each of the next m values of i:

$$\tau_i = (A * \tau_{i-1} + O_\tau) \pmod P$$
$$X_i = X^{\tau_i} \pmod P$$

3. $SA \longrightarrow WS : \prec X_1, ..., X_m \succ$

Surrogates of various customers are sent together in a batch. At the end of this phase SA only retains $\prec X, A, \tau_i, O_\tau \succ$ for every subscriber.

5.2 Transaction Phase

For transaction i Cathy calculates her i^{th} surrogate K_i^+ . The corresponding private exponent K_i^- of the current surrogate is also generated by Cathy.

$$\tau_i = (A * \tau_{i-1} + O_\sigma) \bmod P - 1$$
$$K_i^- = s_c * \tau_i \pmod{P-1}$$
$$K_i^+ = g^{K_i^-} \pmod P$$

To activate her role for her current transaction Cathy sends the web server her current surrogate K_i^+ and proves knowledge of the corresponding secret key K_i^-. If the web server finds a similar surrogate sent to it by the SA then it grants Cathy access. To be specific the WS activates relevant permissions for Cathy. To ascertain the validity of Cathy's request the WS consults the local trust management engine. An implementation incorporating trust management engines can be found in [12].

4. $Cathy \longrightarrow_m WS : K_i^+$

The next time Cathy reads the electronic newspaper she uses a different surrogate to authenticate herself. The role server can authenticate Cathy anonymously without the issuing authority being online. The issuing authority can periodically send Cathy's new surrogates to the web server. SA cannot masquerade as Cathy to the web server.

6 Conclusion

The surrogates can be verified locally by the web server without the cooperation of the issuing authority (SA). Various transactions of any particular user cannot

be correlated with each other without the active cooperation of the issuing authority. For purposes of dispute resolution the issuing authority SA in our protocol can correlate the actions of the subscribers.

Our protocol supports weaker versions of anonymity, and it is still useful even if anonymity is not required at all, because of the ability to weaken trust assumptions. The web server where principals request access does not need to trust the authentication mechanism of a third party. From the point of individual privacy users are no longer required to trust web servers with credentials linked to long term fixed identity using which an adversary can uniquely identify the user.

Our protocol can also be integrated with web servers which currently use trust management systems [2, 3] to make decisions based on fixed policies and credentials. The trust management systems can be placed in the web server and make decisions based on a surrogate supplied by the requestor and role specific policies specified by the appropriate authority. The trust policies can be communicated to the web server by the appropriate authority.

Our work also has implications in other areas like payment, price discrimination, licensing enforcement. If Cathy avails of service using her surrogates and wishes to pay for the service then there should be ways of tying the surrogates to the payment for auditing purposes. Similarly for price discrimination if a buyer can prove with his/her surrogate which price band he/she belongs to and the buyer is the legitimate owner of the surrogate, then price discrimination can be done with online transient identities, thus protecting offline identities.

References

[1] Yolanta Beresnevichiene. A role and context based security model. Technical report, University of Cambridge, 2003.

[2] Matt Blaze, Joan Feigenbaum, J. Ioannides, and Angelos Keromytis. The keynote trust management system. *Request For Comments Series*, (2704), 1999.

[3] Matt Blaze, Joan Feigenbaum, and M. Strauss. Compliance checking in the policymaker trust management system. *Proceedings of the* 2nd *Conference on Financial Cryptography: Lecture Notes in Computer Science Series*, 1465:251–265, 1998.

[4] Nicholas Bohm, Ian Brown, and Brian Gladman. Who carried the risk of fraud in ecommerce. *Journal of Information Law and Technology*, 2000(3):173–199, 2000.

[5] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. *Proceedings of the* 9th *ACM conference on Computer and Communications Security*, pages 21–30, 2002.

[6] C.B. Crispo. *Delegation of Responsibility.* PhD thesis, University of Cambridge, 1999.

[7] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer. *Proceedings of the* 24th *IEEE Symposium on Security and Privacy*, pages 2–15, 2003.

[8] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:472–492.

[9] David Ferraiolo, Ravi Sandhu, Serban Gavrilla, Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role based access control. *ACM Transactions on Information and Systems Security*, 4(3):224–274.

[10] B. Clifford Neuman and Theodore Tso's. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38.

[11] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[12] Bogdan Popescu, Martin Van Steen, and Andrew Tanenbaum. A security architecture for object based distributed systems. *Proceedings of the* 18th *Annual Computer Security Applications Conference*, 2002.

[13] Paul Syverson and David Goldshlag. Unlinkable serial transactions: Protocols and applications. *ACM Transactions on Information and Systems Security*, 2(4):354–389, 2000.

Uncorrelatable Transactions using Ring Signatures

Partha Das Chowdhury, Bruce Christianson, James Malcolm
Computer Science Department
University of Hertfordshire
England
{P.Das-Chowdhury, B.Christianson, J.A.Malcolm}@herts.ac.uk

January 31, 2005

1 Introduction

The Internet is conducive to large scale privacy invasion, identity theft [4], and target marketing [?]. We have seen instances in the past where people have suffered serious damage to the ready availability of digital dossiers [4]. Any centrally stored information can be abused. The use of fixed credentials (credit cards, key certificates) enables an adversary to correlate all the transactions conducted with the fixed credential. The threats of identity theft, correlatability can be countered using anonymous transaction protocols. Here we present a protocol for uncorrelatable electronic transaction based on ring signatures which also guards against identity theft as well as protects the privacy of the communicating partners. The organization of this paper is as follows. In section 2 we present the threat model which we are considering. This is followed in section 3 by an overview of ring signatures. In Section 4 we present our design goals and assumptions. The protocol which is the contribution of this paper is presented in section 5, which is followed by conclusions in section 6.

2 The Threat Model

When we buy goods or services or simply surf online we end up giving out lots of information about ourselves. All this information goes into databases somewhere. These records can be linked together to build a complete dossier on an individual [?]. Thieves can steal credit card information and use it, terrorists can track their targets using government maintained address records, or servers at the other end can leak sensitive information about us. In many instances in the past people have suffered damage due to the malicious use of sensitive information [1], [2]. When we pay for goods using our credit card we present

our card (which is the payment token) along with our identity (we provide our name and address for authorization on the internet). This information is used to check the validity of the card and the creditworthiness of the customer. In other applications trustworthiness of communicating partners are established using certificates, a good example of which is the Globe System [8]. Using same fixed certificates to establish the trustworthiness of the communicating partners enables the grantor of access to any service to correlate all the transactions of a requestor. What happens in the process is that the merchant, in the case of online shopping, also obtains a unique identifier (the credit card number) which, as well as learning the credit card number, enables the merchant to correlate various transactions conducted under the same credit card. Customer information can be indexed using credit card numbers or they can be sold to marketing companies [6].

The problem is that we do not have a clue about how information about us would be used by the entities at the other end of the communication channel.

3 Ring Signatures

Ring signatures were designed Ron Rivest, Adi Shamir and Yael Tauman [9]. In this signature scheme the verifier doesn't learn who the signer is but can only learn that the signer is a member of group of certain possible signers called a ring. One of the members of the ring actually signs using his/her private key and the public keys of the other members. The example cited in [9] speaks of a situation where a a government minister wants to leak information to a journalist. The journalist knowing the public keys of all the ministers can be sure that one of the ministers signed it without knowing who is the mole in the cabinet. In producing such a signature the signatory doesn't need the cooperation of any member of the group. All the signer needs to know is the public keys of all the members of the group. Let there be r members in the group. The signature is generated as:

1. The signer first computes the key as the hash of the message m to be signed.
2. Then the signer generates a random initialization value v.
3. The signer generates a random value x_i for each member of the group and computes y_i which is x_i encrypted with the public key of that ring member.
4. Then the signer solves the equation $F_{k,v}(y_1, y_2, ..., y_r) = v$ for y_s where F is a combining equation.
5. Now the signer uses his private key in order to invert g_s on y_s to obtain x_s as $x_s = g_s^{-1}(y_s)$
6. The output of the signature is the set of values x_i, the random value v, and the public keys of the group members.

A ring signature can be verified as follows:

1. For each member of the group, we encrypt the the corresponding random value x_i with that member's public key to give y_i.

2. We obtain the hash of the message to obtain the key k as $k = h(m)$.

3. We verify that the combining equation F regenerates the random value v in Z_P.

4 Design Goals and Assumptions

4.1 Design Goals

We present a protocol for Uncorrelatable Electronic Transaction (UET) where we use surrogates. Our approach makes it hard for an adversary to correlate all the transactions conducted by the same customer. The transaction flow is outlined by means of an example as:

1. The bank prepares and sends the information Carol needs to generate her surrogates.

2. Carol goes to a website selling goods she wants to purchase.

3. Carol generates the surrogate for the current transaction.

4. The seller authenticates locally whether or not Carol is a valid customer of the bank.

5. The seller sends the customer information to the bank.

The next time Carol goes to shop with the same seller she uses a different surrogate which can be verified as before but cannot be correlated with a previous surrogate. Our motivation has been that Carol trusts her bank which is quite a practical thing to do. There is no communication between the bank and the seller for authorization of payments and the seller can locally verify the validity of the customer.

4.2 Cryptographic and Infrastructural Assumptions

Communications between the bank and it's clients (the customer and the seller) are not anonymous. We assume the existence of a secure authenticated communication channel between the bank and the seller and between the bank and the customer. This can be implemented by digital signatures where every communication between the bank and the customer and the bank and the seller are digitally signed. This provides authentication. The communication link between the customer, bank and the seller can be secured by for example SSL/TLS. All communications between the bank and the seller and between the bank and the customer are secured in this way.

An anonymous communication channel between the seller and the customer is also assumed for our purposes. This can be implemented by Chaum's mix nets [3] or Mixminion [5]. The mix network makes it harder for an adversary observing the network to gain any additional information about the communicating partners beyond its a priori belief. The communication channel between the customer and the entry point of the mix network should also be secure and prevent traffic analysis. Communications between the customer and the seller are made anonymous in this way.

4.3 Mathematical Assumptions

Our protocol depends on the difficulty to compute discrete logarithms in the multiplicative group Z_P^* where P is a large prime. P should be chosen such that $(P-1)$ has one large prime factor. If $(P-1)$ has small prime factors then computing discrete logarithm is easy [7]. The bank selects $A, O_\sigma \in Z_P^*$ and $\sigma_0 \in \{1..P-1\}$.
The customer selects generator $g \pmod P$ and $s \in Z_P^*$, s is the secret key of the customer. We use the Linear Congruential Method to generate exponents where the offset O_σ and the modulus P are co-prime to each other. All operations are carried out $\pmod P$ when not specified otherwise explicitly. The method we use to generate surrogates is similar to the first group signature scheme presented in [?].

5 Protocol for Uncorrelatable Transactions

There are three parties in the protocol, the bank, the customer (Carol) and the seller. We describe the protocol in three phases:

1. Preparation Phase
2. Transaction Phase
3. Synchronization Phase

5.1 Preparation Phase

Carol sends the bank her public key.

1. $Carol \longrightarrow Bank : X = g^s$

The bank sends Carol the information she needs to generate her surrogates which is :

2. $Bank \longrightarrow Carol : \delta = \prec \sigma_0, O_\sigma, A \succ$

The bank while issuing δ calculates the first surrogate the customer will be using as follows.

$$\sigma_1 = (A * \sigma_0 + O_\sigma) \pmod{P-1}$$
$$S_1 = X^{\sigma_1}$$

The bank retains Δ for every customer where

$$\Delta = \prec O_\sigma, A, X, S_1, \sigma_1, P \succ$$

5.2 Transaction Phase

For transaction i Carol calculates her surrogate S_i as well as the corresponding secret key in the following manner.

$$\sigma_i = (A * \sigma_{i-1} + O_\sigma) \pmod{P-1}$$
$$S_i^- = s * \sigma_i \pmod{P-1}$$
$$S_i^+ = g^{S_i^-}$$

1. Carol chooses a subset r of the public surrogates of valid customers of the bank. The subset agreed forms the ring or the group of probable signers.

2. Let m be the transaction description something which uniquely identifies the transaction. Then Carol hashes m to get the key k as:

$$k = h(m)$$

3. She selects a random number $v\ \epsilon Z_p^*$.

4. The seller picks up $x_{1..r}$ for all the members of the group uniformly and independently from $\{0,1\}^b$ and sends that to Carol. She computes $y_i s$ from the $x_i s$ as:

$$Seller \longrightarrow_{mix} Carol : x_{(1..r)}$$
$$y_i = g_i(x_i)$$

5. Then the customer solves F for y_s where F is the same combining equation used in ring signatures.

$$F_{k,v}(y_1, y_2, \cdots, y_r) = v$$

6. The customer then signs m and sends it to the seller as in the original ring signature scheme.

$$Customer \longrightarrow_{mix} Seller : (S_1, S_2, ..., S_r; v; x_1, x_2, ..., x_r)$$

where $S_{1..r}$ are the surrogate DH keys of the r valid customers of the bank who are members of the ring.

7. The seller then verifies the signature as mentioned in section 3. While verifying the seller gets the S_i which is the surrogate of the customer. The seller submits S_i to the bank. The bank locates the customer account S_i.

5.3 Synchronization Phase

We saw in the preparation phase that the bank retains the value of the first surrogate while issuing Δ to the customer. The bank, after it receives a surrogate for a customer, calculates and stores the surrogate the customer will be using for the next transaction. This helps the bank to locate the appropriate account after it receives a surrogate from the seller. The bank retains Δ for every customer as we have seen in the preparation phase. Both the bank and the customer uses the same method to generate surrogates. This enables the bank to calculate the correct surrogate for every customer. This calculation of surrogates is done in this phase.

$$\sigma_{i+1} = (A * \sigma_i + O_\sigma) \pmod{P-1}$$
$$S_{i+1} = X^{\sigma_{i+1}}$$

The bank replaces in Δ.

$$\sigma_i \longleftarrow \sigma_{i+1}$$
$$S_i \longleftarrow S_{i+1}$$

The bank updates its current list of valid surrogates with a new surrogate.

6 Conclusions

The use of various surrogates cannot be correlated with each other but at the same time the validity of the surrogates can be determined. It is also not possible for the bank to masquerade as the customer as the bank doesn't know s and cannot generate the private exponent. Surrogates cannot be transferred between customers as that requires sharing the secret key s.

References

[1] Ross Anderson. *Security Engineering.* Wiley, Inc, 2001.

[2] BBC. Tax records for sale. *Available at http://news.bbc.co.uk/1/hi/business/2662491.stm*, 2003.

[3] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[4] David Chaum. Achieving electronic privacy. *Scientific American*, pages 96–101, 1992.

[5] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer. *Proceedings of the* 24th *IEEE Symposium on Security and Privacy*, pages 2–15, 2003.

[6] Andrew Odylzko. Privacy economics and price discrimination on the internet. *Proceedings of the* 5th *International Conference on Electronic Commerce*, pages 355–366, 2003.

[7] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[8] Bogdan Popescu, Martin Van Steen, and Andrew Tanenbaum. A security architecture for object based distributed systems. *Proceedings of the* 18th *Annual Computer Security Applications Conference*, 2002.

[9] Ronald Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *Proceedings of the* 7th *International Conference on the Theory and Applications of Cryptology and Information Security; Advances in Cryptology: Lecture Notes in Computer Science Series*, 2248:552–565, 2001.

Authentication in an Object-based Distributed System

Partha Das Chowdhury, Bruce Christianson, James Malcolm
Computer Science Department
University of Hertfordshire
England
{P.Das-Chowdhury, B.Christianson, J.A.Malcolm}@herts.ac.uk

January 6, 2005

Abstract. We live in a world where resources are rarely local and distributed often at various locations. Designers of distributed systems generally use fixed credentials (key certificates) for authentication purposes, which means that servers can identify their clients, and can correlate their actions. An authentication mechanism based on ring signatures does not reveal the identity of a client to the process performing the access control decision.

1 Introduction

We live in a world where resources are hardly ever local but are often distributed across various locations. There is seldom a central authority that manages access to these resources. Designers of distributed systems generally use fixed credentials (key certificates) for authentication purposes. This means that servers can identify their clients, and can correlate their actions. In this paper we present an alternative authentication mechanism for distributed systems, using the Globe [2] object based distributed system, which does not reveal the identity of a client to the process performing the access control decision. We discuss our approach using a case study of an electronic newspaper.

Globe is a wide area distributed system where objects are physically shared between users and also physically replicated at several locations. The principal construct in Globe is a Distributed Shared Object (DSO) which consists several local objects residing in local address spaces. The local objects storing a part of a DSO's state are known as replicas. A replica consists of the following subobjects:

- Semantics subobject, which is the only subobject written by the application developer, contains the code that implements the DSO.

- Communication subobject is responsible for communication between local objects residing at local address spaces.
- Replication subobject is responsible for keeping the replica's state consistent with the other replicas.
- Control subobject is responsible for taking care of the invocations from the client process on the host.
- Security subobject is responsible for implementing the security policies of the DSO.

For invoking a method on a DSO a user has to create a local object in his/her own address space and this object often acts as a proxy routing requests to appropriate replicas. The Globe Location Service facilitates finding of replicas. A user willing to run a replica or a user proxy needs a Globe Object Server in his/her computer either stand alone or integrated with other application.

A distributed system like Globe presents lots of interesting problems but in this paper we will concentrate on access control. The current proposals by the developers [2] of Globe are based on certificates issued by the DSO owner and follows a role based authorisation model. We present here an alternative authorisation model which preserves the privacy of the requestor but at the same time satisfies the requirements like verifiability and unforgeability.

2 Access Control in Globe

In order to specify the security policies of the DSO its owner must identify all the meaningful roles of that object. This involves careful examination of the application concerned, which is outside the scope of this paper. Once the user role set is identified for an object then with each user role is associated a bit vector indicating the methods the role is allowed to execute. Grouping the bit vectors for all the roles of a particular object forms the access control matrix for the DSO. This matrix is stored in the security subobject and when a replica is created it gets the matrix from the owner of the DSO. When a role wants to invoke a method on an object the security subobject consults this matrix to verify whether or not this role is allowed to invoke this particular method.

The DSO owners sign certificates binding users to their public key, this certificate is then used by the user to authenticate while requesting access. If there are users with the power to delegate then the security subobject verifies the chain of certificates till the one signed by the DSO owner. The replicas on the other hand also authenticate to the user and the replica and the user can start talking to each other after they have authenticated to each other. Replicas authenticate themselves to the user using replica certificates issued by the DSO owner.

The problem of using fixed credentials by the user to authenticate himself/herself arises from the fact that an adversary can correlate all the actions

of any particular user. This will leak information about any user's online activities to an adversary which can be used in a way harmful to the user. We present an alternative authentication model here in this paper where by an user can authenticate himself/herself to the replica anonymously and an adversary cannot correlate all the actions of any user. This we believe is new because previous approaches to authentication rely on fixed credentials. Similarly trust management systems like Keynote [4] make decisions based on fixed credentials and policies. Our approach doesn't require significant change from the way we do authentications at present. The trust management systems can be queried using the surrogates [5, 6] one uses in our approach and decisions can be made based on fixed policies. We talk about surrogates in the next section.

3 Design Goals

In our approach we view principals as members of a group which may be a subset of all the members of a particular role. To access a resource the requestor has to prove that he/she is a member of the group that is allowed access to the requested resource. For example if an electronic newspaper wishes to allow only its subscribers to read the paper and there is a DSO modelling the electronic newspaper then in the Globe proposal the DSO owner would create a role of subscriber and issue certificates to paid subscribers. The paid subscribers in turn would authenticate themselves to the replica using their certificate and access the newspaper.

In our approach subcribers just prove that they are valid subscribers without revealing their key certificates. The DSO owner doesn't need to issue key certificates using our approach. The subscribers authenticate themselves as member of a group of valid subscribers. In the alternative we propose here the subscribers have an offline identity which is their public key and an online identity which is their surrogate generated from the parent public key. The surrogate has a corresponding secret key like the parent public key. This secret value is generated from the secret key corresponding to the parent public key and together the public surrogate and its corresponding secret value forms the surrogate pair. The surrogate pair for a public private key pair can only be generated and used by the legitimate owner of the corresponding parent public key. In our approach while authenticating the user doesn't need to reveal its surrogate to the replica.

However the DSO owner can correlate the public value of the surrogate with the parent public key. The DSO owner can never masquerade as the owner of the public key as the DSO owner doesn't have the knowledge of the corresponding secret key. Our approach also allows delegation of credentials contrary to the popular belief that non-transferrability is a must for anonymous transaction systems. If a subscriber wants to allow someone to read the newspaper using his/her credentials he/she can always lend his/her surrogate pair. The delegatee however cannot calculate the secret key from the public and private values of the current surrogate. So learning one pair of surrogates doesn't help an attacker

in any way. In non anonymous systems if the private key corresponding to the parent public key is compromised then there comes a need for revocation of keys. In our approach the legitimate owner, in the event of leakage of the secret part of a surrogate pair, just needs to generate another surrogate pair but doesn't need to revoke his/her parent public and private key pair.

4 The Protocol

The existence of a secure anonymous communication channel between the customer and the DSO owner is assumed for our purposes. A anonymous communication channel [3] between the client and the replica is also assumed for our purposes. The protocol is based on ring signatures designed by Ron Rivest, Adi Shamir and Yeal Tauman [1]. They call a set of possible signers a ring. One of the members of the ring actually signs using his/her own private key and the public keys of the other members. In this signature scheme the verifier doesn't learn who the signer is but can only learn that the signer is a member of a certain group of possible signers. In producing such a signature the signatory doesn't need the co-operation of any other member of the group. They don't even have to agree to be in the group. All the signer needs to know is the public keys of all the members of the group.

In our example a valid subscriber creates a signature using the public surrogates of the other subscribers and in doing so the signer doesn't need the active cooperation of the other subscribers. The replica verifies the signature and if the signature is valid then the verifier is sure that the signature was generated by a valid ring member without knowing the actual online identity of the signer. The online identity of the signer is his/her surrogate and it is hard to deduce the offline identity (which is the parent public key from which is the surrogate is generated) of the subscriber from his/her online identity. The transaction flow can be outlined as:

1. A customer, Cathy, registers with the DSO owner as a subscriber. This registration requires Cathy to prove that she owns the corresponding secret key.

2. The DSO owner prepares and sends the information Cathy needs to generate her surrogate pair.

During the protocol run:

1. Cathy authenticates to the replica that she is a valid subscriber using her surrogates.

2. On authentication by Cathy to the replica as well as by the replica to Cathy she is allowed access to the electronic newspaper.

Every time Cathy authenticates herself to a replica that she does so using ring signatures which preserves her privacy and at the same time the replica is also satisfied that the signer is a valid subscriber.

5 Conclusion

In this paper we have presented an alternative authentication model for distributed systems which is particularly useful for situations where all the members of a particular role have the same level of permissions and access priviledges. The signature in our approach can only be generated by legitimate parties and someone who is not a subscriber cannot gain access to the newspaper without the active collusion of a paid subscriber.

Our approach also makes the problem of delegation easier. A delegator who doesn't use surrogates to authenticate needs to share his/her secret key with the delegatee but in our approach the delegator only shares surrogates. Learning one pair of surrogates doesn't help an attacker in any way.

References

[1] Ronald Rivest, Adi Shamir, Yael Tauman, How to Leak a Secret,Proceedings of Asiacrypt, 2001, pp 552-565

[2] Bogdan Popescu, Martin Van Steen, Andrew S. Tanenbaum, A Security Architecture for Object Based Distributed System, Proceedings of the 18^{th} Annual Computer Security Applications Conference, December 2002

[3] George Danezis, Roger Dingledine, Nick Mathewson, Mixminion: Design of Type III Remailer, IEEE Security and Privacy Conference, 2003

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, A.Keromytis, The Keynote Trust Management System, RFC 2704

[5] Partha Das Chowdhury, Bruce Christianson, James Malcolm, Anonymous Authentication, Accepted at the 12^{th} International Workshop on Security Protocols, Cambridge, England, 2004

[6] Bruno Crispo, Delegation of Responsibility, PhD thesis submitted to University of Cambridge, 1999

Uncorrelatable Electronic Transactions - Technical Report 392

Partha Das Chowdhury, Bruce Christianson, James Malcolm
Computer Science Department
University of Hertfordshire
England
{P.Das-Chowdhury, B.Christianson, J.A.Malcolm}@herts.ac.uk

January 31, 2005

1 Introduction

The Internet is conducive to large scale privacy invasion, identity theft [7], and target marketing [10]. When we buy goods or services or simply surf online we end up giving out lots of information about ourselves. All this information goes into databases somewhere and records can be linked together to build a complete dossier on an individual [7]. Thieves can steal credit card information and use it, terrorists can track their targets using government maintained address records, servers can leak sensitive information about us: in many instances people have suffered damage due to the malicious use of sensitive information [1, 2]. When we pay for goods using our credit card we present our card (which is the payment token) along with our identity (we provide our name and address for authorization on the Internet). This information is used to check the validity of the card and the creditworthiness of the customer. What happens in the process is that the merchant, in the case of online shopping, also obtains a unique identifier (the credit card number) which as well as learning the credit card number enables the merchant to correlate various transactions conducted under the same credit card. The problem is that we do not have a clue about how information about us would be used by the entities at the other end of the communication channel. Customer information can be indexed using credit card numbers or they can be sold to marketing companies [10]. To deal with this threat to individual privacy in the networked world methods to preserve anonymity were proposed in various scientific literature some of which we discuss below in section 2.

We present a protocol for uncorrelatable electronic transactions using payment tokens and surrogates. The payment tokens and surrogates satisfies the properties like uncorrelatability, unforgeability and verifiability. This paper is organized as follows. In section 2 we present an account of some previous re-

search in this area. This is followed in section 3 by what we mean by surrogates and how we intend to use them in the protocol for uncorrelatable electronic transactions presented in this paper. Section 4 gives an overview of our design goals and assumptions. The main contribution of this paper is presented in section 5, which is followed in section 6 by conclusions.

2 Previous work

Idemix [5] is a anonymous credential management system which allows the user to be anonymous to the credential issuing organization. In this system there has to be a global pseudonym authority which issues pseudonyms to individual users and a trusted party through which all transactions are carried out.

In the anonymous credential management system proposed by Syverson et. al. [15] users get credentials which can be used only once. After every transaction the user has to go back to the issuer to obtain another credential.

There is another approach by the name of Independent Unbiased Trust Entities. These trust entities hold all sensitive information pertaining to the users and then issue them with one time payment tokens to the users. Such systems depend more on self regulation and also participate directly in dispute resolution [11]. Trust entities are never a reasonable approach as any centrally stored information can be abused. Most attacks on centrally stored information are by insiders and not outsiders [1].

The approach we present in this paper is different from the above mentioned approaches. First of all we do not propose another credential management system. We use surrogates (short lived electronic identities) [8] which are generated from a parent (credit card number in this case). In our approach users are not anonymous to the issuer as in *idemix*. We also do not depend on a global pseudonym authority. Deployment of a system like *idemix* we feel difficult as the users, credential issuers and verifiers are all required to be part of the system. David Chaum [7] and Stefan Brands [4] proposed protocols which use digital cash. The payment tokens we present here are not the digital equivalent of cash. Double spending is against the interest of the customer but for our system double spending is not against the interest of the bank. To preserve its anonymity customers should refrain from reusing their payment tokens. Since all payments are verified so double spending can also be detected in our protocol. The approach we present here doesn't require the users to contact the issuer for new payment tokens after every transaction. The anonymous credential management systems emphasize on non-transferability. This we feel is not desirable where we need delegations. For example James might allow his son to use his credit card but make sure that he cannot use it more than once. Using our protocol controlled delegation can be achieved but at the same time stealing someone's credential is not going to help the thief in any way.

3 Surrogates

Surrogates are transient numbers generated from a parent number (such as a bank account number, credit card number, or social security number) by some mathematical function. They are used to preserve the anonymity of the holder of the parent number. Surrogates have some intrinsic limitations due to the way they are generated and the purpose they are used for. If they are not verified at the point when they are used then random numbers can be passed as surrogates. Although one could use a probabilistic verification model to counter the threat of fake surrogates, for the purposes of the present paper we assume the issuing authority to be online.

There is another party in this whole system: the one who accepts the surrogates in exchange of goods or services. The acceptor of surrogates bears the risk of fake surrogates [3]. Moreover the power to verify individual surrogates cannot be delegated to the acceptors. We feel that this is the difference between blinded credentials and the way we generate surrogates here. Blinded credentials can be verified by the acceptor if the acceptor knows the public key of the issuer. To verify the surrogates the verifier has to know the mathematical function as well as the input (which is the parent number) to the functions. If the power to verify individual surrogates is delegated to the acceptors (*i.e.* acceptors are given the inputs) then the whole purpose of having surrogates is compromised.

Credit cards are verified in the real world by a process in which the card issuing bank is involved. The Point of Sale (PoS) terminal first contacts the issuer. The card issuer verifies that the PoS has a valid merchant account and then verifies the card information. The issuer verifies that the card is valid *i.e.* not revoked and the card balance when added to the current purchase is not over the approved limit. Then an authorization decision is conveyed to the merchant. The merchant might ask for other form of authentication from the customer like signature of the customer etc [?]. For our protocol we assume the issuer to be online similar to the credit card authorization infrastructure mentioned above.

4 Design Goals and Assumptions

4.1 Design Goals

We present a protocol for Uncorrelatable Electronic Transaction (UET) where we use payment tokens and surrogates. Our approach makes it hard for an adversary to correlate all the transactions conducted by the same customer. At the same time the merchant and the bank can verify the validity of the payment token. The transaction flow is outlined by means of an example as:

1. The customer Carol requests payment tokens from her bank.
2. The bank prepares and sends the information Carol needs to generate her payment tokens and surrogates.

3. Carol goes to a website selling goods she wants to purchase.
4. Carol generates the payment tokens and surrogates.
5. The seller authenticates locally whether or not Carol is the legitimate owner of the surrogate.
6. Carol pays with her payment tokens which the seller validates with the bank.

The next time Carol goes to shop with the same seller she uses different surrogates and payment tokens which can be verified as before but cannot be correlated with a previous surrogate or payment token. Our motivation has been that Carol trusts her bank which is quite a practical thing to do. By combining a surrogate key with features from the information checking protocol we are able to produce an anonymous payment protocol that helps preserve the privacy of the customer as well as assures the merchant of their money.

4.2 Cryptographic and Infrastructural Assumptions

We assume the existence of a secure authenticated communication channel between the bank and the seller and between the bank and the customer. This can be implemented by digital signatures where every communication between the bank and the customer and the bank and the seller are digitally signed. This provides authentication. The communication link between the customer, bank and the seller can be secured by for example SSL/TLS. All communications between the bank and the seller and between the bank and the customer are secured by SSL/TLS and are denoted by $\longrightarrow$ in the protocol.

An anonymous communication channel between the seller and the customer is also assumed for our purposes. This can be implemented by Chaum's mix nets [6] or Mixminion [9]. The mix network makes it harder for an adversary observing the network to gain any additional information about the communicating partners beyond its a priori belief. The communication channel between the customer and the entry point of the mix network should also be secure and prevent traffic analysis. Communications over the mix nets are denoted by $\longrightarrow_m$ in our protocol.

4.3 Mathematical Assumptions

Our protocol depends on the difficulty of computing discrete logarithms in the multiplicative group Z_P^* where P is a large prime. P should be chosen such that (P-1) has one large prime factor. If (P-1) has small prime factors then computing discrete logarithm is easy [12]. We know that Z_P^* has $\phi(P-1)$ primitive roots [?]. The bank selects generators $S_0, B_0, Y_0 \pmod P$ and $\tau_0, O_\sigma, O_\tau \in Z_P^*$ and $\sigma_0 \in \{1..P-1\}$.
The customer selects generator $g \pmod P$ and $s \in Z_P^*$, s is the secret key of the customer. We use the Linear Congruential Method to generate exponents τ_i

and σ_i where the offsets O_σ and O_τ and the modulus are co-prime to each other. All operations are carried out modulo P when not specified otherwise explicitly. τ_i and σ_i are the exponents used for generating the tokens and surrogates respectively. A is the constant used in the Linear Congruential Method.

5 The Protocol

There are three parties in the protocol, the bank, the customer and the seller. The bank is online and the seller communicates with the bank for authorization decisions. The bank selects S_0, B_0 and Y_0 and shares with the customer only B_0. The seller knows none of these values. On the other hand only the customer knows s which is not known either to the bank or the seller. The Protocol is described under three sections namely

1. Preparation
2. Transaction
3. Synchronization

5.1 Preparation Phase

Our protocol builds on a new tool called Information Checking which was first proposed in [14]. The bank in our protocol selects S_0, B_0 and Y_0 and generates the subsequent S_i, B_i and Y_i values. For each set of S_i, B_i and Y_i values generated the bank calculates

$$C_i = S_i + B_i * Y_i \pmod{P}$$

The payment token that the customer presents consists of the B_i and C_i corresponding to a particular S_i and Y_i value (known to the bank). The bank verifies the equation. The customer cannot generate fake B and C values as he/she has no knowledge of the corresponding S and Y values. During the preparation phase the customer first sends the bank its public key.

1. $Customer \longrightarrow Bank : X = g^s$

So that the customer doesn't need to communicate with the bank for every single transaction the bank prepares m payment tokens. It sends customer only the C values along with the information Δ it needs to prepare the corresponding B values and the surrogates (which are generated from X). Only the bank can generate the S and Y values.

2. $Bank \longrightarrow Customer : \Delta = \prec C_{1..m}, B_0, \sigma_0, \tau_0, O_\sigma, O_\tau, A, P \succ$

The bank calculates the batch of C values by repeating the following for each of the next m values of i:

$$\tau_i = (A * \tau_{i-1} + O_\tau)$$
$$S_i = S_0^{\tau_i}$$
$$B_i = B_0^{\tau_i}$$
$$Y_i = Y_0^{\tau_i}$$
$$C_i = S_i + B_i * Y_i$$

During the preparation phase the bank also generates the surrogate K_1^+ the customer will be using in its first transaction. We will see later that the bank always stores the surrogate the customer will be using next. When the seller submits the customer information for authorization to the bank the bank locates the proper customer information using this surrogate.

$$\sigma_1 = (A * \sigma_0 + O_\sigma) \pmod{P-1}$$
$$K_1^+ = X^{\sigma_1}$$

At the end of this phase the bank does not retain all the S, Y and B values but only the initial generators τ_0 and σ_1 values, and the offsets and the constant A to generate subsequent σ_i and τ_i values. The bank retains the information Γ which is:

$$\Gamma = \prec S_0, B_0, Y_0, O_\sigma, O_\tau, A, K_1^+, \sigma_1, \tau_0, X, P \succ$$

The K_1^+, σ_1, and τ_0 values will change after every transaction and is shown in the Synchronization phase.

5.2 Transaction Phase

For transaction i the customer calculates its ith surrogate K_i^+ . The corresponding private exponent K_i^- of the current surrogate is also generated by the customer.

$$\sigma_i = (A * \sigma_{i-1} + O_\sigma) \pmod{P-1}$$
$$K_i^- = s * \sigma_i \pmod{P-1}$$
$$K_i^+ = g^{K_i^-}$$

The exponent τ_i used to generate the B_i for the current transactions is calculated by the customer from the information provided by the bank and using the exponent used for generating B_{i-1} (except for B_1). Then the B_i value is calculated.

$$\tau_i = (A * \tau_{i-1} + O_\tau)$$
$$B_i = B_0^{\tau_i}$$

The seller generates a transaction description T and sends the customer signed with the seller's private key K^-_{seller}. T is agreed between both the seller and the customer. If the customer wants to pay the customer authenticates to the seller that it is the legitimate owner of the current surrogate by signing the transaction description T. T typically includes date, sequence number, details of the seller, and the amount to be paid. Along with the signed transaction description the customer sends the seller the ith payment tokens and the current surrogate.

1. $Seller \longrightarrow_m Customer : [T]_{K^-_{seller}}$

2. $Customer \longrightarrow_m Seller : [T]_{K_i^-}, K_i^+, B_i, C_i$

The seller contacts the bank, the bank locates the appropriate customer with the help of the surrogate. We have mentioned in the preparation phase that the bank calculates in advance the surrogate a particular customer will be using next. This is done by the bank during the synchronisation phase at the end of every transaction. For the first transaction the bank retains the value of the first surrogate while issuing the tokens. The bank calculates the corresponding S and Y value and solves the equation $C_i = S_i + B_i * Y_i$ for transaction i. If the equation is satisfied by the B and C values presented by the customer the bank agrees to the transaction. This process ensures that the seller receives his payment.

3. $Seller \longrightarrow Bank : \prec K_i^+, [T]_{K_i^-}, B_i, C_i \succ$

4. $Bank \longrightarrow Seller : \prec T, authorizationDecision \succ$

Our protocol allows for one time delegation of payment tokens which might be desirable in some situations. A user James can allow his son to use his credit card but make sure that he cannot use it more than once. James can do it by lending a set consisting of $\prec K_i^+, K_i^-, B_i, C_i \succ$ to his son. His son cannot figure out his secret key s from either K_i^+ or K_i^-.

5.3 Synchronization Phase

This phase doesn't involve any communication between the bank and the customer or between the bank and the seller. During this phase the bank updates the record for the customer whose transaction request was last approved. The surrogate the customer will be using for the next transaction is calculated in this phase. For this the bank needs to calculate the proper exponent which can be done from the exponent that was used to generate the last surrogate using the Linear Congruential Method. The bank also generates the exponent that it will use to calculate the next S and Y value for the B and C value the customer will submit during the next transaction. Customer records are updated by the bank by replacing the current surrogate with the new surrogate and the old τ value is replaced with the new τ value. The bank calculates the following.

$$\sigma_{i+1} = (A * \sigma_i + O_\sigma) \pmod{P-1}$$
$$\tau_{i+1} = (A * \tau_i + O_\tau)$$
$$K_{i+1} = X^{\sigma_{i+1}}$$

The bank replaces in Δ.

$$\sigma_i \longleftarrow \sigma_{i+1}$$
$$\tau_i \longleftarrow \tau_{i+1}$$
$$K_i \longleftarrow K_{i+1}$$

6 Conclusions

The surrogates cannot be correlated with each other neither can the surrogates be linked to a parent. The payment tokens cannot be forged. The customer has no knowledge of S and Y corresponding to a B and C. The customers can generate several private and public key pairs from a single parent public and private key pair. Since there are no fixed credentials correlation is not possible using our protocol. We do not need any hierarchical pseudonym authority as in *idemix*. The customers can generate their pseudonyms locally and do not need to contact the issuer after every transaction. This protocol we believe is simpler than the credential management systems both in terms of set up and deployment. Previous credential management systems stressed non-transferability which we also do but in a more flexible way. There might be situations when controlled delegation is desirable and these are supported by our protocol. We agree that this flexibility is not good for credentials like a driving license or a prescription but that requirement can be achieved with small modifications to our protocol. In our protocol the customer's privacy is protected and the merchant doesn't lose money. There are other applications where we can use this protocol. Role based access control we feel is an application where we can use this protocol. In some other applications trustworthiness of communicating partners are established using certificates, a good example of which is the Globe System [13]. Globe is also an application where we can use our protocol.

References

[1] Ross Anderson. *Security Engineering.* Wiley, Inc, 2001.

[2] BBC. Tax records for sale. *Available at http://news.bbc.co.uk/1/hi/business/2662491.stm*, 2003.

[3] Nicholas Bohm, Ian Brown, and Brian Gladman. Who carried the risk of fraud in ecommerce. *Journal of Information Law and Technology*, 2000(3):173–199, 2000.

[4] Stefan Brands. Offline cash transfer by smart cards. Technical report, Centrum voor Wiskunde en Informatica, 1994.

[5] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. *Proceedings of the* 9th *ACM conference on Computer and Communications Security*, pages 21–30, 2002.

[6] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[7] David Chaum. Achieving electronic privacy. *Scientific American*, pages 96–101, 1992.

[8] C.B. Crispo. *Delegation of Responsibility.* PhD thesis, University of Cambridge, 1999.

[9] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer. *Proceedings of the* 24th *IEEE Symposium on Security and Privacy*, pages 2–15, 2003.

[10] Andrew Odylzko. Privacy economics and price discrimination on the internet. *Proceedings of the* 5th *International Conference on Electronic Commerce*, pages 355–366, 2003.

[11] Mary Ann Patton and Audung Josang. Technologies for trust in ecommerce. *Proceedings of the IFIP working conference on Ecommerce*, 2001.

[12] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[13] Bogdan Popescu, Martin Van Steen, and Andrew Tanenbaum. A security architecture for object based distributed systems. *Proceedings of the* 18th *Annual Computer Security Applications Conference*, 2002.

[14] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. *Proceedings of the* 21st *ACM Symposia on Theory of Computing*, pages 73–85, 1989.

[15] Paul Syverson and David Goldshlag. Unlinkable serial transactions: Protocols and applications. *ACM Transactions on Information and Systems Security*, 2(4):354–389, 2000.