ADVANCES IN INFORMATION SECURITY

Secure Multi-Party Non-Repudiation Protocols and Applications

José Onieva
Javier Lopez
Jianying Zhou

Springer

Secure Multi-Party Non-Repudiation Protocols and Applications

Secure Multi-Party Non-Repudiation Protocols and Applications

by

José A. Onieva
University of Malaga
Spain

Javier Lopez
University of Malaga
Spain

Jianying Zhou
Institute for Infocomm Research
Singapore

Springer

Authors:
José A. Onieva
University of Malaga
Computer Science Dept.
E.T.S. Ingenieria Informaica
Campus de Teatinos
29071 Malaga, Spain
onieva@lcc.uma.es

Javier Lopez
University of Malaga
Computer Science Dept.
E.T.S. Ingenieria Informaica
Campus de Teatinos
29071 Malaga, Spain
jlm@lcc.uma.es

Jianying Zhou
Institute for Infocomm Research (12R)
21 Heng Mui Keng Terrace
Singapore 119613, Singapore
jyzhou@i2r.a-star.edu.sg

ISBN-13: 978-0-387-75629-5 e-ISBN-13: 978-0-387-75630-1

Library of Congress Control Number: 2008938604

To my parents, for being always there.
– Jose A. Onieva.

Preface

Currently, it seems that each day the world is less reluctant to accept the truth: communications will be shifted to Internet (whatever the generation is when it takes place). Doubtlessly, this will result in a large number of advantages, speeding up the way we have to make businesses and be in touch with each other. Some solutions are already part of our lives and either enhance or provide new methods for communicating, purchasing, selling, reporting and in general transmitting information. Some of these are: VoIP, Instant Messaging, E-Mail, E-commerce, Internet Payment methods, P2P for file sharing, etc.

In some of these methods efficiency and functionality are of major concern. Furthermore, tools (or protocols) providing this functionality have gained a place or become a de-facto standard because they perform better at providing this key components. Nevertheless, other solutions need to consider security as a key component from the very beginning of their design. Moreover, fulfilling this important and at the same time difficult to achieve property, some solutions have failed to accomplish their purpose as users are still unenthusiastic about adopting new solutions if either security is important or otherwise this is not provided with a 100% assurance. In other words, trust is not easily assumed by users in e-payment methods, e-commerce, e-banking, contract signing applications, Internet biddings, etc.

But as security practitioners learn from the very beginning of their career, there's no unconditional and complete secure system. What's more, it will never exist. There is always something that could not work in the system, be broken up, accessed without authorization, leaked information, etc. And if not, the human component (which will be present as long as we do) would need to be totally trusted not to put in risk all the implemented security measures.

This is the reason for which in traditional paper-based procedures (as with contract signing, money transfers, etc.) accountability as a means to solve possible disputes among users has always played part of the procedure itself. Here the handwritten signature plays the main role. It allows for (non-)repudiation of acts such that, for instance, a bank cannot transfer our money from one account to another if it does not have an application with our consent (signature). And of course, it exists the digital counterpart: the digital signature. And even better, most of the countries already (or are in the way to) have legislated its use.

But unfortunately, the network communication grounds (and among them, distance and lack of trust) make translation of paper-based procedures to networked digital ones hard to achieve. Thus, in order to realize security in Internet (or any other networked including mobile) applications, special protocols are needed to ensure that any dispute could be solved between users if the network fails or an entity misbehaves. In the computer security field, these protocols are known as non-repudiation protocols.

Research oriented to non-repudiation protocols has been active since late 90's; considering in most occasions Alice and Bob as the players of the protocol design scenario. It is time to give a step forward and realize that in many applications there are more than two entities and that two-party protocols are not appropriate for scenarios in which multiple entities are involved. This is the main aim of this book.

This book is mainly targeted to professional audiences with in-depth knowledge of information security and a basic knowledge of applied cryptography. This book briefly settles the state of the art in non-repudiation protocols and gives insight of its applicability to e-commerce applications. It organizes the existing scant literature regarding non-repudiation protocols with multiple entities' participation. It provides to the reader with sufficient grounds to understand the non-repudiation property and its applicability to real applications. Security practitioners will find it very useful in the design of secure applications with multiple entities, helping them to envisage the basics of multi-party non-repudiation as a necessary tool when trust is not present among players. Additionally it could serve as text book for postgraduate students who wish to understand the non-repudiation service and mechanisms in the presence of an undefined number of players.

Malaga,
September 2008

Javier Lopez
Jianying Zhou
Jose A. Onieva

Contents

List of Figures

List of Tables

Acronyms

Lists of abbreviations, symbols.

ANX	Automotive Network Exchange
API	Application Programming Interface
B2B	Business-to-Business
B2C	Business-to-Consumer
C2C	Consumer-to-Consumer
CA	Certification Authority
CEM	Certified Electronic Mail
CEMBS	CErtificate of a Message Being a Signature
CRL	Certificate Revocation List
DES	Data Encryption Standard
DoS	Denial of Service
DRM	Digital Rights Management
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
EOD	Evidence of Delivery
EOO	Evidence of Origin
EOR	Evidence of Receipt
ERP	Enterprise Resource Planning
EU	European Union
FTP	File Transport Protocol
GPRS	General Packet Radio Service
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IN	Intermadiary ageNt
IP	Internet Protocol
IPSec	Internet Protocol Security
ISO	International Standarization Organization

IT	Information Technology
ITU	International Telecommunications Union
L2TP	The Layer Two Tunneling Protocol
LAN	Local Area Network
MPNR	MultiParty Non-Repudiation
MAC	Message Authentication Code
MNO	Mobile Network Operator
MPCS	Multi-Party Contract Signing
NRD	Non-repudiation of Delivery
NRO	Non-Repudiation of Origin
NRR	Non-Repudiation of Receipt
NRS	Non-repudiation of Submission
OCSP	On-line Certificate Server Protocol
ODR	On-line Dispute Resolution
OMA	Open Mobile Alliance
OS	Operating Systems
OSI	Open System Interconnection
P2P	Peer-to-Peer
PC	Personal Computer
PCS	Private Contract Signature
PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PPTP	Point-to-Point Tunneling Protocol
PRAC	Partial Result Authentication Code
QES	Qualified Electronic Signature
QoS	Quality of Service
RSA	Rivest, Shamir and Adelman
RO	Right Object
RI	Rights Issuer
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
SSH	Secure SHell
S/MIME	Secure/Multipurpose Internet Mail Extensions
TSA	Time-Stamp Authority
TCP	Transport Control Protocol
TLS	Transport Layer Security
TPD	Trusted Personal Device
TTP	Trusted Third Party
URL	Uniform Resource Locator
VPN	Virtual Private Network
WEP	Wireless Equivalent Privacy
WTLS	Wireless Transport Layer Security
XML	eXtensible Markup Language

Part I
Introduction and Fundamentals

Chapter 1
Introduction

Abstract This chapter sets the scope of the book, beginning with an introduction to e-commerce. Security issues and services which arise from e-commerce systems are defined and identified, giving special attention to *non-repudiation*. Finally, it establishes the main goals of this book.

1.1 Introduction to E-commerce

A crucial factor in the rapid growth of the Internet is electronic commerce: the ability to advertise goods and services, search for suppliers, compare prices and make payments, all being conducted at the click of the mouse button and without the existence of frontiers. In the EU, according to the Eurostat [16], more than 1 out of 4 ordered goods or services over the Internet in year 2006 compared with 1 out of 5 in year 2004. That means a growth of more than 26% in 2 years. With these numbers, it is not too difficult to support that e-commerce is not a new concept or a virtual technology any more.

Electronic commerce could be defined in several ways and it is hard to find out which of them is the most appropriate. We could define *e-commerce* as any commercial activity which is performed (at least partially) making use of open networks, especially the Internet. E-commerce embraces all aspects of the traditional commerce such as catalogue, on-line purchase, customer service, product specification, etc. The user perceives that all the services are just a few seconds further after switching on the computer thanks to the integration of e-commerce. New concepts like e-payment, e-banking, e-lottery, etc., belong to this area.

What seems to be clear is that e-commerce helps businesses to expand their strategy and market, getting global presence and, because of that, most of them are being shifted to the Internet or taking advantages of other digital sources. Thus, traditional paper-based transactions must be transformed into digital procedures. Such traditional documents are contracts, payment invoices, certified mails, cheques and any other documents related to business transactions.

Not only enterprises, but also customers are being profited from e-commerce. These customers can be other businesses (B2B) or final users (B2C and C2C). From their side, e-commerce allows them to interact with companies in a more efficient and economic way than ever, since digital networks are faster and do not have to pass frontiers.

Another area often classified inside e-commerce is *e-government*. In this area citizens get services and information from the government institutions via the Internet. Hence, the relationship between government institutions and citizens is improved in the same way as in e-commerce. E-government does not mean putting scores of government forms on the Internet. It is about using technology to its fullest to provide services and information that is centered around citizen groups. Like its father e-commerce, e-government is not a new concept any longer, as more and more institutions are taking advantage of services offered by the Internet, and implementing these activities in a regulated way. Some examples are E-Gov [119] in the United States and the i2010 program [1] for e-government [11] in the EU.

E-commerce is a conglomerate of applications (see Figure 1.1) powered by the Internet, the World Wide Web and other innovations. In Europe, Internet-based systems dominate the e-commerce profiles of most firms [124], that is, there is a decisive shift towards web-based environments. However, the main foundation role for the Internet in e-commerce among companies is to enhance overall interconnection capabilities - to enable more kinds of systems interacting with other systems within and between companies.

Actually, for an e-commerce application to work, multiple elements are involved in the process. Firstly, for a user to have access to this application, it needs a device with the integrated connectivity (e.g. a PC or a PDA) to access different resources. These devices need some software installed (e.g. e-wallet for payment purposes and an Internet browser) in order to perform all the operations of an e-commerce application. Similarly, at the other side, there is a whole architecture which allows these connections from other entities and through different devices (e.g. web server, private network, secure databases, ...). Standards and well-defined technologies are also needed in order to boost e-commerce among different platforms, businesses and users.

These technologies can be used in e-commerce as *protocols*. Currently, for instance, SSL/TLS (see the Appendix in this chapter for further reference) is being widely used to secure communications between servers and users. This is a protocol developed for transmitting general private documents via the Internet, designated at the transport layer of TCP/IP, and many Web sites use it to obtain confidential user information, such as credit card numbers. Diverse initiatives like 3-D Secure from Visa [19] and SecureCode from MasterCard [71] are developed based on this standard. E-commerce can also rely on *infrastructure* technology like P2P networks. Collaboration and resource sharing can increase productivity and reduce costs. P2P streamlines business supply chains digitally simulating the existing traditional channels.

[1] This programme follows the finished eEurope 2005 program [10].

For an e-commerce system to work, its different functional characteristics that define its behavior have to be identified. Some of the key **functional requirements** [113] are delineated on the new way of doing business that e-commerce demands:

- Support for diverse information appliances – The functionalities provided by the e-commerce platform need to be independent from the appliances used to access its services. This includes smart devices and low capabilities devices.
- Support for varied data and information sources – An e-commerce system should be able to access data sources distributed over the public Internet, internal mainframes, ERP systems data, servers on an enterprise's intranet, and ultimately provide users with a unified view of desired information. For instance, XML standardizes data access and empowers for granular information access.
- Support for rich content – The content to be offered in the platform needs to go beyond text. Animations, images and audio are formats which will enhance and enrich the user experience.
- Varied user interfaces – The user interfaces used to access the e-commerce platforms need to be heterogeneous and customizable.
- Human-like interaction – The better human-like interaction systems that are designed for users to interact in e-commerce applications, the less probability to get users frustrated in using digital sources for their daily commerce activities. Groupware and speech recognition systems are good examples.
- Intelligent learning – For reaching their objectives and offering personalized commerce experiences to their clients, tracking customer behavior and preferences (as far as the legal framework allows) as they interact with the enterprise services is needed, and the system has to automatically learn based on this data.
- Managing information through portal applications.

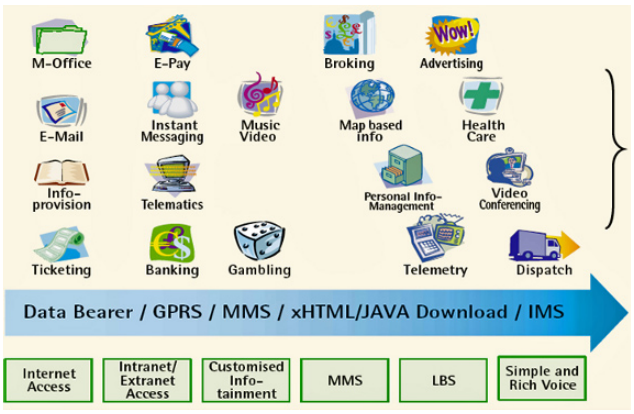

Fig. 1.1 Conglomerate of Applications

- Scalability/global connectivity – The system has to be designed to be scalable to react to increasing customer connectivity.
- Integrate diverse applications – As mentioned before, the system will be likely composed of different devices and technologies, and all of them have to interact for the fulfillment of e-commerce. For example, a web server offering HTML content will have to interact with the payment gateway which could be hosted in a different network and maintained with different technology.
- Support electronic payments and other traditional applications – Traditionally, not only payment has been used in businesses, but also other important applications like contract signing, certified mail, notarization of documents, on-time distribution, supply chain, etc. All these applications have to be imported to the e-commerce system.

At the very beginning we can imagine that every action that can be performed in traditional commerce and administration can be translated into the digital world, and in most of the cases there are few differences in the processes to achieve this objective. One of these differences is **security**, due to the apparently inverse relation between *distance and trust*.

In the actual e-commerce place, users can access the Internet in order to make purchases, bid for products and buy cinema tickets. Soon, more legal activities such as contract signing could be speeded up by the Internet. As it is widely known, one of the main factors for achieving successful e-commerce procedures is trust; that is, e-business will never succeed without the trust from clients. We can analyze the example of eBay [1], which is possibly the most popular bidding site on the Internet. A strong reason for this is its reputation system, which establishes certain trust among users (besides other properties like a secure payment process with Paypal [2] and a complete process policy of the site, etc.).

However, it is obvious that this will not last forever. As more companies come into play and P2P networks spread over the Internet, it is not easy to guarantee trust among participants beforehand. All the users will demand more and more security in their communications and transactions. Therefore, there is a need for the existence of protocols to guarantee that the lack of trust between participants does not stop the success of e-commerce.

1.2 Security Issues in E-commerce

For many years, telecommunications and information technology security was mainly of concern to niche areas such as banking, aerospace and military applications [79]. However, with the rapid and widespread growth in the use of data communications, particularly the Internet, security has become a concern to almost every IT system and, thus, to e-commerce.

As explained before, one of the advantages of e-commerce is the possibility of doing business or purchasing goods anytime and anywhere regardless the distance of participants. But this also supposes a disadvantage because, principals' security has normally been based on trust derived from the physical contact among them, plus other related mechanisms such as hand-written signatures or contracts (often signed in presence of all parties involved). As this is not possible any more (or at least not convenient), security in transactions is not only of paramount importance but also the obstacle that has been continuously delaying e-commerce steps towards a worldwide reality.

There are three main issues in which e-commerce has to be aware of security: platform, service and network security. The first one depends mainly on the OS upon which the application is running, and has been widely studied. So far, the other two issues constitute the main security part over which e-commerce has to be deployed and different solutions and standards have been designed. This part affects the complete security of a remote transaction, beginning with the availability of a reliable transmission network.

Therefore, the first step is to ensure that the communication network is continuously available, which means physical protection of the network medium, correct routing of messages and appropriate operation of the network management. Secondly, it is needed to ensure the authentication and authorization of participants, so as to provide them with services to which they have subscribed and to guarantee the integrity and confidentiality of the exchanges. It is also necessary to preserve evidence that can help resolve disputes and litigation. Next section formally defines these issues.

All these measures are oriented to prevent threats that arise from distribution and openness of actual networks. As summarized in [121], the ITU identifies the following threats:

- Interception of the identity of one or more of the participants by a third party with a mischievous intent;
- Masquerade, whereby one entity pretends to be another entity;
- Replay of a previous message, entirely or partially;
- Defective or fraudulent manipulation of the content of an exchange by substitution, insertion, deletion or reorganization of user's data exchange in a communication by an unauthorized third party;
- User's repudiation or denial of their participation in part or in all of a communication;
- Denial of service and the impossibility of accessing the resources usually available to authorized users;
- Misrouting of messages from one user to another;
- Analysis of the traffic in order to get identities, data type and volume, etc.

These threats, along with the fact that open networks over which e-commerce needs to be implemented, are co-managed by distinct administrative entities which are often unknown to individuals (*lack of trust*), this causing the need for security solutions that address these problems and particularly in e-commerce.

1.3 Security Services and Protocols

In [78], a fresh look was taken at security architecture for networks. The result is Recommendation X.805, which defines a security architecture for providing end-to-end network security. The architecture can be applied to various kinds of networks where the end-to-end security is a concern independent of the network's underlying technology. The general principles and definitions apply to all applications, even though details such as threats and vulnerabilities and the measures to counter or prevent them vary based on the needs of an application. As we saw in our introduction, e-commerce lumps a wide range of end-to-end applications, making this general security architecture fully applicable to e-commerce.

This security architecture is defined in terms of two major concepts, layers and planes (see Figure 1.2, extracted from [79]. The first axis, Security ***Layers***, addresses requirements that are applicable to the network elements and systems that constitute the end-to-end network. A hierarchical approach is taken in dividing the requirements across the layers so that the end-to-end security is achieved by building on each layer. The three layers are (1) *Infrastructure layer*, (2) *Services layer*, and (3) *Applications layer*. One of the advantages of defining the layers is to allow for reuse across different applications in providing end-to-end security. The vulnerabilities at each layer are different and thus countermeasures are to be defined to meet the needs of each layer. The Infrastructure layer consists of the network transmission facilities as well as individual network elements. Examples of components that belong to the Infrastructure layer are individual routers, switches and servers as well as the communication links between them. The Services layer addresses security of network services that are offered to customers. These services range from basic connectivity, such as leased line services, to value added services, such as instant messaging. The Application layer [2] addresses requirements of the network-based applications used by the customers. These applications may be as simple as email, contract signing, data exchange or as sophisticated as collaborative visualization where very high-end video transfers are used in oil exploration, designing automobiles, etc.

The second axis addresses the security of activities performed in a network. This security architecture defines three Security ***Planes*** to represent the three types of protected activities that take place in a network: (1) *Management plane*, (2) *Control plane*, and (3) *End-User plane*. These Security Planes address specific security needs associated with network management activities, network control or signalling activities, and end-user activities, correspondingly. With Security Layers and Security Planes as the two axes (3 Security Planes and 3 Security Layers), the architecture also defines eight Security ***Dimensions*** that are designed to address network security.

Actually, e-commerce and e-government imitate paper-based transactions in which security issues or dimensions are enclosed using different elements such as contracts, hand-written signatures, notaries, manual audit systems, etc. What the security services do is to translate these properties to the digital and networked world.

[2] This book fits mainly in this layer though the non-repudiation service is defined for all of them.

X.805 builds on some of the concepts of X.800 and the Security Frameworks (X.810-X.816), naming these security services as dimensions. In particular, the functionalities of the basic security services of X.800 (Access Control, Authentication, Data Confidentiality, Data Integrity and Non-repudiation), match the functionalities of the corresponding Security Dimensions of X.805. In addition, Communication Security, Availability and Privacy Security Dimensions of X.805 offer new types of network protection.

From an architectural perspective, these dimensions are applied to each cell of the 3-by-3 matrix formed between the layers and planes that the ITU defines, so that appropriate countermeasures can be determined.

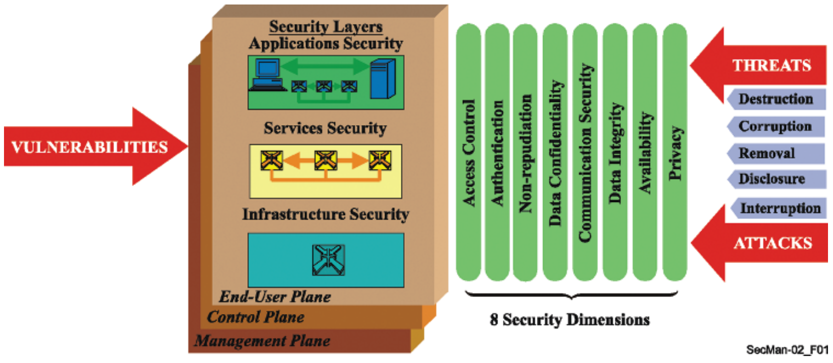

Fig. 1.2 Security Architectural Elements in X.805 ITU-T Recommendation

These dimensions are defined below:

- *Access Control* protects against unauthorized use of network resources. Access Control ensures that only authorized personnel or devices are allowed access to network elements, stored information, information flows, services and applications.
- *Authentication* helps to confirm the identities of communicating entities. Authentication ensures the validity of the claimed identities of the entities participating in communication (e.g., person, device, service or application) and provides assurance that an entity is not attempting a masquerade or unauthorized replay of a previous communication.
- *Non-repudiation* provides means for preventing an individual or entity from denying having performed a particular action related to data by making available proof of various network-related actions (such as proof of obligation, intent, or commitment; proof of data origin; proof of ownership; proof of resource use). Non-repudiation ensures the availability of evidence that can be presented to a third party and used to prove that some kind of event or action has taken place.
- *Data Confidentiality* protects data from unauthorized disclosure. Data confidentiality ensures that the data content cannot be understood by unauthorized enti-

ties. Encryption, access control lists, and file permissions are methods often used to provide data confidentiality.

- *Communication Security* ensures that information flows only between the authorized end points (the information is not diverted or intercepted as it flows between these end points).
- *Data Integrity* ensures the correctness or accuracy of data. The data is protected against unauthorized modification, deletion, creation, and replication and provides an indication of these unauthorized activities.
- *Availability* ensures that there is no denial of authorized access to network elements, stored information, information flows, services and applications due to events (as for instance an attack) impacting the network.
- *Privacy* provides protection of the desired information. Examples of this information include particulars, a user's geographic location, etc.

Non-repudiation is related to authentication but has stronger proof requirements. The major difference is that authentication only needs to convince the other party involved in a communication while non-repudiation should prove to a third party the truth of the event. Its primary purpose is to protect users' communications against threats from other legitimate users, rather than from unknown attackers.

The realization of these security services in electronic commerce applications becomes possible thanks to well-known or standardized protocols. In some cases, a sole protocol covers different security services and, very often, different protocols, with different or alike features, address the same security services. In information technology, a *protocol* is defined as a special set of rules that end points in a telecommunication connection use when they communicate. Protocols exist at several levels in a connection and both end points must recognize and observe the same protocol. For instance, on the Internet, there are the TCP/IP protocols consisting, among others, of TCP that uses a set of rules to exchange messages with other Internet points at the information packet level and IP, which uses a set of rules to send and receive messages at the Internet address level. Other protocols implement (part of) these security dimensions in their design (see the Appendix in this chapter).

In the research area, as well as in the industry arena, most of the security services identified by the standards have been implemented in the form of different protocols, but non-repudiation has not received such attention. As it is demonstrated in the following chapters, although non-repudiation for two entities have been studied in the last years [129], multi-party environments in which several users are involved in a specific e-commerce application (e.g. digital contract signing) have not received sufficient attention.

1.4 The Non-repudiation Service

Repudiation is one of the fundamental security issues existing in paper-based and electronic environments. Dispute of transactions is a common issue in the business world. Transacting parties want to seek a fair settlement of disputes, which brings the need of non-repudiation services in their transactions. The motivation for non-repudiation services is not just the possibility that communicating parties may try to cheat each other. It is also the fact that no system is perfect, and that different and unexpected circumstances can arise in which two parties end up with different views of something that happened. Network failures during the protocol run is a representative example. Hence, we can deduce that transaction and entities (users) play a key role in a non-repudiation service.

We define a *basic transaction* as the transferring of a message M (e.g. electronic goods, electronic cash or electronic contracts) from user A to user B, and represent this event with the following message flow: $A \rightarrow B : M$. Thus, typical disputes that may arise in a basic transaction with a deadline T could be

- A claims that it has sent M to B while B denies having received it;
- B claims that it received M from A while A denies sending it;
- A claims that it sent M before T while B denies receiving it before T.

Fair non-repudiation can be considered as an extended *fair exchange* problem in which non-repudiability is made an integral requirement of the exchange (which in general is not required). Exchange of one data item for another between mutually distrusted parties is usually the difficult part of an electronic transaction. We can find various instances of the general exchange problem in different types of commercial activities: a purchase, contract signing, certified mail or, more generally, in any barter conducted by means of digital networks.

An exchange is said to be *fair* if at the end of the exchange, either each player receives the item it expects or neither player receives any additional information about the others' item. For instance, in payment protocols, fair exchange can ensure that a customer receives the digital goods from a vendor if and only if the vendor receives the payment from the customer. Non-repudiation will additionally enhance the scenario allowing the vendor and the customer to demonstrate **to any other** party that the digital goods was provided and the payment was made respectively.

Non-repudiation acts as a parameter which modifies different requirements in fair exchange. At the same time, non-repudiation augments fair exchange to collecting and processing evidence and resolving disputes over the exchange. To which extent it augments and modifies the requirements of fair exchange transactions depends on the requirements of the non-repudiation service itself.

The transaction's features decide the type of non-repudiation services to be deployed. For any non-repudiation service, evidence is a crucial object playing the main role in resolution of disputes. The processing of evidence usually involves the assistance from *trusted third parties* (TTP). There are different activities at each phase of processing. The non-repudiation policy defines the behavior of these activities. Finally, the eventual success of non-repudiation as a deployed service depends upon technical and legal supports. These and other aspects are analyzed in depth in Chapter 2.

1.5 Challenges and Goals of this Book

During the last years the impressive growth of the Internet and more generally of open networks has originated several security-related problems. Repudiation is one of them. Commercial transactions involve multiple players. Usually, the players mutually distrust one another. Protecting one legitimate player from another is as important as protecting legitimate players from external attackers and a non-repudiation protocol must generate cryptographic evidence to support dispute resolution. Non-repudiation is especially important in electronic commerce and secure Internet applications to protect customers and merchants. Other users, such as online taxpayers or administration users for a secure paperless office, would also need a non-repudiation service.

Research oriented to non-repudiation protocols has been active since the late 90's; considering in most occasions only two parties (*Alice* and *Bob*) as the players of the protocol design scenario. It is time to give a step forward and realize that in many applications there are more than two entities and that two-party protocols are not appropriate for scenarios in which multiple entities are involved. This is the main aim of this book.

In general, protocol requirements definition is independent of the topology which exists in the multi-party scenario, but this is not the case of the protocol design. A good example can be found in contract signing or certified email. In the first one, usually ring architectures are used for several entities, and in the second one selective receipts have to be bore in mind when emails are sent from one originator. Since a 1-N topology is often common for multi-party non-repudiation protocols, we consider an entire subsection for this topology.

Extending the non-repudiation service to multiple entities has been the main objective of our new research in this field and, efficiency our main concern. With the participation of several entities, some protocol properties change. Therefore, we also pursue the unified requirements for multi-party non-repudiation protocols, refine the definitions when needed and provide new ones.

The main challenge for efficiency is to reduce the overload of n participants in the protocol, improving TTP's efficiency without augmenting participants' load and impacting fairness. The overload partially depends on the protocol design. Examples are the number of evidence tokens to be generated, number of recipients the TTP needs to contact, number of simultaneous connections to the TTP in order to retrieve evidence, and the TTP's storage capacity, etc. Furthermore, for ensuring efficiency of the new protocols, providing comparison with existing solutions and evaluation of temporal parameters is also covered in this book.

Additionally, participation of several entities in the application scenario requires new models for facilitating users participation and evidence collection. It also introduces new features, as for instance sending customized messages to other entities. At the same time, we could not obviate all the Internet evolving properties and applications. For such a reason, this book addresses multi-party non-repudiation in three specific scenarios: (1) an OMA-based DRM framework, (2) a practical service charge for P2P content distribution, and (3) an intermediary as a free-roaming mobile agent.

Along the new designs introduced in this book, common properties and requirements fulfillment will be examined. For this purpose, protocol elements and policy in use will be explicitly defined.

The book will be of special interest to network security researchers and practitioners, and at the same time will provide a good support for advanced university courses. Precisely, the topic covered within the book makes it relevant to security-specific modules within the general IT programmes, as well as particularly useful to whole courses that are dedicated towards security. However, and in addition to being used as a course text, the book can be used as an educational reference for technical practitioners, who are already active in the IT domain and the electronic commerce area (or indeed a specific niche within the security field) and wish to broaden their knowledge.

This book is divided into six main chapters:

Chapter 2 focuses on one of the previous security dimensions presented in this chapter. It gives a complete study on the fundamental issues in the non-repudiation service. It analyzes in depth the different sorts of a non-repudiation service, its participants and elements, the roles they play, the phases this service is made up of, and the requirements found in each of them. Finally, it reviews the existing related standards and discusses the supporting legal framework.

In Chapter 3, the multi-party non-repudiation problem is analyzed in general, and specifically, the one-to-many non-repudiation problem doing a state-of-the-art study and adopting a critical position toward current solutions which helps us to identify the main technical shortages and tackle them. In this chapter, the reader can find the definition of a multi-party non-repudiation protocol and deal with its properties in depth.

In Chapter 4, based on the work done in the previous chapter, new non-repudiation solutions for several multi-party environments are introduced. More concretely, it presents a 1-N protocol for the distribution of customized messages and a protocol which allows the provision of a non-repudiation service using a new intermediary entity to ease the collection of digital evidence by final users. Additionally, this chapter introduces an event simulation model which could be used for estimating several parameters explained in the previously described solutions.

In Chapter 5, solutions for certified email and contract signing applications are explored. It extends a two-party certified email protocol to allow multiple participants and provide asynchronous timeliness, which means any party can finish the protocol when desired without losing fairness. At the same time it presents a new protocol for multi-party contract signing, and integrates two properties: *abuse-freeness* which avoids any party to be able to demonstrate it can decide the contract outcome before the contract is signed, and *threshold timeliness* which depends on the trust among participants and the contract value.

More specific multi-party applications which need a non-repudiation service are described in Chapter 6. Specific protocols and scenarios are proposed for these real and practical approaches. In this way, this chapter presents the design and implementation of a non-repudiation protocol with intermediary for an OMA-based DRM framework. The protocol integrates with the architecture generating the evidence that the different participants need. This chapter also introduces a P2P service and payment protocol in which the load of the merchant peer is significantly reduced for only distribution of digital contents while a weakly trusted intermediary agent is used for the advertisement and collection of (small) payments. This protocol makes use of a non-repudiation session with an off-line TTP integrated inside the payment protocol.

The conclusion can be found in Chapter 7, in which we summarize the achievements, discussions and insights of all the solutions presented in this book.

Appendix

Let us briefly describe the widely used security protocols at the present time for better understanding of the protocols later presented in the following chapters.

- Secure Socket Layer (SSL) is the leading security protocol on the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The primary goal of the SSL protocol is to provide privacy and reliability between two communicating applications. The protocol is composed of two layers. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP), is the SSL Record Protocol. The SSL Record Protocol is used for encapsulation of various higher level protocols. One such encapsulated protocol, the SSL Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. One advantage of SSL is that it is application protocol independent. A higher level protocol can layer on top of the SSL protocol transparently.

  The SSL protocol provides connection security that has three basic properties:

  – The connection is private. Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption (e.g., DES, RC4, etc.).
  – The peer's identity can be authenticated using asymmetric, or public key, cryptography (e.g., RSA, DSS, etc.).
  – The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions (e.g., SHA, MD5, etc.) are used for MAC .

  When an SSL session is started, the browser sends its public key to the server so that the server can securely send a secret key to the browser. The browser and server exchange data via secret key encryption during that session. Developed by

Netscape, SSL has been merged with other protocols and authentication methods by the IETF into a new protocol known as Transport Layer Security (TLS).

- Secure HyperText Transfer Protocol (HTTPS) is the protocol for accessing a secure Web server. Using HTTPS in the URL instead of HTTP directs the message to a secure port number rather than the default Web port number of 80. The session is then managed by a security protocol.

There are many other protocols related to security and widely used to provide e-commerce with an implemented framework of security services which are briefly identified in the following list.

- Secure Electronic Transaction (SET) provides a tunnel to process electronic payments for e-commerce purchases. Nevertheless SET has been abandoned by the designing consortium (MasterCard and Visa).
- SOCKS is a proxy server protocol which allows the transmission of messages from within a private enterprise server to the public-facing proxy server. It allows secure video-conferencing to be distributed, along with multimedia applications, including streaming media and collaborative messaging programs.
- Pretty Good Privacy (PGP) is a standard format of email encryption. The sender can use the recipient's public key to encrypt the email while the recipient can use his corresponding private key to decrypt the message.
- Internet Protocol Security (IPSec) secures data transmissions across a Virtual Private Network (VPN). This protocol allows sites to protect their traffic while they intercommunicate over a public Internet connection. For further reference on IPSec, see [122].
- Kerberos, a networking protocol, created by the Massachusetts Institute of Technology, provides strong authentication for client/server applications by using secret-key cryptography.
- The Layer Two Tunneling Protocol (L2TP) combines features from both Microsoft and Cisco products and is an extension of PPTP. L2TP allows an Internet Service Provider to operate a Virtual Private Network.
- Point-to-Point Tunneling Protocol (PPTP) provides corporations with the ability to conduct secure communications on Virtual Private Networks across dial-up, local area network, wide area network, or Internet connections.
- Secure/Multipurpose Internet Mail Extensions (S/MIME) encryption standard uses the RSA algorithm to send secure email. It is included in the latest versions of web browsers and email clients.
- Wireless Equivalent Privacy (WEP) was defined as a security protocol in the IEEE 802.11 (Wi-Fi) standard. It was specifically designed for securing wireless LAN. From the specification point of view, it has been demonstrated to be insecure.

- Secure SHell (SSH) is a UNIX-based security protocol for accessing remote computers. This type of access allows network administrators to control application and web servers from one desktop. It uses the RSA public key cryptosystem for authentication.

Chapter 2
Fundamentals of Non-repudiation

Abstract This chapter gives a complete study in the state of the art about standards and elements of the non-repudiation service. It analyzes in depth the different sorts of a non-repudiation service, its participants and elements, the roles they play, the phases this service is made up of and the requirements found in each of them. Finally it reviews the existing related standards and discusses the supporting legal framework.

2.1 Specific Non-repudiation Services

Non-repudiation services help the transacting parties to settle possible disputes over whether a particular event or action has taken place in a transaction. We define a *non-repudiation protocol* as a message flow in which entities exchange digital evidence in order to provide such non-repudiation services.

In an electronic transaction, message transfer is the building block and there are two possible ways of transferring a message (see Figure 2.1).

1. The originator O sends the message to the recipient R directly; or
2. The originator O submits the message to a *delivery agent* D which then delivers the message to the recipient R.

In the direct communication model, because the originator and the recipient potentially do not trust each other, the originator is not sure that the recipient will acknowledge a message it has received. On the other hand, the recipient will only acknowledge messages it has received. In order to facilitate a fair exchange in which neither party will gain an advantage during the transaction, a TTP will usually be involved. Of course, the extent of the trusted third party's involvement varies among different protocols, and as we will see in Section 2.3, this provides for a protocol distinction.

To establish the accountability for the actions of the originator and the recipient, the following non-repudiation services are required.

- *Non-Repudiation of Origin* (**NRO**) is intended to protect against the originator's false denial of having originated the message. *Evidence of Origin* (EOO) is generated by the originator, or a TTP on its behalf, and will be held by the recipient.
- *Non-Repudiation of Receipt* (**NRR**) is intended to protect against the recipient's false denial of having received the message. *Evidence of Receipt* (EOR) is generated by the recipient, or a TTP on its behalf, and will be held by the originator.

In the indirect communication model, a *delivery agent* is involved to transfer a message from the originator to the recipient. In order to support the settlement of possible disputes between the originator and the delivery agent or between the originator and the recipient, the following non-repudiation services are required.

- *Non-Repudiation of Submission* (**NRS**) is intended to provide evidence that the originator submitted the message for delivery. *Evidence of Submission* (EOS) is generated by the delivery agent, and will be held by the originator.
- *Non-Repudiation of Delivery* (**NRD**) is intended to provide evidence that the message has been delivered to the recipient. *Evidence of Delivery* (EOD) is generated by the delivery agent, and will be held by the originator. Similarly, we should be aware that evidence provided by this service cannot be used to make further deductions about the delivery status without some sort of assumption in the communication channel.

EOS and EOD are provided by the delivery agent to the originator and could be used in two possible contexts. They could be used for resolving disputes over the service provided by the delivery agent. In this case, the delivery agent is a party to the dispute and does not act as a TTP, and EOS and EOD establish the accountability for the delivery agent's actions. On the other hand, when EOS and EOD are used outside disputes between the originator and the delivery agent, e.g. for testifying the time of submission and delivery, the delivery agent has to be a trusted third party which will not collude with the originator to provide bogus evidence. We will further discuss this and other problems in Chapter 4, in which a non-repudiation protocol with an *intermediary* or delivery agent is presented and analyzed.

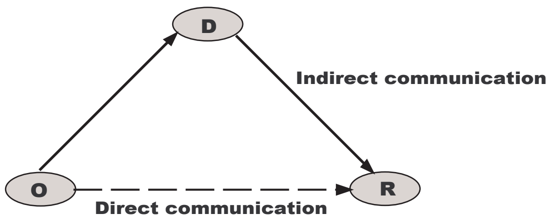

Fig. 2.1 Models of Message Transfer

2.2 Evidence

This is the data that can be used if a dispute arises. It can be either generated and stored by the local user or by a third party. Its format depends on the cryptographic mechanisms agreed in the service. Typically, examples are *digital signatures* (public key cryptography) and *secure envelopes* (secret key cryptography). Whichever the format is, this evidence has to be composed of the common information that helps to clearly identify a transaction and thus resolve a possible dispute in a more deterministic way. Some of these common elements are:

- Non-repudiation service to which evidence is related
- Non-repudiation policy identifier
- Originator identity
- Recipient identity
- Third party identity if evidence generator differs from the originator
- Message or a digital fingerprint
- Message identifier
- Information needed for verifying evidence (i.e. digital certificate, symmetric secret key info) if it is not publicly available
- TTP's identifier and role (see Section 2.3) when involved in the service
- Unique evidence identifier
- Time information (time and date that evidence was generated, expiry date, ...). If this data is certified by a *Time Stamp Authority* (TSA), it could include a time-stamp service identifier.

When a secure envelope is used to provide evidence, data is stamped with a secret key known only by the TTP, thus being the generator and verifier of evidence as requested by the users.

TTP participation can be relaxed through the use of smartcards or manipulation-resistent modules [81] in which secret keys are properly installed. In this case, the smartcard plays the role of a distributed TTP. The smartcard of the generator is used for evidence generation and the verifier's one only for validation. The latter one cannot be used to generate evidence with the secret key (even if it is the same one), such that only the user who owns the generator smartcard could have created the evidence. This is achieved by correctly installing the secret key and the module which controls whether the user can use its smartcard for generation or verification. This module is tamper-proof and different for the generator and the verifier such that it performs just one of the two possible functions.

The secure envelope maintains confidentiality (e.g. symmetric cipher with the secret key) and integrity of the information using a digital fingerprint (i.e. hash function).

When a digital signature is used to provide evidence, information is enclosed in a data structure digitally signed such that only the generator can sign the data and the rest of participants (recipients and TTP) can verify it. Unforgeable digital signatures provide a clear statement of the essential components of handwritten signatures; namely, a user's ability to sign by itself, a universally agreed verification procedure and the assertion that it is unfeasible (or at least very hard) to selectively forge signatures in a manner that passes the verification process without being detected.

In order to bring all of this into reality, digital signatures used as evidence in a non-repudiation service need an infrastructure backing it up. As we mentioned in Section 2.3, there will be a third party certifying participants' link between their identity and public key. Only in this way any recipient can verify the digital signature. Digital signatures introduce a new disrupting element in the non-repudiation service, as the link certified by the TTP (often referred as digital certificate) may have an expiry date. This fact has to be checked when evidence is verified either by the recipient or a TTP (e.g. an adjudicator). If this link has expired, evidence will be valid only if it was generated before. For this reason, time information has to be included in the evidence generated.

In general, it is more efficient, in terms of computation, for users to use secure envelopes with symmetric techniques. Nevertheless, in this case:

a) Principals have to **unconditionally trust** a third party for evidence generation and verification;
b) TTP's on-line availability is needed in order to provide the service when requested;
c) If users are to relax the TTP participation as stated previously, then they need to use dedicated hardware to avoid the TTP being a bottleneck.

So, users would likely prefer to use digital signatures because:

a) There is an **implicit trust** over the TTP computing the digital certificates and an implicit cost derived from the continuous revocation info update needed (see Section 2.3 for further reference), but this trust can be relaxed with legal agreements between users and authorities, audited registration processes and a quite advanced standardization [80];
b) **Trust** imposed over the TTP is **less critical** than in the former case, since the TTP certifies the existence of a binding between a user and a public key, verifying at the same time that this uniquely corresponds to a private key. But this TTP does not need to know the key itself. So, there is no danger of this entity accessing the content or even being able to generate it (as with secure envelopes).

Additionally, Maurer [100] proposed a novel view of digital evidence called *digital declarations*, based on a digital recording of a willful act indicating agreement to a document or contract. This proposal tries to address some of the problems related to digital signatures as mentioned above. It also includes new elements in the digital evidence (like willful acts) to augment the concept of evidence, bringing it nearer to the one used in human judgements. Among all the concepts introduced by Maurer, the semantic of certificates is very important. He proposes that when registering the public key, the user must explicitly commit to be liable for signatures with respect to that public key. Evidence confirming this commitment, designated as *commitment declaration*, is generated and stored by the *Certification Authority* (CA) and can be presented by it if the need arises. This simple change has several important implications.

- The certificate has absolutely no value as evidence in court, only the commitment declaration does.
- Only the recipient of a signature (evidence) must trust the CA.
- An expiration date stated on the commitment declaration must be interpreted differently. It specifies until when evidence can be presented as valid, regardless of when it was generated. In other words, evidence expires, not public keys. As a consequence of this view, the validity period of evidence should be kept short.
- A commitment declaration cannot be revoked. Revocation of a public key is impossible (not needed).

Actually, with these definitions, the signature seems to be more insecure than in the traditional view when revocation is possible while the commitment declaration is valid. But, on the other hand, it seems to be closer to the business model if we consider the discussed users' liability in the traditional approach [1]. Maurer proposes the concept of delegation signatures (digital signatures assisted by TTPs) to strengthen its security. Furthermore, this digital declaration and commitments are a new approach to digital evidence with no implications on how non-repudiation protocols handle the evidence.

2.3 Roles of the TTP

One of the main features which allows us to classify the TTPs is its role in a non-repudiation service. A TTP which does not participate actively in the non-repudiation service; i.e., it will be invoked only when there is something wrong in a transaction, is referred to as an *off-line TTP*. An *on-line TTP* participates in the generation and verification of evidence throughout the protocol instance. An *in-line TTP* acts as an intermediary in all the interactions among the users. The difference between third parties that are used only in case of exceptions and third parties that are actively involved in a protocol was first explained in [47]. Obviously, the first type is preferred if efficiency is the major concern, but in some situations and e-commerce applications, to have a delivery agent or intermediary could be the best practical solution.

Other roles have appeared as a consequence of research achieved in exchange protocols. These new approaches aim at eliminating the involvement of the TTP completely but need strong requirements; either all involved parties must have the

[1] In the current digital signature laws, the "hot potato" does not come from the technical aspects but from the users' liability when it does not understand the technical process or this is done without its knowledge.

same computational power as in *gradual exchange* or fairness depends on the number of protocol rounds [97] as in *probabilistic protocols*.

At the same time there are additional TTPs which provide services needed by the non-repudiation service.

- *Certification Authority* – It provides authentication information. This information will be needed for authentication purposes (e.g. prior to the beginning of the protocol), binding a public key to an identity as described in ITU X-509 Recommendation [80] for digital certificates. It allows to digitally sign messages as well as their verification. It also provides frequent revocation information of these certificates, since if a user accepts a revoked certificate as valid, all security services based on digital certificates can be bypassed. In summary, it provides the needed infrastructure for digital signatures and authentication.
- *Time-Stamp Authority* – The time that in which event occurs can be as important for the e-commerce application as whether it took place or not. It can be used for *QoS* purposes (e.g. to see whether the delivery agent fulfilled its promise to deliver the message in a timely way), or just because it is needed for the dispute resolution process. Since digital signatures can be revoked, it is important to specify the time (slot) in which the signature was generated [69, 99, 131]. If it was generated prior to the revocation of the certificate, then evidence has to remain valid, at least, till its expiration (defined by the non-repudiation policy in effect). This time could be provided by the originator of the evidence. Nevertheless, if it is one of the principals participating in the protocol, as trust is not often assumed among their clocks, then a TTP has to do this task. This entity is the TSA, which includes a time-stamp to evidence and encloses it in a way that maintains its integrity and authentication information (e.g. digital signature).
- *Electronic Notary* – A notary is trusted by the entities to provide correct evidence on their behalf or verify evidence correctly as well as for registering it [93]. A registration token can be provided to the entities such that they can access and refer to the evidence using the audited information. This audited information can be composed by time, identities of the parties involved, a digital fingerprint of the message and non-repudiation policy used. In practice, a TTP acting in-line or on-line in the protocol can notarize the evidence.
- *Adjudicator* – This is the party which drives the dispute resolution process to a conclusion depending on evidence presented by the entities and optionally contacting participating entities. In order to facilitate its task, a well defined dispute resolution process in accordance with the non-repudiation policy must exist. This dispute resolution process has to take into consideration the legal framework in which it is defined. New or established ODR processes can be used [41].

2.4 Non-repudiation Phases

Non-repudiation services establish accountability of an entity related to a particular event or action to support dispute resolution. Provision of these services can be divided into different phases such as: evidence generation, evidence transfer, evidence verification, evidence storage, and dispute resolution.

Evidence Generation

Evidence generation is the first phase in the provision of a non-repudiation service. Depending on the non-repudiation service being provided and the non-repudiation protocol being used, evidence could be generated by the originator, the recipient, and/or the TTP. The elements of non-repudiation evidence and the algorithms used for evidence generation are determined by the non-repudiation policy in effect. When non-repudiation of origin and receipt services are required, evidence of origin and receipt are usually generated by the originator and the recipient, respectively, if digital signature is used for evidence generation. When non-repudiation of submission and delivery services are required, evidence of submission and delivery will be generated by a TTP like a notary or a delivery authority. If a secure envelope is used for evidence generation, it should always be generated by a TTP on behalf of the originator or recipient.

A TTP may also generate and provide supporting evidence in a non-repudiation service. For example, in a fair non-repudiation protocol [132], the notary will digitally sign the message key provided by the originator and make the confirmed message key available to both the originator and the recipient. The confirmed message key will serve as part of non-repudiation evidence to prove that the message key was sent from the originator (via the notary), and is available to the recipient.

Evidence Transfer

Evidence transfer is the most challenging phase in the provision of a fair non-repudiation service. It mainly consists of the sending and reception of evidence among participants. Actually, it represents the core of a non-repudiation protocol. It is greatly influenced by communication channel properties. The different options are as follows:

1. The communication channel is *unreliable*. In this case, data can be lost.
2. The communication channel is *resilient* (also called asynchronous network). In this case, data is delivered after a finite but unknown amount of time.
3. The communication channel is *operational* (also called synchronous network). In this case, data is delivered after a known, constant amount of time.

An unreliable channel will in most cases be transformed into a resilient channel by the use of an appropriate transport protocol (e.g. retransmissions).

Evidence Verification

Newly received evidence should be verified to gain confidence that the supplied evidence will indeed be adequate in the event of a dispute arising. The verification procedure is closely related to the mechanism of evidence generation.

Evidence generated through a digital signature can be verified by any party to which the public key certificate and the revocation information (e.g. the CRL or OCSP server [2]) are available. According to the requirements on non-repudiation evidence defined in Section 2.2, the origin, integrity and validity of a digital signature should be verified. In order to verify the integrity of a digital signature, the verifier needs to use the verification key to check whether the digital signature is the result applied to the expected message. For verification of the origin of the digital signature, the verifier needs to check whether the verification key is bound to the identity of the expected signer in the public key certificate. To verify the validity of the digital signature, the verifier needs to check whether the verification key had not expired and not been revoked at the time that the signature was generated. The last step implies that the verifier needs to check:

1. the trusted time stamp which is applied on the digital signature to identify the time of signature generation;
2. the expiration date of the verification key specified in the public key certificate;
3. the revocation information of the public key certificate.

If evidence is generated through a secure envelope, it should be verified by a trusted third party at the request of the user because the secret key for evidence generation and verification is only held by the TTP. Obviously, the extra communication between the user and the TTP will cause a substantial delay which might be unacceptable for many on-line electronic transactions.

Evidence Storage

Because the loss of evidence could result in the loss of future possible dispute resolution, the verified evidence needs to be stored safely. The duration of storage will be defined in the non-repudiation policy. For extremely important evidence aimed at long term non-repudiation, it could be deposited with a TTP.

There are different associations and organizations for long term archives like ARMA [8] and ERA [18], and this is an active research and development area [111]. It is specially important to mention the IETF Working Group Long-Term Archive and Notary Services (LTANS) [14] since non-repudiation of access to documents is one of the design principles. The objective of the LTANS working group is to define requirements, data structures and protocols for the secure usage of the necessary

[2] Certificate Revocation List (CRL) is a data structure which maintains reference to all the revoked certificates. On-line Certificate Status Protocol (OCSP) is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate.

archive and notary services. Up to date, the Long-Term Archive Service Requirements [125] and Evidence Record Syntax [63] have been published as RFCs.

Dispute Resolution

Dispute resolution is the last phase in a non-repudiation service. This phase will not be activated unless disputes related to a transaction arise. When a dispute arises, an adjudicator will be invoked to settle the dispute according to the non-repudiation evidence provided by the disputing parties and the non-repudiation policy in effect. This policy should be agreed in advance by the parties involved in the service.

The adjudicator needs to verify the evidence, probably with the assistance from other TTPs, e.g. from a notary when evidence was generated through a secure envelope. Nowadays, different on-line arbitrator platforms [3] exist which allows for dispute resolution processes through document and evidence transactions as well as the cooperation of on-line parties [49, 9]. The dispute resolution process can either be registered in one of these platforms and use its services or use its own rules for the definition of an on-line arbitrator.

2.5 Non-repudiation Requirements

Different targets of each non-repudiation service may influence the protocol design. Nevertheless, there are several common requirements on the design of a good non-repudiation protocol:

- *Fairness*: Repudiation can only be prevented when each party is in possession of proper evidence and no party is in an advantageous position during a transaction. The reliability of communication channels affects evidence transfer. Moreover, a dishonest party may abort a transaction, which could leave another party without evidence. Various fair non-repudiation protocols with different features have been proposed. Some of them can be found in [90].

  Asokan defined two levels of fairness [23]. A protocol fulfills *strong fairness* when the exchange is completed, the sender A can prove to an arbitrator that the recipient B has received (or can still receive) the item, without any further intervention from A. On the other hand, a protocol fulfills *weak fairness* when the exchange is completed, A can prove to an arbitrator that B has received (or can still receive) the item, or otherwise an affidavit can be presented to demonstrate that B misbehaved or a network failure occurred.
- *Efficiency* is another criteria. TTPs will usually be involved in non-repudiation services and its involvement will be essential in order to determine the efficiency of the protocol. Fair non-repudiation protocols proposed in [28, 27, 133, 112, 24,

[3] Note that these platforms themselves may need to implement a non-repudiation service.

98, 102] meet the criteria of efficiency and are often called *optimistic* protocols. Some authors define this property as *effectiveness*; that is, if no error occurs and no party misbehaves, then the TTP should not intervene.

- *Timeliness* is also desirable in evidence transfer. For various reasons, a transaction may be delayed or terminated. Hence, the transacting parties may not know the final status of a transaction on time, and would like to unilaterally bring a transaction to completion in a finite amount of time without losing fairness.
- *Policy* has to perfectly define all the parameters needed by the non-repudiation service, some of which can be: rules for evidence generation and verification, rules for evidence storage, evidence use and the dispute resolution process. More specific parameters to be defined by the non-repudiation policy in effect are identified throughout this chapter as, for instance, the algorithms needed for evidence generation and verification.

  In [129], a general criteria is presented as a set of questions that can be used as a guideline. Here, we refine this set with more questions that should be addressed by the non-repudiation policy.

  Related to evidence generation:

  - What evidence should be generated in the non-repudiation service?
  - Which TTP should be involved in evidence generation?
  - What elements should be included in the evidence?
  - Are all of them mandatory?
  - Which cryptographic algorithms will be used?
  - What is the encoding format?
  - What is the length of cryptographic keys?
  - Which parties will be involved in the generation process?

  Related to evidence transfer:

  - Which non-repudiation protocol will be used?
  - Which TTP will be involved in evidence transfer?
  - What are the channel assumptions?

  Related to evidence verification and storage:

  - What mechanism will be used for maintaining the validity of evidence?
  - Which TTP will be involved in evidence verification?
  - How long should the evidence be stored?
  - Which cryptographic algorithms will be used for verification?
  - Under which circumstances is the evidence regarded as valid?
  - Does the evidence need to be confidential?
  - What are the access control rules for accessing the evidence?

  Related to dispute resolution:

- Which entity will play the role of adjudicator?
- Which parties should be involved in dispute resolution? And which TTP?
- What evidence should be provided?
- What are the rules and steps followed in the resolution process itself?
- What is the expiry date of evidence?
- Which law should be referred to enforce the arbitration?

All of these requirements will be further discussed and used as guidelines on the design and analysis of the different multi-party non-repudiation protocols developed in this book. Nevertheless, there are optional requirements depending on the application itself. If the application requires them, they turn out to be as critical as the common ones previously defined. These optional requirements are as follows.

- *Verifiability of TTP* – This property adds one level of security to the protocol itself when it does not exist a strong trust relationship among participants with the TTP which collaborates in the protocol. If the TTP misbehaves resulting in a loss of fairness for any participating entity, all harmed parties will be able to prove it to an arbitrator or verifier. It can be very useful during the initial setup of a non-repudiation infrastructure as well as in those scenarios in which the TTP has to be selected by the entities on the fly (e.g. in an ad-hoc network). It usually assumes that when the TTP misbehaves, the rest of the entities are honest.
- *Transparency of TTP* – It also appears in the literature as *invisible TTP*. If the TTP is contacted to help in the protocol, the resulting evidence will be the same as the one obtained in the case the TTP is not involved. This is especially important in practical cases, in which an institution does not wish to change the existing processes to accommodate the new signatures or affidavits generated by TTPs. At the same time, this property helps on the privacy of users with respect to the use or not of a TTP during the protocol run.

Unfortunately, these last two properties are more often incompatible (achieving one of them increases the difficulty to fulfill the other one) and a trade-off has to be assumed when designing the protocols.

2.6 Analysis of Standards

The ISO and ITU standards provide a guideline for engineering and should reflect the state-of-the-art of science and technology. Non-repudiation is one of the security services in the ISO/OSI security framework, and is especially important for securing electronic commerce. Many efforts have been devoted to the standardization of non-repudiation services and mechanisms. However, some issues have not yet been well addressed.

There are two international standards dealing with non-repudiation: ISO/IEC 10181-4 [73] [4] and ISO/IEC 13888 [76, 75, 74]. ISO/IEC 10181-4 refines and extends the concept of non-repudiation services as described in ISO 7498-2 and provides a framework for the development and provision of these services. In this framework, the goal of non-repudiation and types of non-repudiation services are defined. The basic mechanisms for non-repudiation services and general management requirements for these services are identified. The roles that a TTP plays in non-repudiation services are listed. The relationship of non-repudiation services to other security services is explained. As a general framework, this standard does not include specific non-repudiation mechanisms. This remains as an open issue treated in [75, 74].

ISO/IEC 13888 "Information technology - Security techniques - Non-repudiation" is composed of three parts. ISO 13888-1 [76] [5] serves as a general model for subsequent parts specifying non-repudiation mechanisms using cryptographic techniques. It establishes two main types of evidence, the nature of which depends on cryptographic techniques employed: the *secure envelopes* generated by an evidence-generating authority using symmetric cryptographic techniques, and *digital signatures* generated by an evidence generator (which can be the user itself) or an evidence generating authority using asymmetric cryptographic techniques. It also describes non-repudiation mechanisms generic to the various non-repudiation services for the following phases of non-repudiation (see Section 2.4): evidence generation, transfer, verification, storage and retrieval, and dispute resolution. Those mechanisms are then applied to a selection of specific non-repudiation services (see Section 2.1) such as non-repudiation of origin, non-repudiation of delivery, non-repudiation of submission, and non-repudiation of transport. The standard also recommends how a smartcard (which is called a manipulation-resistent module) can be used to generate, store and validate evidence.

In ISO 13888-2 and ISO 13888-3 [75, 74], a set of non-repudiation mechanisms based on symmetric and asymmetric cryptographic techniques are identified. All of them are final international standards in the different phases that ISO/IEC apply to its documents. The history of this multipart standard which is being developed by ISO/IEC JTC1/SC27 dates back to August 1991 [72]. Zhou's book [129] analyzes the ISO/IEC 13888 non-repudiation mechanisms, and points out their weaknesses and limitations. It also discusses the problems on defining the roles of time stamps in the ISO/IEC 13888 non-repudiation evidence.

In 2006, in response to a request of ISO Subcommittee 27 (SC27) Secretariat, the Working Group 2 (WG2) of SC27 agreed to revise ISO/IEC 13888-2 as well as ISO/IEC 13888-3. The same happened later with ISO/IEC 13888-1. Drafts of the three parts have been circulated within WG2, but no final documents are available yet, though up to the moment it seems that changes will not be dramatic.

On the other side, ITU defines a general framework for the provision of non-repudiation services in X.813 [81] similar to ISO/IEC 10181-4. It defines non-

[4] It has been revised by 10181-4:1997 Information technology – Open systems interconnection – Security frameworks for open systems: Non-repudiation framework.

[5] This document revises ISO/IEC 13888-1:1997, which is withdrawn.

repudiation as "the ability to prevent entities from denying later that they performed an action". The Non-repudiation Framework extends the concepts of non-repudiation security services as described in X.800 and provides a framework for the development of these services. It also identifies possible mechanisms to support these services and general management requirements for non-repudiation.

2.7 Supporting Legal Framework

When non-repudiation services are enforced in e-commerce, its legal framework should be considered. The legal framework of e-commerce has been traditionally provided by governments in order to foster the growth of the digital economy, giving customers and companies greater confidence in their on-line transactions. This legal framework has been focused on the lawfulness of e-commerce transactions, such as evidence collection in order to protect different participants.

In December 1999, the European Union (EU) approved a Directive [52] giving digital signatures on contracts agreed upon over the Internet the same legal status as their handwritten equivalents. This was regarded as a crucial step in the struggle to put Europe ahead in electronic commerce. However, this Directive is not exempt from difficulties [48].

As a sample, in Spain, the *Real Decreto 14/1999* (September 17th) gave a first step towards regulation of digital signatures. A complete review with inclusion of digital national IDs and total renewed equivalence between digital signature and hand-written signature appeared in the Law 59/2003 of Digital Signature (December 19th). The new equivalence introduces the concept of "acknowledged digital signature" as the one generated using acknowledged digital certificates and secure ciphering devices, allowing at the same time to unify this type of certificates for QES among the countries of the EU. Nevertheless, any digital signature will be regarded as legal evidence (conclusive or not). This law seems to be quite important whenever the type of evidence has to be decided, since it is stated that

> The digital signature constitutes an instrument which is able to check the origin and integrity of messages, offering the foundation for avoiding repudiation if time-stamps are used.
>
> Art. 8 – The support in which the electronic data appears digitally signed is legitimate as documental evidence in a judgement.

So, it is clearly described that in case that disputes have to be resolved by an official judge, digital signatures will be more likely regarded as evidence than its counterpart secure envelope for the Spanish law and, in general, for the European one.

Nevertheless, as we have seen for the European Directive on Electronic Signatures, treatment of digital evidence from a legal point of view is not trivial, and there is a very interesting discussion in [100] about legal issues of evidence. Despite of the efforts from governments on bringing digital signatures for non-repudiation services into law, there is still a lack of internationally applicable law, lack of viable business models for fostering the creation of a global PKI, problems with the integration into business processes and the abstractness and complexity of the subject matter resulting in slow user acceptance.

As stated in different laws, digital signatures will have to be treated as handwritten ones are. Actually, the value of handwritten signatures is not their security (indeed, they are easy to forge), but rather that it creates a situation in which a person knows whether or not she signed, thus guaranteeing her awareness of performing a conscious and wilful act. Due to this guaranteed awareness, the denial of having signed a document is a precise and meaningful claim, equivalent to the serious claim that the signature is forged. Whether it happens with digital signatures is a crucial issue in the legal framework accepting digital evidence.

We assume (it is stated so in most laws for e-commerce) that digital evidence (e.g. digital signature) alone implies liability. However, legislators have recognized the problem that if only digital signatures were relevant in a dispute, then users would have no possibility to defend themselves in presence of a correct digital signature. This raises the question of whether the presence of a digital signature involves a willful act from the alleged creator. In other words, it must be solved when electronic evidence is regarded as *documentary evidence* [115] - whether it is expected to 'stand on its own' and requires no context or interpretation by expert witnesses, or it is just electronic evidence as supporting evidence (where independent explanation of its relevance is necessary). Basically, it consists of finding an intersection between digital evidence and willful act. Two possible solutions can be found.

1. A digitally recording of a willful act (e.g. a video recording of the user clicking 'OK' in its digital signature generation application over the intended digital document) is included in the digital evidence.
2. The digital signature systems ensure that there is no way in which a digital signature can be generated without the user knowledge and consent.

From both solutions, the latter one has usually been used by law. Law generally expects a highly secure system for digitally signing documents will be available (and even bring into law general features about them, as in the EU Directive). The other solution is also contemplated by law in some cases in which the presence of witnesses or additional handwritten signatures are needed. At the same time, both of them have their disadvantages. Digitally recording a willful act and transmitting it over the network to the recipient could rise some practical problems. Equally, the use of highly secure systems (and highly 'digitally' educated users), which ensures that only the user can consciously digitally sign a document is far from reality.

From the above paragraphs, a notion of *admissibility* of electronic evidence can be extracted: evidence must have been lawfully collected, it must be collected in accordance with formal requirements and must respect privacy. The crucial point is that integrity and authenticity of material should be established in court. This requires standard techniques and methods for the collection, preservation and presentation of stored material. At the same time, digital material that is not readily admissible as documentary evidence (e.g. sound and image files) may require some form of presentation technique or technology and other supporting documentation to explain its relevance.

The key issue which summarizes the discussion above is the slight difference existing between the definitions of **Cryptographic** non-repudiation (the one considered in previous sections) and **Law** non-repudiation (the traditional view of this term as defined by laws). There are several subtle but important divergences.

1. When using the digital signature as evidence, the crypto definition does not consider the gap existing between the user's will to sign a document and the layers of hardware and software placed in the device which finally performs the mathematical and computational process of signing a digital document. In the law definition this gap cannot exist as the evidence must be generated with the physical presence and will of the user.
2. There is a shift in the responsibility of proving the signature link to a user. In the ITU definition (crypto-based), non-repudiation of origin, for instance, means the originator being unable to deny its participation in the transmission. In case of a dispute, it may need to demonstrate the evidence (e.g. digital signature) was not generated by him or the signatory has no right to repudiate a digital signature at all. On the other side, in the traditional law definition, if the alleged signatory disputes the signature as belonging to him or her then the onus falls upon the relying party to prove that the signature is in fact that of the alleged signatory.

Whether only a digital signature itself can be regarded as evidence in court or additional components (e.g. a willful act of agreement digitally recorded with respect to the document signed) are needed, remains negligible for the application of non-repudiation protocols, which includes not only the type of evidence, but also other phases as generation, distribution, validity, storage and dispute resolution.

Part II
Multi-Party Non-repudiation

Chapter 3
Multi-Party Non-repudiation: Analysis

Abstract Non-repudiation service has received attention in an end-to-end fashion (typically between two parties Alice and Bob), and this environment has been discussed in the previous chapter. This chapter gives a complete view of non-repudiation protocols in the multi-party environments. We first define what the MPNR problem is, and revisit the requirements extracted for a non-repudiation service. Previous works are presented and analyzed, specially in the one-to-many situation which seems more natural for MPNR protocols. We also sketch their weaknesses and how they will be improved in the successive chapters. Finally we define the model and primitives to be used in the solutions presented in this book.

3.1 General MPNR Problem

As commerce applications like e-voting, e-bidding, etc., usually involve several parties, this monograph focused on non-repudiation protocols when several parties are involved. For identifying the multi-party non-repudiation problem, we study several existing approaches of multi-party scenarios in the related topics such as fair exchange, contract commitment, etc. that can be seen in the following chapters. We also pay attention to existing frameworks (e.g. ANX) and solutions (e.g. DRM) to see how multi-party non-repudiation protocols could fit in such scenarios.

3.1.1 Definitions

Extracted from the different multi-party applications and protocols, let us define our view of a MPNR scenario:

Definition 3.1. In a general MPNR scenario, $n \mid n > 2$ entities agree to use a non-repudiation protocol for exchanging messages (general or specific purpose) and collecting evidence of the transactions performed for the exchange of those messages.

Of course, different topologies are possible (e.g., one-to-many, ring, mesh, ...), but some of them seem to be more natural than others. For instance, sending the same message to several entities is more related to existing Internet applications than one entity receiving the same message from different originators [1]. Nevertheless, all the topologies need to be considered as long as an application exists for them. There are applications in the collaborative and e-learning area in which many to many messages are a reality. For example, in [26], a *simultaneous payment for receipt* is presented as an instantiation of a many-to-one application, and in [24] a many-to-many contract signing protocol is depicted.

Let us imagine the following scenario which sketches a virtual application for managing market shares. Several users (or machines) which are market share holders meet up in order to bid for each other stocks, and some users could share one or several stocks. We can represent the bids as a set of messages $Set_i = m_1, \dots, m_k$. For a user U_i, there are $n-1$ entities to which offers from the set can be sent. The same offer can be submitted to all of them or to a subset of the entities. Also different offers can be sent to all or any combination of them. Once the offers are sent, user U_i will wait for a response. This response could be made individually or could require several recipients gathering for replying an offer. This is a typical example of a many-to-many application as seen in Figure 3.1.

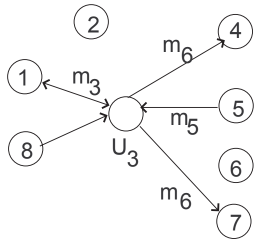

Fig. 3.1 n2n Scenario

This simple figure can be represented by a binary matrix, in which "1" in the (i,j) position indicates that user U_i sends a bidding message to user U_j :

$$\begin{pmatrix} 0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,\dots\dots\dots\dots \\ 1\,0\,0\,1\,0\,0\,1\,0 \\ 0\,\dots\dots\dots\dots \\ 0\,0\,1\,0\,\dots\dots \\ 0\,\dots\dots\dots\dots \\ 0\,\dots\dots\dots\dots \\ 0\,0\,1\,0\,\dots\dots \end{pmatrix}$$

[1] This is the reason for which one entire section is devoted to this environment.

Each user U_i could need evidence of receipt of the message sent while receivers need evidence of origin of the offer received. Many other multi-party applications and protocols can be represented using a matrix. For instance, in the case of the *MPCS* protocol mentioned above, a matrix in which all elements except the diagonal are "1" represents mathematically the topology used.

This scenario could be seen as a typical multi-party fair exchange scenario as described in [26]. Even though general MPNR and multi-party fair exchange protocols have a common design goal (*fairness*), several differences can be found.

- In a fair exchange protocol each entity offers a priori known item (i.e., something is known about the item a priori but not its precise content) and receives another item, also known a priori. In a multi-party fair exchange protocol one can imagine sending an item to one entity and receiving an item from a different one. In non-repudiation it does not make sense that one entity receives some data and a different entity sends the corresponding receipt of that data.
- With a general message M, non-repudiation is more related with a certified email service (see 5.1 for a formal definition) in which the receipt has to be sent by the receiver in order to be able to disclose the message received and optionally obtain an evidence of origin for it. Therefore, a ring topology is not applicable in a *general* MPNR service.
- Due to the fact that a non-repudiation service is not an exchange of items, MPNR protocols have to continue and finish even if some parties do not reply. Only parties following the protocol correctly should be able to disclose the message and obtain evidence of origin for it.
- In some MPNR applications, the message content is previously revealed and known by the recipients. In these applications it is more important the origin and occurrence of the transmission, that is, the fair exchange of EOO and EOR.
- There is no *exclusion-free* property as defined in [66] for MPNR protocols. In other words, since there is no need for a setup phase to agree on participants and items, no danger of excluding participants exists.

The reason of these differences is also due to the classification of fair exchange and non-repudiation as services or applications. Although it seems clear non-repudiation is a service [2], sometimes it is referred in the literature as an application and compared with fair exchange. Actually, fair exchange can also be considered a service. As an example, Asokan has traditionally considered fair exchange as a service for electronic commerce applications. While non-repudiation might provide a service to applications like certified electronic mail, fair exchange can provide service to other upper-layer applications as well like payment protocols or digital contract signing. Although non-repudiation should be a security service in this kind of applications, they can be designed without it (see Figure 3.2). Note that this view is not contrary, but complementary, to that provided in [95].

Other type of applications in which multi-party protocols appear (either for fair exchange, non-repudiation, contract signing, certified electronic mail, or any other

[2] In fact, as mentioned in the previous chapter, it has been standardized as a service.

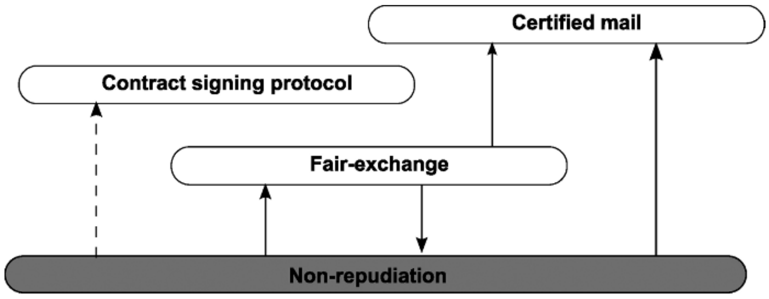

Fig. 3.2 Non-repudiation Service

evidence-generating exchange) are those in which the participants play different roles in the same application. Imagine an electronic shopping application which involves a customer, a merchant, a credit card company and a delivery company. Different existing two-party non-repudiation protocols could be selected for providing a non-repudiation service to this or the previous proposed scenario. Nevertheless, several questions, mainly regarding to efficiency, arise. The solution achieved is not optimal for a real application since the notion of a unique transaction is lost.

The ground for focusing on the multi-party problem as a different one comes from the fact that the problem approaches change. The first step towards multi-party environments consists of revising the requirements defined in Section 2.5.

Definition 3.2. *Fairness*. A multi-party protocol is said to be fair if at the end of the protocol **all honest** parties receive what they expect or none of them receive any valuable information.

It is very important to detail what is *valuable information*. Most existing definitions do not specify the valuable elements and it is not always straightforward to identify these elements. In a general fair MPNR protocol, the recipients either already know the message to be received and thus only evidence are considered valuable elements for the exchange (i.e. they can refuse to run the protocol even if they get the message and we say the protocol fulfills *light fairness*), or they must send a NRR in order to get the message. Thus, in the latter case, instantiating the definition above, we say that a MPNR protocol is *fair* if, at the end of the protocol, either the originator receives EOR and the recipient(s) receive(s) the message and the corresponding EOO or none of them obtains any of these items.

Note that all the participants must be in the same state at the end of the protocol. That is, corrupted parties should receive their outputs if and only if the honest parties also receive them [92].

Definition 3.3. *Confidentiality*. A multi-party protocol is said to be confidential if only the aimed honest recipients can disclose the message.

This means that the TTP cannot disclose the message either. Nevertheless, though it is a must in multi-party protocols that others, especially those participating ones that do not finalize the protocol, do not disclose the message, it is optional whether the TTP is able to do it.

Definition 3.4. *Efficiency*. A multi-party protocol is said to be efficient if, assuming participating entities of the protocol are honest, the TTP does not intervene.

Different instances of a non-repudiation protocol in which different entities participate could make the TTP become a bottleneck. An optimistic protocol in which an off-line TTP participates only in the case of an exception seems to be the solution, as we have already explained for two-party protocols in the previous chapter. Nevertheless, many entities could be participating in each instance of the protocol. In this multi-party environment, even in case of exception, only the TTP needs to act in a light way such that it does not become a bottleneck.

Definition 3.5. *Timeliness*. A multi-party protocol is said to respect timeliness if all honest entities are able to terminate the protocol in a finite amount of time without losing fairness.

The honest transacting parties may not know the final status of a transaction on time, and would like to unilaterally bring a transaction to completion at any time without losing fairness. It should be noted that from the definition of timeliness two versions of this property appear.

- *Asynchronous timeliness*: A multi-party protocol is said to respect asynchronous timeliness if all honest entities are able to terminate the protocol at any time without losing fairness. In this case, there are no deadlines for participants in the protocol, but a serious practical implication makes it hard to achieve: for this property to be fulfilled, an infinite state (or at least until evidence expiry date if it is the case) has to be maintained by the TTP until parties fetch evidence (*statefull TTP*). Otherwise, if responsible of distributing evidence, it needs to retry till recipients of evidence acknowledge reception. An alternative solution can be to extend the channel reliability between TTP and users (in this direction).
- *Synchronous timeliness*: A multi-party protocol is said to respect synchronous timeliness if all honest entities are able to terminate the protocol in a finite *and known* amount of time without losing fairness. In this case, deadlines are used and the TTP clock is assumed as the reference time (i.e., users' clocks need to be synchronized with TTP's clock). Though more difficult for users, the TTP does not need to maintain evidences for long-time periods (*stateless TTP*).

Note that the type of timeliness provided by the protocol does not depend on the network synchronicity (see Definition 3.8). The requirement of asynchronous timeliness is specially difficult to achieve on ring architecture as we will see in a multi-party implementation of contract signing in Section 5.2.

Definition 3.6. *Policy*. A MPNR protocol needs a complete non-repudiation policy that supports the participation of several entities in possible dispute resolution processes.

As with two entities, this has to define perfectly all the parameters needed by the service. There is no important distinction in multi-party environment except for the fact that multiple disputes and agreements can arise among the participants. The arbiter to judge the dispute needs to be explicitly mentioned in the policy definition.

3.1.2 State of the Art Analysis

The first work that appears in general MPNR problems can be found in [87, 96]. Kremer et al. propose generalizations of two protocols: one based on the on-line approach [132] and the other based on an optimistic approach (also known as with off-line TTP) [88]. As stated before, other related works existed and are still under research in the field of multi-party fair exchange [26, 60, 32, 65, 86, 66], contract signing [24, 35, 61, 36, 57, 34, 43, 59], and certified email [29, 58]. Due to its practicality, works on the MPNR and CEM use a one-to-many topology, whereas multi-party fair exchange and contract signing appears either on mesh or on ring configurations. Nevertheless, [60] proposes that a mesh fair exchange can be reduced by the composition of cyclic exchanges using rings until all items expected are exchanged by parties.

3.1.2.1 Optimistic Multi-Party Fair Exchange

The general optimistic multi-party fair exchange protocol in [26] is the first work which achieves a scenario with multiple entities in which non-repudiation evidence needs to be generated together with the fair exchange of the items. Asokan et al. proposed the use of a matrix of descriptions as information of the items to be exchanged for all entities. Depending on the type of items (confidential data, public data and payment) and number of entities (one-to-many, etc.), the generic service described can be instantiated as a particular application; e.g., one-to-many topology in which confidential data is exchanged for public data corresponds to a reliable certified broadcast application.

In this way, the basic idea of the generic multi-party fair exchange protocol is that each party signs the expected global description or local view (i.e., the whole description matrix) of the exchange and commits to the items he will send to every other party. If all parties signed the same description matrix, the parties send the promised items. If someone does not receive what was expected, two-party recovery procedures are started with the TTP.

In this case, if some non-repudiation of receipt tokens of any pairwise exchange, or some expected items or keys to open the commitments, are not received correctly within a certain period [3], the TTP is invoked with necessary information to raise an

[3] Each party may decide independently when to time out.

exception. Each party can initiate recovery at most once. For each recovery request, the TTP starts a recovery phase by requesting the pairs of parties involved to repeat the exchange while being observed by the TTP. In this case, each honest party sends its item first and then waits for the items and tokens it expects to receive.

The exceptions are resolved in the recovery protocol by the TTP. One exception is that all honest partners send their consistent local views (matrix descriptions) but some messages are not delivered. Some partners will receive a complete set of signed messages and some will not. Only the partners who received a complete set will start the exchange. Therefore, the party which received the complete consistent view and has sent its items will complain to the TTP that the party not having received a consistent view did not respond to the items sent. During recovery, the party having an incomplete view will receive a consistent view from the TTP and will now participate in the exchange.

Another exception is the presence of inconsistent local views: One dishonest party sends different messages to some partners, i.e., some partners receive a consistent view and some do not. Since every honest party signs the matrix description only once, the inconsistencies relate to dishonest parties signing multiple versions of it. The recovery from this exception is as follows: when the TTP is invoked by a honest party presenting the only consistent set of signed messages, it tries to enforce these exchanges, and thus forward the consistent view to all parties. If the party invoking the TTP is dishonest; i.e., it has sent another message to the other parties different from the one sent to the TTP, the provided view may be consistent but different from the view of the honest party forced to participate. In this case, the honest party may abort the recovery by proving the dishonesty of the party invoking the TTP: it sends an inconsistency proof *inconsistency()* which contains the message which is signed by it but different to the message which has been sent by the TTP. The recovery is aborted only if the dishonesty of the party initiating the recovery is proven.

The only critical timeout of the generic protocol is the active-time limit t specifying the absolute time at TTP when the protocol ends. This time-limit ensures a consistent view of all honest participants: the state at time t is final time; all parties are guaranteed that the status is not changed after t. Variable t is expressed in terms of the local clock at TTP since the TTP is the only entity that makes decisions based on the active-time limit in a way that has an impact on the correctness of the protocol: if the TTP does not accept recovery requests after a certain time t, i.e., if the TTP decides that a recovery request came too late, no fairness may be provided to the party requesting recovery. In practice, however, all parties have to know the time on TTP's clock in order to agree on the active-time limit as well as to compute local timeouts within rounds. Hence, a model in which clocks of all parties are synchronized (i.e., all parties have real-time clocks, and the differences between all local clocks of honest parties are limited by a constant) is required.

Besides, the authors introduced the notion of *revocable* (e.g., the payment could be cancelled by the bank even when the order was already transmitted) and *generatable* items (e.g., the bank could order on behalf of one of its clients, a payment to another client). In this protocol, the TTP can guarantee strong fairness if the items involved are either all revocable or all generatable, otherwise weak fairness is achieved.

The resolution process of this protocol, though complete, is not efficient. Having the TTP observing every exchange between two parties when a problem exists and if not sufficient, running recovery resolution processes is not a practical solution for an application aimed at the Internet.

In [60] Franklin and Tsudik also develop a classification of types of multi-party fair exchange schemes and present new protocols which assume the presence of a *semitrusted neutral party* in ring topologies. A malicious semitrusted neutral party must be unable to cheat as long as the other parties remain honest. For achieving fairness the protocol makes use of a special mathematical function f which allows to make checks without revealing content. This is similar to the *verifiable encryption scheme* appeared in [32]. The TTP acts in an in-line manner (i.e., it is involved in every exchange) and does not learn about the items exchanged as well as the topology (it knows it is a ring but does not know the sequence).

In [32], Bao et al. propose a multi-party fair exchange protocol with a ring topology making use of an off-line TTP. In this protocol, no honest party is left in an unfair situation no matter how maliciously the dishonest parties behave. Nevertheless, some parties could be excluded from the ring in the exchange (not affecting however fairness). The main design, which uses a verifiable encryption scheme as a tool for commitment, is linear with respect to the number of rounds (i.e., $2n$ rounds where n is the number of parties) and the TTP needs to contact the originator of the protocol. This makes efficiency an important issue to be improved. In fact, two main improvements are proposed for enhancing efficiency; reducing the number of sequential rounds and avoiding the contact of the TTP with the initiator of the protocol. In [65], Markowitch et. al improve the protocol such that participants only need to trust the TTP (and not the initiator). Moreover, under certain circumstances, if there are participants excluded from the exchange, they can prove that a problem occurred to an external adjudicator.

Markowitch tackles the exclusion-freeness property in multi-party fair exchange protocols one year later in [66]. This time, they demonstrate the exclusion problem in Franklin and Tsudik's protocol. They provide a formal definition of the exclusion-freeness property and propose a multi-party fair exchange protocol with on-line TTP which respects the strongest definition of the property defined.

In [86], the authors propose a protocol for multi-party fair exchange with a ring topology. Although it is not optimal, it presents an extensive complexity study with respect to [26]. The protocol consists of 3 phases in which a matrix of elements are exchanged from the initiator to the next party in the ring where the sequence is predefined. Upon receiving the message, each party checks it and pulls out the information components destined to it from the matrix. It may load the next sequenced components into the matrix with regard to the accepted components at this point. The TTP participates in the exchange in abnormal cases. The TTP tries to recover from the abnormality by observing and controlling all the transferred messages. In those cases each message from a party to the next party is not transmitted directly but relayed by the TTP.

With this approach, the protocol achieves $3n$ messages in the best case and $7n$ in the worst. In comparison, the protocol proposed by Asokan with the mesh topology needs $4n(n-1)$ messages in the best case and $8n^2-n-10$ in the worst. Even so, time complexity for the ring topology increases because it needs $O(n^2)$ rounds (for distributing all the $3n$ messages) while the mesh topology only needs $O(n)$ rounds for distributing all messages. More precise protocols for multi-party fair exchange with ring topology exist as depicted previously. Nevertheless, property definitions are not very accurate.

3.1.2.2 Multi-Party Contract Signing

A particular application of multi-party fair-exchange protocols is MPCS; although MPCS protocols can also be used to design non-repudiable multi-party fair exchange protocols when the signed contract is used as evidence (*verifiable commitment*). In [24], Asokan et al. propose the first optimistic MPCS for synchronous networks. Assuming that the TTP is not corrupted, in the all-honest case only two rounds of communication are needed. In the first round, each party who wishes to sign the contract broadcasts a signed "promise to sign". In the second round, each party who receives all n promises from the previous round signs the contract and broadcasts its real signature. Obviously, this works if all parties wish to sign. If at least one party does not wish to sign, it will not send the signed promise, and thus no party will sign in round 2.

If some party cheats, two more rounds are added to the protocol, such that everybody who has n signed promises from round 1 can get them converted into a valid contract by the TTP. If the TTP issues an affidavit, it broadcasts it to all parties in round 4. Thus, each party who did not receive all n promises in round 1 waits until round 4. If it receives an affidavit from the TTP, the decision is `signed`, otherwise `failed`. On asynchronous networks, the last "otherwise" would not be effective, since a party could not decide whether an affidavit was not sent, or just not delivered yet. In fact, due to this synchronicity, termination of the protocol is ensured by the network itself in a fixed number of rounds (4 in the all-honest case and 6 in the worst case). Using this protocol as a building block, Asokan et al. designed a generic multi-party fair exchange and a multi-party certified electronic mail protocol (see next section for further reference).

In [35], the first optimistic MPCS for asynchronous networks is presented. Again, assuming that the TTP is not corrupted, $n+4$ rounds of communication and $O(n^2)$ messages are needed in the worst case, taking into account that the number of dishonest parties is not known a priori. It consists of $n+1$ rounds in the all honest case. In round 1 each party that starts with `sign` signs a "promise" to sign the contract and broadcasts this promise. In each subsequent round each party collects all signatures from the previous round, countersigns this set of n signatures, and broadcasts it. The result of the (n+1)-th round becomes the real contract. Three additional rounds are needed if the TTP is contacted.

A party who becomes tired of waiting for some signatures in some round (i.e., there is a local timeout) can call the third party. The TTP analyzes the situation and decides either failed or signed: if the first request received comes from a party in the first round, then the TTP must decide failed (the TTP cannot know whether some parties might have started the protocol with `reject`). If the TTP receives a request from a party in the last round, then the TTP must decide signed because other parties might already have the signed contract. Besides, if the TTP receives multiple requests somewhere in the middle between the first and the last round, the TTP might have to change the decision from failed to signed.

The problem is that the TTP can do this only if all parties that received failed before are probably dishonest. Therefore, the TTP needs $n+1$ rounds in order to check this behavior. If dishonest parties call the TTP one after the other starting with round 1 then first request must be answered with failed, as already explained. After that, for $i > 2$ [4], the request by party i shows that party $i-2$ to first party are all dishonest: party i calling in round i means that it has seen all messages of round $i-1$, because otherwise, party i would have called the TTP already then. Those include messages from party $i-2$ to the first one, which could not exist if those would have stopped in round $i-2$ to the first, as supposed. Thus, whenever the TTP receives a request for a certain round $i, i \geq 2$, such that it has not answered a request for round $i-1$ yet, then it can safely switch from failed to signed (since parties who previously obtained failed token are clearly dishonest).

As a consequence of the number of rounds, asynchronous timeliness is achieved; i.e., no deadlines are used and each party can finish the protocol when desired. Using this protocol as a building block, Waidner et al. designed a scheme which provides fairness to any general secure multi-party function evaluation protocol. These protocols allow several users to calculate the output of a function f without revealing to the rest of participants their inputs. With the added fairness, dishonest parties have no advantage over the honest parties in learning the function outcome (basically composing this outcome in such a manner that a signed contract protocol is used for exchanging one of the components needed for learning the other component which is the real function output).

In [61], the authors construct a general multi-party optimistic asynchronous contract-singing protocol which requires $O(n^3)$ messages in $O(n^2)$ rounds. The protocol is also *abuse-free*, meaning that at no point can a participant prove to others that he is capable of choosing whether to validate or invalidate the contract. This is the first abuse-free optimistic contract-signing protocol that was developed for $n > 2$ parties. They also showed a linear lower bound of n rounds of any n-party optimistic contract-signing protocol.

For this purpose, a crypto tool denominated PCS is used. Upon receiving a PCS, a party convinces himself of its validity, but cannot convince anybody else, and he also knows that the TTP appointed by the signatory can convert it into a regular

[4] Here i establishes the order of rounds and a party i is the party requesting the TTP in round i.

(self-authenticating) signature. The full formal definition described in [61] presents *invisibility*; i.e., no one can determine if a conversion was performed by the original signer or the TTP. This, of course, allows the TTP to be transparent.

The protocol is designed as a recursive algorithm. In a recursive manner, parties engaged in the multi-party contract signing protocol exchange different level promises. For i participants P_1 through P_i to sign a contract, P_i indicates its willingness to sign the contract to P_1 through P_{i-1}, then participants P_1 through P_{i-1} come to an agreement about the contract (with promises, not signatures), letting P_i know about this (again with promises). Then, all participants exchange promises to sign the contract. Only when all recursive levels are finished do the participants exchange signatures on a contract. This process takes $O(n(n-1))$ rounds. Party P_i can abort the protocol when waiting for recursive level $i-1$ to finish and resolve in the rest of rounds. The TTP attends these requests and can change from `cancel` to `finished` state (and never viceversa) following a similar criteria as the one explained in [35].

Nevertheless, the more important contribution of this work is not the protocol itself, which is rather complicated and not optimal, but the theorem presented at the end of the paper:

> Any complete and optimistic asynchronous contract-signing protocol with n participants requires at least n rounds in an optimistic run.

Describing the theorem, Garay *et al.* stated that "…for each party P_i out of n, when it sends a message that can be used (together with other information) by other entities to obtain a valid contract, as the protocol is fair, it must have received in a previous round, a message from the rest of participants in order to be able to get a valid contract too, no matter how others behave (probably with TTP's help)". By an inductive argument, they showed the number of rounds is at least n.

Another argument exists, which perhaps, better explains the theorem stated above from a more practical point of view. Bearing in mind the technical requirements for fulfillment of fairness in an asynchronous contract signing protocol, the argument's base is in the number of rounds a TTP needs to determine whether a party is misbehaving when requesting resolution (i.e., it requests cancel to the TTP but continues the main protocol):

> The TTP cannot determine whether a party is misbehaving until $round = round_{current} + 1$, since the TTP needs to wait to the next round to see whether this entity cheated and continued the protocol. That means, if $n-1$ dishonest parties exist in the worst case, and each of them requests TTP's cancel sub-protocol in a different round, n rounds are the minimum required to satisfy fairness in an asynchronous optimistic contract signing protocol.

This is the argument used by other solutions to reduce the number of rounds and steps when the number of dishonest parties t, is limited and known a priori.

This is the case of [34]. The protocols achieve an improvement in comparison with the state of the art. The number of rounds is reduced from $O(t)$ to $O(1)$ for all $n \geq 2t+1$, and for $n < 2t+1$, it grows slowly compared with numbers of rounds in $O(t)$: If $t \approx \frac{k}{k+1}n$ then number of rounds $\approx 2k$. Nevertheless, in the worst case ($t = n-1$), the number of rounds is (obviously) still $t+2 = n+1$. Theoretically, the main unrealistic assumption of this work is assuming a known a priori set of dishonest parties. Anyhow, from a practical point of view, those protocols with a determined tolerance could work in real applications that assume the risk on the limited number of dishonest parties (specially in relative trust environments). Furthermore, these protocols could be used in combination with others, such that they are used when at least 1/2 of the participating entities have previously demonstrated their good behavior, and thus, taking advantage of its better efficiency. As these protocols use a threshold, we will refer them as *threshold protocols.*

In [57], Ferrer et al. propose an optimistic contract signing protocol which nevertheless is flawed (see [107] for further details). It is so because it tries to overcome the previous theorem to reduce the number of rounds to a constant number.

There also exists MPCS protocols based on a probabilistic approach. In [37], Ben-Or *et al.* presented an optimistic contract signing protocol which is said to be (ν, ε)-fair if for any contract C, when signer A follows the protocol properly, if the probability that signer B is privileged to validate the contract with TTP's help is greater than ν, the conditional probability that "A is not privileged", given that "B is privileged", is at most ε.

3.1.2.3 Composition of Secure Multi-Party Protocols

When research in multi-party protocols is conducted, there is another important issue to be taken into account: Composition of Secure Multi-Party Protocols. In [92], it is demonstrated that obtaining security under (even rather weak notions of) composition can be strictly harder than the stand-alone execution of the protocols. That is, interactions between different executions of the same multi-party protocol more often reduce security. MPNR protocols themselves need to be studied to see how different executions interact with each other and how it affects security. For instance, MPNR protocols might make use of multi-party primitives as broadcasts, and it has been demonstrated that it is impossible to achieve broadcast if a third or more of the parties are corrupted. And this is under parallel self-composition, which is weaker than the concurrent self-composition we foresee for MPNR protocols.

A protocol is said to be *self compose* if it remains secure when it alone is executed many times in a network. It is *parallel self-composed* if all executions begin at the same time and proceed at the same rate (i.e., in a synchronous fashion). It is *concurrent self-composed* if several executions of the same protocol start and finish in an arbitrary way or determined by the adversary (i.e., in an asynchronous fashion).

Nevertheless, composition of secure multi-party protocols is a large area of research out of the scope of this book. Furthermore, we can base the design of the protocols presented in this book in the fact that Universal Composition (UC) protocols can be achieved if a reference string is used in the composition (i.e. a setup phase). Thus, we assume this setup phase for allowing the composition of these multi-party protocols, even though we do not raise this issue any more. As an example, in this setup phase, transaction identifiers (or labels) have to be perfectly defined in order to distinguish among different executions of the same protocol and participants must be aware of not being engaged in an exchange in which it has been already participating. Details for protocols for UC realizing any multi-party functionality in the *common reference string model* can be found in Chapter 4 in [92].

3.2 1-N MPNR Problem and State of the Art

Several applications like email and multicast of content in general, present a 1-N topology. In some of these applications, the originator sends a message which should not be revealed unless the recipients confirmed its reception.

Let us define what is our view of a 1-N MPNR scenario.

Definition 3.7. A 1-N MPNR application is an instantiation of the general MPNR problem; one entity O is willing to send a (several) message(s) to a set of recipients R such that a subset $S \subseteq R$ obtain the message and an evidence of origin if and only if they sent an evidence of receipt. This topology is found in different transactions (with multicast) done in the Internet.

Kremer et al. proposed generalization of two protocols: one based on the on-line approach and another based on an optimistic approach (also known as with off-line TTP). For both of them, the originator sends an initial commitment (message encrypted with key k) and once a group of recipients reply to this commitment it sends the message to which it was committed to (i.e. it sends the key k which allows the recipients to decipher the expected message) and waits for evidence of receipt. As the same message is distributed, only one key needs to be distributed to those recipients which replied, and a group encryption scheme based on the Chinese Reminder Theorem is used for this purpose [45].

The on-line based approach uses the TTP in a light-weight manner to distribute the key for each execution of the protocol. In this way, the originator submits the key together with an evidence of submission and the TTP publishes this key along with its final signature needed by all entities to complete their evidence. As explained before, only those who behaved correctly can decipher the key and, thus, the message.

In the optimistic approach, and if no exception arises, the TTP does not take part. Otherwise, entities can recover the protocol if needed after a deadline fixed by the originator with respect to the TTP clock. The originator will contact the TTP when not receiving final evidence from recipients to which it sent the key. Recipients will contact the TTP when not receiving the key while having sent evidence of receipt of the commitment. Once the TTP is contacted for resolving the protocol, and assuming it happens after the fixed deadline, the TTP needs to access an originator database in which recipients who replied are marked. The originator needs to be notified of TTP's successful access and stop immediately the main protocol. Of course the authors assume that the channel between originator and TTP is bidirectionally resilient. If TTP receives all necessary information, it sends evidence of confirmation to all entities, which substitute evidence not received (maybe due to a channel failure) by participating entities. They showed that the improvement with respect to *n* parallel executions of a two-party non-repudiation protocol is in terms of number of messages as well as number of needed digital signatures.

For multi-party certified email applications, different protocols can also be found. Asokan et al. instantiated (and optimized some steps) their generic multi-party fair exchange explained in previous section to design a reliable broadcast application. Besides, Asokan et. al presented an optimistic multi-party certified email protocol as an application of a MPCS protocol presented in [24]. That is, the MPCS protocol is used as a building block. The block can be asynchronous or synchronous depending on the type of network the certified email application is running on. It follows the following schema.

1. The sender prepares the message M and tid (a transaction id) to be sent to all the recipients encrypted with the TTP's public key. This $cipher = E_{TTP}(M,tid)$ turns out to be a contract to be signed by all parties.
2. All participants sign the contract using the MPCS building block and tid serves as the contract signing transaction identifier. Only if the final decision of the signing process is *signed* the protocol continues. The signed contract will be the receipt for the sender.
3. The sender sends to all the recipients the cleartext mail M. If any of the recipients does not receive this message, it contacts the TTP with the signed contract $(tid, cipher)$. The TTP checks the consistency of tid after decrypting $cipher$. If the contract is valid and decryption succeeds then the TTP signs the cleartext M for the recipient, otherwise it signs a null message.

In the original paper, a proof can be found based on two properties: no information is leaked on M unless the sender has a receipt. And the sender cannot get a receipt without revealing M. In this work, Asokan et al. also define other possible types of multi-party certified email applications with many-to-one and mesh topologies. All of them can be solved following the same schema described above.

The only apparent difference with respect to the general MPNR protocols described in the previous section is that, as the authors indicated, honest recipients should receive the message even if some dishonest parties do not follow the protocol correctly and thus the MPCS block outputs *failed*. The main privacy problem of this solution is that since it uses a MPCS as a building block, each recipient can identify the rest of addressees in the protocol (unless only the sender can verify signatures from recipients, which is not possible by definition of MPCS).

With this in mind, Ferrer et al. presented a realistic protocol for multi-party certified electronic mail in [58] which can be seen as an instantiation of a MPNR service. If we compare this protocol with Kremer's optimistic proposition already discussed, the former is more efficient (only 3 steps in the main exchange which will be the only one executed in the majority of executions) and solves some of the problems found in the latter one as identified in next section.

In this protocol the sender commits itself to the message, and sends the necessary information (encrypted with the TTP's public key) to open this commitment together with evidence of origin. Then, honest recipients send evidence of receipt and finally the sender opens the commitment and sends the required evidence for those recipients who replied. With respect to the MPNR properties, the protocol is efficient and can be timely terminated thanks to the existence of two sub-protocols (*cancel* and *finish*). A sufficient dispute resolution process is also defined, and the TTP's behavior is verifiable (unless colluding with other entity, the TTP cannot misbehave without being discovered).

If the sender does not get some receipts from some participants, it initiates a cancel sub-protocol in which those recipients are included in a cancelled set if they did not finish the protocol yet. On the other hand, if some recipients do not receive step 3 of the main exchange protocol, they contact the TTP in order to finish it and, those who have not been cancelled yet by the sender, receive the key to open the commitment and therefore, obtain the cleartext message. Otherwise, they receive a cancel token. In this sub-protocol, recipients need to send their receipts to the TTP, which will store and forward them to the sender if required (see Figure 3.3). A weakness can be found in the fact that in the dispute resolution process, the arbiter needs to contact both disputing parties in order to be able to maintain fairness; i.e., an entity and its evidence could not be sufficient to prove its claim.

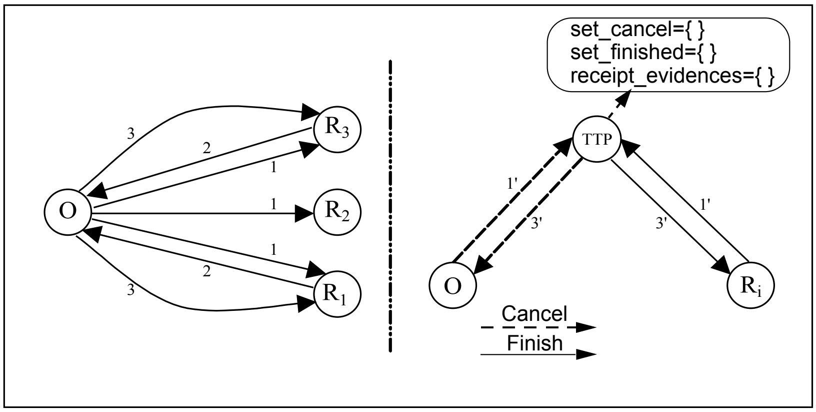

Fig. 3.3 Ferrer et al. Solution for Multi-party Certified Email

In [91], an agent-based commerce system, ABECOS, is proposed to achieve non-repudiation over electronic transactions. In this system, three principal entities are identified: a buyer, a seller, and a directory agent. The directory agent keeps information about other entities and acts as an intermediary broker that helps an entity to find other entities or agents that possess certain required capabilities. In this scenario, the directory agent is not involved in the non-repudiation activity (see Figure 3.4). This motivated us to design a non-repudiation protocol with intermediary as depicted in Section 4.2.

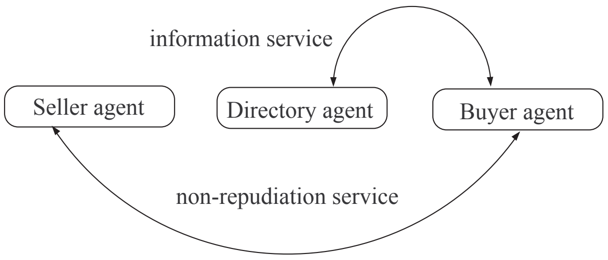

Fig. 3.4 E-commerce Scenario

Once we have precisely defined the concept of MPNR protocols and made a brief introduction on the related work of the 1-N topology, we extract the problems and face the improvements needed to maintain the properties defined.

3.3 General Contribution to Multi-Party Problem

After presenting and analyzing the state of the art, this section presents some of the general and specific (or application-oriented) problems we found and aimed at solving. The general problems which motivate us are:

- *Efficiency*. Some of the protocols can be improved, either at the number of rounds, messages or crypto operations. This is a very important issue, since non-repudiation will be an added security service for a multi-party protocol which per se needs several communication messages and thus, time. Entities will judge efficiency as a discriminating parameter. For such a reason we provide and implement a simple simulation model which gives us the possibility of obtaining time values once the parameters depending on time in the protocol are instantiated.
  Also, part of this aim is to remove the TTP's responsibility of distributing final evidence when possible. This also allows us to relax channel requirements between users and TTP. Though it is not desirable for asynchronous timeliness purposes, it is more practical if a deadline time is established and users are held responsible for accessing required evidence. The use of deadline is something practical and widely used in nowadays digital transactions.
- *Collusion*. With the participation of multiple entities, the probability of collusion among them is higher. Recipients could collude for getting the message without sending a receipt in a certified email application or some signatories may also misbehave for getting a signature over a contract while others do not. Collusion among participants may have an impact on fairness or simply exclude some users from the protocol. This type of misbehaving pattern is not possible in two-

party environments. All this needs to be solved when specifying new designs. However, sometimes, it might be needed to assume that some collusions are not possible in order to improve efficiency and viability. Occasionally, agreement among participants may be of help for threshold solutions.

- *Selective receipt*. This problem appears in the 1-N topology when not all receipts are a prerequisite for successfully completing a protocol run with the rest of recipients. In this case, confidentiality becomes an integral property of the protocol and needs to be guaranteed.
- *New structures and group operations*. With the presence of a group of recipients (or originators), messages which are distributed to all of them require to be structured, and confidential when information intended for each participant is multicasted. For this, new group operations might be studied and analyzed. At the same time, evidence must be arranged in the TTP database directory when it belongs to the same transaction but to different participants. For instance, the label may be used as a search index and its construction also depends on the participation of multiple entities.
- *Comparisons*. Not clear comparisons and efficiency studies are made. But this becomes more important as more participants play a part in the protocol. We have to define a methodology for measurement as well as give clear statement about the results achieved. Model simulations in order to get approximate time results help us to realize the costs of introducing MPNR protocols in nowadays multi-party transactions.
- *Protocol design*. Two main things affect non-repudiation protocol designs. Firstly, research descriptions are sometimes too simplified [5], and secondly new attacks appear everyday and solutions need to be incorporated. The label l present in most of the non-repudiation protocol designs for identifying each instance of a protocol is a good example. It has been an, often, abandoned element and its correct definition is essential for avoiding replay attacks and allowing the users to properly identify its protocol instance. More examples can be found in [135] where the protocol presented in [58] is analyzed and different flaws which affect the fairness property can be found.
- *Timeliness*. It is necessary the design of protocols which do not ensure this property just by assumptions on the channels. In all protocol designs a trade-off between timeliness as a desired property and the hardness of achieving it has to be studied. If necessary, threshold schemes for achieving a *conditional* timeliness property will be designed.
- *TTP's verifiability vs. TTP's transparency*. These are desirable properties for all protocols but unfortunately it is not possible to achieve both of them. The reason is that a transparent TTP will not leave proof of its participation in the protocol and thus, it will not be possible to verify it intervened correctly. It is up to us to

[5] Different causes force authors to describe their research ideas and protocols without realizing that a simple element could open the door for attacks or misunderstandings. Some of these causes are size restrictions in their communications, tight deadlines, lack of a profound analysis of their designs, etc.

select the more important property in each case and whether the selected property is worth the computational load, as it will depend on the final application the multi-party non-repudiation service is applied to.

The specific problems which motivate us are:

- *Different messages*. As mentioned in Section 3.2, Markowitch and Kremer proposed a generalization of the two-party non-repudiation protocol with a 1-N topology. We further generalize their multi-party non-repudiation protocol such that the originator is able to send different (customized) messages to multiple recipients, and more importantly, in an optimistic approach. Section 4.1 will introduce applications which need this type of interactions, and each of this interaction should be performed in the same protocol run, thus getting all the needed evidence and avoiding contacting the TTP several times.
- *Online accesses*. In the existent optimistic MPNR protocol described, the TTP needs to access back to a public directory maintained by the originator in order to be able to resolve the protocol. Furthermore, the originator needs to be notified of TTP's successful access and stop the protocol immediately. This is a strong requirement, since a DoS attack could be performed such that either the originator cannot appropriately update the directory or the TTP cannot access it when needed. We propose that the TTP which counts with more resources should maintain this public directory. This is so because a one-way resilient channel is easier to implement than a bidirectional resilient channel. That is, assuming that the TTP will eventually access the public directory maintained by the originator could be a strong assumption.
- *Intermediary*. We aim at widening the intermediary agent's role for non-repudiation purposes, thus liberating the originator of the non-repudiation protocol from part of the computation load to obtain evidence. Since non-repudiation is an added service and originators will be involved in any case, outsourcing some tasks of this service, when this is possible and secure, is a good approach for Internet applications. Furthermore, the security implications of this approach need to be studied as well as a possible extension to multiple participants.
- *Certified email*. It is a value-added service of ordinary email, in which a sender wants to obtain a receipt from a recipient. We have studied several multi-party approaches and tried to improve existing solutions with timeliness and multicasting.
- *MPCS*. Multi-party Contract Signing protocols need the participation of all entities and due to its nature (the contract can have a value of millions of euros), it is very important to ensure fairness, and thus, analyze the different existing solutions. Nevertheless, efficiency must be also optimized and, when possible, threshold solutions could exist for contracts that are not of great value or certain kind of trust exists among participants.
- *Related applications*. It is necessary to translate all the properties defined and the protocols designed to real applications in order to study its feasibility and convenience.

3.3.1 Model Definition

In Section 2.4, the communication channel properties are explicitly defined. Along the design of different protocols, unless otherwise stated, we assume that channels among users are unreliable and that the channel with the TTP is resilient. When possible, only one-way resilient channel is assumed; i.e., the user will eventually contact the TTP but this does not imply that the latter will eventually contact the user. Although we did define operational channels as well, we will not make use of them as they make very strong assumptions on the network services and are rather unrealistic in heterogeneous networks. Two definitions for distinguishing the type of multi-party protocol are:

Definition 3.8. A *synchronous* protocol is used in synchronous networks in which there is a limited time for a message to reach its destination (otherwise it has been lost and the appropriate transport layer manages these events) even if an attack occurs. Thus, a party can determine that a message has not been sent by other party when this did not arrive within the limited time. Users' clocks are assumed to be synchronized.

Definition 3.9. An *asynchronous* protocol is used in asynchronous networks in which there is no limited time for a message to reach its destination. Loss and unsorted arrival of messages are possible and have to be managed by the protocol itself. Clocks are not assumed to be synchronized among users.

When not using end-to-end communications but multicast, we assume Kremer's definition in which a multicast is a sending from one entity to a set of entities. This kind of sending corresponds to a one-to-many communication and not to a one-to-any communication (i.e. broadcasting). With respect to the technical infrastructure, influenced by [86], we will assume that the user will be using the standard TCP/IP stack, being the multi-party protocols located at the application level over security layers as SSL. An OSI-like protocol stack (see Figure 3.5) represents the protocols' location.

In the literature, different protocols make use of different terms. Throughout these solutions, authors use the terms *round* and *step* without clearly defining them, which often brings on confusion with respect to the metric to be used for its efficiency evaluation. For this reason we explicitly define these terms for multi-party protocols as follows.

- *Round* is understood as the existing time slot in which messages are distributed in synchronous networks. In asynchronous networks, the entities need to wait a local time before going to the next round (this local time can also be triggered when previous round is completed).
- *Step* refers to the action of sending or receiving a message. It is the operation performed by a participating entity. Each round means one step (when all the messages from all entities are distributed or multicasted in the same time slot, usually in matrix topologies) or several (when messages from the same round are distributed from one entity to another, usually in ring topologies).

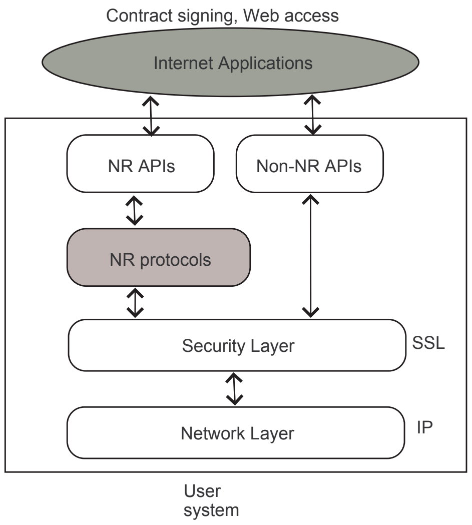

Fig. 3.5 Layered Architecture of Non-repudiation Protocols

It has to be noted that there is some confusion in the literature analyzed with respect to the term 'round'. Some authors explain that when next message to be sent depends on the previous one, then the protocol goes into a different round. But we claim two different cases can be found: (1) message to be sent depends on the previous one because the entity needs to compute/verify it before sending the next one, or (2) message to be sent depends on the previous one because there is a distribution order to be respected (as on ring topologies). We consider a round occurs in case (1).

3.3.2 Cryptographic Primitives

Let us transcribe contextual definitions for the cryptographic primitives and concepts which appear throughout this book in the design of different multi-party protocols. Note that better and more precise definitions could be found in general cryptography studies [101].

Definition 3.10. A *symmetric encryption* scheme is a scheme where the deciphering key can be derived from the ciphering key using a polynomial time (with respect to the key length) algorithm. This implies that both the ciphering and the deciphering keys have to remain secret [89].

Candidates for its use in implementations are DES [55] [6] (and its variants) and the new standard for symmetric encryption AES [56].

Definition 3.11. An *asymmetric encryption* scheme is a scheme where no polynomial time (with respect to the key length) algorithm able to derive the deciphering key from the ciphering key exists. The ciphering key can be public and the deciphering key (also known as private) has to remain secret [89]. When possible, a deterministic asymmetric encryption algorithm will be used. Otherwise, if a non-deterministic algorithm is used (e.g. ElGamal cryptosystem [50]), either verifier would need assistance, or the originator of the encryption should provide the verifier with the random seed used in encryption.

To prevent attacks on the resulting ciphertext, the encryption scheme should provide *non-malleability*, i.e. it should be impossible to modify this ciphertext to construct a different meaningful related ciphertext.

Definition 3.12. A label l is a bit string which uniquely identifies a protocol instance. It must contain all necessary information to avoid replay attacks.

Definition 3.13. A hash function $h()$ is a deterministic function which maps a bit string of an arbitrary length to a hashed value which is a bit string of fixed length. Properties which are assumed to hash functions used throughout this book are: (1) *collision resistance*; i.e., it is computationally infeasible to find two inputs x, y with $x \neq y$ such that $h(x) = h(y)$, (2) *preimage resistance*; i.e., given a hashed value h, it should be computationally infeasible to find an input string x such that $h = h(x)$ and (3) *practical efficiency*; i.e., given input string x, the computation of $h(x)$ can be done in time bounded by a small-degree polynomial in the size of x [94].

Definition 3.14. A *digital signature* scheme is the main tool for providing evidence in a non-repudiation protocol. $S_P(X)$ is the signature of entity P over the message X. The receiver of this signature maintains proof of origin and data integrity of the message. It serves as evidence that a determined transaction (in which message X is sent) occurred. As many digital signature schemes exist, unless otherwise stated, we assume the use of an asymmetric scheme (with public and private keys) such as RSA [94, p. 309].

It is important to seriously take into account the state of the art when selecting cryptographic primitives in order to implement the protocols and policy in effect in the non-repudiation service, since its strength and weaknesses are under continuous research. Most of the hash functions, for instance, used at the time of writing (MD5, SHA1,..) are subject to attacks, some of them found in recent research [127].

For the definition of interactions among parties, the following notation will be used.

- x, y : concatenation of messages x and y

[6] Withdrawn May 19, 2005.

- u_P : public key of a user P in an asymmetric (encryption or signature) scheme
- $S_P(X)$: digital signature of user P over message X
- $E_K(X)$: symmetric encryption of message X with key K
- $E_P(X)$ or $E_{u_P}(X)$: asymmetric encryption of message X with P's public key
- $h(X)$: one-way hash function over a message X
- f : a flag indicating the *purpose* of a message
- $O \rightarrow P$: entity O sends a message to entity P
- $O \leftrightarrow P$: entity O fetches a message from entity P
- $O \Rightarrow R$: entity O multicasts a message to a set of entities R

With respect to the **threat model**, we assume a strong attacker. The Dolev-Yao threat model has become a standard for crypto protocols, and it is implicitly applied to protocols presented in the state-of-the-art sections. Nevertheless, we consider that the threat model must be defined explicitly beforehand, such that analysts can know a priori what an attacker can do. It formalizes the following attacks:

- An attacker can intercept messages transmitted between hosts.
- An attacker can parse any message comprising separately encrypted fields, and the attacker can extract the separate fields.
- An attacker can decrypt a message or field if and only if that attacker has previously obtained the appropriate encryption key.
- An attacker can construct and send new messages using information intercepted from old messages.

The first three elements characterize *passive* attacks; the fourth element exemplifies an *active* attack. We concede more power to the attacker, since by definition, such an attacker in a non-repudiation protocol is usually a user participating in the protocol. That is, it is authenticated as a legitimate user and thus, it can do whatever other user can.

Independently of the threat model, other security problems will not be addressed in this book (at most, mentioned when by context is needed to emphasize the existence of a different security problem as a consequence of the protocol design), as there exists multiple works addressing other aspects of security in depth. One of these examples is denial-of-service attacks which will not be treated here, and which are possible as long as users need to access a directory in the network. This corresponds to the availability service as defined in [79] and mentioned in the first chapter.

Another security threat worth to mention is the user's code integrity when executing software which implements the non-repudiation layer. Of course, this problem is general to software and will not be treated in this book. We will assume that the honest user's software executes only the code in order to implement the protocol. Nevertheless, it is important to bear in mind that this kind of threats could turn out into computer crashes when the user is executing the protocol. The computer is, from an abstract point of view, the extreme of the communication channel. Therefore, considering that a computer might crash while executing a non-repudiation protocol is equivalent to consider channel failure; that is, protocols in this monograph will deal with users' computer crash-tolerance by means of the channel properties. Obviously, and as we will see later, it is easier to design protocols when assumptions on the channels (crash-tolerance) are relaxed.

Note that in every case, in order to support computer crashes or channel failures, logging messages received need to be supported for recovering from undesired situations. If this is implemented so in the protocols presented in this monograph, fault-tolerance will be in a very general way provided as well. If deadlines are fixed at startup, time for crash and failure recoveries needs to be allocated or otherwise fairness cannot be ensured. For a better understanding of which are the protocol requirements and changes when converted to crash-tolerant protocols, see [54]. As the authors explained, a countermeasure against non-repudiation user's code tampering is the use of smartcards.

On the other hand, the TTP is assumed to be reliable and secure against intrusions and Trojan horse attacks, thus not being vulnerable to crashes. When it is vulnerable to crashes and the protocol is designed such that TTP's verifiability is provided, the rest of participants are assumed to behave honestly.

3.4 Summary of MPNR Protocol Properties

Tables 3.1 and 3.2 summarize the different advantages, properties, disadvantages and functionalities of the multi-party non-repudiation protocols presented so far (as well as their topologies), as well as new solutions to be presented in the following chapters.

Protocols	App	Propert.	Advant.	Drawbacks	Top.
Asokan'96 [26]	MPFE	Optimistic, generic service, revocable and generatable items	Asynchronous timeliness, NR generated, strong and weak fairness	Not efficient	N-N
FraTsu'98 [60]	MPFE	in-line TTP	Semitrusted neutral party, Confidentiality	No exclusion-free	Ring
Bao'99 [32]	MPFE	Off-line TTP, Verifiable Encryption Scheme	Asynchronous timeliness	No exclusion-free, Not efficient	Ring
GDM'02 [66]	MPFE	On-line TTP	Exclusion-free		Ring
Asokan'98 [24]	MPCS	Optimistic, Synchronous	Four rounds	Synchronous timeliness	Ring
BW'98 [35]	MPCS	Optimistic, Asynchronous	Asynchronous Timeliness	n+4 rounds, $O(n^2)$ messages	Ring
Garay'99 [61]	MPCS	Optimistic, Asynchronous, PCS crypto tool	Timeliness, Transparent TTP, Abuse-free	$O(n^2)$ rounds, $O(n^3)$ messages	Ring
BW'01 [34]	MPCS	Optimistic, Asynchronous, Threshold	Optimal efficiency		Ring
KM'00 [87]	MPNR	Group Encryption Scheme	Light TTP, One key	Same message, On-line TTP	1-N
MK'00 [96]	MPNR	Group Encryption Scheme	Off-line TTP, One Key	Synchronous timeliness, Same message	1-N
Asokan'98 [24]	CEM	Based on MPCS protocol, A/Synchronous	Optimistic	No privacy	1-N
Ferrer'02 [58]	CEM		Optimistic, Efficient, Timeliness, Verifiable TTP	Arbitrator needs to contact disputing parties	1-N

Table 3.1 Multi-party Protocols' Properties Summary

Protocols	App	Propert.	Advant.	Drawbacks	Top.
MPNR-OZCL [106]	MPNR	different messages for different recipients	privacy, lightweight TTP	on-line TTP	1-N
MPNR-OZL [106]	MPNR	optimistic, different messages for different recipients	privacy, transparent TTP, asynchronous timeliness, efficient		1-N
CEM-ZOL [135]	CEM	optimistic, no split of message in delivery	asynchronous timeliness, very efficient, recipient collusion resistance	more TTP overhead when TTP involved	1-N
MPCS-ZOL [136]	MPCS	optimistic, threshold cancel for weak asynchronous timeliness	abuse free, very efficient, 3 rounds and $3(n-1)$ steps		in-and-out

Table 3.2 New Solutions' Properties Summary

Chapter 4
New Design Approaches for MPNR

Abstract We have stressed non-repudiation features as an ITU standardized service into e-commerce and Internet applications. Once we have realized that several applications, as for instance notification systems and contract signing scenarios, need the participation of several entities, we targeted our research towards multi-party non-repudiation protocols. After studying the existing solutions and isolating the main general and specific drawbacks in the previous chapter, we focus on proposing new approaches with one principle design: efficiency.

In this chapter we give a deep explanation on our work for general MPNR protocols. We extend an existing MPNR protocol to allow an originator to send customized messages to many recipients in a single transaction. This protocol uses a light-weight on-line TTP in every transaction. However, there are situations in which the final entities may be willing to run a non-repudiation protocol without the TTP's assistance unless an error or channel failure occurs. For these situations, we design an optimistic solution that uses only three steps in its main protocol. The performance of our protocols with enhanced functionalities is still promising in comparison with existing multi-party non-repudiation protocols.

However, in some applications, efficiency needs to be traded-off with functionality. With this design principle in mind, we tackle the area of non-repudiation services in which an intermediary or agent is involved. We demonstrate the advantages for end users in the utilization of an intermediary service on reducing the evidence storage requirements and gathering different recipients.

In the design of these new non-repudiation protocols, we introduce parameters whose values are not easy to specify. Some of them are directly related to the TTP operation. Thus, we explore an event-oriented simulation model for determining estimations on these parameters. At the same time, the implementation and tests performed serve as a proof-of-concept.

4.1 MPNR Protocol for Different Messages

As stated in the previous chapter, Kremer et al. presented the generalization on non-repudiation protocols to the multi-party case in which a message can be multicasted to several recipients. In that way no personal and confidential messages can be sent to these parties without loss of privacy. In this section we propose new solutions. We also discuss the group encryption cryptographic tool used in the protocols described. The work in this section has been partially published in [106].

4.1.1 On-line MPNR Protocol for Distribution of Several Messages

Sending (same or different) messages to several recipients could mean a single transaction in a specific application. Therefore, it would be better to store the same key and evidence in the TTP record for every protocol run. In those types of applications, the storage and computation requirements of the TTP are reduced and it will be easy to distinguish between different transactions, regardless of how many entities are involved. Notification systems is an example of application which can get advantage of these kinds of security protocols. Those systems notify different users with customized messages and need evidence of having notified them.

Some useful notation in the protocol description which appears throughout this chapter is as follows.

- O : an originator
- R : set of intended recipients
- R' : subset of R that replied to O with the evidence of receipt
- M_i : message being sent from O to a recipient $R_i \in R$
- n_i : random value generated by O for each recipient
- $v_i = E_{u_{R_i}}(n_i)$: encryption of n_i with R_i's public key
- k : key being selected by O
- $k_i = k\ xor\ n_i$: a key for R_i
- $c_i = E_{k_i}(M_i)$: encrypted message for R_i with key k_i
- $l_i = h(O, R_i, TTP, h(c_i), h(k))$: label [1] of message M_i
- L' : labels of all the messages sent to R'
- t : a timeout chosen by O, before which the TTP has to publish some information
- $E_{R'}(k)$: a group encryption scheme that encrypts k for the group R' (see Section 4.1.4.1 for further details)
- $EOO_i = S_O(feoo, R_i, TTP, l_i, t, v_i, u_{R_i}, c_i)$: evidence of origin for R_i
- $EOR_i = S_{R_i}(feor, O, TTP, l_i, t, v_i, u_{R_i}, c_i)$: evidence of receipt from R_i
- $Sub_k = S_O(fsub, R', L', t, E_{R'}(k))$: evidence of submission of the key to the TTP

[1] There might be a potential attack [68] when the label l is constructed as $h(m,k)$ in the early literature, so we make the label unique in each run and verifiable by any party. Note that the labels can be computed off-line.

- $Con_k = S_{TTP}(fcon, O, R', L', t, E_{R'}(k))$: evidence of confirmation of the key by the TTP

In this extension, the use of the same key for all users creates a new problem that did not appear in Kremer's on-line multi-party non-repudiation protocol. Because messages are different, when the same key k is used for encryption, and after the key k is published, any recipient will be able to read the messages addressed to the other recipients (by eavesdropping the messages that are transmitted between O and R). More importantly, R_i could get c_i in the initial steps of the protocol and quit. Then, by colluding with any other party and getting the unique key k, it could decrypt c_i without providing any evidence of receipt. These problems are solved in our extended multi-party non-repudiation protocol, introducing some extra cost for the extended functionality over [87].

4.1.1.1 Protocol

Here, we describe the protocol (see Figure 4.1, where a dotted line indicates a *fetch* operation).

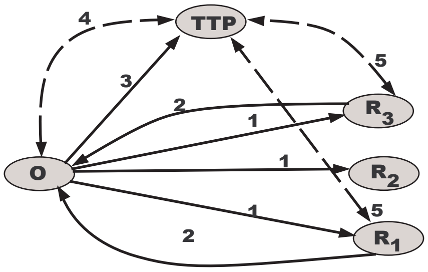

Fig. 4.1 Protocol for Distribution of Different Messages

1. $O \rightarrow R_i$: $feoo, R_i, TTP, l_i, h(k), t, u_{R_i}, v_i, c_i, EOO_i$ for each $R_i \in R$
2. $R_i \rightarrow O$: $feor, O, l_i, EOR_i$ where $R_i \in R$
3. $O \rightarrow TTP$: $fsub, R', L', t, E_{R'}(k), Sub_k$
4. $O \leftrightarrow TTP$: $fcon, O, R', L', E_{R'}(k), Con_k$
5. $R'_i \leftrightarrow TTP$: $fcon, O, R', L', E_{R'}(k), Con_k$ where $R'_i \in R'$

The protocol works in the following way.

Step 1: O sends to every R_i evidence of origin corresponding to the encrypted message c_i, together with v_i. In this way, O distributes $|R|$ messages in a batch operation and each R_i gets the encrypted message as well as n_i. O selects the intended public key u_{R_i} being used in the encryption of n_i. If R_i disagrees (i.e., its authentication digital certificate has expired or been revoked), it should stop the protocol at this step. There is no breach of fairness if the protocol stops at step 1 because c_i cannot be obtained without key k, and it cannot be derived from message 1.

Step 2: Some entities (or all of them) send evidence of receipt of c_i back to O after checking evidence and labels. Again, there is no breach of fairness if the protocol stops.

Step 3: O sends k and Sub_k to the TTP in exchange for Con_k. The key k is encrypted using a group encryption scheme where the group of users is R'. Hence, only those entities belonging to R' will be able to decrypt and extract the key. Alike, O will obtain evidence only for the recipients included in the set R' that submit to the TTP. Note that, in this way, O can exclude some recipients which replied, but fairness is maintained.

Before confirming the key, the TTP checks that

- $|R'| = |L'|$ holds
- current time $< t$

Step 4: O fetches Con_k from the TTP and saves it as evidence to prove that k is available to R'.

Step 5: Each R_i fetches $E_{R'}(k)$ and Con_k from the TTP. They will obtain k_i by computing k xor n_i. Also, they save Con_k as evidence to prove that k originated from O.

At the end of the protocol, if successful, the participants get the following evidence.

- NRO for honest recipient R_i: EOO_i, Con_k
- NRR for originator O: EOR_i, Con_k for all honest recipients $R_i \in R'$

Even though O can send a different deadline time t, this is not an interesting option for it. Since Con_k will not match the rest of evidence obtained, no party will obtain valid evidence, but the recipients could learn the message.

4.1.1.2 Dispute Resolution

Two kinds of disputes can arise: repudiation of origin and repudiation of receipt. Repudiation of origin arises when a recipient R_i claims having received a message M_i from an originator O who denies having sent it. Repudiation of receipt arises when the originator O claims having sent a message M_i to a recipient R_i who denies having received it.

Repudiation of Origin. If O denies sending M_i, R_i can present evidence EOO_i and Con_k plus $(TTP, t, u_{R_i}, v_i, c_i, n_i, k, E_{R'}(k), M_i, R', L')$ to the arbitrator. The arbitrator will check

- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i

- $c_i = E_{k_i}(M_i)$
- $l_i = h(O, R_i, TTP, h(c_i), h(k)) \wedge l_i \in L'$
- O's signature EOO_i
- TTP's signature Con_k

Repudiation of Receipt. If R_i denies receiving M_i, O can present evidence EOR_i and Con_k plus $(TTP, t, u_{R_i}, v_i, c_i, n_i, k, E_{R'}(k), M_i, R', L')$ to the arbitrator. The arbitrator will check

- $R_i \in R'$
- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- $l_i = h(O, R_i, TTP, h(c_i), h(k)) \wedge l_i \in L'$
- R_i's signature EOR_i
- TTP's signature Con_k

4.1.1.3 Efficiency

This protocol uses an on-line TTP which intervenes in every execution in a lightweight manner. We compare our approach with the one where an n-instance of a two-party protocol [132] is used in order to send messages to the intended parties. The efficiency of the three principal entities participating in the protocol is analyzed, using an operation comparison. For this comparison we will use the following basic operations.

- signature generation and verification
- generation of random numbers
- asymmetric encryption and decryption
- store and fetch operation

Depending on which algorithm is chosen for each of these operations, the bit complexity (as well as the bandwidth requirements) of each of the participating entities will change, although the relation going between them remains. Furthermore, the efficiency analysis also depends on how the group encryption primitive is used along the description of the protocol implemented. We will further analyze this issue in Section 4.1.4.1; for the time being, we will assume the complexity for a group encryption for n parties is equivalent to the cost of n asymmetric encryption operations.

We denote

$|R| = N$
$|R'| = N'$ (with $N' \leq N$)
$\approx$ roughly equal
$>$ or $<$ greater or smaller
$\gg$ or $\ll$ much greater or smaller

n-instanced two-party		Our approach
Evidence of origin EOO_i	$=$	EOO_i
N signatures		N signatures
Generation of k_i	$\approx$	**Generation of** n_i **plus** k
Evidence of submission Sub_{k_i}	$\gg$	Sub_k
N’ signatures		1 signature
Encrypted key $E_{u_{R_i}}(k_i)$	$\ll$	**Encrypted key** $E_{R'}(k)$ **plus** $E_{u_{R_i}}(n_i)$
N’ asymmetric encryptions		N’+N asymmetric encryptions
N fetches operations of Con_{k_i}	$\gg$	**One fetch operation of** Con_k

Table 4.1 O’s Computation Complexity

n-instanced two-party		Our approach
Evidence of receipt EOR_i	$=$	EOR_i
Fetch k_i **and** Con_{k_i}	$=$	**Fetch** k **and** Con_k
Obtain k_i	$<$	**Obtain** k **plus** n_i
Decrypts $E_{u_{R_i}}(k_i)$		Decrypt $E_{u_{R_i}}(k)$
		Decrypt $E_{u_{R_i}}(n_i)$

Table 4.2 $R_i's$ Computation Complexity

n-instanced two-party		Our approach
Store N’ keys	$\gg$	**Store only one key**
Generation of N’ evidences Con_{k_i}	$\gg$	**Generation of only one evidence** Con_k

Table 4.3 TTP’s Computation Complexity

Hence we can see in Table 4.3 the TTP’s efficiency is improved when it is generalized to multiple entities. Since communicating entities will usually pay for the TTP services, we achieve a more efficient and cheaper TTP service. In addition, we can see in Tables 4.1 and 4.2 that O’s efficiency is improved too, while R_i’s is slightly increased. However, if the originator and the recipients have any kind of previous relation between them and they all share a secret, then the encryption of n_i could be avoided in each protocol run though it should be still included in evidence. Furthermore, the asymmetric encryption of n_i can be prepared off-line but, on the other hand, the fetch operations must be performed during the protocol run.

4.1.2 *Optimistic MPNR Protocol for Exchange of Different Messages*

There is an optimistic approach in non-repudiation protocols where the entities are likely to behave honestly, thus giving priority to the main protocol and running sub-protocols only in case that an exception arises. Here we present an optimistic multi-party non-repudiation protocol based on [58] [2], and use the same solution described in the previous section for the privacy of different messages.

Markowitch et al. proposed an optimistic protocol for distribution of the same message to several parties with a non-transparent TTP. Summarizing the study done in the previous chapter, four steps are required in the main exchange, which is not optimized. Their protocol also makes use of a pre-defined time constraint, thus does not achieve asynchronous *timeliness*. In addition, their protocol employs an inefficient ftp operation with the originator acting as a server.

Some additional notation in this protocol description is as follows.

- $R'' = R - R'$: a subset of R (in plaintext) with which O wants to cancel the exchange
- $R''\_finished$: a subset of R'' that have finished the exchange with the finish sub-protocol
- $R''\_cancelled = R'' - R''\_finished$: a subset of R'' with which the exchange has been cancelled by the TTP
- $l = h(M_1, M_2, .., k)$: label [3] that identifies the protocol run computed as the hash outcome of the concatenation of every message plus the key k
- $k_T = E_{u_{TTP}}(k)$: key k encrypted with TTP's public key
- $EOO_i = S_O(feoo, R_i, TTP, k_T, l, v_i, u_{R_i}, h(c_i))$: evidence of origin for R_i
- $EOR_i = S_{R_i}(feor, O, TTP, k_T, l, v_i, u_{R_i}, h(c_i))$: evidence of receipt from each R_i
- $Sub_k = S_O(fsub, R', l, k)$: evidence of submission of the key to recipients
- $Cancel_{req} = S_O(TTP, R'', l)$: evidence of request of cancellation issued by the originator to the TTP
- $Cancel_O = S_{TTP}(O, l, R'', R''\_cancelled, Cancel_{req})$: evidence of cancellation issued by the TTP to the originator
- $Cancel_{R_i} = S_{TTP}(R_i, l, EOR_i, Cancel_{req})$: evidence of cancellation issued by the TTP to R_i
- $Con_k = S_{TTP}(R_i, l, k)$: confirmation evidence of k issued by the TTP

4.1.2.1 Protocol

The protocol consists of a *main* protocol (which will be the only one executed by the entities in the normal situation) and two sub-protocols: *cancel* and *finish* (see Figure 4.2). The TTP is only involved in the sub-protocols in case of any participant's misbehavior or channel failure between the originator and the recipients. Any participant can initiate the corresponding sub-protocols to terminate a protocol run at any time without loss of fairness.

The *main* protocol executed by the final entities is as follows.

[2] The protocol [58] has some security errors as being identified in [130]. Those problems have been corrected in the design of our new protocol.

[3] Note that with the reduction to 3 steps in the main protocol, the attack proposed in [68] does not work in our approach.

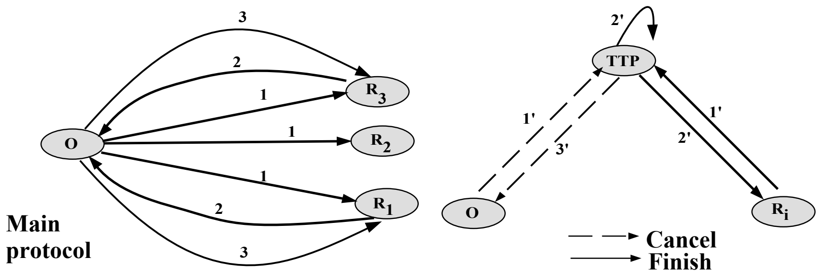

Fig. 4.2 Optimistic Protocol with Different Messages

1. $O \rightarrow R_i : feoo, R_i, TTP, k_T, l, v_i, u_{R_i}, c_i, EOO_i$ for each $R_i \in R$
2. $R_i \rightarrow O : feor, O, l, EOR_i$ where $R_i \in R$
3. $O \Rightarrow R' : fsub, l, E_{R'}(k), Sub_k$

In step 1, the originator sends to each recipient its message encrypted with k_i. A recipient can derive k_i from k and a random number n_i which are also sent in this step (in a confidential way). Note that the TTP is included in this step, thus there is no confusion about which TTP to use in case they have to launch any of the sub-protocols. The originator picks each receiver's public key. If any recipient does not want to use such a key (e.g., the correspondent public key certificate has been revoked), then it stops the protocol. Otherwise, after verifying the data obtained, the recipient sends to the originator evidence of receipt at step 2, and the originator sends to the set of recipients who replied after a reasonable amount of time at step 3, the key and evidence of submission of that key, as a second part of evidence of origin. If O did not receive a correct message 2 from some of the recipients R'', O may initiate the following *cancel* sub-protocol.

```
1'. $O \rightarrow TTP : TTP, R'', l, Cancel_{req}$
2'. $TTP$      FOR (all $R_i \in R''$)
                 IF ($R_i \in R''\_finished$) THEN retrieves $EOR_i$
                 ELSE appends $R_i$ into $R''\_cancelled$
3'. $TTP \rightarrow O$ : <all retrieved $EOR_i$ >, $R''\_cancelled$, $Cancel_O$
```

In this case, the originator communicates the TTP its intention of revoking the protocol with entities contained in R'' and for the protocol run labelled l. After verifying O's cancel request, the TTP checks which entities previously resolved the protocol and retrieves their proofs of receipt. The TTP generates an evidence of cancellation for the rest of entities and includes everything in a message addressed to the originator. We do not send any information about the message, keys and EOO as [58] does because we use a well-defined label for indexing purposes on the TTP side.

If some recipient R_i did not receive message 3, R_i may initiate the following *finish* sub-protocol.

$1'.\ R_i \rightarrow TTP : TTP, k_T, l, v_i, u_{R_i}, h(c_i), EOO_i, EOR_i$
$2'.\ TTP \rightarrow R_i :$ IF ($R_i \in R''\_cancelled$) THEN $R_i, l, R'', Cancel_{req}, Cancel_{R_i}$
ELSE $\{R_i, l, E_{u_{R_i}}(k), Con_k$ AND
appends R_i into $R''\_finished$ and stores $EOR_i\}$

The recipient sends to the TTP all the information that it has already got from the originator along with its evidence of receipt. If this entity does not belong to the group of entities with which the originator has cancelled the exchange, the TTP verifies all the information (digital signatures) and decrypts k_T, obtaining the key for the recipient. It also stores EOR_i. Note that if the protocol has been cancelled, it should be impossible for the recipient to cheat the TTP in a way that the TTP reveals the key k for that protocol run. For such a reason, the TTP must verify O's signature in the first step and check that l and k_T provided by the recipient fits with the information contained in EOO_i. This becomes above all important when confidentiality should be provided, since it prevents using the TTP as a decryption oracle (e.g., a recipient sending in other execution message $1'$ trying to get k_T decrypted). It also avoids replay attacks as discovered in some optimistic protocols [120].

Otherwise, the TTP sends a cancellation evidence to the recipient such that the latter can easily demonstrate to an arbitrator that the exchange was cancelled in case a dispute arises. This evidence includes the request of cancellation, such that the TTP's behavior is verifiable while the TTP need not to store all the request evidences from the originator.

Note that the originator has no interest in sending a subset $\bar{R}'' \neq R''$ when requesting to the TTP for cancellation. If $\bar{R}'' \supset R''$, then O will cancel the exchange with those $R_i \subset \bar{R}'' - R''$ that have replied with EOR_i. If $\bar{R}'' \subset R''$, then $R_i \subset R'' - \bar{R}''$ may invoke the *finish* sub-protocol to get the key k but O will not obtain EOR_i from the TTP at the *cancel* sub-protocol.

At the end of the protocol, if successful, the participants get the following evidence.

- NRO for honest recipient R_i: EOO_i, Sub_k or Con_k
- NRR for originator O: EOR_i for all honest recipients $R_i \in R'$

4.1.2.2 Dispute Resolution

As we have mentioned, two kinds of disputes can arise. Here we further discuss the rules for their resolution.

Repudiation of Origin. If O denies sending M_i, R_i can present evidence EOO_i and Sub_k (or Con_k) plus $(TTP, l, u_{R_i}, v_i, c_i, n_i, k, k_T, M_i)$ to the arbitrator. The arbitrator will check

- $v_i = E_{u_{R_i}}(n_i)$

- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- O's signature on EOO_i
- O's signature on Sub_k, or TTP's signature on Con_k
- R_i is into the signed token Con_k or $R_i \in R'$ as signed in Sub_k.

Repudiation of Receipt. If R_i denies receiving M_i, O can present evidence EOR_i plus $(TTP, l, u_{R_i}, v_i, c_i, n_i, k, k_T, M_i)$ and $(R'', R''\_cancelled, Cancel_O)$ if it has. The arbitrator will check

- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k$ xor n_i
- $c_i = E_{k_i}(M_i)$
- R_i's signature on EOR_i
- (TTP's signature on $Cancel_O) \wedge (R_i \notin R''\_cancelled)$

O will win the dispute if all the above checks are positive. If all the checks but the last are positive and O cannot present evidence $Cancel_O$, the arbitrator must further interrogate R_i. If the latter cannot present $Cancel_{R_i}$ (or this token is not properly constructed including $Cancel_{req}$ from O), O also wins the dispute. Otherwise, R_i can repudiate having received the message M_i.

We can also see that evidence provided by the TTP is *self-contained*, that is, the TTP need not to be contacted in case a dispute arises regarding the occurrence or not of the *cancel* sub-protocol launched by O. Thus, the TTP is efficiently verifiable. For instance, if the TTP cheats and distributes evidence of cancellation to R_i when O did not cancel the protocol for it, TTP's misbehavior will be visible (either the cancellation token will not be properly constructed with O's evidence of request embedded or it will reveal TTP's misconduct since $R_i \notin R''$). Additionally, it is important to know that the adjudicator does <u>not</u> need to check that $k_T = E_{u_{TTP}}(k)$ holds.

4.1.2.3 Protocol Extensions

Transparent TTP. Our protocol can be modified such that the recipient can obtain the same evidence of origin even in case it needs to launch the *finish* sub-protocol. For this, we only have to make possible for the TTP to send O's signature Sub_k whenever the recipients try to fetch the key. In this way, external parties will not be able to distinguish if the recipient launched the *finish* sub-protocol, since evidence obtained is the same. It helps to preserve O's reputation in case of channel failures, and on the other hand also eases the dispute resolution process. We simply redefine k_T as follows.

$$k_T = E_{u_{TTP}}(k, Sub_k)$$

In the *finish* sub-protocol, the TTP decrypts k_T and additionally checks that Sub_k is O's signature on $(fsub, R, l, k)$. If the TTP succeeds in all the checks, then it provides Sub_k to the recipient instead of Con_k. Note that in this case, Sub_k will include

R because the originator does not know R' in the first step of the protocol. Therefore, the dispute resolution process also changes since now the arbiter can just check that $R_i \in R$ in case of repudiation of origin. Nevertheless, this does not damage fairness since the TTP only provides R_i with Sub_k in the finish protocol if R_i is correct in EOO_i and $R_i \notin R''\_cancelled$.

Message Confidentiality. Our protocol has already fulfilled the confidentiality requirement regarding external attackers. However, if we also want to keep the message confidential to the TTP, the originator needs to transmit c_i to the recipients in a private way (e.g., via private channel such as SSL).

4.1.2.4 Efficiency

This protocol is very efficient in the normal case with a good behavior of the participating entities. In fact, three steps is the minimum number of steps we could reach without breaking fairness in non-repudiation protocols. Even with multiple recipients for exchange of different messages, it manages to use only one key for evidence distribution, thus decreasing the computation and verification requirements for the originator and the TTP. For this new feature, public key encryption and decryption of temporal random numbers are the main extra cost added.

It is straightforward to see that this protocol is more efficient than any combination of two-party protocols, since it allows to send different messages in a confidential way to multiple entities as well as to cancel the protocol for a group of entities R'' in only one run of the *cancel* sub-protocol.

In addition, this protocol achieves asynchronous timeliness, because each entity can terminate the protocol, if needed, at any time at its own discretion while maintaining fairness. Nevertheless, as indicated in Section 3.3, asynchronous timeliness, though desirable for users, augments requirements on the channel communication between TTP and users. With the model we assumed, the TTP could setup an evidence server (e.g., FTP server) such that users participating in the protocol can access (with the read permission only) and retrieve evidence. By policy, there must exist a deadline after which evidence cannot be retrieved [4]. With this transformation the protocol is adapted with respect to timeliness in a similar way as the on-line version is.

It seems very important to us to compare this off-line solution with the one proposed by Markowitch et. al even though the latter one did not address multiple messages. On the other hand, this is the only one we can compare with as an off-line multi-party non-repudiation protocol. Since the protocol has already been briefly described in the previous chapter, we present here several tables to make the comparison straightforward. For efficiency comparisons, the following measurement units have been taken into account (when several actions were possible we considered the worst case).

[4] This deadline can match evidence validity depending on the TTP's storage capacity; see Section 4.3 for further considerations.

- *Public key operations* (containing encryption and decryption as well as digital signature and verification).
- Number of items in the *storage*.
- Number of *steps* as number of messages which need to be sent over the network.
- P as the number of recipients which need to contact the TTP for help and succeed.
- $N = |R|$ as the number of participating recipients.

	Our approach	**Markowitch's**
1. Number of steps in main protocol	3	4
2. Number of steps in sub-protocol	2 (finish) 2 (cancel)	2 (if recovered or early) 4 (if not recovered yet)
3. Timeliness	Async	Sync
4. Items to be stored in the TTP	P+1	1
5. Additional interrogation needed in the dispute resolution process	Yes	No
6. TTP network access (reliability in the inverse direction)	No	Yes
7. Timely drawbacks	No	Yes (stop main protocol)
8. Transparency	Yes	No

Table 4.4 General Off-line Solution Comparison

	Our approach	**Markowitch's**
1. O's main protocol	4+2P+N	4+2P
2. R_i's main protocol	3	4
3. O's cancel/recovery protocol	2+P (cancel)	1 (recovery)
4. R_i's finish protocol	2	2
5. TTP's cancel protocol	2 (once)	-
6. TTP's finish protocol	5	5

Table 4.5 Public Key Operations Comparison

In Table 4.4, we can observe that generally and when possible, our protocol improves Markowitch's version. Note that, for instance, it is not possible to reduce the number of items the TTP needs to store as a consequence of dealing with different messages and, thus, different evidence of receipt (since we maintain the transparency property for O). This storage capacity request could be easily avoided by allowing the TTP to issue an affidavit for finished entities, but then, transparency would be lost. There is an improvement on the number of steps needed in the main protocol, which fortunately will be the most frequently used, but the cost is the need of rarely having to interrogate an additional entity in case of disputes.

In Table 4.5, we can observe how, in general, our protocol seems to perform worse. But if we analyze the table, we will discover that this is due to the new functionality introduced and thus, unavoidable. Specifically, operations of O's main protocol are augmented in N because the number of different messages to be sent. For the same reason mentioned above, O needs to verify P different evidence of receipt from honest entities which contacted the TTP for finishing the protocol on time. For the rest of steps, both protocols perform in a similar way.

4.1.3 Fairness vs. Collusion

As it has been introduced in this book, there exist different levels of fairness. This protocol provides two different levels of fairness depending on the initial assumptions and thus, depending on these assumptions the applications which will make use of them vary.

In the description of the previous protocols we assume that no collusion is possible between recipients. This will preserve a strong fairness property and applications benefited of these kind of protocols can be of any type.

Nevertheless, R_i could get c_i in the initial steps of the protocol and quit. Then, colluding with any other party and getting the unique key k, it could decrypt c_i without providing any evidence of receipt. Note that this issue is very difficult to solve as it is not possible to prevent one (semi) honest entity from sharing a secret once it decrypts the secret. However, it is also important to note that when a recipient misbehaves and colludes with a (semi) honest recipient in order to get the key and thus (only) the message intended to him, it will not in any case obtain evidence of origin, unless the originator obtains evidence of receipt. This is what has been defined as light fairness.

There are multiple applications which only need light fairness in their transactions. A common feature in these applications is that it is more important for the exchange of evidence than the message content itself. That is, it is critical that both or none of the entities obtain evidence. For instance, some notification systems only need light fairness. This happens when the message itself is known by the recipient (e.g. birthday certificate) but in the notification, the receiver wants to be able to demonstrate to a third party its origin (e.g. official administration office) and the originator wants to be able to demonstrate if necessary that the message was indeed delivered to the intended receiver.

4.1.4 Further Discussions

The proposed approaches for multi-party non-repudiation protocols considerably improve the number of messages exchanged as well as the amount of evidence collected by the final entities involved. However, they are only the first effort to generalize the non-repudiation service to multiple entities and messages, and might be reviewed to further improve their efficiency.

4.1.4.1 Group Encryption

Along the description of the protocols presented previously, we have been using the notation $E_{R'}(k)$ to define an encryption operation over the key k intended for a group R'. In the multi-party non-repudiation protocol proposed in [87], a group encryption scheme proposed by Chiou and Chen [45] is used. It is based on a public key encryption scheme and on the Chinese Remainder Theorem (CRT). As the authors explained, it is efficient only when the number of users is small, since the time to compute the CRT and its length (hence transmission time) is proportional to the number of users. This method is generic because it can use any public key cryptosystem.

- Let u_{R_i} and $\overline{u}_{R_i}$ be the public and private keys of R_i, respectively (where i corresponds to all parties that belong to R').
- Each recipient of R' receives a random integer $P_i > E_{u_{R_i}}(k)$ such that all P_i are pair-wise relatively prime. (When choosing randomly large primes or multiplications of distinct primes for example, the probability of obtaining two numbers that are not relatively primes is negligible.)
- O computes $X \equiv E_{u_{R_i}}(k)\ mod\ P_i$. Because all of P_i are prime integers, using the CRT, only one solution is obtained from this equation. Hence, $E_{R'}(k) \equiv X$. Each recipient R_i can obtain k by computing $X \equiv E_{u_{R_i}}(k)\ mod\ P_i$ using her private key $\overline{u}_{R_i}$.

From a computational point of view, we know that the CRT is an additional effort to the encryption with each user's public key. Analyzing the CRT properties we realize that X can be of the same magnitude as $\prod_{i=1}^{m} P_i$ with m the number of users in the group R'. Besides, from the second step above we know that $P_i > E_{u_{R_i}}(k)$ for every i, so the length of the message distributed to the recipients is still long. Although it is straightforward to prove that the length of a message M such that $M = M_1 M_2 \cdots M_n$ (a simple concatenation of messages) with M_i of the same length is greater than M' such that $M' = M_1 * M_2 * \cdots M_n$, it is also true that P_i needs to be much greater than $E_{u_{R_i}}(k)$ to avoid possible problems in future encryptions, thus making M and M' approximately of the same size.

As a result, the distribution length improvement of message X with the CRT-based group encryption, if any, is not significant. Furthermore, in that scheme, the authors assume a model in which each recipient R_i has already got the random numbers P_i. However, if we assume that the originator had no prior contact with the recipients, the random numbers have to be distributed by the originator in each protocol run, thus losing any possible advantage. For these reasons we remove the CRT operations and define the group encryption operation as a straightforward concatenation of public key encryptions to each final recipient as follows.

$$E_{R'}(k) = E_{u_{R_1}}(k), E_{u_{R_2}}(k), \cdots, E_{u_{R_m}}(k)$$

Furthermore, the reason proposed by Chiou and Chen for which a group encryption scheme using the CRT is needed instead of a simple concatenation of asymmetric encryptions is that the latter needs a way for recipients to identify their message and thus some kind of labels. This would increase the length of the message. Nevertheless, and as analyzed, we do not consider this a sufficient reason for introducing the associated overhead into a MPNR protocol.

With respect to policy requirements, as stated in Section 2.5, there must exist a policy document (which could be XML-specified), as well as a policy identifier which needs to appear in every protocol run. This policy defines algorithms and rules to be followed by all users, and should be agreed in advance by the parties involved in the service, in the setup phase. We intentionally did not define a policy because its elements will depend on the application itself. Nevertheless, along the description of the protocols, we identified the different possibilities to keep in mind when defining the policy.

4.1.4.2 Optimization based on VPN

Since each topology has its own requirements, our protocols should be adapted to fulfill these requirements, and moreover, adapted to take advantage of the new features.

Basically, a VPN is a private network that uses a public network (usually the Internet) to connect remote sites or users together. Instead of using a dedicated connection such as a leased line, a VPN uses "virtual" connections routed through the Internet. Remote access to VPNs permits secure, encrypted connections between a private network and remote users through a third-party service provider (e.g., *Virtual Private Dial-up Network*) or between more private networks (e.g., *Intranet-based and Extranet-based VPN*).

Inside a VPN, there is typically an AAA (*Authentication, Authorization and Accounting*) server that distributes symmetric keys to a user for confidential communications with other users in the VPN. Therefore a sender can use the symmetric keys to encrypt the messages for the recipients, and all the encryption operations with the recipient's public key used in our previous protocols can be replaced. For example, v_i was defined as $v_i = E_{u_{R_i}}(n_i)$. Now, n_i can be sent to each recipient without public key encryption since the channel between a pair of entities is ciphered in a VPN. This improvement on the computational overhead is applied $|R|$ (number of recipients) times in one protocol run for each public key encryption that we used. In addition, the group encryption scheme can also be changed to take advantage of the symmetric keys in the VPN, i.e., $E_{R'}(k) = E_{sk_1}(k), E_{sk_2}(k), \cdots, E_{sk_m}(k)$ where sk_i is the key shared between the sender and each recipient. With such an optimization, efficiency of final entities in our protocols is further improved.

Of course, this will not be the normal case, because a VPN provides a trusted network for users who already enjoy some kind of relation and thus, trust. Nevertheless there are specific applications of non-repudiation which are also used and appropriate for trusted environments, as for instance, CEM (see Section 5.1.1 for a CEM protocol).

4.2 Agent-Mediated Non-repudiation Protocols

Although optimistic approaches are more efficient from a theoretical point of view in the case of non-repudiation services for payment schemes, on-line approaches could be desirable. In [82], Jakobsson stated that "practically on-line technique is, in fact, not a step backwards. This is the case given the current trend in banking, suggesting that only on-line payment schemes will be considered by banks, in order to minimize the potential losses made possible by global and instantaneous commerce capabilities ... On-line payment model has been abandoned for a long period of time, and much effort has been spent in developing off-line payment schemes, but now, the on-line paradigm is seeing a renaissance again, given the commercial preference for it."

In commercial transactions, an intermediary (or agent) might be involved to help transacting parties to conduct their business. Nevertheless, such an intermediary may not be fully trusted (by the users, but yet by banks ... or none at all). In this section, we propose agent-mediated non-repudiation protocols, and analyze their security requirements. We first present a simple scenario with only one recipient, followed by a more complicated framework where multiple recipients are involved and collusion between them is possible. We also identify applications that could take advantage of these agent-mediated non-repudiation protocols. The work in this section has been partially published in [108].

The intermediary or *agent* is an on-line entity which either uses mobile code agents or local programs for its operations. On the other hand, for a more focused solution on security into the field of mobile agents, Section 6.3 will present a solution which preserves non-repudiability of agents' messages and better explain general concepts related with mobile agents' code.

The use of an intermediary (or agent) to improve electronic transactions is not novel and can be found in [85, 104]. Nevertheless, no agent-mediated non-repudiation protocol exists to the best of our knowledge. Although two-party non-repudiation protocols could be used to implement an agent-mediated non-repudiation protocol, we will propose a new approach to improve the efficiency of such an implementation. In our new approach, a distrusted intermediary entity (different from the TTP) is introduced to facilitate the collection, verification, and storage of evidence on behalf of the final entities. We demonstrate that the use of such an intermediary entity satisfies the security requirements expected in an e-commerce transaction, as well as non-repudiation requirements as defined in Section 2.5.

4.2.1 Model

In our system, an evidence database is maintained by the intermediary entity to store securely evidence for each transaction. Depending on the application, and thus, its policy, the evidence records may have an expiry date (and then, the dispute would not be settled after this date). Our system is flexible, and if the originator requests so, evidence can be transferred to it during the protocol run (or even afterwards). The non-repudiation policy defines who assumes which responsibility. This framework can also be extended to multiple recipients taking advantage of an intermediary acting as a hub. Figure 4.3 shows the model for which our protocol is designed.

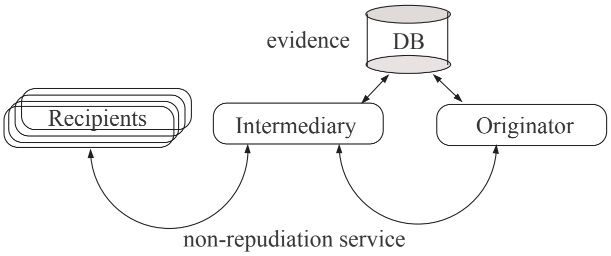

Fig. 4.3 E-commerce Scenario with an Active Intermediary Agent

As we will see, fairness of a non-repudiation protocol depends overall on the behavior of this intermediary agent. The behavior of such an entity is usually related to its interests in the e-commerce scenario. Thus, we can suppose this entity wishes to establish business relations with the participants and earn more profits by providing satisfactory services. Even so, we still do not treat the intermediary as a fully trusted entity in our model. Evidence of transactions carried out with this entity will be collected by the originator and recipients.

The main motivation behind this model is that when designing non-repudiation protocols, we have found that in many Internet applications, this model with a third party which could not be totally trusted (payment gateway, brokers, digital commerce mall, etc.) is used. It is natural to adapt new designs to existing applications, because trying to change business-established models has been demonstrated not to succeed.

4.2.2 First Solution

An intuitive solution to our agent-mediated non-repudiation model is to use two-party non-repudiation protocols for each two-party transactions taking place on it. It can be described as a three-step scenario, where the originator first runs a fair non-repudiation protocol with the intermediary (IN), then the IN executes the protocol with the recipients, and finally the originator collects evidence from the IN (see Figure 4.4). If the fair non-repudiation protocol of [132] is used for the first and third steps, and its extended version for the exchange of multiple different messages, as explained in the previous section, is used for the second step, at least 17 messages are required to complete a transaction (including an on-line use of a TTP service), without even considering the number of encryption operations and digital signatures.

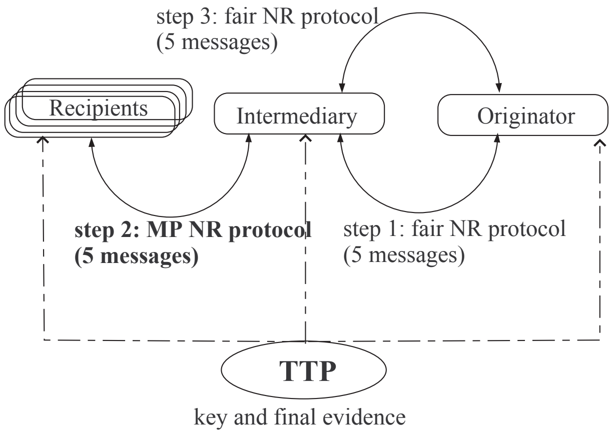

Fig. 4.4 An Intuitive Solution to Non-repudiation

With the intuitive solution, a fair non-repudiation protocol (5 message flows) is executed at Step 1 in which the originator requests the intermediary agent to deliver the messages to the recipients. As a result, the IN obtains evidence of origin about the service requested by the originator, and the latter obtains the promise of the IN to do its best to deliver exactly those messages to the recipients indicated by the originator; that is, evidence of receipt.

Then, a multi-party non-repudiation protocol (5 message flows) is used at Step 2 in which the IN delivers the messages to the recipients. As a result, the recipients receive the service provided by the IN while the IN obtains the recipients' confirmation of the service. Again, as in Step 1, an exchange about the result of the requested service is carried out between the originator and the IN at Step 3 (5 message flows). This step allows the originator to obtain evidence about the result, and the IN to obtain the originator's acknowledgement.

At least 2 more message flows are needed to complete the transaction. The originator lodges the keys of the commitments with the TTP, and all the entities collect the keys and final evidence from the TTP. It is thus clear that, introducing an intermediary or agent which is not fully trusted in a business transaction, involves numerous steps in the non-repudiation service provision. So, we provide a new approach which improves this service provision in these scenarios.

4.2.3 Simple Agent-Mediated Protocol

In this section, we present our new approach for agent-mediated non-repudiation by first introducing a simple protocol with an intermediary agent and a single recipient. In a gradual manner we will extend this approach.

4.2.3.1 A Simple Protocol

As we already noted, the intermediary agent plays a critical role in this scenario, so it is important to analyze its behavior. If the IN has any interest, because, for instance, it charges the originator or the recipients or it is just a party owned by the bank or payment gateway into a commercial transaction, then it will be willing to reach a successful operation. But occasionally, the IN may collude with another (external or internal) entity and, for instance, hide some evidence. Therefore, we assume the intermediary agent is not fully trusted.

Here we presume that the IN, selected by the originator, is not going to hide the initial messages from the originator to the intended parties (in Section 4.2.4.3 we will explain how to get rid of this assumption). The simplest approach comes when the originator wishes to send a message to a single recipient. In this scenario, the IN does not play the role of a hub. Nevertheless, it communicates directly with the recipient, and could help the originator not only in the non-repudiation protocol itself but also in the preliminary steps, such as search for a recipient and a product, price agreement, etc. For this purpose, we introduce a new term *request* that gives the IN some information about the service to be provided. The following notation is used in the protocol description.

- O, R, and IN : originator, recipient, and an intermediary agent
- $All = O, IN, R$: set of entities that will contact the TTP
- M : message being sent from O to R
- k : key being selected by O
- $c = E_k(M)$: encrypted message for R with key k
- $l = h(O, IN, R, TTP, h(c), h(k))$: label of message M
- t : a timeout chosen by O, before which the TTP has to publish some information
- $EOOc = S_O(feoo, IN, R, TTP, l, t, h(request))$: evidence of origin of c generated by O (*note that c is included into label l*)
- $EOOI = S_{IN}(feooi, R, O, TTP, l, t)$: evidence of origin of c issued by the IN for R
- $EORc = S_R(feor, TTP, IN, O, l, t, u_R)$: evidence of receipt of c generated by R
- $EORI = S_{IN}(feori, TTP, O, R, l, t, request, u_R)$: evidence of receipt of c issued by the IN for O

- $Sub_k = S_O(fsub, TTP, IN, R, l, t, E_{u_R}(k), u_R)$: evidence of submission of the key to the TTP generated by O
- $Con_k = S_{TTP}(fcon, All, l, t, E_{u_R}(k), u_R)$: evidence of confirmation of the key issued by the TTP

The protocol is as follows.

1. $O \rightarrow IN$: $feoo, IN, R, TTP, l, h(k), t, request, c, EOOc$
2. $IN \rightarrow R$: $feooi, R, O, TTP, l, h(k), t, c, EOOI$
3. $R \rightarrow IN$: $feor, IN, O, l, u_R, EORc$
4. $IN \rightarrow O$: $feori, O, R, l, u_R, EORI$
5. $O \rightarrow TTP$: $fsub, TTP, IN, R, l, t, E_{u_R}(k), u_R, Sub_k$
6. $All \leftrightarrow TTP$: $fcon, All, l, E_{u_R}(k), Con_k$

The protocol works in the following way.

Step 1: O sends to the IN the request information [5] and evidence of origin corresponding to the encrypted message c. The encrypted message c may be some sensitive information, for instance bank account data, that O is not intended to reveal to the IN. There is no breach of fairness if the protocol stops. Note that $h(k)$ is only sent such that IN can verify label l, which is included in the signature.

Step 2: The IN distributes O's information (maybe after a negotiation or agreement with R), and sends to R evidence of involvement in the transaction. Again, fairness is maintained if the protocol stops.

Step 3: R replies with evidence of receipt of encrypted message c. R's public encryption key u_R is included in $EORc$ to make it undeniable when O uses it at Step 5 to distribute key k. In this way, the originator does not need to verify or retrieve any public key information about the recipient. The protocol still remains fair if it stops.

Step 4: The IN replies to O, indicating that R agreed the transaction. At the same time evidence of receiving *request* and c is given to O. O will check this evidence carefully before proceeding to the next step, since this is the only evidence O will collect from the IN and will be used by O in case of disputes to prove the IN's responsibility of the exchange. The IN will store R's evidence of receipt in its evidence database, and O can retrieve it later if needed. The IN cannot claim that it did not store this evidence since $EORI$ demonstrates it did if a dispute arises. No party is benefited if the protocol stops at this step.

Step 5: O submits the key (encrypted with R's public key) to the TTP, such that only the intended recipient will be able to disclose the message. The TTP will process it only if the submission is received before deadline t.

Step 6: The TTP releases the encrypted key. O fetches Con_k as evidence that it submitted the key on time to complete the transaction. The IN fetches Con_k as evidence that O accepted $EORI$ and, thus, the service offered. R obtains the key to decrypt c and fetches Con_k as evidence to prove its origin.

[5] If confidentiality for the request information is required, it needs to be encrypted.

At the end of the protocol, each party will hold the corresponding evidence [6].

- The originator should collect $EORI$ and Con_k as evidence of receipt.
- The IN should collect $EOOc$, $EORc$, and Con_k as evidence of origin and evidence of receipt, respectively, which allows the IN to demonstrate its good behavior during the protocol.
- The recipient should collect $EOOI$ as evidence of origin of c issued by the IN. Con_k must also be collected as evidence of origin of the key.

Our protocol takes only 6 steps, improving the first intuitive solution we presented in Section 4.2.2 on the number of messages sent over the network. In our protocol, *anonymity* could be preserved. That is, unless the originator is willing to communicate with a pre-selected recipient, neither the originator nor the recipient needs any knowledge (i.e., authentication digital certificates) about each other in order to reach a successful protocol end. As we can see above, only the IN needs the final entities' digital certificates in order to verify their digital signatures, while the final entities only need the IN's digital certificate during the protocol execution.

4.2.3.2 Dispute Resolution

In our model, disputes might arise between any pair of three parties. If the evidence has an expiry date, the disputes should be settled with the help of an arbitrator prior to that date.

Disputes between Originator and Recipient

If O denies sending message M, R shows the arbitrator evidence $EOOc$ and Con_k. With $EOOc$, the arbitrator checks whether O originated c. With Con_k, the arbitrator checks whether $E_{u_R}(k)$ is encrypted with u_R and published by O via the TTP. The arbitrator will also check the validity of label l in $EOOc$ and Con_k. If all checks are positive, the arbitrator settles that message M is from O.

In order to obtain $EOOc$, the recipient must retrieve this evidence from the IN's evidence database. But if the IN precludes the recipient's access to evidence (or it is not valid), the recipient should present $EOOI$ to the arbitrator, proving that it received the encrypted message c from the IN and now the responsibility of submitting $EOOc$ lies on the latter. Of course, IN in $EOOI$ must be the same as the one contained in Con_k.

If R denies having received a message M from O, the latter shows the arbitrator evidence Con_k and EOR_c. With EOR_c, the arbitrator checks whether the recipient received the encrypted message $c = E_k(M)$ and extracts R's public key u_R as well. With Con_k, the arbitrator checks whether the key k was published before deadline t and whether the correct R's public encryption key was used.

[6] Note that c has been eliminated from digital signatures because it is already present in label l which is, at the same time, included in evidence. Therefore, for resolving disputes, an arbitrator needs always to verify label's consistency.

In order to obtain $EORc$, the originator must retrieve this evidence from the IN's evidence database. But if the IN precludes the originator's access to evidence (or it is not valid), the originator should present $EORI$ to the arbitrator, proving that it received the proof that IN received repudiation of receipt evidence from recipients and now the responsibility of submitting $EORc$ lies on the latter. Of course, IN in $EORI$ must be the same as the one contained in Con_k.

Again, the arbitrator needs to check the validity of label l in Con_k. If all checks are positive, the arbitrator settles that O published key k.

Disputes between Originator and Intermediary

If the IN denies having received any request labelled l from O, O presents $EORI$ and the arbitrator checks the IN's signature on it. If successful, the arbitrator settles that O sent the request to the IN.

Note that IN could try to cheat the originator by sending a different u_R. However, this behavior will be immediately found in case of dispute. The arbiter will check the incongruence which exists between u_R into $EORI$ and into $EORc$, both connected by the label l. Of course, in the policy defined by this protocol, the IN is held responsible for storing the related $EORc$ once it sent $EORI$ to the originator.

If O denies having received a response from the IN for a labelled transaction l, the IN presents Con_k and the arbitrator checks the TTP's signature on it. If successful, the arbitrator settles that O published the key due to receipt of a response from the IN.

Disputes between Recipient and Intermediary

If the IN denies delivering message c to R, R presents evidence $EOOI$ and the arbitrator checks the IN's signature on it. If successful, the arbitrator settles that c, originated from O, is delivered by the IN to R.

If R denies having received message c, the IN presents $EORc$ and the arbitrator checks R's signature on it. If successful, the arbitrator settles that the IN delivered c to R.

Of course, in all dispute resolution processes depicted in this section, the arbiter needs to check consistence of all parameters included in different signatures.

4.2.4 Extension to Multiple Recipients

The participation of an intermediary becomes more interesting when multiple recipients are involved in a transaction. In this scenario, the IN acts as a hub, that is, the originator sends the transaction information to the IN, and the IN transacts with multiple recipients according to the originator's *request*. A new protocol that combines a simple agent-mediated non-repudiation protocol presented in Section 4.2.3 and a multi-party non-repudiation protocol in [87] is introduced in this section.

In order to release the key only to the honest parties, a group encryption mechanism as discussed in Section 4.1.4.1 is needed that allows the encryption of a message to be decrypted by the intended group of recipients.

4.2.4.1 A Multi-Recipient Protocol

Some useful new notation in the protocol description is as follows.

- R': subset of R that replied to the IN with evidence of receipt
- $All = O, IN, R'$: set of entities that will contact the TTP
- $u_{R'}$: set of public encryption keys of the recipients belonging to R'
- $E_{R'}(k)$: a group encryption scheme that encrypts k for the group R'
- $EOOI = S_{IN}(feooi, R, O, TTP, l, t, EOOc)$: evidence of origin of c issued by the IN for R
- $EORc_i = S_{R_i}(feor, TTP, IN, O, l, t, u_{R_i})$: evidence of receipt of c generated by R_i
- $EORI = S_{IN}(feori, TTP, O, R', l, t, request, u_{R'})$: evidence of receipt of c issued by the IN for O containing the recipients who replied
- $Sub_k = S_O(fsub, TTP, IN, R', l, t, E_{R'}(k), u_{R'})$: evidence of submission of the key to the TTP generated by O
- $Con_k = S_{TTP}(fcon, All, l, t, E_{R'}(k), u_{R'})$: evidence of confirmation of the key issued by the TTP

The protocol is as follows.

1. $O \rightarrow IN$: $feoo, IN, R, TTP, l, h(k), t, request, c, EOOc$
2. $IN \Rightarrow R$: $feooi, R, O, TTP, l, h(k), t, h(request), c, EOOc, EOOI$
3. $R_i \rightarrow IN$: $feor, IN, O, l, u_{R_i}, EORc_i$
4. $IN \rightarrow O$: $feori, O, R', l, u_{R'}, EORI$
5. $O \rightarrow TTP$: $fsub, TTP, IN, R', l, t, E_{R'}(k), u_{R'}, Sub_k$
6. $All \leftrightarrow TTP$: $fcon, All, l, E_{R'}(k), u_{R'}, Con_k$

Minor changes are introduced with respect to the previous one. In this situation, the IN should store all evidence collected from the honest recipients in the evidence database. In case of disputes, the resolution process remains unchanged (see Section 4.2.3.2) except for the fact that when verifying the label, it has to verify that $R' \in R$ for group $R \in l$. Although the recipients receive evidence of origin $EOOc$ from O, we assume that they do not store this evidence, even though they may. We explain the need of this evidence later in this section. Public encryption keys are included by each recipient to make them undeniable when O uses them at the group encryption scheme to distribute key k. In this way, the originator does not need to verify or retrieve any public key information about recipients.

In this scenario, we should prevent the IN from sending a $R'' \neq R'$ to O in Step 4. If $R'' \supset R'$, then the IN claims that some recipients replied but they actually did not. Some solutions exist depending on the transaction's type. If the disclosure of the message can be brought back or the transaction can be cancelled after a dispute resolution, O can request to settle the dispute and the IN will not be able to present all evidence of receipt. If it is not possible (i.e., for more critical transactions or exchanges), the IN should send all evidence of receipt to O at Step 4. But O only needs to keep $EORI$, and may not keep the evidence of receipt generated by each recipient after verifying that $R'' = R'$. O will terminate the protocol run if $R'' \neq R'$.

If $R'' \subset R'$, then the IN hides some evidence of receipt from some of the honest recipients. Here, the solution requires a recovery sub-protocol which allows these honest entities communicate directly to the TTP about their commitment to the transaction.

Note, however, that in no case is the fairness property overcome, but the normal functioning of the protocol is yet affected if the IN misbehaves in such a way. At the same time, the IN may try to cheat the originator by sending a different u'_R. However, this behavior will be immediately found in case of dispute. The arbiter will check the incongruence which exists between u'_R into $EORI$ and every u_{R_i} into $EORc_i$, for all $i|i \in R'$. Of course, in the policy defined by this protocol, the IN is held responsible for storing all correspondent $EORc_i$ once it sent $EORI$ to the originator. We need to bear in mind that an intermediary entity has no interest on misbehaving when this is going to be easily demonstrated, since in that case it will also easily lose registered users [7]. Furthermore, IN's misbehavior can be scored in a reputation system.

4.2.4.2 Recovery Sub-protocol

Let $t1 < t$ be a deadline time after which the recovery protocol cannot be launched by any recipient. The previous notation is modified as follows.

- $EOOc = S_O(feoo, IN, R, TTP, l, t, \mathbf{t1}, h(request))$: evidence of origin of c generated by O
- $EOOI = S_{IN}(feooi, R, O, TTP, l, t, \mathbf{t1}, EOOc)$: evidence of origin of c issued by the IN for R
- $EORc_i = S_{R_i}(feor, TTP, IN, O, l, t, \mathbf{t1}, u_{R_i})$: evidence of receipt of c generated by R_i
- $\mathbf{EORER_i} = S_{IN}(feorer, l, EORc_i)$: evidence of receipt of $EORc_i$ issued by the IN for R_i

The main protocol is modified as well.

$$\begin{array}{ll}
1.\ O \rightarrow IN & : feoo, IN, R, TTP, l, h(k), t, \mathbf{t1}, request, c, EOOc \\
2.\ IN \Rightarrow R & : feooi, R, O, TTP, l, h(k), t, \mathbf{t1}, h(request), c, EOOc, \\
 & \quad EOOI \\
3.\ R_i \rightarrow IN & : feor, IN, O, l, u_{R_i}, EORc_i \\
4.\ \mathbf{IN} \rightarrow \mathbf{R_i} & : \mathbf{feorer}, \mathbf{l}, \mathbf{EORER_i} \\
5.\ IN \rightarrow O & : feori, O, R', l, u_{R'}, EORI \\
6.\ O \rightarrow TTP & : fsub, TTP, IN, R', l, t, E_{R'}(k), u_{R'}, Sub_k \\
7.\ All \leftrightarrow TTP & : fcon, All, l, E_{R'}(k), u_{R'}, Con_k
\end{array}$$

[7] Originators and recipients may be paying a small amount of money in order to be able to use this agent or intermediary in their transactions.

The recovery sub-protocol is as follows.

$5.a.\ R_i \rightarrow TTP$	: $frec, IN, O, R, l, h(k), t, t1, h(request), u_{R_i}, EOOc, EORc_i$
$5.b.$ If $current\_time \geq t1$	: TTP ignores the message
Else	
$5.c.\ TTP \rightarrow O$	: $frec, O, R_i, l, u_{R_i}, EORc_i$
$5.d.\ O$	: adds R_i into R'

The recovery sub-protocol will be launched only in case of IN's misbehavior or channel failure. A new step has been introduced in the main protocol, such that the IN must acknowledge each evidence of receipt ($EORc_i$). If R_i receives $EORER_i$ at Step 4 but the IN does not include him in R', the recipient can present $EORER_i$ to the adjudicator in a dispute resolution.

If R_i does not receive $EORER_i$ at Step 4, considerably before $t1$, R_i should launch the recovery sub-protocol to contact the TTP directly. The TTP checks that message $5.a$ arrives before $t1$ and that the same $t1$ is signed by O in $EOOc$. Then, the TTP sends the recovery information to O and the latter will include R_i into R' for the group encryption of key k after validating the evidence and checking that R_i belongs to R. In such a case, (R' in Sub_k) $\wedge$ (R' in $EORI$) will not match. Afterwards, O may need to contact the IN for a corrected $EORI$. Since some recipients may launch the recovery sub-protocol before $t1$, O must wait until the deadline $t1$ has passed before proceeding to Step 6.

R_i may launch the recovery sub-protocol (several times) even when the IN behaves honestly. However, this does not give the recipient any benefit. On the contrary, the recipient may need to pay more when requesting this service from the TTP. On the other hand, it is straightforward to see that the IN is not interested in misbehaving, since the originator will easily realize its behavior and it could be punished by a reputation system. This reputation system, like in other applications [1], will help the originator in deciding which IN is more trustful.

Obviously, this recovery protocol can be optional depending on the application it is running over. As we already explained, this sub-protocol does not fix any fairness drawback, since neither the originator nor the recipients excluded will obtain valid evidence from each other. Therefore, this sub-protocol makes sense only in those network applications in which exclusion problem must be avoided (e.g., the originator saves big sums of money when maximum number of honest recipients participate in the exchange), since the overload obtained is not negligible. Note that not only needs the IN to send in the worst case $|R|$ messages (and digital signatures) more, but also the TTP will need to receive and verify $|R|$ digital signatures more as well as to contact the originator before time $t1$. This means that a resilient channel in that direction is needed and in this case t and $t1$ need a long time slot between them, thus, delaying the total time length of the protocol considerably. Although the TTP could store evidence in a directory network and O could access it at $t1$, thus relying the responsibility on the latter, it is not practically convenient. The reason is that we foresee the sub-protocol will be executed rarely and then, the TTP as well as the originator will execute the protocol normally in almost all instances.

4.2.4.3 Collaboration among Recipients

A problem might arise if the IN sends the messages to the recipients in a selective manner. The IN can always claim that some recipients did not reply. There are two possible scenarios. In one of them, the IN proceeds according to the information contained in *request*, choosing by itself the recipients. In this case, the originator has no other choice but to trust the IN for this service.

In another scenario, as we used in the previous protocol description, the originator will select all the intended recipients prior to the beginning of the transactions. Usually, the IN would not misbehave in such a way as if it has any interest in bringing a transaction to its end. Nevertheless, the IN may collude with another internal or external entity and exclude some recipients from the transaction if it can get more benefits.

In this case, the recipients should collaborate in order not to be excluded. After receiving Step 2 of the protocol, each honest recipient that did not receive this step again can distribute this message to the rest of recipients. Otherwise, it just continues. In order to obtain the group R of recipients before distributing any message, the recipient R_i should verify that the group R sent by the IN matches with the one included in $EOOc$ (in such a case, O will lose its anonymity). At least, one honest entity should receive Step 2 to avoid the IN's misbehavior. However, note that this issue, again, does not affect the fairness property of the protocol and its importance depends on the application scope (exclusion-freeness is not a mandatory requirement in a general non-repudiation protocol).

The collaboration among recipients could be used depending on the transaction type and the network latency since this solution needs more message flows and could overload the network.

4.2.5 *Further Extension to Multiple Messages*

Frequently, in e-commerce applications, the originator needs to send different messages to recipients in the same transaction. A modification can be made to distribute different messages to the intended parties. At the first step, the originator may send these different messages as well as the *request* (including the instructions on how to split them for each recipient) to the IN.

Let R be a group of n recipients and M_i the different plain messages that the IN sends to each R_i, with $i \in \{1..n\}$. The following notation is used in the protocol description.

- n_i : a random value generated by O for each R_i
- $v_i = E_{u_{R_i}}(n_i)$: encryption of n_i with R_i's public key
- $k_i = k\ xor\ n_i$: a key for each R_i
- $c_i = E_{k_i}(M_i)$: encrypted message with a key k_i for each R_i
- $l_i = h(O, IN, R_i, TTP, h(c_i), h(k))$: label of message M_i

- L' : concatenation of labels of the recipients belonging to R'
- $EOOc_i = S_O(feoo, IN, R_i, TTP, l_i, v_i, u_{R_i}, t, h(request))$: evidence of origin of c_i generated by O
- $C = l_1 c_1 v_1 u_{R_1} EOOc_1 ... l_n c_n v_n u_{R_n} EOOc_n$: concatenation of label, encrypted message, encrypted random number, public encryption key, and evidence of origin for each recipient in R
- $L = h(C)$: Label of transaction for IN and O [8].
- $EOOI_i = S_{IN}(feooi, R_i, O, TTP, l_i, v_i, u_{R_i}, t)$: evidence of origin of c_i issued by the IN for R_i
- $EORc_i = S_{R_i}(feor, IN, O, l_i, v_i, u_{R_i}, t)$: evidence of receipt of c_i generated by R_i
- $EORI = S_{IN}(feori, O, R', L, L', t, h(request))$: evidence of receipt of C issued by the IN for O
- $Sub_k = S_O(fsub, TTP, IN, R', L, L', t, E_{R'}(k), u_{R'}, EORI)$: evidence of submission of the key to the TTP generated by O
- $Con_k = S_{TTP}(fcon, All, L, L', t, E_{R'}(k), u_{R'}, EORI)$: evidence of confirmation of the key issued by the TTP

The protocol is as follows.

$$\begin{array}{ll}
1.\ O \rightarrow IN & : feoo, IN, R, TTP, t, h(k), request, C \\
2.\ IN \rightarrow R_i & : feooi, R_i, O, TTP, t, h(k), l_i, c_i, v_i, u_{R_i}, EOOI_i \\
3.\ R_i \rightarrow IN & : feor, IN, O, l_i, EORc_i \\
4.\ IN \rightarrow O & : feori, O, R', L', EORI \\
5.\ O \rightarrow TTP & : fsub, TTP, IN, R', L, L', t, E_{R'}(k), u_{R'}, Sub_k \\
6.\ All \leftrightarrow TTP & : fcon, All, L', E_{R'}(k), u_{R'}, Con_k
\end{array}$$

The originator selects the intended public keys that are going to be used in the encryption of n_i. If the recipient disagrees (e.g., because its authentication digital certificate has expired or been revoked), it should stop the protocol after receiving Step 2.

This protocol has the same properties as the one in the previous section. However, if no trust is placed on the IN, some external mechanism should be found to ensure this entity will distribute all the messages to the intended parties. This problem cannot be addressed by means of collaboration among recipients as previously proposed, because each recipient will receive a different message.

It seems that the uniqueness of the label has been broken, because now O and IN will use L as a label between them and while R_i and IN will use l_i as a label. But, in fact, two different unique labels are used through the protocol run and relation among them can be demonstrated ($l_i \in L' \in L$).

4.2.5.1 Dispute Resolution

In a similar way as we did with the one-recipient protocol, a dispute resolution process needs to be defined, this being part of the policy.

[8] Note that l_i as well as L' are contained in L.

Disputes between Originator and Recipient

If O denies sending message M_i, R_i shows the arbitrator evidence $EOOc_i$ and Con_k. With $EOOc_i$, the arbitrator checks whether O originated c_i. With Con_k, the arbitrator checks whether $E_{R'}(k)$ uses u_{R_i} and was published by O via the TTP. The arbitrator will also check the validity of label l_i (being $l_i \in L'$) in $EOOc_i$ and Con_k. If all checks are positive, the arbitrator settles that message M_i is from O.

In order to obtain $EOOc_i$, the recipient must retrieve this evidence from the IN's evidence database. But if the IN precludes the recipient's access to evidence (or it is not valid), the recipient should present $EOOI_i$ to the arbitrator, proving that it received the encrypted message c_i from the IN and now the responsibility of submitting $EOOc_i$ lies on the latter. Of course, the labelled IN which signs $EOOI_i$ must be the same as the one contained in Con_k.

If R_i denies having received a message M_i from O, the latter shows the arbitrator evidence Con_k and EOR_{c_i}. With EOR_{c_i}, the arbitrator checks whether the recipient received the encrypted message $c_i = E_{k_i}(M_i)$ and extracts R_i's public key u_{R_i} as well. With Con_k, the arbitrator checks whether the key k was published before deadline t and whether the correct R_i's public encryption key was used.

In order to obtain EOR_{c_i}, the originator must retrieve this evidence from the IN's evidence database. But if the IN precludes the originator's access to evidence (or it is not valid), the originator should present $EORI$ to the arbitrator, proving that the IN received evidence of receipt from the recipients and now the responsibility of submitting EOR_{c_i} lies on the latter. Of course, IN in $EORI$ must be the same as the one contained in Con_k.

Again, the arbitrator needs to check the validity of label l_i (being $l_i \in L'$) in Con_k. If all checks are positive, the arbitrator settles that O published key k.

Disputes between Originator and Intermediary

If the IN denies having received any request labelled L from O, O presents $EORI$ and the arbitrator checks the IN's signature on it. If successful, the arbitrator settles that O sent the request to the IN.

If O denies having received a response from the IN for a labelled transaction L, the IN presents Con_k and the arbitrator checks the TTP's signature on it. If successful, the arbitrator settles that O published the key due to receipt of a response from the IN.

Disputes between Recipient and Intermediary

If the IN denies delivering message c_i to R_i, R_i presents evidence $EOOI_i$ and the arbitrator checks the IN's signature on it. If successful, the arbitrator settles that c_i, originated from O, was distributed by the IN to R.

If R denies having received message c_i, the IN presents $EORc_i$ and the arbitrator checks R_i's signature on it. If successful, the arbitrator settles that the IN delivered c to R.

Of course, in all dispute resolution processes depicted in this section, the arbiter needs to check consistence of all parameters included in different signatures. For instance, for validating evidence of receipt EOR_{c_i} of a message M_i, the participating entity provides n_i, u_{R_i}, k, M_i, t and the arbiter checks that the following holds.

- $v_i = E_{u_{R_i}}(n_i)$
- $k_i = k\ xor\ n_i$: a key for each R_i
- $c_i = E_{k_i}(M_i)$: encrypted message with a key k_i for each R_i
- $l_i = h(O, IN, R_i, TTP, h(c_i), h(k))$: label of message M_i
- $EORc_i = S_{R_i}(feor, IN, O, l_i, v_i, u_{R_i}, t, h(c_i))$

4.2.6 Applications

Our approach fits in any software infrastructure for a large, distributed, agent-based commerce system, such as digital auctions, virtual shopping, and supplying chains.

As we have already stated, the intermediary is an entity which can act as a hub either in a client-server manner for connection with recipients, or can travel through the network as a mobile agent. In the latter case, the protocol with only one recipient could be used for instance when the IN is implemented as a mobile code object which visits each of the intended recipients (i.e., several executions of the same protocol are needed). On the other hand, if the IN reaches a network of intended recipients (e.g., ANX network or a network which connects different travel agents), the protocol with multiple recipients is more efficient. Let us describe a virtual shopping scenario where a customer (the originator) is willing to buy some products and for that purpose, it accesses an intermediary entity (see Figure 4.5).

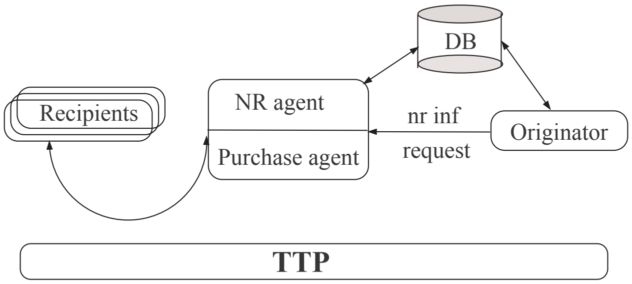

Fig. 4.5 Virtual Shopping

Using the *request* information contained in the first step of our protocols, the intermediary software may search for sellers and products, negotiate price, etc. Important data of the purchase (including banking account data and some purchase and product details) can be enclosed in the encrypted message c such that only the intended recipients have access to it. Then, the agent will redirect the purchase request to the sellers along with evidence of its participation ($EOOI$), and the sellers will reply with evidence of receipt to the agent ($EORc_i$). The agent should verify all evidence it received and store them in its evidence database. If the transaction is revocable, that is, payment can be cancelled, the customer does not need to check the sellers' evidence during the protocol execution.

After collecting the commitments ($EORc_i$) from the sellers, the agent will notify the customer of the list of sellers that have responded to the purchase request. The notification ($EORI$) also proves that the agent has done its work as expected by the customer, and the list will be used by the customer on the delivery of the key k to the sellers through the TTP. The TTP will verify the customer's request (Sub_k) before storing final evidence (Con_k) for downloading by transacting parties of the protocol.

Once the transaction ends, a simple light-weight electronic payment protocol can take place, since each party has got enough evidence for possible dispute resolution.

4.2.7 Requirements Fulfillment

Let us analyze the requirements fulfillment of our agent-based non-repudiation protocols as defined in Section 2.5.

- *Strong Fairness*: As it has been depicted along the description of protocols, each party is in possession of proper evidence and no party is in an advantageous position during a transaction even if the transaction is aborted. Because the TTP acts in a light-weight on-line manner, the communication channel from participants to the TTP has to be resilient.

  In Section 2.1, the concept of evidence of submission and delivery (EOS and EOD, respectively) was introduced. Referring to the TTP, the properties explained for EOS and EOD hold. Referring to the IN (because it acts as a delivery agent) $EORI$ serves at the same time as EOS and EOD and there is no assumption on the trust deposited over the IN nor in the channel requirements due to the help of the TTP.
- *Confidentiality*: This requirement is fulfilled by encrypting the decryption key with recipients' public key as agreed.
- *Efficiency*: The protocols are not optimistic, but the TTP acts in a light-weight manner (i.e., receiving information, processing it and storing digital evidence in a network accessible directory with read-only permissions).
- *Timeliness*: It fulfills synchronous timeliness by the use of a deadline.
- *Policy*: A policy for agent-based protocols needs to be defined targeted to each different application. The minimum elements it needs to contain are those explicitly mentioned through the description of the protocol. As an example, the policy needs to explicitly define the arbiter (or set of) to be used in a dispute resolution process.
- *Verifiability of TTP*: The TTP needs to be trusted, so it is not verifiable. On the other hand, the behavior of the IN is verifiable and can be disputed by the originator and the recipients with the help of an arbiter.

- *Transparency of TTP*: Because the TTP is lightly involved in every protocol execution, there is no transparency property. However, note that, in protocols presented in this section, transparency of the TTP is not a desired property as discussed at the beginning of Section 4.2.

4.3 Event-Oriented Simulation

In order to achieve high performance in a real non-repudiation service implementation, it is necessary to estimate timeouts, TTP features, publication key time, number of originators and recipients, and other relevant parameters we have been referring to along this book. We focus on a basic event-oriented simulation model for the estimation of these parameters. Based on the one-to-many Markowicth's protocol simulation model as a specific example, we have worked out various simulation experiments.

So far, as we have described, most of the (two-party or multi-party) non-repudiation protocols include diverse timeouts in their specifications. We have no reference about any proposed values or a procedure to estimate those timeouts. Due to the fact that these timeouts depend on real system conditions (e.g., network, involved parties, TTP capacity etc.), we propose the use of a simulation model in order to estimate approximated values of the timeout variable, and specially the relation among them.

We demonstrate, by means of a multi-party scenario example, how event-oriented simulation can be considered as a tool to estimate those timeouts, which can be adapted to the real conditions of each implementation. We select Kremer-Markowitch protocol [87] because it is the first multi-party extension and its events are similar to the protocol we designed in the first section of this chapter. We firstly describe the event-oriented simulation model specifications and entities, show modelled events and give different examples with this simulation model. Finally, we present and analyze the valuable results obtained. The work in this section has been partially published in [42].

4.3.1 Protocol

Since we have already explained this protocol in the previous chapter, we do not go into details. Nevertheless, its description is given in order to get an understanding on the parameters used afterwards.

- $l = h(M, k)$: label of message M and key k
- $EOO = S_O(feoo, R, l, t, c)$: evidence of origin
- $EOR_i = S_{Ri}(feor, O, l, t, c)$: evidence of receipt of each R_i
- $Sub_k = S_O\ (fsub, R', l, t, E_{R'}(k))$: evidence of submission of k to the TTP

- $Con_k = S_{TTP}$ *(fcon, O, R', l, t,* $E_{R'}(k))$: evidence of confirmation of *k* by the TTP.

The protocol is as follows.

1. $O \Rightarrow R \quad : R, l, t, c, EOO$
2. $R_i \rightarrow O \quad : O, R_i, l, EOR_i$ where$R_i \in R$
3. $O \rightarrow TTP : R', l, t, E_{R'}(k), Sub_k$
4. $R'_i \leftrightarrow TTP : O, R', l, E_{R'}(k), Con_k$ where$R'_i \in R'$
5. $O \leftrightarrow TTP : O, R', l, E_{R'}(k), Con_k$

The originator O multicasts to all recipients *R* the evidence of origin corresponding to the encrypted message *c* in step 1. Then, some recipients R_i (or all of them) send evidence of receipt EOR_i in step 2. In the next step, O sends *k* and evidence of submission Sub_k to the TTP in order to obtain evidence of confirmation Con_k in step 5. As authors assume that the communication channel between O and the TTP is not permanently broken, O will be eventually able to send *k* and Sub_k to the TTP in exchange for Con_k at any time before timeout t.

In step 4, each recipient R'_i fetches $E_{R'}(k)$ and Con_k from the TTP and stores them together with *EOO* as evidence to prove that message *M* was originated and sent by *O*; and the latter fetches Con_k from the TTP and stores it as evidence to prove that *k* is available to R'.

Timeout *t* constitutes one of the halt conditions in this protocol due to the fact that the TTP cannot publish the encryption key if it receives *k* some time t' after the timeout. Timeout *t* can be used to stop originator's key publication request sent to the TTP if the latter could not be connected before deadline t. Besides, the recipients should halt the protocol if the key has not been published after time t. Both halt conditions for *O* and *R* would avoid useless loops.

The estimation of this timeout *t* depends on the real features and conditions of the implemented scenario including number of originators and recipients, TTP capacity and the network speed. In the next section, we present the simulation model of the protocol described here.

4.3.2 Simulation Model

The following model is useful in order to estimate the timeout and to diagnose some possible problems in the implementation of the protocol, starting from different values of the critical system variables such as: low connection speed, shortage of TTP's storage capacity and delay in the messages due to firewall protection or other security schemes. This diagnosis could be used to model further improvements in the real scenario, to find better implementations, shortest waiting time and adequate TTP features. The model includes 15 different events (see Figure 4.6).

1. Originators send messages to recipients (event 1: message generation, event 2: message arrival to R).

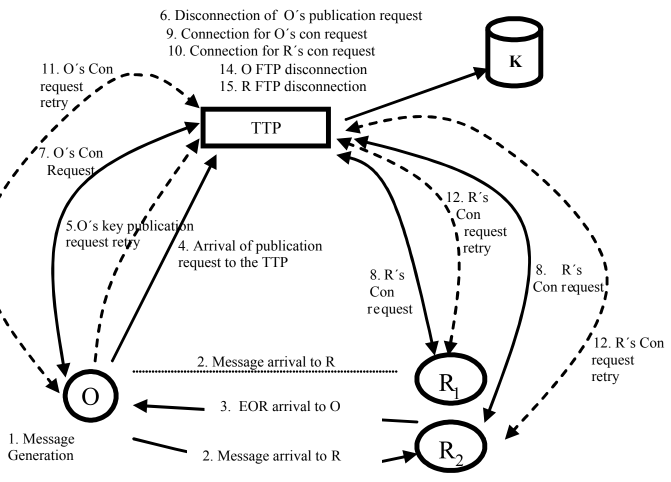
6. Disconnection of O´s publication request
9. Connection for O´s con request
10. Connection for R´s con request
14. O FTP disconnection
15. R FTP disconnection
K
TTP
11. O´s Con request retry
7. O´s Con Request
5.O´s key publication request retry
4. Arrival of publication request to the TTP
12. R´s Con request retry
8. R´s Con request
8. R´s Con request
12. R´s Con request retry
2. Message arrival to R
O
R1
3. EOR arrival to O
1. Message Generation
2. Message arrival to R
R2

Fig. 4.6 Simulation Events

2. These originators will wait for *EOR* and then send a key publication request to the TTP (event 3: *EOR* arrival to O). Note that in the simulation model, O will wait for all *EOR* in order to estimate the delay time when all R_i send *EOR* to O in a real execution, even though the protocol has been designed for terminating also when not all of the recipients reply. We use this model in order to simulate the worst case.
3. The TTP publishes the key if it has enough connection and storage capacity (event 4: arrival of the publication request to the TTP, event 6: disconnection of O's publication request). Obviously, in our model, there is no deadline check, since it is exactly the deadline time, the variable we want to estimate.
4. Otherwise, O should retry the request later (event 5: O's key publication request retry).
5. Once the key is published, the originator and the recipients can start Con_k requests (event 7: O's Con_k request, event 8: R's Con_k request).
6. If allowed by FTP server resources, the TTP opens a connection with the involved entity, verifies authenticity and integrity of the message and outputs an affirmative or negative response to the request (event 9: connection for O's Con_k request, event 10: connection for R's Con_k request, event 14: O's FTP disconnection, event 15: R's FTP disconnection).
7. If FTP resources are exhausted, the involved entity should retry the connection later (event 11: O's Con_k request retry, event 12: R's Con_k request retry).
8. The key is maintained in the TTP's database until its expiration (event 13: key deletion on the TTP).
9. When all involved entities have verified the key, a protocol execution has finished.

Note that network connections have been simulated with two events. This allows us to better represent each different state of the protocol as well as update of the variables (see tables below) used in the simulation.

In a real scenario, the TTP needs to process many protocol executions with similar or different originators. We could imagine several electronic bookshops (originators), during the whole day selling books [9] to many readers (and thus using a multi-party non-repudiation protocol).

Problem P1: Estimate the timeout t that O sends to R in the first step of the protocol according to a real scenario.

Using our simulation model, we have realized that it is important to reduce the key storage (expiration) time in the TTP when resources are limited, as well as to avoid unsuccessful confirmation requests while the number of originators and recipients increase or the TTP features (number of concurrent ftp available connections and storage capacity) change. It is relevant to note that the elimination of unsuccessful confirmation requests guarantees a fair execution of the protocol. Therefore, for this reason, another case is also studied.

Problem P2: Estimation of TTP efficient features without increasing the timeout t while keeping all Con_k requests successful.

Goals: Reduce the delay in the entire system while guaranteeing a complete and fair execution of the protocol. We need to find the influence of modifications on

- number of originators and recipients
- number of messages that the originators send to the recipients
- network speed [10]
- connection capacity to publish in the TTP (TTP throughput as seen by O)
- FTP capacity connection to the TTP server for retrieving Con_k evidence
- storage capacity in the TTP
- key availability time in the TTP
- time between successive retries of connections

We can model the protocol with event-oriented simulation [30] due to the fact that the generation and the reception of messages are asynchronous processes that evolve a finite number of events. Following (see Table 4.6 onwards) we present the entities of the simulation model and its variables.

[9] The stop criteria will be a finish time of the simulation event which is decided in advance.

[10] The network speed influences the different experimental distributions obtained for the delay of messages. In our experiment, we have used a unique value for network speed. If a different network speed is to be used, the delay distributions hold while the delay values change accordingly to the network speed variation.

Entity 1. Simulator (S)	
Variables	**Description**
Input variables	
FinalTime	Final simulation time
Recipients	Number of Recipients (R)
Originators	Number of Originators (O)
MsgGenDist	List of message generation distributions for each O (step 1)
CommunicationOR	Matrix of delay distributions of network messages between O and R (step 1)
CommunicationOTTP	List of delay distributions of network messages between O and the TTP (step 3)
CommunicationRTTP	List of delay distributions of network messages between R and the TTP (step 4)
EORsendDist	Delay distribution of the *EOR* message (step 2)
PUBConnectionDist	Time distribution of O's connection to publish the key in the TTP (step 3)
FTPConnectionDist	FTP connection time distribution of O and R (steps 4 and 5) for getting Con_k
State variables	
CurrentTime	Current simulation time
Lentity	List of entities
Levent	List of events

Table 4.6 Simulator Entity

Entity 2. Message (M): This entity is created by originators. Each originator is able to create many messages.	
Variables	**Description**
Input variables	
IdM	Unique identifier of message *M*
CreationTime	Creation time (step 1)
State variables	
State	States of the message: *St1*: It is being sent to R (step 1). *St2*: O is waiting for all EOR (step 2). *St3*: O is trying to publish the key in the TTP (step 3). *St4*: The key has been published in the TTP. *St5*: The key was deleted from the TTP.
Nbr_EOR	Number of R that sent EOR (step 2)
InitTime_O_Con	Initial time of O's Con_k request (step 5)
Output variables	
WaitRTime	Total waiting time for all EOR (step 2)
PubDelayTime	Key publication delay time (step 3)
DelayTime_R_Con	List of delay time of Rs Con_k requests (step 4)
Nbr_PUBRetries	Number of O's key publication request retries (step 3)

Table 4.7 Message Entity

Entity 3. ORIGINATOR (O)	
Variables	**Description**
Input variables	
Time_btw_PUBRetries	Time between successive retries of O's key publication request (step 3)
Time_btw_FTPRetries	Time between successive retries of O's Con_k request (step 5)
IdO	Originator's unique identifier
State variables	
LMsg	List of messages generated by O (step 1)
Nbr_Msg	Number of messages generated by O (step 1)
Output variables	
Nbr_PUBMsg	Number of published keys (step 3)

Table 4.8 Originator Entity

Entity 4. RECIPIENT (R)	
Variables	**Description**
Input variables	
Time_btw_FTPRetries	Time between successive retries of R's Con_k request (step 4)
IdR	Recipient's unique identifier
State variables	
LReceivedMsg	List of received messages (step 2)
Output variables	
Nbr_ReceivedMsg	Number of received messages (step 2)

Table 4.9 Recipient Entity

4.3.3 Main Model Simulation Events

We now describe the main events. We use *entity.variable* to refer to one variable of the entities (*S*, *M*, *O*, *R* and *TTP*). For each event, we describe the name and the input parameters between brackets, followed by the description of the event using a simple pseudo-language.

- ***Event 1: Message generation (O: originator)***

```
Generate a message at time t=S.CurrentTime
Increase O.Nbr_Msg
M.IdM = O.IdO+O.Nbr_Msg
M.CreationTime = S.CurrentTime
M.State = St1
For i = 1 to S.Recipients do
    Add the event Message arrival to R (O,M,$R_i$) at time
    t=S.CurrentTime+Random value generated with S.CommunicationOR(O,Ri)
Add M to the list O.LMsg
Add the event Message generation (O) at time
t=S.CurrentTime+Random value generated with S.MsgGenDist(O)
```

Entity 5. TTP	
Variables	**Description**
Input variables	
Max_StorageKTime	Key storage time in the TTP
CapacPUBConnection	Publication connection capacity (number of users)
CapacFTPConnection	FTP connection capacity (number of users)
CapacStorage	Storage capacity (number of keys)
State variables	
Current_ConnectedPUB	Current number of publishing connected entities
Current_ConnectedFTP	Number of FTP connected entities
Capac_Occupied	Occupied storage capacity
Output variables	
LPUBMsg	List of messages whose keys were published
Nbr_PUBMsg	Number of messages whose keys were published
Nbr_PUBRetries	Number of retries of O's key publication request caused by the lack of TTP connection capacity (step 3)
Nbr_PUBRetries_Str	Number of retries of O's key publication request caused by the lack of TTP storage capacity (step 3)
Nbr_O_Con_Retries	Total number of O's Con_k request retries (step 5)
Nbr_Successful_O_Con	Total number of successful O's Con_k requests (step 5)
Nbr_UnSuccessful_O_Con	Total number of unsuccessful O's Con_k requests (step 5)
Nbr_R_Con_Retries	Total number of all R_is Con_k request retries (step 4)
Nbr_Successful_R_Con	Total number of successful R's Con_k requests (step 4)
Nbr_UnSuccessful_R_Con	Total number of unsuccessful R's Con_k requests (step 4)

Table 4.10 TTP Entity

- ***Event 2: Message arrival to R (O: originator, M: message, R: recipient)***

```
Add the message to the list R.LReceivedMsg
Increase the number of received messages R.Nbr_ReceivedMsg
Add the event EOR arrival to O (M, R_i) at time
t=S.CurrentTime+Random value generated with S.CommunicationOR(O, R_i)+
Random value generated with S.EORsendDist
```

- ***Event 3: EOR arrival to O (M: message, R: recipient)***

```
Increase M.Nbr_EOR
If M.State=St1
    M.State=St2
If M.Nbr_EOR = S.Recipients
    M.State=St3
    Update M.WaitRTime= S.CurrentTime - M.CreationTime
    Add the event Arrival of the publication request to TTP (O, M, TTP) at time
    t=S.CurrentTime+ Random value generated with S.CommunicationOTTP(O)
```

- ***Event 4: Arrival of the publication request to TTP (O: originator, M: message, TTP: trusted third party)***

```
If TTP.Current_ConnectedPUB+1 > TTP.CapacPUBConnection
    Increase TTP.Nbr_PUBRetries
    Add the event O's key publication request retry (O,M) at time
    t = S.CurrentTime+O.Time_btw_PUBRetries
Else
    If TTP.CapacOccupied+1 >TTP.CapacStorage
        Increase TTP.Nbr_PUBRetries_Str
        Add the event O's key publication request retry (O,M) at time
        t = S.CurrentTime+O.Time_btw_PUBRetries
    Else
        Increase TTP.Current_ConnectedPUB
        Add the event Disconnection of O's publication request (O,M, TTP) at
time
        t = S.CurrentTime+Random value generated with S.PUBConnectionDist
```

- ***Event 5: O's key publication request retry (O: originator, M: message)***

```
Add the event Arrival of the publication request to TTP (O, M, TTP) at time
t = S.CurrentTime+ Random value generated with S.CommunicationOTTP(O)
```

- ***Event 6: Disconnection of O's publication request (O: originator, M: message, TTP: trusted third party)***

```
Update M.PubDelayTime=S.CurrentTime - M.CreationTime
Increase O.Nbr_PUBMsg
Increase TTP.Nbr_PUBMsg
Add the message to the list TTP.LPUBMsg
Increase TTP.CapacOccupied
Decrease TTP.Current_ConnectedPUB
M.State=St4
Add the event O's Con request(M) at time
t = S.CurrentTime
Add the event R's Con request(M) at time
t=S.CurrentTime for each recipient i.
Add the event Key deletion in the TTP (TTP, M) at time
t = S.CurrentTime+TTP.Max_StorageKTime
```

- ***Event 7: O's Con request (M: message)***

```
Add the event Connection for O's Con request (O,M,TTP) at time
t = S.CurrentTime+ Random value generated with S.CommunicationOTTP(O)
```

- ***Event 8: R's Con request (M: message)***

```
Update $M.DelayTime\_R\_Con[R_i]=S.CurrentTime$
Add the event Connection for R's Con request ($R_i$,M,TTP) at time
$t = S.CurrentTime+$ Random value generated with $S.CommunicationRTTP(R_i)$
```

- ***Event 9: Connection for O's Con request (O: originator, M: message, TTP: trusted third party)***

```
If $TTP.Current\_ConnectedFTP+1 > TTP.CapacFTPConnection$
    Increase $TTP.Nbr\_O\_Con\_Retries$
    Add the event O's Con request retry (M) at time
    $t = S.CurrentTime+O.Time\_btw\_FTPRetries$
Else
    Increase $TTP.Current\_ConnectedFTP$
    Add the event O's FTP disconnection (O,M,TTP) at time
    $t = S.CurrentTime+$Random value generated with $S.FTPConnectionDist$
```

- ***Event 10: Connection for R's Con request (R: recipient, M: message, TTP: trusted third party)***

```
If $TTP.Current\_ConnectedFTP+1 > TTP.CapacFTPConnection$
    Increase $TTP.Nbr\_R\_Con\_Retries$
    Add the event R's Con request retry (M) at time
    $t = S.CurrentTime+R.Time\_btw\_FTPRetries$
Else
    Increase $TTP.Current\_ConnectedFTP$
    Add the event R's FTP disconnection ($R_i$,M,TTP) at time
    $t = S.CurrentTime+$Random value generated with $S.FTPConnectionDist$
```

- ***Event 11: O's Con request retry (M: message)***

```
Add the event Connection for O's Con request (O,M) at time
$t = S.CurrentTime+$Random value generated with $S.CommunicationOTTP(O)$
```

- ***Event 12: R's Con request retry (M: message)***

```
Add the event Connection for R's Con request ($R_i$,M) at time
$t = S.CurrentTime+$Random value generated with $S.CommunicationRTTP(R_i)$
```

- ***Event 13: Key deletion in the TTP (M: message, TTP: trusted third party)***

```
Change the state of the message $M.State=St5$
Decrease $TTP.CapacOccupied$
```

- ***Event 14: O's FTP disconnection (O: recipient, M: message, TTP: trusted third party)***

```
If M is in the list TTP.LPUBMsg and M.State=St4
    Increase TTP.Nbr_Successful_O_Con
Else
    Increase TTP.Nbr_Unsuccessful_O_Con
Decrease TTP.Current_ConnectedFTP
```

- ***Event 15: R's FTP disconnection (R: recipient, M: message, TTP: trusted third party)***

```
If M is in the list TTP.LPUBMsg and M.State=St4
    Increase TTP.Nbr_Successful_R_Con
Else
    Increase TTP.Nbr_Unsuccessful_R_Con
Update M.DelayTime_R_Con[Ri]= S.CurrentTime - M.DelayTime_R_Con[Ri]
Decrease TTP.Current_ConnectedFTP
```

Main Program

```
Initialization of Simulator (S)
- Add all entities to the simulator
- Initialize variables (input, state, output)
- Generate the first events of Message Generation(O) for each O
While not empty S.LEvent and S.CurrentTime < S.FinalTime do
- E = The minimum time event in S.LEvent [11]
- Delete E from S.LEvent
- S.CurrentTime = time of E
- Execute the procedure that handles the event
Do the report
- For each entity save the report
```

4.3.4 Output Analysis

In order to check our simulation model, we have implemented an example of the described protocol in a standard LAN network. Therefore, the experimental results refer only to this kind of topology that were performed for studying the relationship among values obtained and not for the sake of the values themselves.

[11] The first one if it is implemented as an ordered list.

Nevertheless, we stuck as much as possible to reality and, for instance, in order to calculate the time recipients delayed in replying *EOR*, we implemented a small application which received a message in one socket, extracted it, verified and signed it using RSA. The software used for monitoring packets in the network was *WhatsUp* [3].

The originators send messages to the recipients with a uniform distribution between $^1\!/_2$ and 1 hours (*S.MsgGenDist*). After a hundred executions of the protocol, sampling the network delays, we observed the following input distributions of the model.

- The network message delay distribution among participating entities is a uniform distribution between 10ms and 17ms (*S.CommunicationOR, S.CommunicationOTTP, S.CommunicationRTTP*).
- The delay distribution of the *EOR* reply is a uniform distribution between 15ms and 20ms (*S.EORsendDist*).
- The time distribution of O's connection to publish the key is a uniform distribution between 30ms and 50ms (*S.PUBConnectionDist*).
- The FTP connection time distribution of the originators and the recipients for getting the evidence Con_k follows a uniform distribution between 25ms and 35ms (*S. FTPConnectionDist*).

For estimating the distribution type over the time values sampled, we used the test of Kolmogorov-Smirnov [126] for proving its uniformity. This test says that being

- N: samples number
- X_i: expected frequency in the interval [0,xi] (being xi one of the delimiters of the interval) according to the uniform distribution.
- f_i: frequency observed in the interval [0,xi] in the samples object to study

$$max|X_i * N - f_i| \tag{4.1}$$

gives us the value for calculating how uniform is the sample set we have. The closer it is to 0, the closest we are to the uniform distribution.

In summary, we estimated the needed parameters for providing answers to problems P1 and P2 with two sets of experiments. In each experiment, we give values to the input variables and analyze the results obtained. The input and output variables match those ones described for each entity in the simulation model, but with abbreviated names for sake of readability. The values chosen for the input variables are traded-off between those ones in a real scenario, and those ones which allow us to study its influence in the overall protocol. Although most of the results are easily foreseen, it is worth to understand the relation among them.

A. Input variables:

1. **NO** – Number of originators (*S.Originators*)
2. **NR** – Number of recipients (*S.Recipients*)
3. **SC** – TTP's storage capacity measured in number of keys (*TTP.CapacStorage*)

4. **FTP** – FTP connection capacity (*TTP.CapacFTPConnection*) and publication connection capacity (*TTP.CapacPUBConnection*)
5. **TS** – Key storage time in the TTP (*TTP.Max_StorageKTime*)
6. **RO** – Time between successive retries of O's Con_k request (*O. Time_btw_FTPRetries*)
7. **RR** – Time between successive retries of R's Con_k request (*R. Time_btw_FTPRetries*)

B. Output variables:

1. **NM** – Number of generated messages in the experiment $\sum_{i=1}^{NO} Oi.Nbr\_Msg$
2. **MP** – Number of messages whose keys were published on the TTP (*TTP.Nbr_PUBMsg*)
3. **CPC** – Number of successive retries of O's key publication request caused by the lack of TTP connection capacity (*TTP.Nbr_PUBRetries*)
4. **CPA** – Number of successive retries of O's key publication request caused by the lack of TTP storage capacity (*TTP.Nbr_PUBRetries_Str*)
5. **CRO** – Number of successive retries of O's Con_k request (*TTP.Nbr_O_Con_Retries*)
6. **CRR** – Number of successive retries of R's Con_k request (*TTP.Nbr_R_Con_Retries*)
7. **SO** – Number of successful O's Con_k requests (*TTP.Nbr_Successful_O_Con*)
8. **SR** – Number of successful R's Con_k requests (*TTP.Nbr_Successful_R_Con*)
9. **UO** – Number of unsuccessful O's Con_k requests (*TTP.Nbr_UnSuccessful_O_Con*)
10. **UR** – Number of unsuccessful R's Con_k requests (*TTP.Nbr_UnSuccessful_R_Con*)
11. **ERT** – Average waiting time of all *EOR*

$$\frac{\sum_{i=1}^{NO}\left(\frac{\sum_{j=1}^{Oi.Nbr\_Msg} Mj.WaitRTime}{\text{Oi.Nbr\_Msg}}\right)}{NO}$$

12. **PKT** – Average key publication delay time

$$\frac{\sum_{i=1}^{NO}\left(\frac{\sum_{j=1}^{Oi.Nbr\_Msg} Mj.PubDelayTime}{\text{Oi.Nbr\_Msg}}\right)}{NO}$$

C. Results:

Recall that the problem P1 consists of estimating the timeout t that O sends in its first message to the recipients (i.e., the time at which the key must be published in the TTP's directory). The most intuitive solution is to set this deadline as far as possible from the protocol start. Nevertheless, this is not practical, as this means all entities need to wait a long time before reaching a protocol finish state. On the other hand, this timeout cannot be too short, since the originator will not have time

	Input variables						
	NO	**NR**	**SC**	**FTP**	**TS**	**RO**	**RR**
A	300	30	10500	9000	1min	20s	20s
B	5000	30	10500	65000	2min	20s	20s
C	10000	10	10500	9000	1min	20s	20s

Output variables							
NM	**MP**	**CPC**	**CPA**	**CRO**	**CRR**	**SO**	**SR**
4672	4669	0	0	0	0	4668	140041
76885	76833	0	0	0	0	76816	2304481
157850	157775	2000	0	0	0	157739	1577370

Table 4.11 Input Test for P1

UO	UR	ERT	PKT
0	0	10.75s	50.85s
0	0	11.93s	51.97s
0	0	10.50	60.20s

Table 4.12 Output Results for P1

to publish the key *k*. Estimating the average time for O to publish the key, we will have a good approximation for this deadline. Therefore, PKT is the value we will inspect in detail in the first set of experiments (see Table 4.12).

- (**A**) 50.85s with 300 originators and 30 recipients. In this simulation of the protocol, the originator would not wait more than 10.75s for all *EOR* in order to send the key publication request to the TTP. With the established conditions on the network delay, 10500 keys space for store, 9000 simultaneous connections available in the TTP FTP server, 1 min. of Con_k availability in the TTP directory and a 20s delay when a retry is needed, O needs approximately 40s for publishing the key once it obtained all EOR.
- (**B**) 51.97s with 5000 originators and 30 recipients. The originator would not wait more than 11.93s for all *EOR* in order to send the key publication request to the TTP. In this example, although the number of originators and time needed for the key to be published in the TTP FTP server has been augmented, the relation is maintained.
- (**C**) 60.20s with 10000 originators, 10 recipients. The originator would not wait more than 10.50s for all *EOR* in order to send the key publication request to the TTP. The decrease on this parameter is due to the reduction in the number of recipients. An increment in the number of participants resulted in an increment in the PKT. The TTP needs to accept more publication connections. Note that, in this example, there will be 10000 originators and the connection maximum capacity for publication has been input as 9000. As we can see (CPC), originators need to retry key's publication in 2000 occasions (note that each originator in our simulation sends several messages, that is, it executes the protocol several times) because TTP's server is overloaded, and this is the reason for the PKT delay.

The following conclusions from the first set of experiments can be deduced.

- As expected, we deduce that originator retries become the main reason for the increase in the time needed for publishing the key (sample C).
- There are no retries of the key's publication request due to lack of capacity in the TTP (CPA) in none of the samples. This is due to an initial bigger keys capacity for the number of originators.

  As time distribution for the generation of messages is [30-60 mins.], when a new execution of the protocol by each originator comes, all previous ones have been finished, and thus the TTP server has been cleaned. This is a very important conclusion for optimization of TTP space, but could at the same time bring into confusion to the designer. In our model, disconnection (failure) of the network is not foreseen, and thus 1 minute is enough for all entities to collect final evidence. Therefore, and very importantly, when implementing a real model, maximum disconnection time (hours, days) will be the only important parameter for calculating key deletion time (TS).

  Furthermore, in our model, all originators start their protocols at the same time and then follow the uniform message distribution. This makes the execution to occur in batches. In a real world, executions of different originators will overlap and then, space and throughput in the TTP needs to be reserved (according to the number of originators (NO) and their expected message distributions) for avoiding public key publication retries due to lack of capacity and connection (CPC and CPA).
- The reason for which there are not successive retries from participants in obtaining final evidence Con_k, is also due to connection parameters chosen in the TTP. These reasons, for which a participant can be denied accessing final evidence Con_k are

  1. Final evidence Con_k is not in the TTP sever; i.e., it has already been deleted by the TTP or not published yet. For the second case, as we implemented the simulation model, the participants did not originate retries, because they only try after deadline t, that by definition is the time by which the final evidence and key must be published (sent by the originator). In the first case, an unsuccessful Con_k request (UO and UR) is generated. Nevertheless, in this first set of experiments, TS is long enough for not producing unsuccessful Con_k requests.
  2. TTP FTP server is overloaded and the user is refused from connecting. This case does not happen in our samples, since only in example **C** occurs that the connection capacity is less than the maximum number of entities which can try to connect in the same instant in order to get final evidence Con_k (FTP<(NO+NR)).

     Nonetheless, as we can see in CPC, some originators are delayed while waiting for a public key publication retry and a question comes up: why is there no delay (retries) in final evidence Con's accesses from participants?

     The answer is the difference existing between RO (time between successive retries of O's key publication requests, i.e., O.Time_btw_PUBRetries) and the distribution which dictates the speed with which recipients and originators access the TTP for Con_k retrieve once the evidence is published (S.CommunicationRTTP(R_i) and S.CommunicationOTTP respectively) which is between 10 and 17 ms. This makes that, while some originators are still waiting for a public key publication retry, and possibly not for the first time, the rest of participants are accessing their final evidence, so TTP FTP server queue is empty when late final evidences are published and intended parties access to retrieve Con_k.

- The timeout (t) estimated is less than 1 minute with the delay time distributions and the message generation distribution selected. The latter could be reduced several times without affecting the timeout estimation even if disconnection of some participants is foreseen. Thus, if the system is initialized from scratch, only the capacity (SC) and connection (FTP) throughput in the TTP will influence the value we obtain in the deadline time [12].

We can do other experiments as well, like the estimation of efficient initial conditions (**SC, FTP, TS, RO, RR**), so that the protocol would operate without unsuccessful Con_k searches with a fixed number of originators and recipients. Obviously, these adjustments can help in the decision-making of a TTP investment process. Bearing this in mind, we have performed different tests (see Table 4.13 [13]) that have helped us to obtain good results for problem **P2**, and efficient conditions for the protocol operation.

1. **Test 1 (A,B)**: In this test, we used a small increment for values of SC and TS, and this has resulted in small changes in the unsuccessful Con_k requests (UO, UR). In this case, we can observe that the number of O's Con_k publication retries due to lack of space in the TTP (CPA) is reduced to 0 in sample B, obviously as a consequence of the increment in the TTP capacity (from 100 to 450, which is enough for the number of originators that try to publish a key at the same moment of time). It is also important to stress the fact that the unsuccessful Con_k requests appear as a consequence of the low value used for TTP's connection capacity (FTP) which, on the other hand, is the main parameter that ultimately explains a refuse from the TTP when trying to access final evidence for all participants.
2. **Test 2 (D,E)**: In this test, we increased the values of SC, FTP and TS. The result is that we get a significant reduction of UO and UR. CPC is reduced to 0 as a result of the considerable increment in the publication connection capacity in the TTP. There are still however a number of retries when accessing the final evidence even for the originators which reached the point in which all final evidence is accessed (UO=0).

[12] Of course, with fixed delay time distributions.

[13] Note that it could seem that there are errors in the table, since not always holds that MP-SO=UO, i.e, the number of unsuccessful O's Con_k requests equals the number of correctly published messages minus the number of successful O's Con_k requests. This is due to the way in which the simulation is stopped: a fixed final time. This avoids originators to access some final evidence after publishing the keys simply because the simulation ends, and thus, no unsuccessful request is even generated.

3. **Tets 3 (F):** In this test, we increased the value of TS and obtained a reduction of UR. The key has been published in one hour, which is reasonable from a practical point of view. When analyzing the results, we deduct that an increment in the capacity of ftp connections (FTP) is necessary to obtain better results.
4. **Test 4 (G)**: In this test, we increased FTP. The result is that the unsuccessful Con_k requests become 0. Additionally, the number of retries of Con_k request becomes 0 too. This guarantees a better execution of the protocol in a real scenario because of the reduction in the number of messages in the network.
5. **Test 5 (H,I,J):** In this test, we performed some estimations of the appropriate TS value. The result is that we get the optimal solution with a value of TS$\simeq$50 seconds. Note that retries on Con_k final evidence access only occurs when TTP's connection throughput is overloaded. When this is not the case, i.e., the participant entity cannot access the key because it has been deleted from the server (or it has not been published), an unsuccessful request is generated.
6. **Test 6 (K):** In this test, we decreased the FTP value. The result is that we get unsuccessful Con_k request retries again, which proves the most appropriate value for FTP is 9000 keys capacity.

	Input variables						
	NO	**NR**	**SC**	**FTP**	**TS**	**RO**	**RR**
A	300	30	100	140	1/2min	20s	20s
B	300	30	450	140	2,5min	20s	5s
D	300	30	3500	3000	5min	20s	5s
E	300	30	4000	3000	20min	20s	5s
F	300	30	4000	3000	1h	20s	10min
G	300	30	1500	9000	30min	20s	1min
H	300	30	1500	9000	1min	20s	20s
I	300	30	1500	9000	1/2min	20s	20s
J	300	30	1500	9000	50s	20s	20s
K	300	30	1500	8000	1/2min	20s	20s

Output variables							
NM	**MP**	**CPC**	**CPA**	**CRO**	**CRR**	**SO**	**SR**
4712	4706	388	4895	399	80388	4701	121540
4720	4718	869	0	353	80502	4715	125108
4628	4621	0	0	122	7985	4620	133068
4655	4652	0	0	116	9239	4652	133461
4593	4587	0	0	99	6523	4587	132864
4734	4733	0	0	0	0	4732	141930
4672	4669	0	0	0	0	4668	140041
4737	4734	0	0	0	0	4730	141916
4646	4641	0	0	0	0	4639	139140
4706	4703	0	0	12	378	4684	140498

Table 4.13 Input Test for P2

An essential issue for the best operation of non-repudiation protocols is to figure out their timeouts. We proposed a simulation model for this purpose, since timeouts

UO	UR	ERT	PKT
5	15051	10.67s	72.96s
3	11026	10.79s	54.77s
0	5502	10.75s	51.10s
0	6099	10.66s	50.40s
0	4746	10.72s	50.77s
0	0	10.62s	50.86s
0	0	10.75s	50.85s
3	74	10.75s	50.56s
0	0	10.85s	50.94s
17	532	10.77s	50.87s

Table 4.14 Output Results for P2

depend on specific scenario features such as network speed, TTP characteristics, number of originators and recipients, etc. This simulation is very useful for a reliable and adequate implementation. Furthermore, we have proposed an estimation of the appropriate values of parameters for an efficient use of a TTP (**P2**) in non-repudiation protocols.

This simulation model could be extended to other security protocols in two-party and multi-party scenarios. We provided important insights on the relation existing among the parameters to be used in the implementation of a multi-party non-repudiation protocol by means of event simulation results.

Part III
Applications

Chapter 5
Multi-Party Non-repudiation Applications

Abstract The previous chapter introduced new approaches for MPNR protocols targeting efficiency as one of the properties to improve. For such a reason, it provided a comparison with existing solutions and explored an event-directed model for estimating time values. Although Chapter 6 will demonstrate that these general approaches can be adapted to real applications and frameworks as a value-added service [1], the present chapter addresses more specific multi-party solutions in which non-repudiation is implicit or explicitly an integral requirement.

Beginning with an existing solution for CEM, it is enhanced with asynchronous timeliness and multicasting, thus allowing certified and timely notification of multiple users without introducing significant complexity, and overall, preserving the properties for a CEM application. Then, we introduce the problem of multi-party contract signing with an analysis and new solutions. A new synchronous MPCS protocol that reaches the lowest bound of steps is presented. We further consider additional features like abuse-freeness and threshold timeliness.

5.1 Multi-Party Optimistic Fair CEM

With the experience acquired in the previous multi-party descriptions, this section analyzes and provides extensions to a fast and simple optimistic fair CEM protocol [102] for achieving timeliness and participation of multiple recipients.

Definition 5.1. A certified electronic mail (CEM) application is a digital mail transaction in which a recipient gets its intended message if and only if the sender obtains evidence of receipt of that message.

[1] Although we have stressed that non-repudiation is a basic security service as defined by the ITU, the current reality is that very few (even secure-named) applications implement it.

5.1.1 A Protocol for Fair CEM with Deadline Time

Although Micali explained different fair CEM protocols in [102], the most complete one that supports confidentiality, fairness and synchronous timeliness is described here. Because it is an optimistic protocol, if both parties behave honestly, the TTP (which is also referred as the post office PO here) will not be involved. Each user in the system has a unique identifier. Before sending the plaintext message M to the recipient, the originator computes a secret Z protected with the TTP's encryption key as $Z = E_{PO}(O,R,E_R(M))$. To reach timeliness, Micali proposed a deadline time solution, where the originator chooses a time t after which the TTP should not help the recipient in the conclusion of the protocol.

1. $O \to R : t, Z, S_O(t,Z)$
2. $R \to O : S_R(Z)$
3. $O \to R : E_R(M)$

Whenever R reaches step 1 and verifies O's signature, it must extract the deadline time t and estimate whether it will have enough time to contact the PO in case of O's misbehavior or channel failure. Variable t_D denotes the maximum possible time discrepancy that R believes may exist between his clock and that of the PO. If R receives step 1 in time t_R (i.e., recipient's local time) such that $t_R + t_D$ is greater than or equal to t, then R halts; otherwise it proceeds to step 2. After verifying R's signature, O sends the message M to the recipient at step 3.

After replying at step 2, if the recipient does not get the message within a reasonable amount of time, or $Z = E_{PO}(O,R,E_R(M))$ does not hold, R contacts the PO in a *resolve* sub-protocol.

$1'$. $R \to PO$: $t, Z, S_O(t,Z), S_R(Z)$
$2'$. IF ($t_{PO} < t$) AND (valid signatures)
    $PO \to R$: X
    $PO \to O$: $S_R(Z)$

In this sub-protocol, the PO verifies whether R's request arrives before O's deadline time and also whether both signatures are correct. If so, the PO decrypts Z with its private key and, if the result is a triplet consisting of O, R, and an unknown string X, it sends X to the recipient and forwards R's signature to the originator.

Note that the semantic of the deadline t is different from the previous deadlines we have used before, where clock shifts between the entities could not lead to a security problem, but to a timeliness problem. Even though the deadline parameter is used for finalization of the protocol and consequently for timeliness, the meaning is different due to the optimistic approach. When the TTP participates in an on-line manner, the deadline establishes a point after which the protocol is resolved and the recipients can contact the TTP. In this case, the deadline indicates the point before which the recipients must contact the TTP if needed, because afterwards it will not help (and thus fairness will be damaged).

It can be argued that, at the end, both solutions need a point of time in which the TTP cannot help any more, and if recipients do not access in advance, fairness will be also damaged. Nevertheless, there is a very important difference in both approaches which will be better appreciated in Figure 5.1.

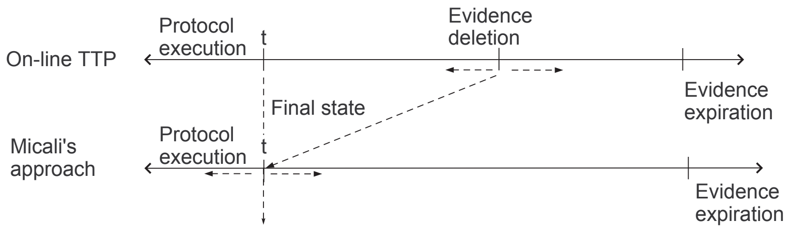

Fig. 5.1 Deadline Time Intervals

In the previous approaches, when evidence deletion occurs at the TTP, the latter will not be able to help entities any more. This event can coincide with evidence expiration or not depending on TTP's storage resources. Nevertheless, time (deadline *t*) at which the state is final (synchronous timeliness) and evidence deletion are different. In this way, entities can know beforehand when the protocol has a definitive state and then, if needed, seek TTP's help. On the other hand, Micali's solution unifies deadline and evidence deletion (there is no deletion event, but the meaning is the same; i.e., TTP's help is not available). If this deadline time is selected too early, mail recipients could not have time enough to resolve the protocol. If greater, entities will need to wait in order to know the final state of the protocol, perhaps more than what is desired.

5.1.2 Extension to Fair CEM with Asynchronous Timeliness

A deadline time is not the best solution for a fairness property in CEM applications. Thus, in this section a different solution, *asynchronous timeliness* (i.e., either party can finish the protocol at any time without loss of fairness) is proposed. In Micali's proposal, even if the recipient approximately calculates in each run the time to contact the PO, there can be always a situation in which the PO is unaccessible for longer. In such a case, the recipient will not get the item from the (un-)fair exchange protocol and it will be difficult to figure out who bears the responsibility for the breach of fairness.

Nevertheless, as mentioned in Definition 3.5, there will always exist advantages and disadvantages in both timeliness solutions (asynchronous vs. synchronous). At the same time, in Micali's protocols, the TTP (or PO) is held responsible for distributing final evidence to participants when requested. It is important to remove this requirement.

A new *cancel* sub-protocol is introduced. With the *resolve* sub-protocol, the recipient will be able to finish the protocol at any time. In this way, if O does not want to wait for the resolution of the protocol it can abort it with the *cancel* sub-protocol at any time too. The revised *main* protocol, in which O's signature [2] is not needed anymore in step 1 due to the omission of the deadline, is as follows.

1. $O \rightarrow R : Z$
2. $R \rightarrow O : S_R(Z)$
3. $O \rightarrow R : E_R(M)$

The revised *resolve* sub-protocol which will be requested by the recipient under the same conditions as the original one is as follows.

$1'.\ R \rightarrow PO : Z, S_R(Z)$
$2'.\ R \leftrightarrow PO :$ IF cancelled THEN $S_O(cancel, Z)$
ELSE $E_R(M)$

When the PO receives such a request, it first checks R's signature on Z. If valid, the PO further decrypts Z and extracts the identities of sender and recipient of Z. If R is the intended recipient of Z and the exchange has not been cancelled by O, the PO marks the exchange status related to $(O, R, h(Z))$ as resolved, sends $E_R(M)$ to R, and stores $S_R(Z)$ (which will be collected by O when it initiates the *cancel* sub-protocol). If the exchange has been cancelled by O, the PO forwards $S_O(cancel, Z)$ to R, and R can use this evidence to prove that O has cancelled the exchange.

The new *cancel* sub-protocol is as follows.

$1'.\ O \rightarrow PO : Z, E_{PO}(S_O(cancel, Z))$
$2'.\ O \leftrightarrow PO :$ IF resolved THEN $S_R(Z)$
ELSE *ack*

When the PO receives such a request from the originator, it first checks O's signature after decrypting with its corresponding asymmetric key O's cancel request [3]. If valid, the PO further decrypts Z and extracts the identities of sender and recipient. If O is the sender of Z and the exchange status related to $(O, R, h(Z))$ is marked as resolved, the PO forwards $S_R(Z)$ to O. Otherwise, the PO marks the exchange status related to $(O, R, h(Z))$ as cancelled, and acknowledges O's cancellation.

Because there is no deadline time for the recipient, the originator can abort the protocol as desired. As we can see, the PO provides access to the items that any of the parties needs. The parties can access the data at any time and it is not the responsibility of the PO whether the users get the messages they expect. The PO is not stateless and needs to maintain (within a reasonable amount of time) a table with entries $(O, R, h(Z), state)$ and to store the signatures for serving the users. Therefore, the server implementing the PO must secure this information (although confidentiality is not needed).

[2] In CEM applications, only non-repudiation of receipt is mandatory.

[3] As pointed out in [64], O needs to encrypt its own signature in order to avoid an attack which allows the recipient to get the cancel token (eavesdropping O's request from message $1'$) when the protocol has been previously resolved by the recipient (and thus making the originator believe the exchange succeeded).

Although in [102] there is no explicit definition of the dispute resolution process, it is, in general, necessary for any protocol generating evidence. In this process, both parties must agree which arbitrator will evaluate the final outcome of the protocol based on the evidence provided by the users. Consequently, if the recipient denies having received a message M in a CEM protocol run, the originator should provide $(Z, S_R(Z))$ and the arbitrator settles that the originator sent the message M to the recipient if

- $Z = E_{PO}(O, R, E_R(M))$ holds;
- the recipient's signature on Z is valid;
- the recipient cannot provide $S_O(cancel, Z)$.

It is easy to notice that integrity of the message is not ensured. Any external attacker (or R itself) can create a message $Z' = E_{PO}(O, R, E_R(M'))$. In this case, the originator will obtain a proof for a message it did not send and will try to cancel the protocol. If it is not possible (R already resolved the protocol), the recipient will have a message the originator did not send but cannot demonstrate it (the process is anonymous because there are no digital signatures). So this protocol does not ensure the service, but it does ensure that the recipient can read the message the originator sent if and only if the originator obtains evidence of receipt. If integrity and authenticity of the message for ensuring the service are properties needed by the originator, then it needs to digitally sign Z at step 1 and the receiver needs to provide O's signature at step 1' of the resolve sub-protocol for PO's verification.

5.1.3 Extension to Multi-Party Fair CEM

There are scenarios in which the participation of multiple entities can result in an important improvement. CEM is one of them. We can easily figure out applications in which sending e-mail to several users is feasible and useful. In Section 3.1 several solutions for multi-party CEM were pointed out. Based on Micali's CEM protocol, a new feature is introduced such that the sender can distribute in a certified manner a message M to several recipients. Some additional notation used in the description of this new multi-party fair CEM protocol (CEM-ZOL) is as follows.

- R : a set of intended recipients.
- $Header$: a header indicating in which position the recipient has to look for its information (e.g. $[R_i, j]$).
- M : message being sent from the originator to the recipients R.
- $R'' = R - R'$: a subset of R (in plaintext) with which the originator wants to cancel the exchange.
- $R''\_resolved$: a subset of R'' that have resolved the exchange with the *resolve* sub-protocol.
- $R''\_cancelled = R'' - R''\_resolved$: a subset of R'' with which the exchange has been cancelled by the PO.

- $u_R = u_{R_1}, u_{R_2}, \ldots$: concatenation of public keys from group R.
- $E_R(M)$: a group encryption of M for group R.
- $Z = E_{PO}(O, R, E_R(M))$: a secret Z protected with the PO's encryption key.

The PO will participate, if requested by any entity, in a mutually exclusive way (i.e., atomic execution of the sub-protocols for each user). Here, the CEM-ZOL protocol is described (see Figure 5.2).

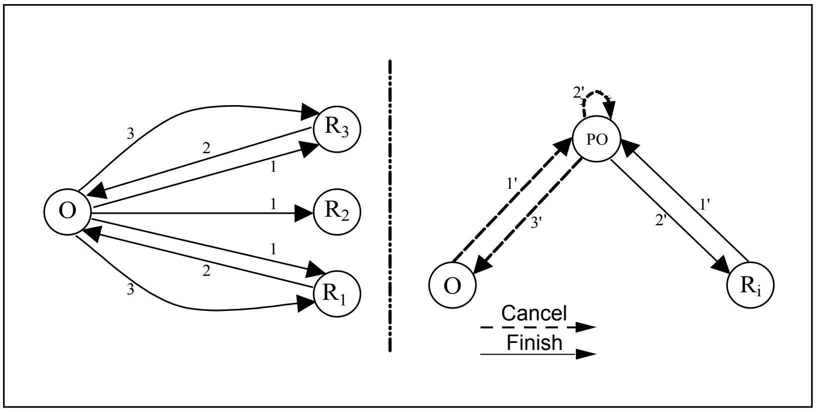

Fig. 5.2 CEM-ZOL Protocol

The *main* protocol executed by the final entities is

1. $O \Rightarrow R$: $Header, u_R, Z$
2. $R_i \rightarrow O$: $S_{R_i}(u_{R_i}, Z)$ where each $R_i \in R$
3. $O \Rightarrow R'$: $E_{R'}(M)$

In step 1, the originator sends the secret Z and all the recipients' public keys such that if any recipient does not agree with its public encryption key (e.g., the corresponding public key certificate has been revoked), then it stops the protocol. Otherwise, after verifying the data obtained, the recipient sends evidence of receipt to the originator at step 2, and the latter sends the (encrypted) message M at step 3 to the set of recipients who replied.

If the originator did not receive a correct message 2 from some of the recipients R'', it may initiate the following *cancel* sub-protocol.

```
1'. $O \rightarrow PO$ : $Z, R'', S_O(cancel, R'', Z)$
2'. $PO$          FOR (all $R_i \in R''$)
                  IF ($R_i \in R''\_resolved$) THEN retrieves $S_{R_i}(u_{R_i}, Z)$
                  ELSE appends $R_i$ into $R''\_cancelled$
3'. $O \leftrightarrow PO$ : all retrieved $S_{R_i}(u_{R_i}, Z)$, $S_{PO}(R''\_cancelled, Z)$
```

In this case, the originator communicates to the PO its intention of revoking the protocol with entities contained in R''. After verifying the originator's cancel request, the PO checks which entities previously resolved the protocol and gets their proofs of receipt. Then, the PO generates an evidence of cancellation for the rest of entities and includes everything in a message destined to the originator.

If some recipient R_i did not receive the message 3 or it was not valid, R_i may initiate the following *resolve* sub-protocol.

$1'.\ R_i \rightarrow PO : Header, u_{R_i}, Z, S_{R_i}(u_{R_i}, Z)$
$2'.\ R_i \leftrightarrow PO :$ IF ($R_i \in R''\_cancelled$) THEN $S_{PO}(R''\_cancelled, Z)$
ELSE $\{E_{R_i}(M)$
appends R_i into $R''\_resolved$ and stores $S_{R_i}(u_{R_i}, Z)\}$

The recipient sends to the PO all the information that it has already got from the originator along with its evidence of receipt. If this entity does not belong to the group of entities with which the originator cancelled the exchange, the PO verifies all the information (digital signatures) and decrypts Z. It also stores $S_{R_i}(u_{R_i}, Z)$. Note that if the protocol has been cancelled, it should be impossible for the recipient to cheat the PO in a way that the PO reveals the message for that protocol run. For that reason, the PO must verify the recipient's signature as well as integrity of Z in the first step.

Note that the recipient can cheat the TTP by manipulating the $Header$ field. Nevertheless, this will not disclose any other message, since it is encrypted with the intended recipient's public key. Furthermore, this action is nonsense due to two reasons: (i) message M is similar for every recipient regardless of the position in $E_R(M)$, and (ii) R_i will bring itself to an unfair situation since it will not be able to get the message while the originator can obtain from the TTP its receipt signature.

If this entity belongs to the group of entities with which the originator cancelled the exchange, the PO sends a cancellation evidence to the recipient such that the latter can easily demonstrate to an arbitrator that the exchange was cancelled in case a dispute arises.

If R_i denies having received M, the originator can present evidence $Z, E_R(M)$, $S_{R_i}(u_{R_i}, Z), S_{PO}(R''\_cancelled, Z)$, and the arbitrator settles that the recipient received the message M from the originator if

- R_i can decrypt $E_R(M)$ with u_{R_i} (see Section 4.1.4.1) and the outcome is M;
- $Z = E_{PO}(O, R, E_R(M))$ holds;
- R_i's signature on Z and its encryption public key is valid;
- PO's signature on $S_{PO}(R''\_cancelled, Z) \wedge R_i \notin R''\_cancelled$.

The originator will succeed on the dispute if all the above checks are positive. If all the checks but the last are positive and it cannot present evidence of cancellation, then the arbitrator must further interrogate R_i. If the latter cannot present $S_{PO}(R''\_cancelled, Z)$ in which $R_i \in R''\_cancelled$, the originator also wins the dispute. Otherwise, R_i can repudiate having received the message M. Therefore, evidence provided by the PO is *self-contained*, that is, the PO does not need to be contacted in case a dispute arises regarding the occurrence of a cancellation sub-protocol launched by the originator.

5.1.4 Requirements Fulfillment

Let us analyze the requirements fulfillment of the CEM-ZOL protocol as defined in Section 2.5.

- *Strong Fairness*: With the extensions, due to the asynchronism, weak fairness is fulfilled; i.e., each party is in possession of proper evidence or an affidavit which demonstrates channel failures or misbehavior of other participants (cancel tokens). On the other hand, since the TTP acts in an optimistic manner in both cases, the only requirement in the communication channel with entities is that they must be able to contact the TTP eventually.
- *Confidentiality*: This requirement is fulfilled by encrypting the message with recipients' public keys as agreed.
- *Efficiency*: The protocols are optimistic, so efficiency reaches its maximum when all entities behave honestly (only 3 steps are needed). Furthermore, the main sub-protocol is elegant even in the multi-party case, and fortunately, this sub-protocol will be the only one executed in almost all occasions (since optimistic approaches target users whose behavior is expected to be honest).
- *Timeliness*: It fulfills asynchronous timeliness.
- *Policy*: Policy about which cryptographic algorithms to use for evidence generation, which arbiter to use in case of disputes etc., needs to be defined for each different application. The minimum elements it needs to contain are those explicitly mentioned through the description of the protocols. Nevertheless, we define here a more detailed policy by answering the questions defined in Section 2.5 for the extension of multi-party fair CEM. For this task, we assume this CEM application is targeted to Internet users and make use of the verbs "SHALL" "MUST" and "MAY" which are interpreted as in [40].

  a) Rules for evidence generation and verification:
     The policy defines the cryptographic algorithms to be used, e.g. RSA with 2048-bits keys. Keys' format and extensions are needed if X.509v3 digital certificates are to be used for verification. The policy MUST explicitly define the structure of every element used in the protocol. It MUST also define the language (e.g. XML) for representing them $(R, Header, R', R'', R''\_cancelled, R''\_resolved)$. Of course, all legal restrictions coming from use of cryptography MUST be inherited.
     - What evidence should be generated in the non-repudiation service?
       Only the originator collects evidence $(S_{R_i}(u_{R_i}, Z))$ for a normal execution of the protocol. If the TTP intervenes, O also collects $S_{PO}(R''\_cancelled, Z)$

  for the group of recipients which did not reply and each of them MUST also obtain the same token if the originator cancel the protocol for them.
- Which TTP should be involved in evidence generation?
  Explicitly stated.
- What elements should be included in the evidence?
  Explicitly stated.
- Which type of evidence should be generated?
  Explicitly stated.
- Which parties are involved in the generation process?
  Participants MAY obtain the support of external trusted parties for the generation of digital evidence. Nevertheless, for these protocols, the participating entities SHALL generate their own digital evidence.

b) Rules for evidence transfer:
   - Which non-repudiation protocol will be used?
     Explicitly defined.
   - What are the channel assumptions?
     Explicitly defined.

c) Rules for evidence storage:
   The TTP SHALL be audited before being able to serve Internet users. The database storage SHALL make at least an hourly backup. Only the TTP SHALL have write permissions and every input, query and operation in general MAY be logged (and the format will be established by the TTP itself). Time-stamps signed by the TTP for evidence validity and logging purposes MAY be issued as well.
   - What mechanism will be used for maintaining the validity of evidence?
     Trusted time-stamping.
   - How long should the evidence be stored?
     Five years.
   - Does the evidence need to be confidential?
     No.
   - What are the access control rules for accessing the evidence?
     Write permissions for the TTP. Read permissions for users. The access control procedure MAY be established by the TTP.

d) Evidence use:
   It MUST explicitly state under which circumstances the digital evidence collected can be used (i.e., in which situation is considered legally useful) and MAY establish a period of validity (see [14] for further research in this area). For example, the evidence is only valid under the judgement of specific arbiters specialized in CEM disputes on the Internet. The validity period of evidence for this service is 5 years (i.e., this service SHALL not be used if the originator needs to be able to demonstrate recipients receipt of the message for a longer period).

e) Dispute resolution process:

- Which entity will play the role of adjudicator?
  Explicitly stated in the protocol policy. If it is a digital arbitrator, X.500 DN (Distinguished Name) [77] MAY be used for uniquely referring to a specific entity.
- Which parties should be involved in dispute resolution?
  Only participants. In normal executions only the originator. If exception occurs, the recipients MAY also be involved. PO is not involved.
- What evidence should be provided?
  Explicitly defined.
- What are the rules followed in the resolution process itself?
  Explicitly defined.
- What is the expiry date of evidence?
  Explicitly defined
- Which law should be referenced to enforce the arbitration?
  Depends on the country, e.g. LSSI in Spain.

• *Verifiability of TTP*: The TTP (or PO) needs to be trusted, so it is not verifiable. However, if this is a preferred property, it can be straightforwardly achieved. For this to occur, the PO needs to modify step 3' in the cancel sub-protocol and step 2' in the resolve sub-protocol.

1. Signing all messages for entities
2. Including O's cancel request into cancel tokens

With those two modifications, the PO becomes *verifiable* when all the entities behave honestly but, unfortunately, transparency of the PO is lost. The solution chosen depends on the application itself. In our Internet CEM application for users, we MAY select a verifiable PO without transparency. In an email application, the users would not care on the receipt format as long as it is valid in case of dispute, and would rather prefer not to deposit full trust over the PO, specially when a lot of PO entities exist on the Internet.

For a demonstration on TTP's verifiability with the proposed modifications, we first enumerate the possible misbehaving actions the PO can commit.

a. Cancel the protocol without O's request
b. Cancel the protocol for recipients the originator did not request
c. Do not cancel the protocol for recipients the originator requested
d. Store the message for R_i when the protocol has been cancelled by O for that entity
e. Cancel the protocol for entities which have already resolved
f. Store R_i's evidence of receipt for O when it provided a cancel token to R_i

By the modification (2), (a) and (b) will be later demonstrated by the originator. In (c) if the PO does not cancel the request for an entity R_i, it needs to provide O with R_i's evidence of receipt. If it does not so, the originator can demonstrate it with the modification introduced in (1). Similarly in (d), (e) and (f) there will exist an incongruence between the signature provided to O and to the recipients.

- *Transparency of TTP*: Even when the PO intervenes, its participation is *transparent*, i.e., if the protocol can be resolved successfully, evidences obtained by the originator (i.e., mail receipt) do not change.

We have informally studied the properties the protocol should fulfill in a scenario with Internet users willing to send a certified email to several recipients. The fulfillment of these properties would be different if different applications are addressed.

5.2 Multi-Party Contract Signing

There is no doubt that contract signing is a fundamental service in doing business. As pointed out in the state-of-the-art study of Section 3.1, a number of two-party contract signing protocols have been proposed with various features. Nevertheless, in some applications, a contract may need to be signed by multiple parties. Less research has been done on multi-party contract signing. This section will describe a new synchronous multi-party contract signing protocol (MPCS-ZOL) that, with n parties, reaches a lower bound of $3(n-1)$ steps in the all-honest case and $4n-2$ steps in the worst case (i.e., all parties contact the TTP). This is so far the most efficient synchronous multi-party contract signing protocol in terms of the number of messages required. We further consider additional features like timeliness and abuse-freeness in an improved version.

Since contract signing is a particular case of fair exchange, any fair exchange protocol found in the literature in which digital signatures are exchanged can be considered as related work and some of these schemes have been mentioned in Section 3.1.2. On the other hand, also fair-exchange protocols can be constructed by means of contract signing protocols (using digitally signed contracts as evidence tokens).

In all practical schemes, contract signing involves a TTP which plays the role of a notary in paper-based contract signing and somehow shares the legal duties the former ones have. In fact, designing and implementing a contract signing protocol using an in-line TTP should not be a complicated task. In this case, if Alice and Bob wish to enter into a contract, they each sign a copy of the contract and send it to the TTP through a secure channel. The TTP will forward the signed contracts only when it has received valid signatures from both Alice and Bob.

Nevertheless, in our continuous search for speeding up our daily life activities, it is desirable not using a TTP in a contract signing protocol. Additionally, if the TTP is not involved, the notary fee could be avoided. Some protocols appear in the literature trying to eliminate the TTP's involvement using *gradual or probabilistic exchange* of signatures [38, 53]. In [53], Even et al. propose a contract signing protocol which makes use of *oblivious transfer*. This allows the transfer of a recognizable (e.g. signed) message M such that the recipient can read it with a probability $1/2$ while the originator has no way of knowing whether the recipient could read it or not. With this tool, they provided a probabilistic contract signing protocol with a very high success (fairness) probability.

Nevertheless, and specially for contract signing protocols, a signer would not like to risk one million dollars when the deal is done. Therefore, these solutions may not be accepted by signatories. Furthermore, users already deal with the presence of TTPs when important contracts are to be signed (e.g., the act of selling a house in which usually a notary is involved) in the paper-based world. This section is focused on deterministic optimistic contract signing protocols.

We compare these solutions in terms of efficiency with [24] because only Asokan *et al.* addressed the MPCS problem in synchronous networks. Some authors also considered the abuse-freeness property in [61, 36].

As asynchronous networks impose less requirements on the network, asynchronous protocols need more rounds to fulfill fairness. In order to find a lower bound in the number of rounds of an asynchronous MPCS protocol, let us repeat Garay's theorem.

Any complete and optimistic asynchronous contract-signing protocol with n participants requires at least n rounds in an optimistic run.

In the synchronous model, messages sent among participants can be lost in the network, but a message from a participant reaches the TTP in a finite and known amount of time. Attackers can insert, delete and modify messages, but it is assumed that attackers cannot break the clock synchronization of the network and cannot forge digital signatures. Under this model, the number of rounds can be made independent of the number of participants.

5.2.1 A New Synchronous MPCS Protocol

Let us first present a simple synchronous protocol for multi-party contract signing. As in Asokan's approach, this is also based on two differentiated phases: a promise to sign, and a real signature that a party releases only after receiving all promises from the rest of participants. Again, in the same manner, this protocol reaches a lower bound of $4(n-1)$ steps in the all-honest case and $5n-3$ steps in the worst case when all parties contact the TTP. This result will be further improved in the optimal version by reducing the number of steps to $3(n-1)$ in the all-honest case and $4n-2$ in the worst case.

5.2.1.1 A Simple Version

Let us consider the following simple solution which uses verifiable encryption of signatures based on a ring architecture for achieving *transparency* of the TTP. Assume that the channel between any participant and the TTP is functional and not disrupted. The following notation is used in the protocol description.

- $C = [M, P, id, t]$: a contract text M to be signed by each party $P_i \in P(i = 1, \cdots, n)$, a unique identifier id for the protocol run, and a deadline t agreed by all parties to contact the TTP.
- $Cert_i$: a certificate with which anyone can verify that the ciphertext is the correct signature of the plaintext, and can be decrypted by the TTP (see Section 5.2.1.3).

A simple linear protocol for multi-party contract signing is sketched as follows.

1.	$P_1 \rightarrow P_2$	$: m_1[= C, e_{TTP}(S_{P_1}(C)), Cert_1]$
2.	$P_2 \rightarrow P_3$	$: m_1, m_2[= C, e_{TTP}(S_{P_2}(C)), Cert_2]$
$n-1$.	$P_{n-1} \rightarrow P_n$	$: m_1, .., m_{n-1}[= C, e_{TTP}(S_{P_{n-1}}(C)), Cert_{n-1}]$
n.	$P_n \rightarrow P_{n-1}$	$: m_n[= C, e_{TTP}(S_{P_n}(C)), Cert_n]$
$n+1$.	$P_{n-1} \rightarrow P_{n-2}$	$: m_{n-1}, m_n$
$2(n-1)$.	$P_2 \rightarrow P_1$	$: m_2, m_3, .., m_n$
$2n-1$.	$P_1 \rightarrow P_2$	$: S_{P_1}(C)$
$2n$.	$P_2 \rightarrow P_3$	$: S_{P_1}(C), S_{P_2}(C)$
$3(n-1)$.	$P_{n-1} \rightarrow P_n$	$: S_{P_1}(C), S_{P_2}(C), .., S_{P_{n-1}}(C)$
$3n-2$.	$P_n \rightarrow P_{n-1}$	$: S_{P_n}(C)$
$3n-1$.	$P_{n-1} \rightarrow P_{n-2}$	$: S_{P_{n-1}}(C), S_{P_n}(C)$
$4(n-1)$.	$P_2 \rightarrow P_1$	$: S_{P_2}(C), S_{P_3}(C), .., S_{P_n}(C)$

The above main protocol is divided into two phases. The parties first exchange their commitments in an "in-and-out" manner. Note that P_1 can choose t in the first message (and others can halt if they do not agree). Only after the first phase is finished at step $2(n-1)$, the final signatures are exchanged. Following this simple approach, only $4(n-1)$ steps are needed.

If there is no exception (e.g., network failure or misbehaving party), the protocol will not need TTP's help. Otherwise, the following resolve sub-protocol helps to drive the contract signing process to its end. P_i can contact the TTP before the deadline t.

```
1. $P_i \rightarrow TTP$ : $resolve_{P_i} = m_1, .., m_n$
2. $TTP$          : IF $NOT\ resolved\ AND\ resolve_{P_i}$ is received before $t$ THEN
                        decrypts $m_1..m_n$
                        publishes $S_{P_1}(C), .., S_{P_n}(C)$
                        $resolved$=true
```

Boolean variable *resolved* is initialized to false. If the main protocol is not completed successfully, some parties may not hold all the commitments $(m_1, .., m_n)$. Then, they just wait until the deadline t and check with the TTP whether the contract has been resolved by other parties. If not, the contract is cancelled. Otherwise, they get the valid contract $(S_{P_1}(C), .., S_{P_n}(C))$ from the TTP.

If a party has all the commitments [4] when the main protocol is terminated abnormally, it could initiate the above sub-protocol. Then the TTP will help to resolve the contract if the request is received before the deadline t, and the contract will be available to all the participants (even after the deadline t). In this case, the TTP needs to

[4] Note that the TTP does not need to receive $Cert_i$ in order to resolve the protocol. It has not been indicated for the sake of simplicity in the description of the resolve sub-protocol.

check whether all the signatures from the participants match; i.e., they signed over the same contract C[5]. After the deadline, the TTP will not accept such requests any more. In other words, the status of the contract will be determined the latest by the deadline t. Note that no party can cheat the TTP using a distinct deadline because in that case they will be cancelling or resolving a different contract. If parties do not want to advertise this data in the contract itself since timeout is not part of the text agreed, then the deadline can be somehow (hashed together with) included in the unique *id* which is the main variable used by the TTP to distinguish between different protocol instances as a contract reference. In this protocol, the TTP intervention is simple and elegant.

In this case, the dispute resolution process is straightforward. If a party holds all the signatures the contract is assumed to be valid and signed.

5.2.1.2 An Optimal Version

The above protocol has two clearly differentiated phases: exchange of commitments and exchange of digital signatures. The number of steps can be further reduced if more available information is sent at each step and thus merge both phases. This will result in an improvement to the previous simple version of the MPCS-ZOL protocol.

Using the same notation, an optimal synchronous protocol for multi-party contract signing is outlined as follows.

1.	$P_1 \rightarrow P_2$	$: m_1[= C, e_{TTP}(S_{P_1}(C)), Cert_1]$
2.	$P_2 \rightarrow P_3$	$: m_1, m_2[= C, e_{TTP}(S_{P_2}(C)), Cert_2]$
$n-1$.	$P_{n-1} \rightarrow P_n$	$: m_1, .., m_{n-1}[= C, e_{TTP}(S_{P_{n-1}}(C)), Cert_{n-1}]$
n.	$P_n \rightarrow P_{n-1}$	$: m_n[= C, e_{TTP}(S_{P_n}(C)), Cert_n], S_{P_n}(C)$
$n+1$.	$P_{n-1} \rightarrow P_{n-2}$	$: m_{n-1}, m_n, S_{P_{n-1}}(C), S_{P_n}(C)$
$2(n-1)$.	$P_2 \rightarrow P_1$	$: m_2, m_3, .., m_n, S_{P_2}(C), S_{P_3}(C), .., S_{P_n}(C)$
$2n-1$.	$P_1 \rightarrow P_2$	$: S_{P_1}(C)$
$2n$.	$P_2 \rightarrow P_3$	$: S_{P_1}(C), S_{P_2}(C)$
$3(n-1)$.	$P_{n-1} \rightarrow P_n$	$: S_{P_1}(C), S_{P_2}(C), .., S_{P_{n-1}}(C)$

The resolve sub-protocol used by participants to request TTP's help does not change. Note that even though the two phases are merged, no party releases its plaintext signature of the contract without having first received all the commitments. If any party decides to quit before releasing its plaintext signature of the contract, the rest of participants can obtain all plaintext signatures of the contract with TTP's help. As the protocol is similar to the previous one, the same requirements are fulfilled (see Section 5.2.2) and identical dispute resolution process is used by the adjudicator.

This optimal version allows overlapping the dispatch of promises with real signatures without loosing fairness. It improves the simple version presented in Section 5.2.1.1 by reducing the number of steps to $3(n-1)$ in the all-honest case and

[5] Although this operation requires time, it is done off-line.

$4n-2$ in the worst case. Note that for $n=2$, three messages are sufficient and optimal, as shown in [112].

5.2.1.3 Certificate of an Encrypted Message Being a Signature

Let us introduce the concept of *Certificate of Encrypted Message Being a Signature* (CEMBS). Detailed mathematics about the certificate are described in [31]. This certificate is a tool which allows users to verify that the ciphertext is the digital signature over a given plaintext and that can be decrypted by a third party. Let P_u/S_u be the pair of public/private keys for a user u in P and p_u/s_u be the pair of public/private keys for a user u in S. The operations allowed in this system are

$$m = S_{s_u}(M);\;\; C = E_{P_u}(m);\;\; Cert = CEMBS(C) \tag{5.1}$$

This generates a certificate *Cert* which allows the user to prove that indeed C is the ciphertext under P of the digital signature (under S).

$$Verify(P_u, p_u, Cert, C, M) = \text{yes or no} \tag{5.2}$$

This predicate allows a user to verify whether *Cert* represents the encryption under P of the digital signature over M (under S).

In this system, it is computationally hard to generate *Cert* in such a way that $Verify(P_u, p_u, Cert, C, M) = yes$ without $C = E_{P_u}(m)$ and $m = S_{s_u}(M)$.

CEMBS can be realized on systems with P = ElGamal public key cryptosystem and S = DSA-like signature scheme. It can also be realized on systems with P = ElGamal public key cryptosystem and S = Gillou-Quisquater signature scheme [67]. For more details on its realization refer to [31].

5.2.2 *Requirements Fulfillment*

Let us analyze the requirements fulfillment of the optimistic multi-party contract signing protocols (MPCS-ZOL) as defined in the above sections.

- *Strong Fairness*: It is straightforward to demonstrate the fulfillment of this property. No party will be in an advantageous situation at the end of the protocol. That is, either all of them possess the contract (or have access to it), or none of them obtains it.

  The concept of evidence of submission and delivery is omitted here because the TTP acts in an off-line manner.
- *Confidentiality*: This requirement is not fulfilled because the contract is not considered to be confidential data.

- *Efficiency*: The protocols are optimistic. If all parties send all the needed messages correctly, the TTP will not have to decrypt any commitment since after the $3(n-1)$ steps all parties have the signed contract (with n signatures).
- *Timeliness*: The status of a contract will be finalized either at the end of the main protocol or the latest by a pre-defined deadline t. As the participants not holding all the commitments cannot determine the status of the contract before the deadline t, the property of asynchronous timeliness is not satisfied. See Section 5.2.4 for further discussions about timeliness.
- *Policy*: The policy for these protocols needs to be defined for each different application (e.g., depending on whether the contract implies considerable amounts of money). The minimum elements it needs to contain are those explicitly mentioned through the description of the protocol. For instance, the policy needs to explicitly define which digital signature algorithms can be used for obtaining a valid signed contract and for how long a valid contract will be published in a TTP media storage after reaching the deadline.
- *Verifiability of TTP*: Let us identify the possible dishonest behaviors of the TTP: (1) the TTP simply does not reply to participants' requests, or replies with invalid messages; (2) the TTP resolves the protocol but does not publish the contract.

  In the first case, some parties could be beneficiated if they got the contract from the main protocol while others did not. A possible solution is using multiple TTPs and a secure media storage. TTPs have only the write privilege over the media storage but do not control it. Participants in the contract signing protocol have only the read privilege over the media storage. A participant can multicast his request to the TTPs before the deadline t. As long as one of the TTPs does not misbehave, the correct response will be available from the secure media storage. A drawback of this design is the existence of a new (hardware) identity over which participants need to place some trust. Nevertheless, this is a solution recommended by the ITU in X.813 [81]. At the same time, replication of the TTP introduces a (minor) cost in the infrastructure which needs to be faced in resolving the protocols.

  In the second case, the TTP could collude with some parties and resolve the contract for them but not publish the contract for other parties. That means some parties not holding all the commitments will not get the valid contract. To detect TTP's misbehavior, the TTP is required to sign the contract when it is resolved, but this overrides TTP's transparency. Again, it is difficult to reach a trade-off between transparency and verifiability.

  In summary, the TTP can only be partially verifiable.
- *Transparency of TTP*: The protocol uses a cryptographic primitive (CEMBS), which allows any user to verify that a bit string is actually the encryption (with the TTP's public key) of the sender's digital signature over the contract C. If the TTP is invoked, it only decrypts the digital signatures and makes them available to all participants. Therefore, after a successful protocol instance, no evidence of the TTP's participation exists.

- *Abuse-freeness*: This is an optional property for contract signing protocols. In the protocol presented, the last participant (P_n) in the ring can decide whether to resolve the protocol after receiving all the commitments from other parties. However, as stated in [36], it is not possible to avoid this participant to control whether the normal flow of the protocol continues or not, but all we can aim to is to avoid that it is able to provide evidence to an outsider about its control over the result of the contract. So, for P_n holding $m_1, \cdots, m_{n-1}$, due to the presence of $Cert_i$ in m_i that anyone can verify, it is possible for P_n to abuse about the state of the contract. Section 5.2.3 shows that the property of abuse-freeness can be achieved.

5.2.3 Achieving Abuse-Freeness

The MPCS protocol presented in Section 5.2.1 improved the lower bound of steps in existing synchronous MPCS protocols. However, it does not satisfy the properties of abuse-freeness and asynchronous timeliness. Here, those properties are addressed.

Although it is not possible to force a participant to keep on following the steps of the protocol, the latter can be designed in such a manner that it has no way to demonstrate to an outsider the contract is under its control. For this purpose, the new protocol uses a *blind commitment* that only the TTP can verify. With this concept of design in mind, we modify the previous protocol to eliminate the "illustrative" information. The main protocol remains the same, but $Cert_i$ is not included in m_i. Instead, evidence of origin of the blind commitment $Commit_i$ is generated.

$$Commit_i = S_{P_i}(h(C), e_{TTP}(S_{P_i}(C)))$$

where $h(C)$ is the hash value of C to be used to establish a unique link between $Commit_i$ and C.

$$\begin{array}{lll}
1. & P_1 \rightarrow P_2 & : m_1[= C, e_{TTP}(S_{P_1}(C)), Commit_1] \\
2. & P_2 \rightarrow P_3 & : m_1, m_2[= C, e_{TTP}(S_{P_2}(C)), Commit_2] \\
n-1. & P_{n-1} \rightarrow P_n & : m_1, .., m_{n-1}[= C, e_{TTP}(S_{P_{n-1}}(C)), Commit_{n-1}] \\
n. & P_n \rightarrow P_{n-1} & : m_n[= C, e_{TTP}(S_{P_n}(C)), Commit_n], S_{P_n}(C) \\
n+1. & P_{n-1} \rightarrow P_{n-2} & : m_{n-1}, m_n, S_{P_{n-1}}(C), S_{P_n}(C) \\
2(n-1). & P_2 \rightarrow P_1 & : m_2, m_3, .., m_n, S_{P_2}(C), S_{P_3}(C), .., S_{P_n}(C) \\
2n-1. & P_1 \rightarrow P_2 & : S_{P_1}(C) \\
2n. & P_2 \rightarrow P_3 & : S_{P_1}(C), S_{P_2}(C) \\
3(n-1). & P_{n-1} \rightarrow P_n & : S_{P_1}(C), S_{P_2}(C), .., S_{P_{n-1}}(C)
\end{array}$$

Each party needs to check whether all the blind commitments it has received are valid before releasing its real signature of the contract. A *valid* blind commitment $Commit_i$ means it is from P_i (by checking its signature), linked to C (by checking $h(C)$), but does not guarantee that $e_{TTP}(S_{P_i}(C))$ in $Commit_i$ matches $S_{P_i}(C)$. $Commit_i$ is *correct* if it is valid and also matches $S_{P_i}(C)$.

If there is no exception (e.g., network failure or misbehaving party), the protocol will not need TTP's help. Otherwise, a modified resolve sub-protocol helps to drive the contract signing process to its end. P_i can contact the TTP before the deadline t.

```
1. $P_i \rightarrow TTP$ : $resolve_{P_i} = m_1,..,m_n$
2. $TTP$               : IF $NOT\ resolved\ AND\ resolve_{P_i}$ is received before $t$
                         AND all $Commit_i$ are valid THEN
                           decrypts & verifies $m_1..m_n$
                           $resolved$=true
                           IF $S_{P_1}(C),..,S_{P_n}(C)$ ok THEN
                             publishes $S_{P_1}(C),..,S_{P_n}(C)$
                           ELSE IF $P_i \notin group_f$
                             publishes $fail, group_f, S_{TTP}(fail, C, group_f)$
```

Boolean variable *resolved* is initialized to false. When a party holding all the *valid* blind commitments initiates the above sub-protocol, the TTP will help to resolve the contract if the request is received before the deadline t. The TTP decrypts and verifies $m_1, \cdots, m_n$. If they are all correct, the TTP will publish $S_{P_1}(C), \cdots, S_{P_n}(C)$. Otherwise, the TTP will invalidate the contract by publishing a *fail* token $S_{TTP}(fail, C, group_f)$ where $group_f$ indicates the parties which misbehaved in generating their commitments.

The dispute resolution process is changed when the *fail* token is introduced. If a party can show this token, the contract is invalid. Therefore, with respect to the simple dispute resolution process defined previously, now the arbiter needs to interview both parties disputing the validity of the contract.

Furthermore, at the end of the main protocol, each party needs to check whether $e_{TTP}(S_{P_i}(C))$ in $Commit_i$ matches $S_{P_i}(C)$ for $i = 1, \cdots, n$ (assuming the encryption algorithm is deterministic). If not, it should initiate the above sub-protocol to get the *fail* token. A party P_i cannot get any advantage by providing different $Commit_i$ in the main protocol and the resolve sub-protocol. If P_i provides correct $Commit_i$ in the main protocol but incorrect $Commit_i'$ in the sub-protocol, P_i will not get the *fail* token, i.e., cannot cancel a protocol instance whose final state is signed. On the other hand, if P_i provides incorrect $Commit_i$ in the main protocol but correct $Commit_i'$ in the sub-protocol, P_i may get the signed contract if other parties did not misbehave; however, any other honest party can initiate the resolve sub-protocol to get the *fail* token, thus the contract is still invalid.

Remark : It seems that an external adversary could generate valid looking or even fake blind commitments himself. This would allow him to abort the protocol when desired, but evidence of origin avoids this situation.

The blind commitment does not allow a participant to demonstrate that the protocol state is under its control. In fact, in this case, getting all m_i does not mean being able to solve the protocol. Hence, it provides an abuse-freeness feature. Proof is straightforward, since there is no point in the protocol in which an entity can ensure, even to itself, that the contract is signed until plaintext signatures are obtained. The solution allows to maintain the same number of steps as the optimal protocol in Section 5.2.1.2. Furthermore, the TTP is still transparent in this sub-protocol because the signed contract published by the TTP is the same as the one obtained in the main protocol.

5.2.4 Achieving Timeliness

In the protocols presented in Section 5.2.1, a deadline t is selected by the first participant. If other participants disagree with the deadline, they can simply abort the execution of the protocol. Of course, this deadline could be negotiated among the participants before the contract signing protocol is initiated.

If the main protocol is not completed successfully, some participants may hold all the commitments while the others may only hold part of the commitments. For those holding all the commitments, they have the freedom to either resolve the contract with TTP's help before the deadline t, or take no action and just let the contract being automatically cancelled after the deadline t.

As already mentioned, asynchronous timeliness is not fulfilled in these protocols, where a deadline t is used, forcing all the participants to approximately synchronize their clocks with the TTP's one. This task has been widely studied and standard solutions exist (such as the *network time protocol* [103]). Nevertheless, asynchronous timeliness fits better with users in MPCS protocols since, for instance, contract conditions can also change with time, not being favorable any more for a group of entities.

For those only holding part of the commitments, they have no options but only wait until the deadline t to know the status of the contract. Obviously, this is unfavorable to these participants in term of timeliness. They should also have the right to decide the status of the contract before the deadline t. As they only hold part of commitments, they are not able to resolve the contract, so they can only choose to cancel the contract. (Note that in the "in-and-out" architecture of commitment exchange, for those participants only holding part of the commitments, even if all of them collaborate, their combined commitments are still incomplete to resolve the contract.)

Here we present a (j,n)-*threshold* cancel sub-protocol. As long as there are at least j out of n participants that wish to cancel the contract before the deadline t, the contract could be cancelled. The cancel sub-protocol is as follows, where $counter$ (initial value equals to zero) records the number of cancel requests received by the TTP, and $group_c$ records the participants which made cancel requests. For simplicity of description, it is built based on the main protocol in Section 5.2.1.2 without considering abuse-freeness (but can be easily merged).

```
1. $P_i \rightarrow TTP$ : $cancel_{P_i} = C, cancel, S_{P_i}(C, cancel)$
2. $TTP$             : IF $cancel_{P_i}$ is received before $t$ AND $C$ is not resolved
                       AND $C$ is not cancelled THEN
                         stores $cancel_{P_i}$; $group_c = group_c + P_i$;
                         $counter{+}{+}$;
                         IF $counter \geq j$ THEN
                           sets $C$ as cancelled
                           publishes $cancel, group_c, S_{TTP}(cancel, C, group_c)$
```

The resolve sub-protocol is modified as follows.

```
1. $P_i \rightarrow TTP$ : $resolve_{P_i} = C, m_1, .., m_n, S_{P_i}(C, m_1, .., m_n)$
2. $TTP$             : IF $resolve_{P_i}$ is received before $t$ AND $C$ is not cancelled
                       AND C is not resolved
                         decrypts $m_1..m_n$
                         sets $C$ as resolved
                         publishes $S_{P_1}(C), .., S_{P_n}(C)$
```

With the above cancel and resolve sub-protocols, each participant has at least one option to determine the status of the contract before deadline t if the main protocol is not completed successfully. Thus timeliness is achieved, and the extent of timeliness depends on the threshold value j: strong timeliness when $j = 1$, and weak timeliness when $j = n$.

However, the threshold value j should be selected carefully. If j is too small, a few parties may collude to invalidate a contract. If j is too big, it might be hard to establish a valid cancel request among j parties. A possible option is $j = [n/2] + 1$, with a weak majority to "vote" for the validity of a contract.

In the dispute resolution, the *cancel* token issued by the TTP has the top priority. In other words, if a participant presents the *cancel* token, then the contract is invalid. This implies that if there are at least j out of n participants who want to cancel the contract before the deadline, even if they have released their plaintext signatures in the main protocol, they together can still change their mind before that deadline. This is a reasonable scenario in the real world because the situation defined in the contract may change with time, even during the process of contract signing, and each participant wishes to pursue the maximum benefit by taking appropriate actions (resolve or cancel).

As the *cancel* token from the TTP has higher priority than the signed contract, those parties that have got the signed contract in the main protocol may need to double check with the TTP about the status of the contract by the deadline t. (Note that the double check does not mean the involvement of the TTP itself, but just a query to a public file maintained by the TTP.) If they do not want to wait until that deadline, they can send the resolve request to the TTP instead, thus blocking other parties to enable the TTP to issue the *cancel* token.

With no special requirements and more importantly without introducing additional steps in the protocol, the abuse-freeness property is achieved. In addition, by introducing the concept of threshold cancel sub-protocol, a threshold asynchronous timeliness property in the MPCS-ZOL protocol is accomplished as well. Both are very important properties for digital contract signing protocols.

Chapter 6
Scenarios Supported by MPNR Services

Abstract After studying and designing protocols in which non-repudiation is an integral requirement, we give a step forward looking towards real applications in which non-repudiation can be integrated as an additional security service. The main idea is to make use of those general MPNR protocols in these scenarios. The first scenario in which MPNR has been integrated is DRM. DRM is an umbrella term for any of several arrangements which allows a vendor of content in electronic form to control the material and restrict its usage in various ways that can be specified by the vendor. These arrangements are provided through security techniques, mainly encryption, and the distribution, in a detached manner, of content and rights. This allows free access to the content by the consumers, but only those carrying the proper Right Object (RO) will be able to process such content. As a security service considered in different layers of the security framework defined by ITU X.805, almost all applications need to consider non-repudiation at the very beginning of their design. Unfortunately, this has not been done so far in DRM specifications due to practical issues and the type of content distributed. Section 6.1 analyzes this service for a DRM framework and provides a solution which allows the right objects acquisition to be undeniable.

This content could be shared among different users if the proper digital right is purchased. With emerging decentralized technologies, peer-to-peer (P2P) content distribution arises as a new model for storage and transmission of data. In this scenario, one peer can be playing different roles, either as a distributor or as a receiver of digital contents. In order to give an incentive to the legal distribution of these contents and prevent the network from free riders, Section 6.2 introduces a charging model where distributors become merchants and receivers become customers, and where a non-repudiation service is integrated. To help in the advertisement of digital contents and collection of payment details, an intermediary agent is introduced in the P2P framework. An underlying P2P payment protocol presented in [21] is applied to this scenario *without* total trust on the intermediary agent.

It is possible to implement intermediary entities, as those we have described, as Mobile Agents. Mobile agents play an important role in electronic commerce. But there is still a lot of research to be done in this field. Security in free-roaming

agents is especially hard to achieve when the mobile code is executed in hosts that may behave maliciously. Some schemes have been proposed to protect agent data (or computation results) and thus, allowing us to implement intermediary entities in MPNR protocols with mobile agents. However, a known vulnerability of these techniques is the *truncation attack* where two visited hosts (or one revisited host) can collude to discard the partial results collected between their respective visits. This and other properties, requirements and features are discussed in Section 6.3.

6.1 Extension of an OMA-based DRM Framework

The traditional industry for multimedia contents has used classical technologies for distribution and consumption. Nevertheless, with the introduction of digitalized multimedia and the use of telecommunication networks, content production and distribution has become easier and faster than ever before. These contents demand more protection against theft and prying eyes. This increasing need of content protection is driven by two trends. The first is mass piracy and theft of intellectual property and proprietary information. The second is that more "sensitive information" such as financial statements, medical records, and contracts are available in digital form and must be securely stored, shared, or distributed within and between organizations.

This is precisely the niche in which DRM comes out to offer us a solution. Technically, DRM is defined as a set of technologies and systems that can collectively support the entire life cycle of contents (creation, manipulation, distribution and consumption) by preventing illegal copying, imposing fees, processing payments, tracking contents, and protecting each principal's right and profit.

In these systems, content and rights are distributed in a detached manner. This technique simplifies the download of content and its management. No protection of the content is needed, such that any user can download it. But, of course, in order to consume it, a user needs to access (purchase) the corresponding digital right object. Here, two possible approaches for rights management exist.

- *Centralized*: A user needs to access the corresponding right from a central manager each time it wants to consume content. It is very effective against malicious users, but not so much against malicious rights managers. Additionally, this approach suffers from scalability problems.
- *Distributed*: A user keeps its rights and just makes use of them when needed. It overcomes the existing drawbacks of centralized systems, but nevertheless, in order to avoid illegal use of the rights, a tamper-resistant hardware or *Trusted Personal Device* (TPD) is needed (that locally manages the rights in a certified and tamper-proof way).

With the advent of cellular networks, the distributed approach allows the convergence of user and industry needs. Combining DRM solutions with mobile networks, users can access the digital rights by using their mobile handset as a TPD. Telecommunication operators can drive the users for accessing or purchasing digital rights as well as certifying the secure management of digital rights in the handset (see Figure 6.1).

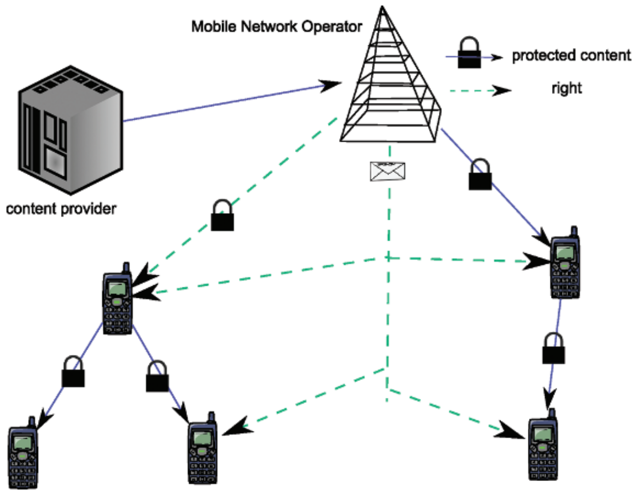

Fig. 6.1 Content Distribution

The modified platform shown in the figure is based on the OMA DRM specification 2.0 [109]. The modified scheme proposed for the distributed rights management in the European project UBISEC [1] will enable a more secure framework for charging on the digital rights acquisition, taking into account important issues such as anonymity and efficiency (see Figure 6.2).

The distribution of the RO to the user through a *Mobile Network Operator* (MNO) comes out as a final important step on the fair distribution of digital content (see Figure 6.3). Anonymous purchase of rights is supported, since the Content Provider and the *Rights Issuer* (RI) do not require privacy details of consumers. Consumer billing is performed through the MNO to whom the consumer is subscribed. Evidence will be generated, such that, if any dispute arises among the parties, they will be able to demonstrate their participation in the DRM scenario. Even though this solution strongly relies on trusted third parties (MNO and RI), non-repudiation issues on content distribution have to be considered, without having an impact on all the above mentioned properties.

Considering the user as the customer which receives content and rights in order to be able to consume such content, non-repudiation could be a valuable service in the last phase when the customer has to access the Right Issuer (through the Mobile Network Operator) to get the RO in exchange for the payment (the MNO charges the user for the RO value in its monthly bill).

[1] UBISEC - Ubiquitous Networks with a Secure Provision of Services, Access, and Content Delivery (FP6-2002-IST-1-506926).

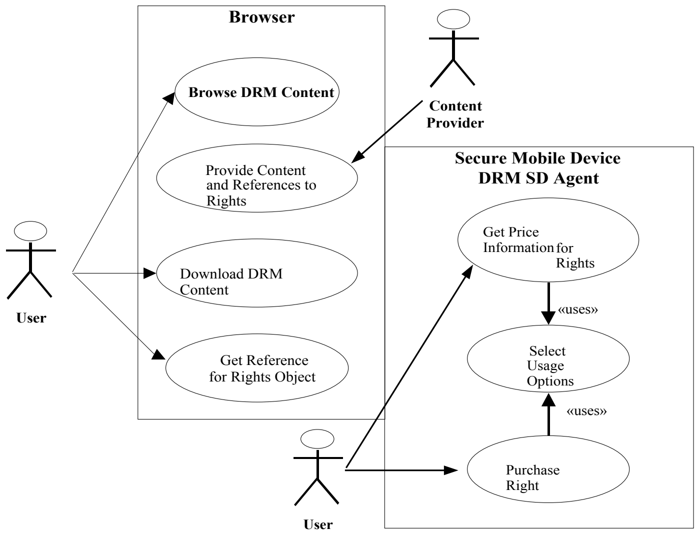
Browser
Browse DRM Content
Provide Content and References to Rights
Download DRM Content
Get Reference for Rights Object
Content Provider
User
Secure Mobile Device DRM SD Agent
Get Price Information for Rights
«uses»
Select Usage Options
«uses»
Purchase Right
User

Fig. 6.2 DRM

Even though the MNO and RI are considered trusted entities, there can be several difficulties in the process (e.g., a network failure or loss of data) which can end up in disputes among parties. Such possible disputes could be as follows.

- The MNO charges the user for a RO it did not purchase or receive. It could also occur that the amount of money charged does not coincide with the one expected by the user.
- The user receives a corrupted RO while already having paid for it.
- The user denies having sent a request (RORequest) for purchasing the RO.
- The MNO denies having received a request from the user.
- Similar disputes between the MNO and the RI.

From this list, and according to the definition of non-repudiation services given by the ITU, the non-repudiation of origin and non-repudiation of receipt services have to be provided between the user and the MNO and between the MNO and the RI, thus establishing a logical non-repudiation channel between the user and the RI.

It must not be possible for a Right Issuer to claim that he sent the RO when he did not. In the same way, it must not be possible for a user to falsely deny having received the RO. Evidence should be collected to resolve these disputes arisen between participating entities in a DRM scenario. Using the agent-mediated protocol described in Section 4.2, we introduce a protocol that is integrated into UBISEC's DRM framework. It uses an intermediary and allows fair exchange of evidence in

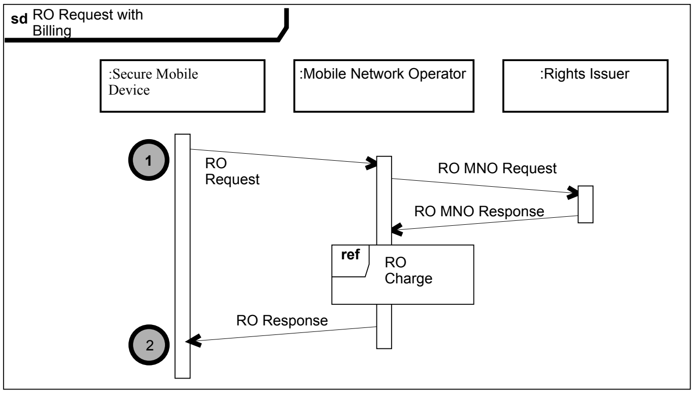

Fig. 6.3 Right Object Acquisition

the RO acquisition phase [2]. For more details on this DRM framework which is based on the OMA DRM v2.0, see [123].

6.1.1 Protocol

Collecting, verifying and storing evidence about an electronic interaction is required, but might be operationally undesirable for final entities. Hence, intermediary entities are useful in such scenarios to help final entities to carry out their protocol exchanges. It is clear that this philosophy matches the Mobile DRM approach in which the Mobile Network Operator serves as an intermediary entity. Users have direct access to the MNO and implicitly place certain degree of trust on it.

The MNO plays a critical role in this scenario, so it is important to analyze its behavior. As the MNO has interest (billing) in a transaction, it will be willing to reach a successful transaction. But occasionally, the MNO may collude with another (external or internal) entity and, for instance, hide some evidence. Therefore, we assume the MNO is not fully trusted and that the MNO is not going to collude with the user against the RI. This seems a reasonable assumption, since collusion between the MNO and a customer against other Third Party Provider with which it will have a more profitable contract, is unlikely to occur.

Prior to the protocol description, let us depict the scenario. The mobile user accesses a Content Provider and obtains the desired content as well as a reference to the Digital Rights needed for consuming that content [3]. Using this reference ob-

[2] Although the requests and responses are XML signed in the DRM specification, this does not ensure fair exchange of items and thus, it does *not* provide a fair non-repudiation service.

[3] Note that we do not make explicit which device is used for downloading the contents.

ject and its TPD (e.g. mobile phone), it can access Price Information depending on the Usage Properties it desires; for instance, one-time play, three times play or an unlimited consumption right. We call this selection $PriceInfo$. With this info collected, the user is ready to access the Rights Issuer (which can be the same entity as the Content Provider depending on the market model) for buying the corresponding right. It is at this moment in which the protocol presented in this section takes place.

6.1.1.1 Protocol Description

Detailed notation for the protocol is as follows.

- $l = h(U, RI, MNO, TTP, t, PriceInfo, RORequest)$: label of message *RORequest*
- t: a timeout chosen by the user U, before which the TTP has to publish some information
- $EOO = S_U(feoo, MNO, RI, TTP, l, t, RORequest)$: evidence of origin of having sent *RORequest*, generated by U
- $EOO_{MNO} = S_{MNO}(feoo, RI, TTP, l, t, ROMNORequest)$: evidence of origin of *RORequest* issued by the MNO for the RI
- $EOR = S_{RI}(feor, MNO, l, t, ROMNOResponse)$: evidence of receipt of *ROMNORequest* generated by the RI
- $EOR_{MNO} = S_{MNO}(feor, U, RI, TTP, l, t, PriceInfo, ROResponse)$: evidence of receipt of *RORequest* issued by the MNO for U, and evidence of origin of *ROResponse* at the same time
- $Con = S_{TTP}(fcon, MNO, RI, l, t, PriceInfo, ROResponse)$: evidence of confirmation issued by the TTP

The protocol is as follows.

$$\begin{array}{ll}
1.\ U \rightarrow MNO & : feoo, MNO, RI, TTP, l, t, PriceInfo, RORequest, EOO \\
2.\ MNO \rightarrow RI & : feoo, RI, TTP, l, t, PriceInfo, ROMNORequest, EOO_{MNO} \\
3.\ RI \rightarrow MNO & : feor, MNO, l, ROMNOResponse, EOR \\
4.\ MNO \rightarrow U, TTP & : feor, U, RI, l, t, RORequest, PriceInfo, ROResponse, \\
 & : EOR_{MNO} \\
5.\ All \leftrightarrow TTP & : fcon, MNO, RI, l, ROResponse, Con
\end{array}$$

The protocol works in the following way.

1. U sends the MNO evidence of origin corresponding to the *RORequest* message and $PriceInfo$ as obtained after browsing for rights. There is no breach of fairness if the protocol stops.
2. The MNO distributes U's information (maybe after a negotiation or agreement with the RI and after having prepared *ROMNORequest* from user's *RORequest*) and sends to the RI evidence of involvement in the transaction. Again, fairness is maintained if the protocol is halted.
3. The RI replies with evidence of receipt of *RORequest* together with *ROMNOResponse*. It is assumed that a secure channel exists between the MNO

and the RI. The protocol still remains fair if it stops, since none entity obtains what they expected. (U needs *ROResponse* whereas the RI and the MNO need final evidence of the transaction performed). Note that *RORequest* is uniquely identified in label l and it can be checked by any party involved in the protocol.

4. The MNO sends to U the Digital Rights Object (*ROResponse* [4]) together with evidence of having received *RORequest* and sends a copy to the TTP. U and the TTP will check all evidence carefully before proceeding to the next step. For U, this is the only evidence it will collect from the MNO and will be used in case of disputes to prove MNO's responsibility of the exchange. The MNO will store RI's evidence of receipt in its evidence database and U can retrieve it later if needed. The MNO cannot claim that it did not store this evidence since EOR_{MNO} demonstrates it did if a dispute arises. U and the TTP check

   - $l = h(U, RI, MNO, TTP, t, PriceInfo, RORequest)$
   - the info received is signed by the MNO in EOR_{MNO}
   - $actual\_time < t$

   If *ROResponse* is the expected object (together with its associated price information), U does not need to continue the protocol (since it got what it needed). Otherwise, i.e., if *ROResponse* or the price information is not obtained or it is corrupted, it goes to the next step. The following step undertaken with an extra entity represents an addition with respect to the steps explained so far in the DRM scenario.

5. The TTP releases the confirmation message. U fetches *ROResponse* (if not satisfied in the previous step) and *Con* as evidence of the digital right purchased. The MNO fetches *Con* as evidence that U received (or could fetch) EOR_{MNO} and RO (and the corresponding charge) offered by the RI. The RI fetches *Con* as evidence to prove its origin. Note that if the MNO proceeds to step 4 with $actual\_time > t$, it will gain no advantage. Furthermore, U could get RO without having to pay for it, since the TTP will not generate *Con*.

   On the other hand, if the MNO tries to cheat the TTP by changing the deadline, then it will obtain evidence *Con* which does not match the rest of evidence collected. Thus, all entities are safe after the deadline time t. This behavior will only favour the user U, but by assumption, it is not possible for a collusion between the user and the MNO.

At the end of the protocol, each party will hold the corresponding evidence.

- The user collects EOR_{MNO} and/or *Con* as evidence from the MNO.
- The MNO collects *EOO*, *EOR*, and *Con* as evidence of origin and evidence of receipt, respectively, which allows the MNO to demonstrate its good behavior during the protocol.
- The RI collects EOO_{MNO} as evidence of origin of *RORequest* issued by the MNO. *Con* must also be collected to complete the evidence.

[4] The RO is embedded in ROResponse by the MNO in such a way that only U can have access to it.

This protocol takes only five steps and anonymity could be preserved, that is, unless the consumer is willing to communicate with a pre-selected Right Issuer, neither the consumer nor the Right Issuer needs any knowledge (i.e., digital certificates) about each other in order to reach a successful protocol end. This feature preserves the anonymity property of the DRM framework, and can be used if the MNO is allowed to select different RIs (e.g., depending on trust deposited or price information).

The main critic on this approach could come from the practical efficiency of having the user U producing digital evidences with its mobile phone. Section 6.1.2 sketches the major details of a proof-of-concept implementation showing that nowadays technology allows using asymmetric cryptography in handheld devices. However, if very limited mobile devices are to be used, the concept proposed by Asokan in [23, chp 4, sect 2] of Server-Supported Signatures can be integrated into this protocol using the MNO as a signature server. In such a way, users do not need to be able to produce digital signatures using asymmetric cryptography, but only need to be able to verify digital signatures which can be much easier (if, for instance, RSA is used with a small public exponent). At the same time, the user will need to generate *hash chains* which is computationally less costly than public key cryptography (see "practical efficiency" property of selected hash functions in Definition 3.13).

In this approach, the mobile user generates a secret key K_U, randomly chosen from the range field of its hash function. Based on it, U computes the hash chain $K_U^0, K_U^1, \dots K_U^n$ where

$$K_U^0 = K_U,\ K_U^i = h^i(K_U) = h(K_U^{i-1})$$

with h^i meaning applying i times hash function h. $V_U = K_U^n$ constitutes *U's root verification key* and will enable U to authenticate n messages. U submits this key to a CA for certification. A certificate for U's root verification key is of the form: $Cert_U = S_{CA}(U, n, V_U, MNO)$. Each MNO can easily access or acquire this type of user's certificate.

With initial $i = n$; i is decreased during each run and the steps of the protocol are slightly modified as follows.

- $EOO_{MNO} = S_{MNO}(feoo, RI, TTP, l, t, ROMNORequest, \mathbf{i}, \mathbf{K_U^i})$: evidence of origin of *RORequest* issued by the MNO for the RI

1. $U \rightarrow MNO$	: $feoo, MNO, RI, TTP, l, t, PriceInfo, RORequest, i, K_U^i$
2. $MNO \rightarrow U$	: EOO_{MNO}
3. $U \rightarrow MNO$	: K_U^{i-1}
4. $MNO \rightarrow RI$	: $feoo, RI, TTP, l, t, ROMNORequest, PriceInfo, EOO_{MNO},$
	: K_U^i, K_U^{i-1}
5. $RI \rightarrow MNO$	: $feor, MNO, l, ROMNOResponse, EOR$
6. $MNO \rightarrow U, TTP$	: $feor, U, RI, l, t, RORequest, PriceInfo, ROResponse,$
	: EOR_{MNO}
7. $All \leftrightarrow TTP$	: $fcon, MNO, RI, l, ROResponse, Con$

In the second step, when receiving the message from the user U, the MNO verifies K_U^i based on U's root verification key and U's certificate obtained from CA, by checking $h^{n-i}(K_U^i) = V_U$. The MNO has to ensure that only one evidence of origin is generated on behalf of user for a given (U, i, K_U^i). In this case, the MNO records K_U^i as consumed and sends signature back to U. This *candidate* non-repudiation token is needed by U for demonstrating a possible MNO's misbehavior.

In the third step, U verifies the received signature and stores it. It also records K_U^i as consumed by replacing i with $i-1$. It also reveals K_U^{i-1} for providing definitive evidence. Note that U must consume each element in the hash chain in sequence and must not skip any of them. In particular, U must not ask for a signature using K_U^{i-1} unless it has received MNO's signature under K_U^i. Otherwise, the MNO could use that to create a fake non-repudiation evidence, which U cannot repudiate during a later dispute.

Dispute resolution process, as explained in below, is not modified. But if the RI needs to demonstrate validity of EOO_{MNO}, i.e., MNO's signature on behalf of U, it should provide EOO_{MNO} together with K_U^{i-1}. The arbiter will do the following [5].

- Extract the root verification key from U's certificate
- As before, verify MNO's signature on EOO_{MNO}
- Verify that K_U^i is in fact a hash of the alleged pre-image K_U^{i-1}
- Verify that the root verification key V_U can be derived by repeated hashing $h^{n-i}(K_U^i) = V_U$

Evne if these checks are successful, U can still repudiate if it can demonstrate MNO's misbehavior by showing a different signature EOO_{MNO} corresponding to the same K_U^i.

Although Asokan also proposed this approach for evidence of receipt, generally the RI is not be a resource-limited entity. At the same time, several other considerations show up, as for instance, how to avoid the need for the user of storing all evidences generated on its behalf by the MNO, which is an important factor in limited-devices. This and other issues will not be considered (see Asokan's thesis in which details are explained for further reference).

6.1.1.2 Dispute Resolution

In this model, common disputes which might arise are depicted below. If evidence has an expiry date, the disputes should be settled with the help of an arbitrator prior to that date. Entities (including the TTP) only store evidence during its lifetime, which usually will not exceed a 2-months period (if bills are paid in a monthly manner).

Disputes between User and MNO

If the user receives a corrupted Right Object while already having paid for it but the MNO denies the fact, the user has to provide $ROResponse$, $PriceInfo$, EOR_{MNO} and/or Con to the arbitrator. The arbitrator will check the validity of label l, and also

[5] Of course, checks are done by the RI as well before sending step 5.

check that $(l, ROResponse)$ is signed by the MNO in EOR_{MNO} or by the TTP in *Con*. If successful, the arbitrator determines that the MNO did not provide a valid Rights Object to the user.

If the MNO charges the user for a Right Object (embedded in *ROResponse*) but the user denies purchasing or receiving it, the MNO has to present *EOO* and *Con* to the arbitrator. The arbitrator will check U's signature on *EOO* (demonstrating its request) and the TTP's signature on *Con*. It will also check they are linked by a valid label l. If successful, the arbitrator settles that U got *ROResponse* (or could fetch it from the TTP), and thus, the Right Object from the MNO.

Disputes between RI and MNO

If the MNO denies delivering message *RORequest* (reformatted as *ROMNO-Request*) to the RI, the RI presents evidence EOO_{MNO} and the arbitrator checks MNO's signature on it. If successful, the arbitrator settles that *RORequest*, originated from U, is delivered by the MNO to the RI. If the RI denies having received message *RORequest*, the MNO presents *EOR* and the arbitrator checks RI's signature on it. If successful, the arbitrator settles that the MNO delivered *RORequest* to the RI.

The RI fetches *Con* to demonstrate the transaction was finished with the user. This is useful in case the RI charges the MNO depending on the number of successful Rights Object distributions.

6.1.2 Design and Implementation

This section briefly sketches the design and proof-of-concept of the system (see Figure 6.4). Firstly, we identify the different operations (either as processes or part of an API) to be implemented and describe in detail its functionality. The major operations are shown as follows.

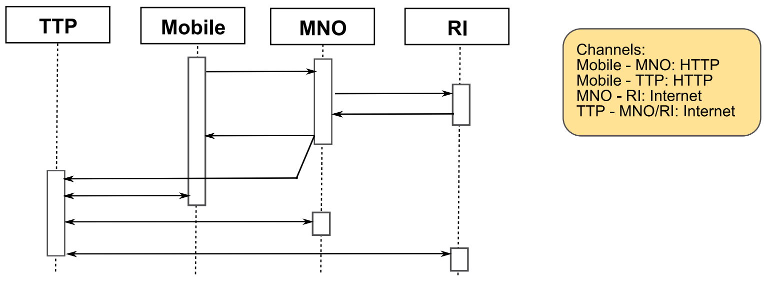

Fig. 6.4 HTTP Communication Flow

U - Mobile Phone User: The user manages the mobile phone, obtaining DRM services. The operations inside the mobile phone are

- *(API)* ***ObtainROResponse***. Enter: *RORequest*. Exit: $[\mathit{ROResponse}|\mathit{Error}]$.
  - *Internal Operation*: Mobile phone negotiates with the MNO (sends *EOO* to the MNO and receives EOR_{MNO}) and with the TTP (fetches *Con* from the TTP), obtaining the rights inside *ROResponse* together with the communication evidence.
  - *Side Effects*: U **must** test and store EOR_{MNO} and/or *Con* as evidence of receipt. Note, U contacts the TTP if EOR_{MNO} is corrupted or lost.

MNO - Mobile Network Operator: It provides service for rights acquisition, by contacting a TSP (Third-party Service Provider) that acts as a RI. The operations are

- *(Process)* ***ManageRORequestFromU***. Triggered by: *EOO*. Halt: on *Error*.
  - *Internal Operation:* The MNO receives *EOO* from U. It creates and sends EOO_{MNO} to the RI, receives *EOR* from the RI, and creates and sends EOR_{MNO} to U and the TTP.
  - *Side Effects:* The MNO **must** test and store *EOO* and *EOR*. Notes, this process must have an interface to access global resources from the mobile network operator infrastructure, such as billing databases and evidence databases.

RI - Rights Issuer: It listens to *RORequest* messages from other entities, and accesses the DRM Objects for obtaining an adequate *ROResponse*.

- *(Process)* ***ManageRORequestFromMNO***. Triggered by: EOO_{MNO}. Halt: on *Error*.
  - *Internal Operation:* The RI receives EOO_{MNO} from the MNO. It calls the DRM *ROResponse* Object with the *RORequest* parameter. It sends *EOR* to the MNO.
  - *Side Effects:* It **must** test and store EOO_{MNO}.

TTP - Trusted Third Party: It receives right objects from mobile phone networks, and distributes them alongside with other evidence information.

- *(Process)* ***ReceiveKeyFromMNO***. Triggered by: EOR_{MNO}. Halt: on *Error*.
  - *Internal Operation:* The TTP receives EOR_{MNO} from the MNO. After testing that the message has been received before the deadline t, it creates *Con* and stores it for later use.
  - *Side Effects:* It **must** store message *Con* alongside with label l. Later, U, the MNO and the RI will fetch the message by using that label l.

For MNO and RI, there is also another process which, if triggered by the previous ones, and after deadline t, it accesses the TTP's directory and fetches and stores *Con* as final evidence.

Although the TTP is a different entity, separated from the MNO, a GPRS connection at network layer may be needed for linking the mobile user with the TTP.

This makes all the traffic between them to be routed through the MNO infrastructure. Therefore, this means the HTTP end-to-end connection between U and the TTP must support SSL.

In this implementation, digital signatures are used as the main tool for managing evidence. Nowadays, generating digital signatures with limited devices is not a restricting operation. For example, some tests done on a mobile phone manufactured in 2005 calculated all the cryptographic operations in 6 seconds.

For the implementation of the Mobile Phone system J2ME-MIDP 2.0 [15] can be used whereas for the rest of components (RI, MNO, TTP) J2SE Java Programming and J2EE-Servlets on the server-side are good candidates. Crypto operations can be implemented (in both J2ME and J2SE/J2EE environments) with the Bouncy Castle Crypto Lightweight Library [17] or use the standard for MIDP, JSR 219 [12]. Finally, for XML-processing in constrained environments, kXML (Lightweight XML library for mobile phones) is a perfect candidate [13].

6.1.3 Requirements Fulfillment

Let us analyze the requirements fulfillment of this multi-party non-repudiation protocol for a OMA-based DRM framework.

- *Strong Fairness*: As it has been depicted along the description of the protocol, each party is in possession of proper evidence and no party is in an advantageous position during a transaction even if it aborts. As the TTP acts in a lightweight on-line manner, the communication channel from participants to the TTP has to be resilient. Note that, even if the MNO and RI collude, fairness in the protocol is preserved, since they will need to present to an arbiter the possession of non-repudiation of origin signed by the user. This fact also holds for the case in which the user makes use of the MNO as a Signature Server as explained in Section 6.1.1.
- *Confidentiality*: This requirement is not needed in this protocol.
- *Efficiency*: The protocol is not optimistic, but the TTP acts in a lightweight manner (i.e., receiving information, processing it and storing digital evidence in a network accessible directory with read-only permissions).
- *Timeliness*: It fulfills synchronous timeliness by the use of a deadline.
- *Policy*: Again, policy about which cryptographic algorithms to use for evidence generation, which arbiter to use in case of disputes etc., needs to be defined for a mobile scenario. The minimum elements it needs to contain are those explicitly mentioned through the description of the protocol as, for instance, the guideline to use when entering a dispute resolution process. Nevertheless, we define here a more detailed policy by answering the questions defined in Section 2.5 for this protocol and considering the limited capabilities of user's device. For this task, we make use of the verbs "SHALL", "MUST" and "MAY" which are interpreted as in [40].

a) Rules for evidence generation and verification:
   The policy defines the cryptographic algorithms to be used. In this case, RSA with a public exponent 3 will help resource-limited devices to easily verify digital signatures without reducing the security of the RSA algorithm if the MNO is used as a digital signature server. If not, keys of 1024 bits are to be selected. Keys format and extensions are needed if X.509v3 digital certificates are to be used for verification. The certificates (from the MNO and the TTP) will be stored on user's mobile memory. Nowadays almost all phones are sold with preloaded digital certificates which can be on WTLS or X.509 format depending on the operating system implemented in the mobile phone. For the proof of concept implementation, X.509 formatted certificates is the most standard option, although for more limited devices, WTLS certificates can be used as well. WTLS defines a compressed certificate format which broadly follows the X.509 v3 certificate structure, but uses smaller data structures. The policy MUST also define the language (XML) for representing them. Of course, all legal restrictions coming from use of cryptography MUST be inherited.
   - What evidence should be generated in the non-repudiation service?
     User *U* generates evidence of origin of the Rights Object request message. The Mobile Network Operator *MNO* generates evidence of U's origin of this request to the Rights Issuer *RI* and evidence of RI's receipt of the request (which serves as evidence of origin of the Right Object Response as well). The RI creates evidence of receipt for the MNO.
   - Which TTP should be involved in evidence generation?
     Explicitly stated.
   - What elements should be included in the evidence?
     Explicitly stated.
   - Which type of evidence should be generated?
     Explicitly stated.
   - Which parties will be involved in the generation process?
     Each participant generates its own digital evidence. Mobile user MAY obtain the support from the MNO for the generation of digital signatures in a verifiable way as explained.

b) Rules for evidence transfer:
   - Which non-repudiation protocol will be used?
     Explicitly defined.
   - What are the channel assumptions?
     Since the TTP acts in an on-line manner, the channel between all entities and the TTP MUST be resilient for ensuring fairness.

c) Rules for evidence storage:
   The database storage SHALL make at least an hourly backup. Only the TTP SHALL have write permissions and every input, query and operation in general MAY be logged (and the format will be established by the TTP itself).

Time-stamps signed by the TTP for final evidence validity and logging purposes MAY be issued as well.

Regarding the mobile user, it MUST store all evidence received (EOR_{MNO} and/or *Con*) for solving future disputes. Nevertheless, because nowadays mobile technology supports the use of bluetooth connections and compacted memory cards, evidence can be off-line downloaded to other devices with less storage constraints. If this is not possible, Asokan also suggested in his thesis an approach for transferring the stored digital evidence to the digital signature server (MNO) as well as an approach for using this entity as a time-stamping server. In case of using this last modification to the introduced protocol, the mobile user MUST store the hash chain corresponding to its root verification key certificate as well.

- What mechanism will be used for maintaining the validity of evidence?
  Trusted time-stamping.
- How long should the evidence be stored?
  Two months for expiration, since the mobile user will have an extra month (assuming monthly billing) for repudiating any Digital Right Object it has been charged for.
- Does the evidence need to be confidential?
  No.
- What are the access control rules for accessing the final evidence (*Con*)?
  Write permissions for the TTP. Read permissions for users. The access control procedure MAY be established by the TTP. Authentication is not really important because evidence is not confidential and only valuable to the intended users.

d) Evidence use: Explicitly stated.

e) Dispute resolution process:
   - Which entity will play the role of adjudicator?
     Explicitly stated in the protocol policy. If it is a digital arbitrator, X.500 DN (Distinguished Name) [77] can be used for uniquely referring to a specific entity. The user *U* MAY be involved in a dispute resolution process using other device different from its handheld device (for which an evidence transfer feature MUST be available).
   - Which parties should be involved in dispute resolution?
     Only the participants; i.e., U, MNO and RI.
   - Which law should be referenced to enforce the arbitration?
     Depends on the country, e.g. LSSI in Spain.

- *Verifiability of TTP*: The TTP needs to be trusted, so it is **not** verifiable. On the other hand, MNO's behavior is verifiable and can be disputed by the originator and RI with the help of an arbiter.
- *Transparency of TTP*: As the TTP is lightly involved in every protocol execution, there is **no** transparency property.

6.2 Practical Service Charge for P2P Content Distribution

Nowadays several factors have lit a fire under the peer-to-peer (P2P) movement: inexpensive computing power, bandwidth, and storage. In a P2P architecture, computers that have traditionally been used solely as clients communicate directly among themselves and can act as both clients and servers, assuming whatever role is needed at each moment. The new P2P networking paradigm offers new possibilities for content distribution over the Internet. Customer peers interchange roles with provider peers, and compete in this new networked economy. A major differentiating factor of P2P from traditional content distribution models is the lack of central management and control. This very important characteristic of P2P systems offers the ability to create efficient, scalable, anonymous - when required, and persistent services by taking advantage of the fully distributed nature of the systems.

If a peer distributing contents gets paid for this distribution, why is this peer going to distribute contents freely? This approach can incentivize a legitimate P2P content distribution, hence avoiding the actual problems of free riders and legal issues for which P2P networks such as eDonkey 2000 and Gnutella have been strongly criticized [20, 62].

Popular software for P2P networking like eMule, Gnutella [4], and Freenet [5] provides everybody with opportunities to exchange low value digital goods. But potential merchants with low value goods (i.e., users inside a P2P network) have no future in such a competitive digital world due to the hardness of collection of payments and advertisement of their goods, compared with profits expected.

For such reasons, new solutions that help the merchant to gain entrance to P2P e-commerce should be provided. Previous work on paid P2P service [70] relies on a fully trusted on-line escrow server, which could be too expensive for those low value transactions. This section introduces a P2P service and payment protocol in which the load of the merchant peer is significantly reduced for only distribution of digital contents while a *weakly* trusted intermediary agent is used for the advertisement and collection of (small) payments [105]. This protocol makes use of a non-repudiation session with an off-line *TTP* integrated inside the payment protocol.

6.2.1 Scenario and Requirements

A market study about P2P commerce is provided in [22], where peers can find evaluation functions and results about the behavior of such a system, facilitating them to take decisions in advance. In that study, different parameters are used to evaluate the market such as cost of transportation, popularity of the contents and competitiveness of the peers. This work intends to reduce costs of transportation, i.e., reduce involvement of the peers in the framework.

Figure 6.5 shows a general scenario we can find in a P2P application. In this scenario, each peer entity desires to earn some money by selling its files (photos, music, videos etc.). But it is hard for every entity to advertise its goods and manage the (probably small) payments with so many entities. Then, such an entity can seek a purchase agent for advertising goods and collecting payments, such that it only needs to provide the digital goods/contents.

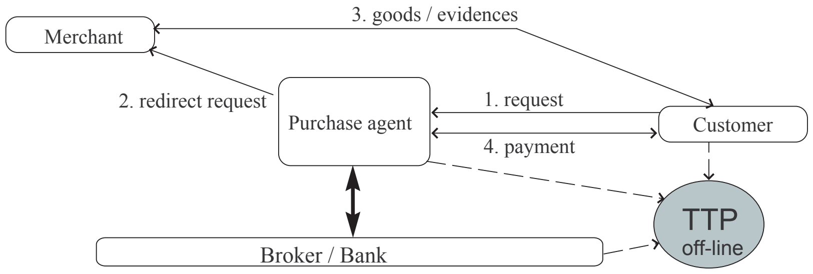

Fig. 6.5 P2P Service and Payment Scenario

We assume that the purchase agent is weakly trusted by the merchant peer in the only sense that collusion with the customer is not possible (see Section 6.2.3.2 for further reference). Tools and reasons for making this type of collusion harder (although not impossible) can be found in reputation issues and incentive schemes. As an incentive for the participation of the agent in this scenario, it could earn a part of each payment or a monthly percentage of each user's successful transactions. On the other hand, collusion with the merchant peer or misbehavior of the agent by itself has to be properly and efficiently treated in the protocol.

Provision of evidence to the peers for later dispute resolution would be important to boost P2P e-commerce where exchanges are carried out between parties that probably have no prior relations and whose identities could be highly volatile.

The following properties are desirable in the above P2P service and payment scenario.

1. *Confidentiality*: The digital goods/contents should be disclosed only to the intended party (i.e., only to customers).
2. *Payer anonymity*: Payers may prefer keeping their everyday payment activities private, i.e., not allowing payees and in some cases even banks to observe and track their payments. There are two levels of anonymity: *untraceability* simply means that an adversary cannot determine a payer's identity in a payment protocol; *unlinkability* means that, in addition, participation of the same player in two different payments cannot be linked.
3. *Fairness*: The customer cannot obtain the digital goods/contents either from the intermediary agent or from the merchant unless a payment is ensured to the merchant.
4. *Timeliness*: The transacting parties always have the ability to reach, in a finite amount of time, a point in which they can stop the protocol without loss of fairness.

5. *Non-repudiation*: It is impossible for a sender peer, after a successful execution of the protocol, to deny having distributed the digital goods. It is impossible for an agent, after a successful execution of the protocol, to deny having received the payment.
6. *Lightweight merchant*: Since the protocol is run in a P2P scenario, merchant peers should not be overloaded with payment issues.

6.2.2 P2P Payment Protocol

General purpose electronic payment systems have been widely studied, which can be classified into two categories: *cash-like* and *check-like* systems. In cash-like systems, special tokens denominated electronic coins (or cash) are used [39, 114]. The payer has been previously taken away an amount of money in a withdrawal protocol, hence they are *pre-paid* payment protocols. In check-like systems, the payer usually issues a form (whether it be a check or a credit card slip) to the payee [25]. There is no previous withdrawal of money, and the payee must ensure that the payer possesses enough money to carry out the payment. So, consulting the payer's bank is necessary prior to accept it. Such systems are also denominated as *on-line* verification payments (e.g. [33]).

An electronic payment system for P2P scenarios was proposed in [21]. In this protocol, three entities are involved: merchant, customer, and broker/bank who is trusted by the other entities. The same notation used in the original paper is listed below for the understanding of this scheme.

- ID_X: identity of entity X
- K_X: secret key used by entity X (and known only to it)
- $KeyedHash_K(inf)$: message inf is hashed using a secret key K
- $h(inf)$: one-way hash function over message inf
- *SerNum*: unique serial number associated with every digital note
- *Value*: value associated with the digital note
- $T0, T1, T2, T3$: time of issue, deadline of redemption, start time of refund, and expiry time for the digital note, respectively

Each digital note is prepared by the merchant as follows.

- $ID_{Material} = ID_M, Value, SerNum, T0, T1, T2, T3$
- $DigitalNote = ID_{Material}, KeyedHash_{K_M}(ID_{Material})$

The main advantage of having the merchant creating its own digital notes is that it can check double spending *before-the-fact* without contacting the broker/bank. The merchant peer then transfers digital notes to the broker who adds a broker stamp such that a stamped digital note format is [*DigitalNote, BrokerStamp, SVC*].

- $BrokerStamp = ID_B, KeyedHash_{K_B}(DigitalNote, ID_B)$
- Stamp Verification Code (SVC) = $h(BrokerStamp)$

Once the broker transfers *SVC* to the merchant, the stamped digital note is ready for circulation. Whenever the customer peer wants to purchase goods from a particular merchant peer, he approaches the broker and obtains a certain amount of digital cash (issued by that merchant peer and stamped and stored by the broker) using macro payment mechanisms such as credit card payment schemes. The stamped unspent digital note should be kept secret and protected by the entity possessing it, either by the broker or by the customer.

A merchant peer can redeem the value of digital cash before time T1 associated. The merchant peer has to reveal the broker stamp to the broker. If the broker stamp is valid, the broker credits the merchant peer's account and marks the digital note as spent. Similarly, the customer peer can refund its unspent digital note before expiration, that is, after time T2 but before time T3. A time is elapsed between T1 and T2 in order to avoid clock synchronization problems.

For the transaction between merchant and customer peers, the customer sends the stamped digital note excluding the broker stamp to the merchant, who verifies that this *SVC* is unspent, and that current time is less than T1. Then, a two-party fair exchange protocol is assumed for the exchange of the digital good and the broker stamp.

6.2.3 A New Approach

This section introduces a novel P2P content distribution and payment protocol in which the load on the peer that plays the role of merchant is significantly reduced, hence motivating the participation of peers in this type of evolving e-commerce. The basic idea is the delegation of the merchant role to the agent during the payment phase. With this change, the customer peer does not need special digital notes for each merchant peer. Instead, he can buy digital contents from several merchant peers by interacting with only one agent who represents these merchant peers [6]. Thus, the payment view of the P2P network changes to a new model (see Figure 6.6).

A prior notation needed for the complete understanding of the protocol is as follows.

- M, C, B: merchant peer, customer peer, and broker/bank, respectively
- IN, TTP: agent and trusted third party, respectively
- $DigitalContent$: digital content that the merchant peer M sells to the customer peer C
- $descr$: description of the digital content that C obtains before starting purchase (e.g. from the agent's web site)
- P_{ID}: identifier of the digital content and its price
- $utsn$: unique transaction serial number
- $L = (utsn, P_{ID})$: label of the current transaction

[6] The more merchant peers this agent represents, the more universal this digital notes will be.

- k_c: session key generated by the agent and used by M to encrypt the digital content
- $Cipher = E_{k_c}(DigitalContent)$: ciphertext of the digital content encrypted with k_c
- $dc = h(DigitalContent)$: digest of the digital content
- $IntegritySign = S_M(dc, descr, P_{ID})$: digital content verification code generated by M and available at the agent's web site
- $k_{TTP} = E_{TTP}(IN, M, k_c)$: ciphertext of the session key encrypted with the TTP's public key

6.2.3.1 P2P Service and Payment Protocol

In this protocol, we assume that each peer (acting as a customer) can set up a secure and confidential channel (SSL or IPSec) with its agent, broker/bank, and the TTP, and the agent can also establish such a channel with the broker and the TTP. The protocol consists of a main protocol and two sub-protocols. In the normal situation, only the main protocol will be executed among the customer peer, the agent, and the merchant peer, while the TTP is off-line and not involved. If there is something wrong in a transaction, the agent can initiate the *cancel* sub-protocol and the customer peer can initiate the *resolve* sub-protocol to terminate the transaction without loss of fairness.

The agent will prepare the digital notes for the merchants that it represents, and send these digital notes to the broker for stamping. The customer can obtain the stamped digital notes from the broker using a macro payment mechanism, and use these digital notes in purchase of digital goods/contents from any merchant (via its agent).

Suppose the customer C has obtained some digital notes from the broker B. At the beginning, C accesses information in the agent IN's web page, and downloads *descr*, P_{ID}, and *IntegritySign*. Then, C launches the following P2P service and payment main protocol.

1. $C \rightarrow IN$: $M, L, DigitalNote, SVC$
   IN checks and IF correct follows
2. $IN \rightarrow M$: $M, L, k_c, k_{TTP}, SVC, S_{IN}(M, L, k_c, K_{TTP}, SVC)$
3. $C \leftarrow M$: $IN, L, Cipher, dc, k_{TTP}, S_M(IN, C, L, h(Cipher), dc, k_{TTP}, SVC)$
4. $C \rightarrow IN$: $BrokerStamp$
5. $IN \rightarrow C$: $L, k_c, S_{IN}(M, C, L, k_c)$

At Step 1, the customer C sends a digital note to the agent IN. IN makes all necessary checks on the digital note before notifying the merchant M at Step 2 that there is a request pending from C. Such checking includes that the current time is earlier than the deadline of redemption T1, and that the digital note has not been spent yet. If correct, IN provides M with its signature which could be used to prove the amount of payment to be credited to M's account (if the transaction is completed) and the session key for encryption of the digital content. IN also encrypts the session key k_c with the TTP's public key (in order to reduce the computational load on the merchant peer host). After verifying the purchase request redirected by *IN*, *M* prepares the encrypted digital content and its signature which could be used to prove the origin of the digital content. *C* retrieves the encrypted digital content from *M* at Step 3. Prior to submit the broker stamp to *IN* at Step 4, *C* verifies whether *M* is committed that the digital content sent in ciphertext is the one as *C* expected (by checking *IntegritySign*). *IN* releases the session key at Step 5 after obtaining the valid broker stamp from *C*. If the above protocol is executed successfully, the customer peer obtains k_c for the decryption of *Cipher*, and thus the digital contents. The agent obtains the broker stamp. Then the agent can send the broker stamp to the broker for redemption.

If *IN* does not receive the broker stamp in a pre-determined amount of time before T1, it can launch the following *cancel* sub-protocol.

$$\begin{aligned}
4'.\ IN \rightarrow TTP &: IN, M, L, SVC, S_{IN}(cancel, IN, M, L, SVC)\\
&\quad \text{IF not resolved THEN}\\
5'.\ IN \leftarrow TTP &: S_{TTP}(cancel, IN, M, L, SVC)\\
&\quad \text{ELSE}\\
5'.\ IN \leftarrow TTP &: BrokerStamp
\end{aligned}$$

In such a case, *IN* sends to the TTP a cancel request. Then, if the protocol has not been resolved, the TTP verifies *IN*'s signature on the request. If correct, the TTP signs a cancel affidavit. If the protocol was resolved by *C*, the TTP gives *IN* access to retrieve the valid broker stamp. Note that the agent can obtain the broker stamp and a cancel affidavit, which results in an unfair situation. Nevertheless this approach considers a *poll* solution to revoke the broker stamp redemption. In this poll solution, the broker has access to the cancel affidavits from the TTP server and then, it will execute that operation (searching for fraudulent redemption operations) before redeeming the agent. Note that, however, a refund operation after T2 is allowed.

If *C* does not get the session key for decryption of the digital content in the main protocol before time T1, it appeals to the TTP in a *resolve* sub-protocol.

$$\begin{aligned}
5'.\ C \rightarrow TTP &: IN, M, L, h(Cipher), dc, k_{TTP}, SVC, BrokerStamp,\\
&\quad S_M(IN, C, L, h(Cipher), dc, k_{TTP}, SVC)\\
&\quad \text{IF not cancelled THEN}\\
6'.\ C \leftarrow TTP &: k_c\\
&\quad \text{ELSE}\\
6'.\ C \leftarrow TTP &: S_{TTP}(cancel, IN, M, L, SVC)
\end{aligned}$$

In such a case, *C* sends to the TTP all the information received from *M* as well as the broker stamp. If the protocol has not been cancelled, the TTP verifies *M*'s signature, such that the TTP is not used as a decrypting oracle by the customer, and checks whether the hash of the broker stamp equals *SVC*. If everything is positive, the TTP decrypts k_{TTP}, verifies that the key k_c is intended for *IN* and *M*, and finally stores k_c for *C*'s access. If the protocol has been revoked by *IN*, the TTP will provide a cancel affidavit.

Some financial issues should be taken into account. The agent could send all broker stamps to the broker in batch mode. Similarly, the broker could credit the agent account in batch mode, giving an elapse time for this operation, such that it can retrieve from the TTP all the cancel affidavits and revoke the broker stamp redemption (and hence the agent's bank account credit operation) if needed. None of these financial assumptions seems to be hard to achieve.

Finally, we would like to state that a complementary design based on a reputation system [46] could help to boost the P2P commerce. Reputation is the only mechanism available to peers in order to evaluate a candidate provider of a requesting service in terms of quality, reliability and correctness and thus plays a significant role in the selection of agents. So, if a situation arises in which an agent is misbehaving, a network of reputation can "mark" this entity, thus preventing the next fraudulent action.

6.2.3.2 Dispute Resolution

Disputes can arise, and here is how the resolution with an arbitrator proceeds for all the entities involved in such a dispute.

Origin of digital content: If M denies having sent a particular $DigitalContent$, then C gives $descr, IntegritySign, IN, L, P_{ID}, Cipher, dc, k_{TTP}, k_c, SVC$, and M's signature to the arbitrator. The arbitrator checks

- $descr$ fits with $DigitalContent$
- $dc = h(DigitalContent)$
- $IntegritySign$ is M's signature on $(dc, descr, P_{ID})$
- $k_{TTP} = E_{TTP}(IN, M, k_c)$
- $DigitalContent = D_{k_c}(Cipher)$
- M's signature on $IN, L, h(Cipher), dc, k_{TTP}, SVC$

If all the above checks are positive, the arbitrator concludes that the digital content is from M. If C receives a wrong digital content, some of the first three checks in the list might be false. However, C can demonstrate the misbehavior of M with $IntegritySign$. If M can present IN's signature on a different session key k_c for the same transaction L, the arbitrator concludes that IN is the misbehaving party.

Payment received by IN: A possible dispute could arise between M and IN if the latter did not credit M's account after transferring a broker stamp to B for redemption. IN could obtain incentives from the merchant peers, and the commission depends on how many successful payments it carries out. However, IN may try to keep the entire payment of the digital good. If IN denies having completed a transaction (L, SVC), M should present to the arbitrator M, L, k_c, k_{TTP}, SVC and IN's signature on it. Then, the arbitrator checks the signature and, if IN cannot present a cancel affidavit signed by the TTP for that transaction (L, SVC), the arbitrator concludes that IN completed the transaction and must pay M for it. Note that if IN tries to misbehave by completing the transaction and obtaining a cancel affidavit, it will eventually succeed. But *B* will prevent it from crediting *IN*'s account if *B* obtains the cancel affidavit from the TTP. In this case, only *C* will be benefited. *C* obtains the k_c and thus the digital content while *IN* is not redeemed. *C* could spend the same stamped digital note later again, or get refunded from *B*.

Invalid broker stamp: If *C* tries to misbehave by sending an invalid broker stamp, two cases are possible.

- *C* sends the invalid broker stamp at Step 4. Assume that *IN* is not going to collude with *C* as was discussed in 6.2.1, *IN* will detect an invalid broker stamp using *SVC* and will reject it.
- *C* stops the protocol at Step 4, and contacts the TTP to resolve. If the transaction has not been cancelled, the TTP will check with *SVC* signed by *M* that the broker stamp provided by *C* is valid before providing k_c to *C*. Therefore, when the agent access the TTP in order to cancel the protocol after waiting a reasonable time without receiving the broker stamp from *C*, it will obtain a valid *BrokerStamp* for the transaction.

Origin of SVC: If *M* colludes with *C* and sends at Step 3 a *SVC* which has already been spent by *C*, then the latter could contact the TTP and successfully get k_c with the resolve sub-protocol. Whenever *IN* tries to fetch the broker stamp from the TTP, it will discover *M*'s fraudulent behavior and go to the arbitrator. If *M* can present *IN*'s signature on the same transaction (L, SVC), the arbitrator concludes that the misbehaving party is *IN*, and *IN* will have to pay for the transaction to *M*. Otherwise, *M* is identified as the colluding party.

6.2.4 Requirements Fulfillment

Now, let us informally analyze whether this new P2P service and payment protocol satisfies the requirements described in Section 6.2.1.

- *Confidentiality*: The digital goods/contents are disclosed only to the intended customer peer. Although the agent knows the deciphering key, it does not have the knowledge about the encrypted digital content if it is transmitted over a private peer-to-peer channel from *M* to *C*. Similarly, the TTP, if involved, cannot get the encrypted digital contents either. There is a bunch of secure P2P software which allows to establish private channels among users (like JXTA technology [6] from SUN, Magi P2P, GNUnet [7]).
- *Payer anonymity*: Only the first level of anonymity is reached, that is, untraceability. A customer peer never needs to reveal its identity except its IP address for receiving messages during the protocol.
- *Fairness*: Fairness is achieved under the assumption of no collusion between the agent and the customer. In the main protocol, fairness will not be lost until Step 3 since neither the agent has obtained the broker stamp nor the customer

gets the key to decrypt the digital content. After Step 3, both of them obtain what they expect (digital good and broker stamp) or none of them obtains any valuable information. As the TTP intervenes, it can produce affidavit tokens for later disputes. The cancel token can be used to solve an unfair situation (i.e., agent obtains the broker stamp but the consumer peer does not get the needed session key for deciphering the digital content). That is, only weak fairness is achieved.

- *Timeliness*: After notifying the merchant of a request for purchase at Step 2, the agent has the ability of cancelling the protocol, if needed, in order to reach the end of the protocol without breach of fairness. On the other hand, the customer can terminate the protocol at any time before releasing the broker stamp, or initiate the resolve sub-protocol after Step 4. However, deadlines used for the underlying P2P payment protocol (T0, T1, T2, T3) provide this solution with a synchronous timeliness property. Due to this synchronicity, as usual, resilient channels are expected between agent and customers and the TTP. This will ensure that digital notes can be redeemed or refunded on time.
- *Non-repudiation*: Proofs of origin of digital contents and payment received by the merchant are discussed in Section 6.2.3.2. If the digital content provided at the end of a successful execution of the protocol does not fit with the description signed by the merchant on *IntegritySign*, the customer can obtain the evidence from Step 3 for a dispute resolution. If the agent cheats the merchant by falsely denying receipt of the payment from the customer, the merchant can get the evidence from Step 2 for a dispute resolution.
- *Lightweight merchant*: For each protocol run, the merchant only needs one signature verification and generation (although *IntegritySign* token is also generated by the merchant, this operation can definitely be carried out in an off-line process). More importantly, the merchant only receives one service request from the agent, and makes the encrypted digital contents available to the customer. The merchant does not need to take care of advertisement and payment.
- *Efficiency*: The non-repudiation protocol is optimistic, therefore the TTP does not need to intervene when, either participants behave honestly, or there are not channel failures.
- *Policy*: Again, policy about which cryptographic algorithms to use for evidence generation, which arbiter to use in case of disputes etc., needs to be defined for a peer to peer application. The minimum elements it needs to contain are those explicitly mentioned through the description of the protocol as, for instance, the guideline to use when entering a dispute resolution process. Guidelines on how to make use of the digital money (redeem and refund operations) are established by the P2P payment protocol explained in this section. We make use of the verbs "SHALL" "MUST" and "MAY" which are interpreted as in [40].

  a) Rules for evidence generation and verification:
  Cryptographic algorithms to be used, e.g. RSA with 1024-bits keys, would be sufficient, since the value of the operations is not too high. Keys format and extensions are needed if X.509v3 digital certificates are to be used for verifica-

tion. The policy MUST explicitly define the structure of every element used in the protocol. Of course, all legal restrictions coming from use of cryptography MUST be inherited.

- What evidence should be generated in the non-repudiation service?
  The agent MUST generate the following evidence: (i) evidence of operation intended to the merchant peer M, (ii) evidence of key k_c intended to the customer, and (iii) evidence of cancel request (for authentication and registry purposes, since we assume the TTP is trusted and will not generate cancel affidavits on its own). The merchant MUST generate evidence of origin to the client peer while the latter MAY not provide any kind of evidence.
- Which TTP should be involved in evidence generation?
  Explicitly stated.
- What elements should be included in the evidence?
  Explicitly stated.
- Which type of evidence should be generated?
  Explicitly stated.

b) Rules related to evidence transfer:
   - Which non-repudiation protocol will be used?
     Explicitly defined.
   - What are the channel assumptions?
     There is no channel assumption other than as stated in the protocol description and discussion.

c) Rules for evidence storage:
   The TTP's database storage SHALL make an hourly backup. Only the TTP SHALL have write permissions and every input, query and operation in general MAY be logged (and the format will be established by the TTP itself). Time-stamps signed by the TTP for evidence validity and logging purposes MAY be issued as well.
   - What mechanism will be used for maintaining the validity of evidence?
     Trusted time-stamping.
   - How long should the evidence be stored?
     Six months.
   - Does the evidence need to be confidential?
     No.
   - What are the access control rules for accessing evidence in TTP's database?
     Write permissions for the TTP. Read permissions for peers and the agent. The access control procedure MAY be established by the TTP.

d) Evidence use: Explicitly stated in the dispute resolution process guidelines.

e) Dispute resolution process:
   - Which entity will play the role of adjudicator?
     It MUST be stated in the protocol policy.

  - Which parties should be involved in dispute resolution? And which TTP?
    Only participants related to the dispute. TTP directory MAY also need to be audited (processed).
  - What evidence should be provided?
    Explicitly defined.
  - What are the rules followed in the resolution process itself?
    Explicitly defined.

- *Verifiability of TTP*: The TTP needs to be trusted, so it is not verifiable. On the other hand, agents's behavior is verifiable and can be disputed by the participating entities in the P2P scenario.
- *Transparency of TTP*: Except when the TTP generates cancel affidavits, it behaves in a transparent way. Thus, we can affirm that the TTP is only partially transparent.

6.2.5 Practical View

In order to give a more practical view of the involvement of the entities in this approach, let us depict an instantiated execution of the protocol. Let's consider a typical P2P scenario where one of the peers tries to purchase a file. This file description and associated advertisement are hosted in an agent server. Note that, if an agent advertises similar files (belonging to different peers), an analysis about the competition between different peers in the distribution of the contents should be undertaken. A preliminary study can be found in [22].

In this scenario (see Figure 6.6), a previous contract or relation exists between a merchant peer and an agent. This is something totally necessary, since at least, the merchant must register to use the agent's hosting services. As analyzed earlier, the merchant peer has a very light participation in the protocol; an important property that will facilitate the involvement of peers distributing, in an exchange for a small amount of money, digital contents over P2P networks.

The different steps of this application are as follows.

1. A peer, who is surfing the web, visits *http://www.curious-papers.com* and once inside, clicks the section "Snakes". He reads the abstract or description *descr* and decides to buy it. So, he clicks on the button "Purchase". This operation forms a transaction label L, downloads the content verification code *IntegritySign* token (whose signature is verified by an applet), and uploads a valid stamped digital note (excluding broker stamp).
2. The agent's server verifies the validity of the digital note, and checks that SVC is unspent. If correct, it redirects this request to another peer who owns the paper, along with the product information received from the customer peer (L), the key needed to encrypt the contents (k_c), and the fingerprint of the broker stamp (SVC).

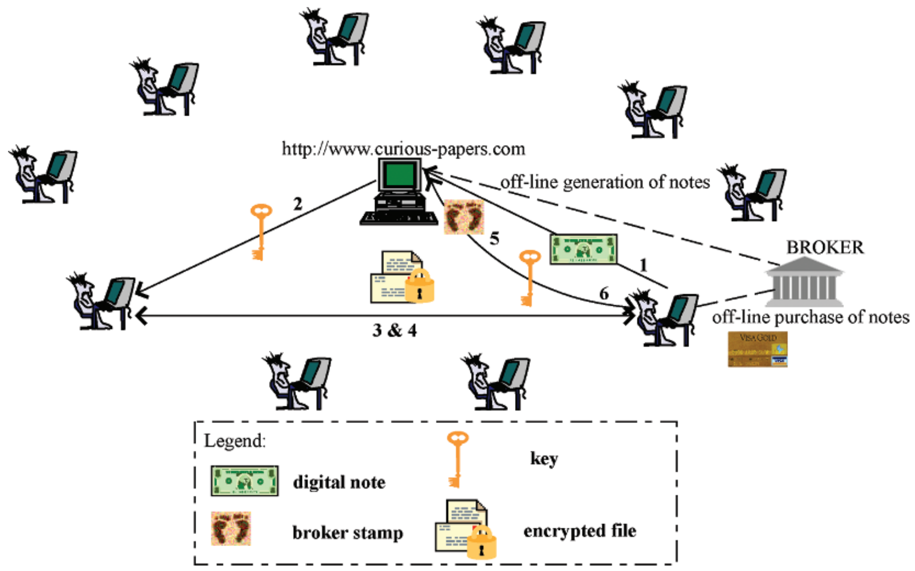
http://www.curious-papers.com
off-line generation of notes
BROKER
off-line purchase of notes
VISA GOLD
1
2
3 & 4
5
6
Legend:
digital note
key
broker stamp
encrypted file

Fig. 6.6 Application Scenario

3. The merchant peer prepares the encrypted version of the paper (*Cipher*), and generates a signature. Then, it notifies the customer peer to retrieve.
4. The customer peer downloads the encrypted paper and the merchant peer's signature. An add-in component in the customer peer's browser verifies the merchant peer's signature and the fingerprint of the broker stamp. If correct, the customer peer is asked for approval of the description signed by the merchant peer by clicking on the button "OK". In this step, the content verification code token is checked using *dc* received.
5. Then the customer peer's computer sends the broker stamp to the agent. A window of notification should pop up to advise the customer that once the broker stamp is sent, it will be the non-return point of the transaction. The add-in component waits for the session key.
6. After receiving the broker stamp, the agent proceeds to send the session key. At the customer peer side, the add-in component will verify the agent's signature, decrypt the paper, and display the paper to the customer.
7. If the agent does not receive the broker stamp within a determined time (before T1) a cancel sub-protocol can be launched, obtaining either a cancel affidavit for the digital note or the valid broker stamp from the TTP.
8. If the customer peer does not receive the session key within a determined time before T2, the add-in component processes the XML non-repudiation policy file to extract the identifier of the TTP and redirect a request to the TTP to resolve

the protocol. If the session key is received, the add-in component will decrypt the paper and display the paper to the customer. If a TTP signed cancel affidavit is received, the add-in component will pop up a window to notify the customer that the transaction has been cancelled.

Redemption and refund phases proceed according to the underlying P2P payment protocol. The broker has the ability of cancelling the redemption phase as stated before.

Note that merchant peers have to prepare (encrypt) the content in each protocol execution, since it depends on the session key k_c generated by the agent which changes from one instance to another. One question arises whether it is possible to distribute a fixed encrypted content. The main drawback of this approach is free-riding. Although the proposal introduced does not avoid this possibility, it is practically more difficult to distribute the decrypted whole digital content than only the deciphering key. That is, if the file remains encrypted with the same key, another user needs just to execute the protocol up to step 3 and halt. Later, it will receive the key from another user. On the other hand, if the file is encrypted each time with a different key, provided by the agent and targeted to a specific client, a free rider needs the decrypted content from other peer, which complicates a bit more free sharing. However, note that free sharing is unavoidable, and only incentive schemes (like this one) can act against it.

6.3 Free Roaming Agent Result-Truncation Defense

Mobile agents are software programs that live in computer networks, performing their computations and moving from host to host as necessary to fulfill their goals [83]. *Free roaming* mobile agents are free to choose their respective next hops dynamically based on the data they acquired from their past journeys.

Mobile agents are especially useful in electronic commerce, and have attracted lot of research interest. Furthermore, these mobile agents could implement the intermediary entity introduced in protocols along this book. Nevertheless, as stated in [84], there are still many security issues on mobile agents to be addressed. Free-roaming agent security goals could be summarized as

- protection of the host from malicious code in the mobile agent, and
- protection of the agent from a malicious host trying to tamper the code and the agent data.

The community has initially placed more attention on the first problem and come out with some solutions, since the problem is similar to the one that already existed with Java and ActiveX technologies in which the host has to run software coming from untrusted sources. The most popular solution for such a problem is rather simple, the *sandbox* solution; i.e., an agent cannot control the machine in which it is executed. The agent is executed in a sandbox that blocks the access of the agent to the real machine. This feature has been implemented in Java Mobile Code, Telescript, and SafeTcl.

With respect to the second problem, we can distinguish two principal subproblems. In the first case, a malicious host tries to tamper the agent's code. To address this problem, computing with encrypted functions such as *homomorphic* encryption schemes is under research [117]. In the second case, a malicious host tries to tamper the data carried by the agent. For instance, in a scenario that a free-roaming agent is used to collect offers for an air-ticket, a malicious host may try to "hijack" or "brainwash" the previously collected data to favour its offer. In our case, assuming the mobile agent carries different digital evidence collected in its path along a multi-party application, we will be willing to protect these evidence from being extracted from the agent without the originator knowledge while other optional properties like confidentiality might be necessary as well.

Let's take the protocol introduced in Section 4.2.4.1 as an example. In this protocol, as shown in the message flow, the intermediary can depart from the originator and exchange the different messages (evidence) with multiple recipients in sequence, visiting all of them. As depicted in the protocol flow, the intermediary or agent will return back to the originator before step 5. In this scenario, among other requirements enumerated in the following sections, we need to preserve the agent from being tampered with.

6.3.1 Security Requirements

Suppose an agent departing from host computer O will obtain a list of encapsulated messages $M_1, \cdots, M_n$ from different hosts $R_1, \cdots, R_n$ selected dynamically when the agent roams over the network. The security properties on the agent data protection defined in [83] and extended in [44] are as follows.

- *Data Confidentiality*: Only the originator O can extract the encapsulated messages $M_1, \cdots, M_n$.
- *Non-repudiability*: R_i cannot deny submitting M_i once O receives M_i.
- *Forward Privacy*: No one except the originator O can extract the identity information of the hosts $R_1, \cdots, R_n$ by examining the chain of encapsulated messages.
- *Forward Integrity*: None of the encapsulated messages M_i can be modified.
- *Publicly Verifiable Forward Integrity*: Anyone can check the integrity of the chain of encapsulated messages. This property is optional.
- *Insertion Defense*: No new message can be inserted in $M_1, \cdots, M_n$ without being detected.
- *Truncation Defense*: No existing message can be removed from $M_1, \cdots, M_n$ without being detected.

Several schemes have been proposed to protect agent data. Yee proposed to use a *Partial Result Authentication Code* (PRAC) to ensure the integrity of the messages acquired from the hosts [128]. In this scheme, an agent and its originator keep a list of secret keys, or a key generating function. The agent uses a key to encapsulate the collected message and then destroys the key. However, a malicious host may keep the key or the key generating function. When the agent revisits the host or visits another host conspiring with it, a previous message or series of messages could be modified, without being detected by the originator.

Karjoth et. al. extended Yee's results [83]. In the KAG scheme, each host generates a signing key for its successor and certifies the corresponding verification key. Using the received signature/verification key pair, a host signs its partial result and certifies a new verification key for the next host. Their scheme could resist the modification attack in Yee's scheme but not a two-colluder truncation attack. In this attack, two visited hosts (or one revisited host) can collude to discard the partial results collected between their respective visits.

Cheng and Wei proposed a scheme to defense against the truncation of computation results of free-roaming agents [44]. Cheng-Wei scheme is effective against such an attack in most cases. However, it still suffers from the truncation attack when a special loop is established on the path of a free-roaming agent. In order to avoid the truncation of computation results we propose several amendments.

6.3.2 Cheng-Wei Protocol

Let us first sketch Cheng-Wei Protocol (N1) presented in [44]. Among all the security properties that the protocol claims to achieve is the truncation defense, and in particular, defense against a *two-colluder truncation attack*. In this scenario, an attacker W captures an agent with encapsulated messages $M_1, \cdots, M_{j-1}, M_j, \cdots, M_n$ and colludes with host R_j trying to truncate all the messages after M_j and insert the attacker's messsages to get the new chain $M_1, \cdots, M_{j-1}, M'_j, \cdots, M_W$.

A public key infrastructure is assumed in the mobile agent environment. Each host has a certified private/public key pair $(\bar{u}_i, u_i)$. Given a signature expressed as $S_i(m)$, we assume that anyone could deduce the host identity. The chain of encapsulated messages $M_1, M_2, \cdots, M_n$ is an ordered sequence. Each entry of the chain depends on some of the previous and/or succeeding members. A chaining relation specifies the dependency.

An important definition given by Cheng and Wei in order to avoid *interleaving* attacks (proposed by Roth [116]) is as follows.

An agent is defined as $A = (I, Co, St)$ where I is the identity, Co is the code and St is the state of the agent. Both I and Co are assumed to be static while St is variable. I is in the form of (ID_A, Seq_A), where ID_A is a fixed identity bit string of the agent and Seq_A is a sequence number which is unique for each agent execution. The originator signs H_A, where $H_A = h(I, Co)$ is the agent integrity checksum and $S_0(H_A)$ is the certified agent integrity checksum. *The agent carries this certified checksum, allowing the public to verify the integrity of I and Co and deduce the identity of the originator.*

In interleaving attacks, the attacker tries to use a previous host as an oracle, running its own agent but with the attacked agent's chain of messages. Since the *certified agent integrity checksum* is sent in each transmission between the hosts, each host can verify the consistency of the chain of messages (specially the one belonging to the agent's owner O) and the code. With this definition, an agent execution is uniquely identified.

Protocol N1 uses a co-signing mechanism in which a host needs the preceding host's signature on its encapsulated message before sending it to the next host. It also depends on the signatures of an integrity checksum generated by the two associated preceding hosts; such that, the current host is able to verify that the preceding host did not insert two messages in a self-looping mode.

The model and cryptographic notation used in the description of the protocol N1 is summarized in Tables 6.3.2 and 6.3.2, respectively.

$O = R_{n+1}$	The originator
$R_i, 1 \leq i \leq n$	A host computer
$m_i, 1 \leq i \leq n$	A message from R_i. The identity of R_i is explicitly specified in m_i
$M_i, 1 \leq i \leq n$	An encapsulated message (cryptographically protected m_i) from R_i
$h_i, 1 \leq i \leq n$	An integrity check value associated with M_i and the next hop
$M_0, M_1, .., M_n$	The chain of encapsulated messages

Table 6.1 Model Notation

r_i	A random number generated by R_i
$(\bar{u}_i, u_i)$	Private and public key pair of R_i
$(\bar{\mu}_i, \mu_i)$	Temporally private and public key pair of R_i
$E_i(X)$	A message X encrypted with the public key u_i of R_i
$S_i(X)$	A signature of R_i on message X with its private key $\bar{u}_i$
$Ver(\sigma, u)$	A signature verification function for signature σ with public key u
$[X]$	Message X sent via a confidential channel

Table 6.2 Cryptographic Notation

Protocol N1 consists of three parts: agent creation (with a dummy message m_0), agent migration at R_1, and agent migration at R_i $(2 \leq i \leq n)$ (see Figure 6.7).

Agent Creation

1. *Message encapsulation*

$O: h_0 = h(r_0, R_1)$
$O: M_0 = S_0(E_0(m_0, r_0), I, h_0, \mu_1)$
$O: \sigma_0 = S_0(h_0)$

2. *Agent transmission*
   $O \rightarrow R_1: \ M_0, [\bar{\mu}_1, \sigma_0]$

Agent Migration at $\mathbf{R_1}$

3. *Agent verification*
   R_1 : receive $M_0, \bar{\mu}_1, \sigma_0$
   R_1 : $Ver(M_0, u_0)$, and recover I, h_0, μ_1
   R_1 : $Ver(\sigma_0, u_0)$

4. *Interactive message encapsulation*
   $R_1: \quad h_1 = h(M_0, r_1, R_2)$
   $R_1 \rightarrow O: temp_1 = E_0(S_1(m_1, \sigma_0), r_1), h_1, \mu_2$
   $O: \quad M_1 = S_0(temp_1)$
   $O \rightarrow R_1: M_1$
   $R_1: \quad Ver(M_1, u_0)$
   $R_1: \quad \sigma_1 = S_1(h_1)$

5. *Agent transmission*
   $R_1 \rightarrow R_2: \ M_0, M_1, [\bar{\mu}_2, \sigma_0, \sigma_1]$

Agent Migration at $\mathbf{R_i}$ $(2 \leq i \leq n)$

6. *Agent verification*
   $R_i: receive\ M_0, \cdots, M_{i-1}, \bar{\mu}_i, \sigma_{i-2}, \sigma_{i-1}$
   R_i : $Ver(M_0, u_0)$, and recover I, h_0, μ_1
   R_i : $Ver(M_1, u_0)$, and recover h_1, μ_2
   R_i : $Ver(M_k, \mu_{k-1})$, and recover h_k, μ_{k+1} $recursively\ for\ 2 \leq k \leq i-1$
   R_i : $Ver(\sigma_{i-2}, u_{i-2})$
   R_i : $Ver(\sigma_{i-1}, u_{i-1})$
   R_i : verify $R_{i-2} \neq R_{i-1}$

7. *Interactive message encapsulation*
   $R_i: \quad h_i = h(M_{i-1}, r_i, R_{i+1})$
   $R_i \rightarrow R_{i-1}: temp_i = E_0(S_i(m_i, \sigma_{i-2}, \sigma_{i-1}), r_i), h_i, \mu_{i+1}$
   $R_{i-1}: \quad M_i = S_{\bar{\mu}_{i-1}}(temp_i)$
   $R_{i-1} \rightarrow R_i: M_i$
   $R_i: \quad Ver(M_i, \mu_{i-1})$
   $R_i: \quad \sigma_i = S_i(h_i)$

8. *Agent transmission*
   $R_i \rightarrow R_{i+1}: \ \{M_k | 0 \leq k \leq i\}, [\bar{\mu}_{i+1}, \sigma_{i-1}, \sigma_i]$

It is assumed that there is an authenticated channel between R_i and R_{i-1} in the co-signing process. R_{i-1} will store the records of the departed agents in such a way

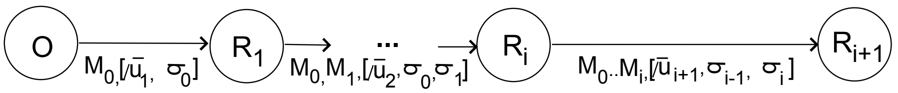

Fig. 6.7 Agent Transmission

that it would only sign once for R_i on the partial results of a particular departed agent.

Each host is responsible for checking and verifying all the messages as well as the chaining relation in the agent state whenever it arrives. If any of the agent verifications fails or R_i receives twice the same agent from R_{i-1} then it rejects the agent.

6.3.3 Security Analysis

This section presents a two-colluder truncation attack on Cheng-Wei Protocol. The protocol bases its security, among other properties, on the impossibility for a host computer R_j, which tries to collude with a host R_W $(W > j)$, to request a signature on another encapsulated message M'_j from the previous host R_{j-1} in order to truncate all of the encapsulated messages collected from the host computers between R_{j+1} and R_W.

Our attack scenario is as follows (see Figure 6.8). A free roaming agent that has visited a host R_j re-visits R_j after visiting another host R_{j+1}, in other words, $R_{j+2} = R_j$. Then, no matter how the agent will roam after visiting R_{j+2}, R_{j+2} can always collude with a host R_W being visited by the agent to truncate all of the encapsulated messages collected after R_{j+2}. R_{j+2} need not collude with R_{j+1} in order to get a new co-signed encapsulated message M'_{j+2} from R_{j+1}. Instead, R_{j+2} can sign it by itself with the temporary private key $\bar{\mu}_{j+1}$ which was generated by R_j where $R_j = R_{j+2}$!

Suppose the agent visited $O, R_1, \cdots, R_j, R_{j+1}, R_{j+2}$ $(= R_j), R_{j+3}, \cdots, R_W$, and collected the encapsulated messages

$$M_0, M_1, \cdots, M_j, M_{j+1}, M_{j+2}, M_{j+3}, \cdots, M_W$$

The processes of interactive message encapsulation and agent transmission at hosts R_j, R_{j+1} and R_{j+2} $(= R_j)$ are as follows.

When the agent is at $\mathbf{R_j}$:

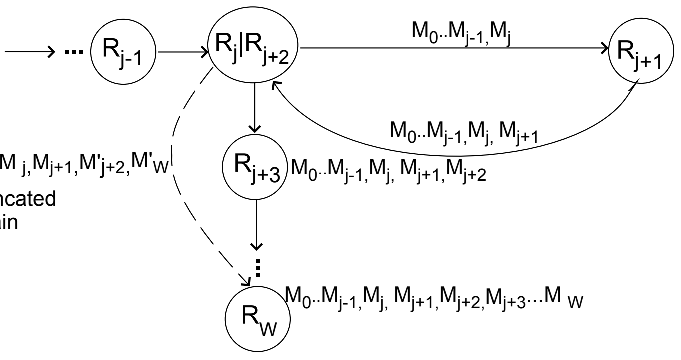
R_{j-1}
R_j|R_{j+2}
R_{j+1}
R_{j+3}
R_W
M_0..M_{j-1},M_j
M_0..M_{j-1},M_j, M_{j+1}
M_0..M_{j-1},M_j, M_{j+1},M_{j+2}
M_0..M_{j-1},M_j, M_{j+1},M_{j+2},M_{j+3}...M_W
M_0..M_{j-1},M_j,M_{j+1},M'_{j+2},M'_W
truncated chain

Fig. 6.8 Attack Scenario

$R_j :\quad h_j = h(M_{j-1}, r_j, R_{j+1})$
$R_j \rightarrow R_{j-1} : temp_j = E_0(S_j(m_j, \sigma_{j-2}, \sigma_{j-1}), r_j), h_j, \mu_{j+1}$
$R_{j-1} :\quad M_j = S_{\bar{\mu}_{j-1}}(temp_j)$
$R_{j-1} \rightarrow R_j : M_j$
$R_j :\quad Ver(M_j, \mu_{j-1})$
$R_j :\quad \sigma_j = S_j(h_j)$
$R_j \rightarrow R_{j+1} : \{M_k | 0 \leq k \leq j\}, [\bar{\mu}_{j+1}, \sigma_{j-1}, \sigma_j]$

When the agent is at $\mathbf{R_{j+1}}$:

$R_{j+1} :\quad h_{j+1} = h(M_j, r_{j+1}, R_{j+2})$
$R_{j+1} \rightarrow R_j :\quad temp_{j+1} = E_0(S_{j+1}(m_{j+1}, \sigma_{j-1}, \sigma_j), r_{j+1}), h_{j+1}, \mu_{j+2}$
$R_j :\quad M_{j+1} = S_{\bar{\mu}_j}(temp_{j+1})$
$R_j \rightarrow R_{j+1} :\quad M_{j+1}$
$R_{j+1} :\quad Ver(M_{j+1}, \mu_j)$
$R_{j+1} :\quad \sigma_{j+1} = S_{j+1}(h_{j+1})$
$R_{j+1} \rightarrow R_{j+2} : \{M_k | 0 \leq k \leq j+1\}, [\bar{\mu}_{j+2}, \sigma_j, \sigma_{j+1}]$

When the agent is at $\mathbf{R_{j+2}}$ $(= \mathbf{R_j})$:

$R_{j+2} :\quad h_{j+2} = h(M_{j+1}, r_{j+2}, R_{j+3})$
$R_{j+2} \rightarrow R_{j+1} : temp_{j+2} = E_0(S_{j+2}(m_{j+2}, \sigma_j, \sigma_{j+1}), r_{j+2}), h_{j+2}, \mu_{j+3}$
$R_{j+1} :\quad M_{j+2} = S_{\bar{\mu}_{j+1}}(temp_{j+2})$
$R_{j+1} \rightarrow R_{j+2} : M_{j+2}$
$R_{j+2} :\quad Ver(M_{j+2}, \mu_{j+1})$
$R_{j+2} :\quad \sigma_{j+2} = S_{j+2}(h_{j+2})$
$R_{j+2} \rightarrow R_{j+3} : \{M_k | 0 \leq k \leq j+2\}, [\bar{\mu}_{j+3}, \sigma_{j+1}, \sigma_{j+2}]$

Suppose the agent roams to $R_{j+3}, \cdots, R_W$ after visiting R_{j+2}, and then R_W colludes with R_{j+2} $(= R_j)$. R_{j+2} can initiate the truncation attack with the following processes of interactive message encapsulation and agent transmission.

$\mathbf{R_{j+2}}$ $(= \mathbf{R_j})$ prepares:

$R_{j+2}:\quad h'_{j+2} = h(M_{j+1}, r_{j+2}, R_W)$
$R_{j+2}:\quad temp'_{j+2} = E_0(S_{j+2}(m'_{j+2}, \sigma_j, \sigma_{j+1}), r_{j+2}), h'_{j+2}, \mu_W$
$R_{j+2}:\quad M'_{j+2} = S_{\bar{\mu}_{j+1}}(temp'_{j+2})$
$R_{j+2}:\quad \sigma'_{j+2} = S_{j+2}(h'_{j+2})$
$R_{j+2} \rightarrow R_W: \{M_k | 0 \leq k \leq j+1, M'_{j+2}\}, [\bar{\mu}_W, \sigma_{j+1}, \sigma'_{j+2}]$

When the agent is at $\mathbf{R_W}$:
$R_W:\quad h'_W = h(M'_{j+2}, r'_W, R_{W+1})$
$R_W \rightarrow R_{j+2}:\quad temp'_W = E_0(S_W(m'_W, \sigma_{j+1}, \sigma'_{j+2}), r'_W), h'_W, \mu_{W+1}$
$R_{j+2}:\quad M'_W = S_{\bar{\mu}_{j+2}}(temp'_W)$
$R_{j+2} \rightarrow R_W:\quad M'_W$
$R_W:\quad Ver(M'_W, \mu_{j+2})$
$R_W:\quad \sigma'_W = S_W(h'_W)$
$R_W \rightarrow R_{W+1}: \{M_k | 0 \leq k \leq j+1, M'_{j+2}, M'_W\}, [\bar{\mu}_{W+1}, \sigma'_{j+2}, \sigma'_W]$

In the above attack, R_{j+2} and R_W can collude to truncate the encapsulated messages $M_{j+2}, \cdots, M_W$. Then, R_W forwards the truncated chain of encapsulated messages

$$M_0, M_1, \cdots, M_j, M_{j+1}, M'_{j+2}, M'_W$$

to the next host R_{W+1}. As we can see, further checks made by R_{W+1}, and even the originator of the agent, will not be able to detect such a truncation attack.

6.3.4 Amendments

The cause for the attack is that a host may possess sufficient information to forge a signature that is supposed to be generated by its predecessor to establish a publicly verifiable chain of encapsulated messages. To preserve the forward privacy property, each encapsulated message is not signed with a certified private key. Instead, it is signed with a temporary private key generated by a host being just visited. Then, if a loop of $R_i \rightarrow R_{i+1} \rightarrow R_i$ is formed by a free-roaming agent, the truncation attack will succeed.

A simple solution to avoid this attack is to disallow such a roaming path. In Figure 6.8, when S_{i+1} is preparing the interactive message encapsulation, it knows the identity of its predecessor R_i by verifying σ_i and should not select R_i as the next host to be visited. In principle, this solution will suffice for a MPNR protocol in which recipients are visited by the intermediary agent only once in each message exchange.

However, a more generic solution exists to avoid the truncation attack even if a loop of $R_i \rightarrow R_{i+1} \rightarrow R_i$ is formed by a free-roaming agent. To protect against the impersonation of R_{i+1} by R_i in generating an encapsulated message with a temporary private key $\bar{\mu}_{i+1}$, the key pair $(\bar{\mu}_{i+1}, \mu_{i+1})$ should be generated by R_{i+1} itself rather than by R_i. Then R_{i+1} sends μ_{i+1} to R_i to testify that μ_{i+1} is R_{i+1}'s temporary public key. As $\bar{\mu}_{i+1}$ is only known to R_{i+1}, impersonation is prevented while the forward privacy property of the protocol is preserved. The protocol is modified as follows (see Figure 6.9).

Agent Creation

1. *Message encapsulation*

$$O : h_0 = h(r_0, R_1)$$
$$O : M_0 = S_0(E_0(m_0, r_0), I, h_0)$$
$$O : \sigma_0 = S_0(h_0)$$

2. *Agent transmission*

$$O \rightarrow R_1 : \ M_0, [\sigma_0]$$

Agent Migration at $\mathbf{R_1}$

3. *Agent verification*

R_1 : receive M_0, σ_0
R_1 : $Ver(M_0, u_0)$, and recover I, h_0
R_1 : $Ver(\sigma_0, u_0)$

4. *Interactive message encapsulation*

$$\begin{array}{ll} R_1 : & h_1 = h(M_0, r_1, R_2) \\ R_1 \rightarrow O : & temp_1 = E_0(S_1(m_1, \sigma_0), r_1), h_1, \mu_1 \\ O : & M_1 = S_0(temp_1) \\ O \rightarrow R_1 : & M_1 \\ R_1 : & Ver(M_1, u_0) \\ R_1 : & \sigma_1 = S_1(h_1) \end{array}$$

5. *Agent transmission*

$$R_1 \rightarrow R_2 : \ M_0, M_1, [\sigma_0, \sigma_1]$$

Agent Migration at $\mathbf{R_i}$ $(2 \leq i \leq n)$

6. *Agent verification*

$R_i : receive\ M_0, \cdots, M_{i-1}, \sigma_{i-2}, \sigma_{i-1}$
R_i : $Ver(M_0, u_0)$, and recover I, h_0
R_i : $Ver(M_1, u_0)$, and recover h_1, μ_1
R_i : $Ver(M_k, \mu_{k-1})$, and recover h_k, μ_k $recursively\ for\ 2 \leq k \leq i-1$
R_i : $Ver(\sigma_{i-2}, u_{i-2})$
R_i : $Ver(\sigma_{i-1}, u_{i-1})$
R_i : verify $R_{i-2} \neq R_{i-1}$

7. *Interactive message encapsulation*

$R_i:$ $\quad h_i = h(M_{i-1}, r_i, R_{i+1})$
$R_i \rightarrow R_{i-1}: temp_i = E_0(S_i(m_i, \sigma_{i-2}, \sigma_{i-1}), r_i), h_i, \mu_i$
$R_{i-1}:$ $\quad M_i = S_{\bar{\mu}_{i-1}}(temp_i)$
$R_{i-1} \rightarrow R_i: M_i$
$R_i:$ $\quad Ver(M_i, \mu_{i-1})$
$R_i:$ $\quad \sigma_i = S_i(h_i)$

8. *Agent transmission*

$R_i \rightarrow R_{i+1}: \{M_k | 0 \le k \le i\}, [\sigma_{i-1}, \sigma_i]$

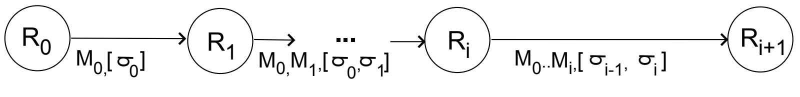

Fig. 6.9 Amended Agent Transmission

With the above change, even if the agent revisits one host, the host still has to request to the previous host for a signature over its new encapsulated message, and the co-signing process cannot be forged since each host's temporary private key is known to that host only.

6.3.5 Security Requirements Analysis

This section gives a brief analysis of the protocol depicted with respect to the security requirements outlined in Section 6.3.1, which are also desirable in every MPNR protocol or service with an intermediary agent involved.

Data Confidentiality. Each message m_i ($i = 1, \cdots, n$) that is encapsulated in M_i is encrypted with the originator O's public key u_0. Only O can decrypt it to extract the message, thus confidentiality is preserved.

Non-repudiability. Each message m_i ($i = 1, \cdots, n$) that is encapsulated in M_i is signed by R_i with its private key. Therefore, R_i cannot deny its message m_i once the agent carrying M_i returns to the originator O.

Forward Privacy. Each message m_i ($i = 1, \cdots, n$) that is encapsulated in M_i is first signed by R_i but then encrypted with O's public key u_0. Therefore, the identity of R_i will not be disclosed to others (except O) by examining M_i. In addition, as a random number r_i is used in computing the checksum h_i, it reveals no identity information by examining h_i. However, as σ_i will be sent to R_{i+1} and R_{i+2} in order to verify that two adjacent hosts are different on the agent migration path, the identity of R_i will be disclosed to R_{i+1} and R_{i+2}.

Forward Integrity. Each message m_i ($i = 1, \cdots, n$) that is encapsulated in M_i is signed by R_i. Any change to the signed message will be detected. Furthermore, even R_i cannot change its own encapsulated message M_i in the chain $M_1, \cdots, M_i, \cdots, M_n$ without being detected. Suppose R_i wants to replace m_i with m_i'. To make this change undetected, R_i needs to get a new counter-signature $M_i' = S_{\bar{\mu}_{i-1}}(temp_i')$ from R_{i-1} which should also satisfy $h(M_i, r_{i+1}, R_{i+2}) = h(M_i', r_{i+1}, R_{i+2})$. Even if R_{i-1} is willing to collude on generation of M_i', the equation will not be satisfied under a collision-free hash function.

Publicly Verifiable Forward Integrity. Each encapsulated message M_i ($i = 1, \cdots, n$) contains R_i's temporary public key μ_i that is certified by R_{i-1} with its temporary private key $\bar{\mu}_{i-1}$. With M_i, anyone can obtain μ_i and use it to verify M_{i+1}. Therefore, the integrity of $M_1, \cdots, M_n$ is publicly verifiable.

Insertion Defense. Since all encapsulated messages $M_1, \cdots, M_n$ are chained, if a new encapsulated message M_W is inserted between M_i and M_{i+1} without being detected, some chaining relations have to be changed in M_i and M_{i+1}. Suppose an attacker R_W tries to insert M_W as follows.

$$
\begin{aligned}
&h_W = h(M_i, r_W, R_{i+1}) \\
&temp_W = E_0(S_W(m_W, \sigma_{i-1}, \sigma_i), r_W), h_W, \mu_W \\
&M_W = S_{\bar{\mu}_i}(temp_W) \\
&\sigma_W = S_W(h_W)
\end{aligned}
$$

This implies that R_W at least needs to ask R_i to counter-sign $temp_W$, and ask R_{i+1} to replace its signature in $temp_{i+1}$ as $S_{i+1}(m_{i+1}, \sigma_i, \sigma_W)$ and R_{i+2} to replace its signature in $temp_{i+2}$ as $S_{i+2}(m_{i+2}, \sigma_W, \sigma_{i+1})$. Furthermore, the Forward Integrity property explained does not allow any change in the messages without being detected by the agent's originator. So it is impossible for R_W to insert M_W.

Truncation Defense. The chaining mechanism used in the insertion defense also works for the truncation defense. Suppose an attacker R_W tries to truncate the encapsulated messages $M_1, \cdots, M_i, M_{i+1}, \cdots$ from M_{i+1} thereafter and may also add M_W after M_i. Then, R_W needs to revise M_i as follows.

$$
\begin{aligned}
&h_i' = h(M_{i-1}, r_i, R_W) \\
&temp_i' = E_0(S_i(m_i, \sigma_{i-2}, \sigma_{i-1}), r_i), h_i', \mu_i \\
&M_i' = S_{\bar{\mu}_{i-1}}(temp_i') \\
&\sigma_i' = S_i(h_i')
\end{aligned}
$$

Obviously, R_W is unable to make the above revisions without collusion with R_i and R_{i-1}. In other words, this technique defends against truncation attacks if there are no more than two colluders. A straightforward extension of this protocol is possible, in order to defend truncation attacks with more colluders. That is, if defending against L colluders, L-1 co-signers for the M_i (from S_{i-L+1} to S_{i-1}) encapsulated message are needed.

Part IV

Chapter 7
Conclusions

A handshake might be enough in some traditional businesses, but for most business transactions simply relying on trust does not work. Unfortunately, today's computer systems often require that we do just that. It would be better if we were able to prove when a party is trying to cheat in a transaction, particularly in complex environments such as today's and tomorrow's Internet. Non-repudiation mechanisms provide us with the means to generate irrefutable evidences for our transactions. They can be used to prove who communicated, what about, and whether or not a transaction was successfully completed.

Even though the non-repudiation service has not been taken into account as it deserves, due to an evident difference and misfit of the digital world, it will definitely need more attention if we really want to talk about the "digital era". We just do not foresee a future in which digital exchanges will not mandatorily request the generation of evidence tokens, as it is done in nowadays paper-based life. In fact, non-repudiation does not substitute trust-based solutions and frameworks, but complement them.

Therefore, a naive solution would be to force parties to include an evidence token in all their communications (when originating and acknowledging). But this simple approach will not avoid unfair situations, the impossibility of demonstrating a fact/transaction which certainly occurred, when network failures and fraudulent behaviors are possible. So, the design of new protocols covering all possible situations is needed, and approaches with different properties exist for Alice and Bob.

In a typical scenario, Alice wants to send a message (digital content, payment, ...) to Bob such that Bob consumes (reads, processes, ...) the message only if Alice gets a token of receipt as evidence. At the same time, Bob could also require an evidence token of origin for demonstrating that indeed, Alice sent that message. Unfortunately there does not exist a deterministic solution without the intervention of a trusted third party. Here again, a straightforward approach is possible. Each entity sends its needed information (messages and evidence tokens) to a TTP and the latter only distributes the information when its view is complete for both entities.

But this assumes too much trust and computational and network load over the TTP. Thus, solutions, which try to reduce the TTP's participation to minimum, have been studied. During this process, it has been always assumed that there are two entities in the application (whether it is CEM or contract signing). But undeniably, there are a lot of applications designed for simultaneous participation of several entities.

In this case, not only previously defined requirements change but, at the same time, the former methods remain inefficient. Consequently, a new step is needed in the non-repudiation service, that is to design multi-party non-repudiation protocols with an abstraction on the number of participants.

The main focus of this monograph has been multi-party non-repudiation protocols, although other more specific multi-party protocols for fair exchange, certified email and contract signing protocols have been studied as well. New approaches to multi-party non-repudiation protocols have come in mainly three ways.

1. n entities participate in the protocol
2. two entities are involved in an exchange but they seek the help of an intermediary or agent.
3. n entities participate in the protocol with the help of an intermediary or agent

This book introduced electronic commerce applications as the main area which can be benefited by the non-repudiation security service as defined by the ITU. This is mainly due to the fact that security is and needs to be a cornerstone in electronic commerce deployment and operation. This is the reason for which standard security services and protocols in electronic communications, and thus in electronic commerce, are presented at the beginning, highlighting the non-repudiation service among all of them.

The reader will find very useful the foundations of non-repudiation services divided in the following elements: *type* of service, *evidence* as the main tool or token to provide the service, the different *roles* of a TTP as an entity/system which must participate in the service, the several *phases*, the *requirements* the service must or can fulfill, the *legal* framework and analysis of the existing *standards*.

When introducing the service applied to multi-party scenarios and applications, we not only gave a state of the art study but also an in-depth analysis underlining the merits and weaknesses. The reader can also see a comparison of the studied solutions and the new schemes.

The properties that a non-repudiation protocol is required to respect are clearly defined. Following the first research conducted on multi-party non-repudiation protocols proposed in [87, 96], we extended that scenario such that sending different messages to different entities in only one transaction is possible. This protocol uses a light-weight on-line TTP in every transaction. However, there are situations in which the final entities could be willing to launch a non-repudiation protocol without the TTP's assistance (either because the TTP's service is expensive or because the entities have some kind of trust to each other) unless an error or channel failure occurs. For these situations we designed an optimistic protocol that uses only three steps in its main protocol. We further discussed the dispute resolution process and the efficiency matters of each design.

Collecting, verifying and storing evidence about the transactions is an additional task required as part of the non-repudiation service, but might be undesirable for final entities when these transactions are undertaken with multiple entities and the volume is considerable. Hence, intermediary entities are useful in such scenarios to help final entities to carry out their business transactions. In addition, these intermediary entities can act as 'hubs', increasing the market and opportunities for both customers and merchants.

We analyzed a new entity that takes part in the non-repudiation protocol. This intermediary entity can be just another module in an existing agent-based system, facilitating the originator to carry out an e-commerce transaction. We introduced different scenarios in which this approach can be easily fitted into, and demonstrated the advantages for end users in the use of an intermediary service on reducing the evidence storage requirements and gathering different recipients. In these agent-mediated non-repudiation protocols, the originator can be kept anonymous to the recipients, and vice versa, as long as the originator and the recipients do not need to verify each other's evidence. The intermediary agent can be distrusted and the scheme maintains the security requirements for a non-repudiable e-commerce transaction. In general, the intermediary entity cannot collude with the entity initiating the protocol if the other party is to be provided with a valuable object in exchange for evidence (or other object).

It is widely known that the role of the TTP is essential for many Internet security protocols. On the other hand, we have seen that most of non-repudiation protocols include parameters whose values are not easy to specify, and some of those parameters are directly related to the TTP operation. We demonstrated how event-oriented simulation can be considered as a tool to estimate the timeouts of non-repudiation protocols. We proposed a simulation model in order to estimate the appropriate values of the parameters for an efficient use of a TTP in multi-party non-repudiation protocols. The model has been validated with some tests that helped to estimate the most appropriate values for the simulated parameters.

Two-party certified email has become an important value-added messaging application. If an extension to multiple entities is made, it could be used for certified and timely notification of users. We extended a CEM protocol to support timeliness and multiple entities. Although a new cancel sub-protocol is added, the main protocol remains unchanged. The additional messages included improve the timeliness property in an efficient and elegant way. At the same time the extension for the distribution of the same mail (message) to several entities in a certified manner is made in such a way that only those who reply with evidence of receipt will obtain the message. Timeliness is also considered here and though the Post Office has to deal with several entities if any of them launches a sub-protocol, no significant complexity is introduced.

Contract signing is a fundamental service for business transactions. Previous work mainly focused on two-party contract signing. In some applications, however, a contract may need to be signed by multiple parties. In fact, we daily sign documents [1] and, in many occasions, these documents need the signatures from several entities. We presented a new multi-party contract signing protocol. This is so far the most efficient synchronous multi-party contract signing protocol in terms of the number of messages required. Besides, we further considered additional features like abuse-freeness and threshold timeliness.

As the technology evolves, content downloading will be an inexpensive operation. In order to protect Intellectual Property Rights, distributed DRM appears as a very good approach and it has already been used by the main handset manufactures (e.g. Nokia mobile telephones implement at least OMA DRM v1.0). Furthermore, DRM frameworks will be enriched by the implementation of security services from the very beginning. Non-repudiation is one of them. We presented a non-repudiation protocol for a DRM platform that takes into account all participants in the acquisition of rights, namely, the user, the Mobile Network Operator and the Rights Issuer, thus providing all of them with sufficient evidence to be used in case a dispute arises. The protocol has been implemented, tested and integrated with the Mobile DRM framework designed for the European project UBISEC.

With the emergence of wireless technology, grid computing, and other technologies where the storage and transmission of data are carried out without a centralized server, it is clear that new models of charging and distribution should not only comply with the requirements of this new topology but also provide with an efficient and practical solution. We introduced a new entity that without being totally trusted, acts as a hub of the topology, helping the distributors in collection of possibly small payments of the digital contents. An underlying P2P payment protocol is applied to the P2P content distribution scenario. In this scenario a merchant peer's workload is largely shifted to the intermediary agent, such that each peer can be easily involved in distributing digital contents and receiving payment via the agent. We also discussed the trustworthiness presumed to each of the entities and integrated a non-repudiation security service in the design of the protocol.

Along this monograph, we have brought up the concept of intermediary entities or agents. Although we omitted implementation details, one possibility is to implement them as mobile agents (in fact its functionality is independent of how it is implemented). In mobile code systems more attention is needed in the design of protocols to fulfill the desired properties. As a consequence of dynamically selected hosts on the path of a free roaming agent, a host can be revisited, and hence, we must ensure that the information it possesses is not enough to carry out a truncation attack by colluding with another party, or otherwise, properties like non-repudiation which has been considered in the initial design will be flawed as well. Furthermore, since digital evidence could be truncated, user could deny having sent a specific offer (or message). We proposed a new scheme that is effective in defending against the truncation attack. We also gave a brief analysis to demonstrate that the scheme satisfies other security requirements on the protection of agent data.

We should point out that there is still a need for further research on the adaptation of non-repudiation to the new applications scenarios appearing nowadays (pervasive

[1] By doing so we commit to the document's content which automatically transforms it to a contract that means: ... "I am aware of what the document says and totally agree on its content ...".

and ubiquitous computing scenarios in which multiple highly dynamic users are involved) in order to make the service as proposed by the ITU more efficient and robust. In other words, mobile telephones and other resource and power restricted devices are the main candidate to undertake users daily transactions. This will obviously change some problem requirements. For instance, efficiency will be of primary concern and other schemes for avoiding the use of asymmetric cryptography as evidence need to be considered. Some works [134, 51] already exist, but the area remains generally unexplored.

As identity documents (passports, national cards) and mobile phones are two items that users usually carry with, and in some countries, both integrate the necessary digital material for electronic signatures, fair mobile contract signing applications should be also studied and designed when several signatories participate. With the advent of the electronic identity documents, we have the opportunity to materialize new MPCS protocols for mobile environments.

Finally, formal methods for verification of the presented protocols is another topic which needs to be investigated. Although some methods exist [118, 68, 110, 43], most of them relate to two-party protocols. Similarly, it could be worthwhile to apply the verification approach presented in [43] to the new synchronous multi-party MPCS protocol detailed in Section 5.2. See the appendix below for a brief example.

Appendix

Some of these formal methods for two-party non-repudiation protocols can be applied to the protocol introduced in Section 4.1 for its on-line version. The CSP (Communicating Sequential Processes) used by Schneider in the analysis of [132] can be used since the multi-party adaptation does not introduce new flows in the protocol. In particular (read [118] for better understanding), evidence of origin can be specified as

$$\begin{array}{lll}\mathrm{NRO(tr)} & = & \mathit{evidence}.R_i.S_O(\mathit{feoo},R_i,TTP,L_i,t,v_i,u_{R_i},c_i) \;\; \text{in tr} \\ & & \bigwedge \mathit{evidence}.R_i.S_{TTP}(\mathit{fcon},O,R',L',t,E_{R'}(k)) \;\; \text{in tr} \\ & & \Rightarrow \\ & & \text{O } \textbf{sent } S_O(\mathit{feoo},R_i,TTP,l_i,t,v_i,u_{R_i},c_i) \bigwedge \\ & & \text{O } \textbf{sent } S_O(\mathit{fsub},R',L',t,E_{R'}(k))\end{array}$$

and evidence of receipt as

$$\begin{array}{lll}\mathrm{NRR(tr)} & = & \mathit{evidence}.O.S_{R_i}(\mathit{feor},O,TTP,l_i,t,v_i,u_{R_i},c_i) \;\; \text{in tr} \\ & & \bigwedge \mathit{evidence}.O.S_{TTP}(\mathit{fcon},O,R',L',t,E_{R'}(k)) \;\; \text{in tr} \\ & & \Rightarrow \\ & & \text{B } \textbf{sent } S_{R_i}\mathit{feor},O,TTP,l_i,t,v_i,u_{R_i},c_i) \bigwedge \\ & & \mathit{ftp}.R_i.TTP.S_{TTP}(\mathit{fcon},O,R',L',t,E_{R'}(k)) \quad \notin \quad X\end{array}$$

The reasoning for which the correctness of evidence is verified using CSP is similar and the same lemmas and corollaries can be applied. Regarding fairness and taking into account the same premises (e.g. an agent is only entitled to expect fairness if he behaves in accordance with the protocol) our protocol can be verified against CSP to fulfill this property except in FAIR1 when collusion is allowed among recipients. It happens because the collusion attack can only be avoided with new requirements over the channel (no possible communication among R_i). The accomplishment of FAIR2 and FAIR3 properties ensures light fairness in the protocol.

References

1. http://www.ebay.com
2. http://www.paypal.com
3. http://www.ipswitch.com/Products/WhatsUp/
4. http://www.gnutella.com
5. http://freenet.sourceforge.net
6. http://www.jxta.org
7. http://www.gnunet.org
8. Arma international. http://www.arma.org
9. Bbb dispute resolution. http://www.dr.bbb.org
10. e-Government eEurope 2005.
11. i2010 initiative. http://ec.europa.eu/information_society/eeurope/i2010/index_en.htm
12. JSR 219: Foundation Profile 1.1. http://jcp.org/en/jsr/detail?id=219
13. kXML. http://kxml.sourceforge.net
14. Ltans-charter. URL `http://www.ietf.org/html.charters/`
15. Mobile Information Device Profile. http://java.sun.com/products/midp/
16. Statistical office of the european communities. http://epp.eurostat.ec.europa.eu/
17. The Legion of the Bouncy Castle. http://www.bouncycastle.org
18. The us national archives electronic records archives (era). http://www.archives.gov/era
19. 3-D Secure Team: 3-D Secure impementation Guide. Visa Internatitonal Service Association, 1.0.2 edn. (2004)
20. Adar, E., Huberman, B.: Free riding on gnutella (2000). URL `citeseer.nj.nec.com/adar00free.html`
21. Anantharaman, L., Bao, F.: An efficient and practical peer-to-peer e-payment system (2002). Manuscript
22. Antoniadis, P., Courcoubetis, C.: Market models for P2P content distribution. In: AP2PC'02 (2002)
23. Asokan, N.: Fairness in electronic commerce. Ph.D. thesis, University of Waterloo, Computer Science (1998)
24. Asokan, N., Baum-Waidner, B., Schunter, M., Waidner, M.: Optimistic synchronous multi-party contract signing. Tech. Rep. RZ 3089, IBM Zurich Research Lab (1998)
25. Asokan, N., Janson, P.A., Steiner, M., Waidner, M.: The state of the art in electronic payment systems. IEEE Computer **30**(9), 28–35 (1997)
26. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for multi-party fair exchange. Tech. Rep. RZ 2892 (# 90840), IBM, Zurich Research Laboratory (1996)
27. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 7–17. ACM Press (1997). DOI http://doi.acm.org/10.1145/266420.266426

28. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE J. Sel. Area. Comm. **18**(4), 593–610 (2000). URL `citeseer.nj.nec.com/asokan98optimistic.html`
29. Ateniese, G., de Medeiros, B., Goodrich, M.T.: TRICERT: A distributed certified E-mail scheme. In: Network and Distributed System Security Symposium Conference Proceedings (2001)
30. Banks, J., Carson, J., Nelson, B., D.Nicol: Discrete-event system simulation. Prentice Hall (2000)
31. Bao, F., Deng, R., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: IEEE Symposium on Security and Privacy, pp. 77–85. IEEE (1998)
32. Bao, F., Deng, R., Nguyen, K., Varadharajan, V.: Multi-party fair exchange with an off-line trusted neutral party. In: Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on, pp. 858–862 (1999)
33. Bao, F., Deng, R., Zhou, J.: Electronic payment systems with fair on-line verification. In: IFIP TC11 16th Annual Working Conference on Information Security: Information Security for Global Information Infrastructures, pp. 451 – 460. IFIP TC11, Kluwer Academic Publishers (2000)
34. Baum-Waidner, B.: Optimistic asynchronous multi-party contract signing with reduced number of rounds. In: F. Orejas, P. Spirakis, J.V. Leeuwen (eds.) Automata, Languages and Programming, pp. 898–911. 28th International Colloquium, ICALP 2001, Springer-Verlag (2001)
35. Baum-Waidner, B., Waidner, M.: Optimistic asynchronous multi-party contract signing. Tech. Rep. RZ 3078, IBM Zurich Research Lab (1998)
36. Baum-Waidner, B., Waidner, M.: Round-optimal and abuse-free multi-party contract signing. In: 27th International Colloquium on Automata, Languages and Programming, *LNCS*, vol. 1853, pp. 524–535. Springer (2000)
37. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.: A fair protocol for signing contracts. In: IEEE Transactions on Information Theory, vol. 36, pp. 40–46 (1990)
38. Blum, M.: Three applications of the oblivious transfer: Part i: Coin flipping by telephone; part ii: How to exchange secrets; part iii: How to send certified electronic mail. Tech. rep., Department of EECS, University of California (1981)
39. Boly, J.P., Bosselaers, A., Cramer, R., Michelsen, R., Mjolsnes, S.F., Muller, F., Pedersen, T.P., Pfitzmann, B., de Rooij, P., Schoenmakers, B., Schunter, M., Vallee, L., Waidner, M.: The ESPRIT project CAFE - high security digital payment systems. In: ESORICS, pp. 217–230 (1994)
40. Bradner, S.: Rfc 2119. key words for use in rfcs to indicate requirement levels (1997)
41. Brannigan, C.: Beyond e-commerce: Expanding the potential of Online Dispute Resolution. Interaction **16**(4), 15–17 (2004)
42. Carbonell, M., Onieva, J.A., Lopez, J., Zhou, J.: Timeout estimation using a simulation model for non-repudiation protocols. In: Fourth International Conference on Computational Science and Its Applications ICCSA, *LNCS*, vol. 3043, pp. 903–914. Springer (2004)
43. Chadha, R., Kremer, S., Scedrov, A.: Formal analysis of multi-party contract signing. In: Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04), pp. 266–279. IEEE Computer Society Press (2004)
44. Cheng, J.S., Wei, V.K.: Defenses against the truncation of computation results of free-roaming agents. In: Fourth International Conference on Information and Communications Security, *LNCS*, vol. 2513, pp. 1–12 (2002)
45. Chiou, G., Chen, W.: Secure broadcasting using the secure lock. IEEE Transaction on Software Engineering **15**(8), 929–934 (1989)
46. Damiani, E., Vimercati, S.C.D., Paraboschi, S., Samarati, P., Violante, F.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: V. Atluri (ed.) Computer and Commmunications Security, pp. 207–216. ACM (2002)
47. DeMillo, R.A., Merritt, M.: Protocols for data security. IEEE Computer **16**, 39–50 (1983)

48. Dumortier, J., Kelm, S., Nilsson, H., Skouma, G., Eecke, P.V.: The legal and market aspects of electronic signatures. Study for the european comission - dg information society, Interdisciplinary centre for Law & Information Technology, Leuven Universiteit (2003)
49. E-Arbitration-T: Online arbitration: What technology can do for arbitral institutions. Seminar (2003)
50. ElGamal, T.: A public-key cryptosystem and a signature scheme based on discrete logarithms. In: IEEE Transactions on Information Theory, vol. IT-31, pp. 469–472 (1985)
51. Eschenauer, L., Gligor, V.D., Baras, J.: On trust establishment in mobile ad-hoc networks (2002). Submitted for publication 2002
52. EU Information Society: DIRECTIVE 1999/93/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL (1999)
53. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Communications of the ACM, vol. 28, pp. 637–647 (1985)
54. Ezhilchelvan, P., Shrivastava, S.: A family of trusted third party based fair-exchange protocols. IEEE T. Depend. Secure. **2**(4), 273–286 (2005)
55. Federal Information, N.B.o.S.: Data encryption standard (des). Tech. Rep. fips-46-3, NIST (1999)
56. Federal Information, N.B.o.S.: Advanced encryption standard (aes). Tech. Rep. fips-197, NIST (2001)
57. Ferrer-Gomila, J.L., Payeras-Capellà, M., Huguet-Rotger, L.: Efficient optimistic n-party contract signing protocol. In: Proceedings of the 4th International Conference on Information Security, pp. 394–407. Springer-Verlag (2001)
58. Ferrer-Gomila, J.L., Payeras-Capellà, M., Huguet-Rotger, L.: A realistic protocol for multiparty certified electronic mail. In: Information Security ISC 2002, *LNCS*, vol. 2433, pp. 210–219 (2002)
59. Ferrer-Gomila, J.L., Payeras-Capellà, M., Huguet-Rotger, L.: Optimality in asynchronous contract signing protocols. In: 1st International Conference on Trust and Privacy in Digital Business, vol. 3184, pp. 200–208. Springer-Verlag (2004)
60. Franklin, M., Tsudik, G.: Secure group barter: Multi-party fair exchange with semi-trusted neutral parties. In: Proceedings of Financial Cryptography 1998, *Lecture Notes in Computer Science*, vol. 1465, pp. 90–102. Springer (1998)
61. Garay, J.A., MacKenzie, P.D.: Abuse-free multi-party contract signing. In: Proceedings of the 13th International Symposium on Distributed Computing, pp. 151–165. Springer-Verlag (1999)
62. Golle, P., Leyton-Brown, K., Mironov, I.: Incentives for sharing in peer-to-peer networks. In: EC '01: Proceedings of the 3rd ACM conference on Electronic Commerce, pp. 264–267. ACM Press (2001). DOI http://doi.acm.org/10.1145/501158.501193
63. Gondrom, T., Brandner, R., Pordesch, U.: Evidence Record Syntax (ERS). RFC 4998 (Proposed Standard) (2007). URL `http://www.ietf.org/rfc/rfc4998.txt`
64. González-Deleito, N.: Trust reletionships in exchange protocols. Ph.D. thesis, Faculté des Sciences, Université Libre de Bruselles (2005)
65. González-Deleito, N., Markowitch, O.: An optimistic multi-party fair exchange protocol with reduced trust requirements. In: Proceedings of the 4th International Conference on Information Security and Cryptology, *Lecture Notes in Computer Science*, vol. 2288, pp. 258–267. Springer-Verlag (2001)
66. González-Deleito, N., Markowitch, O.: Exclusion-freeness in multi-party exchange protocols. In: Lecture Notes in Computer Sciences, pp. 200–209. 5th International Conference on Information Security (ISC 2002), Springer-Verlag (2002)
67. Gullou, L.C., Quisquater, J.: A paradoxical indentity-based signature scheme resulting from zero-knowledge. In: Advances in Cryptology, *LNCS*, vol. 403, pp. 216–231. Springer (1988)
68. Gürgens, S., Rudolph, C.: Security analysis of (un-) fair non-repudiation protocols. In: Formal Aspects of Security, *LNCS*, vol. 2629, pp. 99–114. Spinger-Verlag (2002)
69. Haber, S., Kaliski, B., Stornetta, S.: How do digital time-stamps support digital signatures? Cryptobytes Newsletter **1**(3), 14–15 (1995)

70. Horne, B., Pinkas, B., Sander, T.: Escrow services and incentives in peer-to-peer networks. In: Proceedings of the 3rd ACM conference on Electronic Commerce, pp. 85–94. ACM Press (2001). DOI http://doi.acm.org/10.1145/501158.501168
71. International, M.: MasterCard SecureCode Merchant Implementation Guide. MasterCard International Incorporated (2004)
72. ISO/IEC: 1st WD 13888-2. Non-repudiation using a symmetric key algorithm. JTC1/SC27/WG2 N83 (1991)
73. ISO/IEC: DIS 10181-4. Information technology - Open systems interconnection - Security frameworks in open systems - Part 4: Non-repudiation (1996)
74. ISO/IEC: 2nd CD 13888-3. Information technology - Security techniques - Non-repudiation - Part 3: Using asymmetric techniques. JTC1/SC27 N1379 (1997)
75. ISO/IEC: 3rd CD 13888-2. Information technology - Security techniques - Non-repudiation - Part 2: Using symmetric encipherment algorithms. JTC1/SC27 N1276 (1998)
76. ISO/IEC: 13888-1. Information technology - Security techniques - Non-repudiation - Part 1: General model. JTC1/SC27 (2004)
77. ITU-T: Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services (1997)
78. ITU-T: Security architecture for systems providing end to end communications (2003)
79. ITU-T: Security in Telecommunications and Information Technology (2006)
80. ITU-T X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks (2000)
81. ITU-T X.813: Information technology - Open Systems Interconnection - Security frameworks for open systems: Non-repudiation framework (1996)
82. Jakobsson, M.: Mini-cash: a minimalistic approach to e-commerce. In: Proceedings of PKC'99, *LNCS*, vol. 1560, pp. 122–135. Springer (1999)
83. Karjoth, G., Asokan, N., Gülcü, C.: Protecting the computation results of free-roaming agents. In: Mobile Agents, *LNCS*, vol. 1477, pp. 195–207 (1998)
84. Karjoth, G., Posegga, J.: Mobile agents and telcos' nightmares. Tech. Rep. 55(7/8):29-41, IBM (2000)
85. Ketchpel, S., Garcia-Molina, H.: Distributed commerce transactions. Tech. rep., Standford University, Computer Science Department (1997)
86. Khill I.and Kim, J., Han, I., Ryou, J.: Multi-party fair exchange protocol using ring architecture model. Computers & Security **20**(5), 422–439 (2001)
87. Kremer, S., Markowitch, O.: A multi-party non-repudiation protocol. In: Proceedings of SEC 2000: 15th International Conference on Information Security, pp. 271–280. IFIP World Computer Congress (2000)
88. Kremer, S., Markowitch, O.: Optimistic non-repudiable information exchange. In: J. Biemond, editor, 21st Symp. on Information Theory in the Benelux, pp. 139–146. Werkgemeenschap Informatie- en Communicatietheorie (2000)
89. Kremer, S., Markowitch, O.: Fair multi-party non-repudiation protocols. International Journal of Information Security **1**(4), 223 – 235 (2003)
90. Kremer, S., Markowitch, O., Zhou, J.: An intensive survey of fair non-repudiation protocols. Computer Communications **25**(17), 1606–1621 (2002)
91. Liew, C.C., Ng, W.K., Lim, E.P., Tan, B.S., Ong, K.L.: Non-repudiation in an agent-based electronic commerce system. In: Proceedings of 1999 DEXA International Workshop on Electronic Commerce and Security, pp. 864–868. Florence, Italy (1999)
92. Lindell, Y.: Composition of Secure Multi-Party Protocols. Springer (2003)
93. Lopez, J.: Diseño de una Infraestructura de Notarización para Comercio Electrónico. Ph.D. thesis, E.T.S.I. Informatica, Malaga, Spain (2000)
94. Mao, W.: Modern Cryptography: Theory and Practice. Hewlett-Packard Books (2004)
95. Markowitch, O., Gollmann, D., Kremer, S.: On fairness in exchange protocols. In: Springer-Verlag (ed.) 5th International Conference on Information Security and Cryptology, *LNCS*, vol. 2587, pp. 451–464 (2002)

96. Markowitch, O., Kremer, S.: A multi-party optimistic non-repudiation protocol. In: Proceedings of 3rd International Conference on Information Security and Cryptology, *LNCS*, vol. 2015, pp. 109–122. Springer-Verlag (2000)
97. Markowitch, O., Roggeman, Y.: Probabilistic non-repudiation without trusted third party. In: Second Workshop on Security in Communication Network 99 (1999). URL `citeseer.nj.nec.com/markowitch99probabilistic.html`
98. Markowitch, O., Saeednia, S.: Optimistic fair-exchange with transparent signature recovery. In: Proceedings of Financial Cryptography 2001, *LNCS*, vol. 2339, pp. 339–350. Springer-Verlag (2001)
99. Massias, H., Quisquater, J.: Time and cryptography. Tech. rep., Project Timesec Digital Timestamping and the Evaluation of Security Primitives (1997)
100. Maurer, U.: New approaches to digital evidence. In: Proceedings of the IEEE, vol. 92, pp. 933–947. IEEE (2004)
101. Menezes, A.J., Oorschot, P.C.V., Vanstone, S.A.: Handbook of Applied Cryptography, 5 edn. CRC Press (1996)
102. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: Proceedings of the twenty-second annual symposium on Principles of distributed computing, pp. 12–19. ACM Press (2003). DOI http://doi.acm.org/10.1145/872035.872038
103. Mills, D.L.: Network time protocol (version 3) specification, implementation and analysis. Tech. Rep. RFC 1305, IETF Working Group (1992)
104. Mullen, T., Wellman, M.: The auction manager: Market middleware for large-scale electronic commerce. In: Proceedings of the 3rd USENIX Workshop on Electronic Commerce, pp. 37–48. Boston, Massachusetts (1998)
105. Onieva, J.A., Zhou, J., Lopez, J.: Practical service charge for P2P content distribution. In: Proceedings of 2003 Fifth International Conference on Information and Communications Security, *LNCS*, vol. 2836, pp. 112 – 123 (2003)
106. Onieva, J.A., Zhou, J., Lopez, J.: Non-repudiation protocols for multiple entities. Computer Communications **27**(16), 0140-3664 (2004)
107. Onieva, J.A., Zhou, J., Lopez, J.: Attacking an asynchronous multi-party contract signing protocol. In: Proceedings of 6th International Conference on Cryptology in India, *LNCS*, vol. 3797, pp. 311–321. Springer (2005)
108. Onieva, J.A., Zhou, J., Lopez, J., Carbonell, M.: Agent-mediated non-repudiation protocols. Electronic Commerce Research and Applications **3**(2), 1567-4223 (2004)
109. Open Mobile Alliance: DRM Specification, 2 edn. (2006)
110. Pancho-Festin, S., Gollmann, D.: On the formal analyses of the zhou-gollmann non-repudiation protocol. In: Formal Aspects in Security and Trust, pp. 5–15 (2005)
111. Peterson, M.: The coming archive crisis. Tech. rep., Storage Networking Industry Association (SNIA) (2006)
112. Pfitzmann, B., Schunter, M., Waidner, M.: Optimal efficiency of optimistic contract signing. In: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing, pp. 113–122. ACM Press (1998). DOI http://doi.acm.org/10.1145/277697.277717
113. Rajput, W.E.: E-Commerce Systems Architecture and Applications. Artech House (2000)
114. Rivest, R.L., Shamir, A.: Payword and micromint: Two simple micropayment schemes. In: Security Protocols Workshop, pp. 69–87 (1996)
115. Robinson, N.: Handbook of Legislative Procedures of Computer and Network Misuse in EU Countries. RAND Europe, Brussels (2003). Study for the European Commission Directorate-General Information Society (2002)
116. Roth, V.: Programming satan's agents. Electr. Notes Theor. Comput. Sci. **63**, 1–16 (2001)
117. Sander, T., Tschudin, C.F.: Protecting mobile agents against malicious hosts. In: Mobile Agents and Security, *LNCS*, vol. 1419, pp. 44–60 (1998)
118. Schneider, S.: Formal analysis of a non-repudiation protocol. In: 11th Computer Security Foundations Workshop (1998)
119. Senate, of Representatives, H.: Public law 107347dec. 17 2002 (2002)

120. Shao, M.H., Zhou, J., Wang, G.: On the security of a certified e-mail scheme with temporal authentication. In: Proceedings of 2005 ICCSA Workshop on Internet Communications Security, vol. 3482, pp. 701–710. Springer (2005)
121. Sherif, M.H.: Protocols for Secure Electronic Commerce. CRC Press (2000)
122. Smith, R.E.: Internet Cryptography. Addison-Wesley (1997)
123. Tacken, J., Flake, S., Zoth, C.: Mobile DRM in pervasive networking environments. In: Workshop on Trust and Security in Pervasive Networking, Pervasivetrust 2005, p. N.A. IEEE (2005)
124. Verhoest, P., Hawkins, R., Desruelle, P., Martinez, C., Lopez-Bassols, V., Vickery, G.: Electronic business networks: An assessment of the dynamics of b2b electronic commerce in eleven oecd countries. Tech. rep., IPTS (2003)
125. Wallace, C., Pordesch, U., Brandner, R.: Long-Term Archive Service Requirements. RFC 4810 (Informational) (2007). URL `http://www.ietf.org/rfc/rfc4810.txt`
126. Walpole, R.E., Myers, R.H.: Probabilidad y estadística, Fourth edn. McGraw-Hill (1994)
127. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: R. Cramer (ed.) Advances in Cryptology, *Lecture Notes in Computer Science*, vol. 3494, pp. 19–35. EUROCRYPT, Springer (2005)
128. Yee, B.S.: A sanctuary for mobile agents. In: Secure Internet Programming, pp. 261–273 (1999)
129. Zhou, J.: Non-repudiation in electronic commerce. Computer Security Series. Artech House (2001)
130. Zhou, J.: On the security of a multi-party certified email protocol. In: Information and Communications Security: 6th International Conference, vol. 3269, pp. 40–52 (2004)
131. Zhou, J., Deng, R.: On the validity of digital signatures. ACM SIGCOMM Computer Communication Review **30**(2), 29–34 (2000). DOI http://doi.acm.org/10.1145/505680.505684
132. Zhou, J., Gollmann, D.: A fair non-repudiation protocol. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 55–61. IEEE Computer Society Press (1996)
133. Zhou, J., Gollmann, D.: An efficient non-repudiation protocol. In: PCSFW: Proceedings of The 10th Computer Security Foundations Workshop, pp. 126–132. IEEE Computer Society Press (1997). URL `citeseer.nj.nec.com/article/zhou97efficient.html`
134. Zhou, J., Lam, K.: Undeniable billing in mobile communication. In: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 284–290 (1998)
135. Zhou, J., Onieva, J.A., Lopez, J.: Optimised multi-party certified email protocols. Information Management & Computer Security Journal **13**(5), 350–366 (2005)
136. Zhou, J., Onieva, J.A., Lopez, J.: A synchronous multi-party contract signing protocol improving lower bound of steps. In: 21st IFIP International Information Security Conference Security and Privacy in Dynamic Environments, *IFIP*, vol. 201, pp. 221–232. IFIP SEC, Springer (2006)

Index

Biographies

Dr. Jose A. Onieva received his M.S. in Computer Science in 2002 in the University of Malaga (UMA), Spain, actively collaborating with the Computer Science Department. Afterwards, he stayed as a ”Research fellow” in Infocomm Research Institute (I2R), Singapore, period in which initiated a research in the areas of non-repudation, mobile agents and P2P. Funded by the Junta de Andaluca government, he joined the Security Group of the Computer Science department at UMA where he received his PhD degree in 2006. Among other activities he has been actively involved in the IST European project from the VI Programme Framework - UBISEC (Ubiquitious Networks with a Secure Provision of Services, Access and Content Delivery) and has actively collaborated in security-related National funded projects. He has published several international journal and papers in the field of Security for Information Technologies. Currently, he is an Assistant Professor at the Computer Science Department and participates as a research collaborator in several European and National Projects and Programmes. He is an involved member of the ARES (Advanced Research on Information Security and Privacy) National Program (Ingenio 2010) whose objective is to advance the state of the art and the practice of information security and privacy in Spain. He also served as a General Chair in WISTP'08 (Workshop in Information Security Theory and Practices) and as a member of several Conference Programme Committees.

Dr. Javier Lopez received his M.S. and Ph.D. in Computer Science in 1992 and 2000, respectively, from University of Malaga, where he currently is Full Professor. His research activities are mainly focused on information and network security, leading some international research projects in those areas. Prof. Lopez is the Co-Editor in Chief of Springer's International Journal of Information Security (IJIS) and member of the Editorial Boards of Computer Networks (COMNET), Wireless Communication and Mobile Computing (WCMC), Journal of Network and Computer Applications (JNCA), Security and Communications Network Journal (SCN), Information Management and Computer Security Journal (IMCS) and the International Journal of Internet Technology and Secured Transactions (IJITST). Additionally, Prof. Lopez is the Spanish representative in the IFIP Technical Committee 11 on Security and Protection in Information Systems, a member of the Steering Committee of ERCIM's Working Group on Security and Trust Management, and Chair of the IFIP Working Group on Trust Management.

Dr. Jianying Zhou is a senior scientist at Institute for Infocomm Research (I2R), and heads the Network Security Group. Dr. Zhou worked in China, Singapore, and USA before joining I2R. He was a security consultant at the headquarters of Oracle Corporation, and took an architect role on securing e-business applications. He was a project manager at Kent Ridge Digital Labs, and led an R&D team to develop network security technologies. He was a post-doctoral fellow in National University of Singapore, and involved in a strategic research programme on computer security funded by National Science and Technology Board. He was formerly employed in Chinese Academy of Sciences, and played a critical role in a couple of national information security projects.

Dr. Zhou obtained PhD degree in Information Security from University of London, MSc degree in Computer Science from Chinese Academy of Sciences, and BSc degree in Computer Science from University of Science and Technology of China. His research interests are in computer and network security, cryptographic protocol, digital signature and non-repudiation, mobile and wireless communications security, and secure electronic commerce.

Dr. Zhou is actively involved in the academic community, having served over 100 times in international conference committees as general chair, program chair, publication chair, publicity chair and PC member, having been in the editorial board and as a regular reviewer for over 20 international journals. He has published over 120 referred papers at international conferences and journals, of which the top 10 publications received over 1000 citations. He is a world-leading researcher on non-repudiation, and authored the book Non-repudiation in Electronic Commerce which was published by Artech House in 2001. He is a co-founder and steering committee member of International Conference on Applied Cryptography and Network Security, and served as program chair of ACNS'03 and general chair of ACNS'04. He is also a co-founder and coordinating editor of Cryptology and Information Security Series published by IOS Press. He received National Science and Technology Progress Award from Ministry of Science and Technology in 1995 in recognition of his achievement in the research and development of information security in China.