Universitat de les
Illes Balears

FAIR EXCHANGE IN E-COMMERCE AND CERTIFIED E-MAIL, NEW SCENARIOS AND PROTOCOLS

GERARD DRAPER GIL

Departament de Ciències Matemàtiques i Informàtica

Palma, Juliol 2013

Josep Lluís Ferrer Gomila, professor del Departament de Ciències Matemàtiques i Informàtica de la Universitat de les Illes Balears

FA CONSTAR:

que la present memòria *Fair Exchange In E-Commerce And Certified E-Mail, New Scenarios And Protocols* presentada per *Gerard Draper Gil* per optar al grau de Doctor en Informàtica, ha estat realitzada sota la seva direcció i compleix els requisits per a ser considerada com a tesi doctoral.

Firma i data

M. Francisca Hinarejos Campos, professora del Departament de Ciències Matemàtiques i Informàtica de la Universitat de les Illes Balears

FA CONSTAR:

que la present memòria *Fair Exchange In E-Commerce And Certified E-Mail, New Scenarios And Protocols* presentada per *Gerard Draper Gil* per optar al grau de Doctor en Informàtica, ha estat realitzada sota la seva direcció i compleix els requisits per ser considerada com a tesi doctoral.

Firma i data

CONTENTS

LIST OF FIGURES

ACRONYMS

ABSTRACT

We are witnessing a steady growth in the use of Internet in the electronic commerce field. This rise is promoting the migration from traditional processes and applications (paper based) to an electronic model. But the security of electronic transactions continues to pose an impediment to its implementation. Traditionally, most business transactions were conducted in person. Signing a contract required the meeting of all interested parties, the postman delivered certified mail in hand, and when paying for goods or services both customer and provider were present. When all parties are physically present, a transaction does not require a complex protocol. The participants acknowledge the presence of the other parties as assurance that they will receive their parts, whether a signature on a contract, or a receipt, etc. But with e-commerce growing in importance as sales and business channel, all these transactions have moved to its digital counterpart. Therefore we have digital signature of contracts, certified delivery of messages and electronic payment systems. With electronic transactions, the physical presence is not required, moreover, most of the times it is even impossible. The participants in a transaction can be thousands of kilometers away from each other, and they may not even be human participants, they can be machines. Thus, the security that the transaction will be executed without incident is not assured *per se*, we need additional security measures. To address this problem, fair exchange protocols were developed. In a fair exchange every party involved has an item that wants to exchange, but none of the participants is willing to give his item away unless he has an assurance he will receive the corresponding item from the other participants. Fair exchange has many applications, like digital signature of contracts, where the items to be exchanged are signatures on contracts, certified delivery of messages, where we exchange a message for evidence of receipt, or a payment process, where we exchange a payment (e-cash, e-check, visa, etc.) for digital goods or a receipt.

The objective of this dissertation is the study of the fair exchange problem. In particular, it presents two new scenarios for digital contracting, the Atomic Multi-Two Party (AM2P) and the Agent Mediated Scenario (AMS), and proposes one optimistic contract signing protocol for each one. Moreover, it studies the efficiency of Multi-Party Contract Signing (MPCS) protocols from their architecture point of view, presenting a new lower bound for each architecture, in terms of minimum number of transactions needed. Regarding Certified Electronic Mail (CEM), this dissertation presents two optimistic CEM protocols designed to be deployed on the

current e-mail infrastructure, therefore they assume the participation of multiple Mail Transfer Agents (MTAs). In one case, the protocol assumes untrusted MTAs whereas in the other one it assumes each User Agent (UA) trusts his own MTA. Regarding payment systems, this dissertation presents a secure and efficient electronic bearer bank check scheme allowing the electronic checks to be transferred fairly and anonymously.

RESUM

L'ús d'Internet en l'àmbit del comerç electrònic està experimentant un creixement estable. Aquest increment d'ús està promovent la migració de processos tradicionals i aplicacions (basades en paper) cap a un model electrònic. Però la seguretat de les transaccions electròniques continua impedint la seva implantació. Tradicionalment, la majoria de les transaccions s'han dut a terme en persona. La firma d'un contracte requeria la presència de tots els firmants, el carter entrega les cartes certificades en mà, i quan es paga per un bé o servei ambdós venedor i comprador hi són presents. Quan totes les parts hi són presents, les transaccions no requereixen un protocol complex. Els participants assumeixen la presència de les altres parts com assegurança que rebran el que esperen d'elles, ja sigui la firma d'un contracte, un rebut d'entrega o un pagament. Però amb el creixement del comerç electrònic com a canal de venda i negoci, totes aquestes transaccions s'han mogut al seu equivalent en el món electrònic. Així doncs tenim firma electrònica de contractes, enviament certificat de missatges, sistemes de pagament electrònic, etc. En les transaccions electròniques la presència física no és necessària, de fet, la majoria de vegades és fins it tot impossible. Els participants poden estar separats per milers de kilòmetres, i no és necessari que siguin humans, podrien ser màquines. Llavors, la seguretat de que la transacció s'executarà correctament no està assegurada *per se*, necessitem proporcionar mesures de seguretat addicionals. Per solucionar aquest problema, es van desenvolupar els protocols d'intercanvi equitatiu. En un intercanvi equitatiu totes les parts involucrades tenen un objecte que volen intercanviar, però cap de les parts implicades vol donar el seu objecte si no té la seguretat que rebrà els objectes de les altres parts. L'intercanvi equitatiu té multitud d'aplicacions, com la firma electrònica de contractes, on els elements a intercanviar son firmes de contractes, enviament certificat de missatges, on s'intercanvien un missatge per una evidència de recepció, o un procés de pagament, on intercanviem un pagament (e-cash, visa, e-xec, etc.) per bens digitals o per un rebut.

L'objectiu d'aquesta tesi és estudiar el problema de l'intercanvi equitatiu. En particular, la tesi presenta dos nous escenaris per a la firma electrònica de contractes, l'escenari *multi-two party* atòmic i l'escenari amb agents intermediaris, i proposa un protocol optimista per a cada un d'ells. A més, presenta un estudi de l'eficiència dels protocols de firma electrònica multi-part (Multi-Party Contract Signing (MPCS) protocols) des del punt de vista de la seva arquitectura, presentant una nova fita per a cada una, en termes de mínim nombre de transaccions

necessàries. Pel que fa al correu electrònic certificat, aquesta tesi presenta dos protocols optimistes dissenyats per a ser desplegats damunt l'infraestructura actual de correu electrònic, per tant assumeix la participació de múltiples agents de transferència de correu. Un dels protocols assumeix que cap dels agents de transferència de correu participants és de confiança, mentre que l'altre assumeix que cada usuari confia en el seu propi agent. Pel que fa a sistemes de pagament, la tesi presenta un esquema de xec bancari al portador, eficient i segur, que garanteix que la transferència dels xecs es fa de manera anònima i equitativa.

Before starting I would like to thank everyone who helped me to get here. This document is the result of some years of work, but even though it bears my name, the merit is not just mine.

The merit goes to my own family, who always supported me... now you're going to Belgium on an Erasmus, now you're going to Barcelona to finish what?, now you're going on an Erasmus again, to Norway. You start working, about time! And now you start studding again, a PhD, What's that?. But it doesn't end here, now you're going to Singapore, and now to Austria! Son, will you ever finish? Well, dear parents, uncles, brothers and nephews, with this document I've reached a milestone, but I warn you... It doesn't end here.

The merit goes to my thesis directors, my colleagues, my supervisors and friends abroad. They have given me infinite patience. Always open to discuss my new *idea*, and to right me every time I've been wrong... which never happens!

The merit goes to my friends who have been with me as student fellows, coworkers, flatmates, and over all as friends. Giving me advice (thanks but I don't need it), even offering me a job position... (thanks but I refuse).

And It's better to leave it here, lest some of the *acknowledged* comes to claim a piece of PhD!

THANK YOU ALL!!!

Abans de començar voldria donar les gràcies a tothom que m'ha ajudat a arribar fins aquí. Aquest document és el resultat d'uns quants anys de feina, però tot i que porta el meu nom, el mèrit no és només meu.

El mèrit és de la meva família, que sempre m'ha recolzat... ara te'n vas d'Erasmus a Bèlgica, ara a Barcelona a acabar el què?, ara te'n tornes d'Erasmus a Noruega. Comences a fer feina, ja era hora! I ara tornes començar a estudiar, un doctorat, i això què és?. Però no s'acaba aquí, ara te'n vas a Singapur, i ara a Àustria! Nen, acabaràs algun dia? Doncs bé, papas, tiets, germans i nebots, amb aquest document he complert una fita, però aviso... això no s'acaba aquí.

El mèrit és dels meus directors de tesi, dels meus companys de feina, dels meus supervisors i amics "allende los mares". La paciència infinita que han tingut amb mi (espero que encara els hi quedi una mica). Sempre oberts a discutir la meva darrera *idea*, a corregir-me sempre que m'he equivocat, que és... mai!

El mèrit és dels amics que m'han acompanyat com a companys d'estudis, companys de feina, companys de pis, companys de... gresca (no tot es estudiar i fer feina). Però sobretot com amics de *la vida y el deporte*, donant-me consells (gràcies però no em fa falta), fins i tot oferint-me feina.

I millor que ho deixi aquí, no sigui que algun d'aquests *agraïts* em reclami un troç de doctorat!

GRÀCIES A TOTS!

INTRODUCTION

The expansion of Internet has been fueled by the evolution of fixed broadband technologies first (ISDN, xDSL, Fiber,...), and the progress of mobile broadband technologies later, along with the mobile terminals (smartphones, tablets, etc.). In 2001 only 8% of the world population had access to Internet, while in 2011 this number increased to a 32,5% (figure 1.1). Governments have also played an important role in the growth of Internet, developing policies to improve the competitiveness of the electronic communications market. With the Universal Service Directive [1, 2] the European Union (EU) defines universal service as: *minimum set of services of specified quality to which all end-users have access, at an affordable price in the light of specific national conditions, without distorting competition.* In particular, the eighth paragraph of the Directive 2002/22/EC [1], states that all users must be able to have Internet access with a minimum quality, limited to narrowband connections for primary location/residences. This limitation was eliminated in a later modification made by Directive 2009/136/EC (fifth paragraph) [2], delegating to the Member States of the EU the decision of the minimum quality requirements. In the case of Spain, the article 52 from the *Ley 2/2011 de 4 de marzo* [3], defines the minimum quality of the Internet Universal Service must be at least 1Mbit per second (download), and 256Kb per second (upload).

E-commerce can be defined as the object of conducting business transactions over electronic means, including: Internet, fax, sms, Electronic Data Interchange (EDI), etc. A company selling books by phone (typically a landline) or through infomercials or teleshopping is *doing e-commerce*. It is a typical example of e-commerce before the boom of Internet, but nowadays most e-commerce transactions are, at some point, conducted over the Internet. The first image that comes to our mind when we think of e-commerce is retail shopping. As customers, we access

to some online store and we order some products or services. But there are many other activities that can also be considered as e-commerce, like online advertising, online auctioning, etc. If we focus on the participants in a commercial transaction, a typical classification of e-commerce would be the following:

- Business to Business (B2B) where both participants are companies. An example could be a company buying travel services for their employees' business trips.

- Business to Consumer (B2C) where the business transaction is between a company and a customer (end-user). The example of retail-shopping, is a B2C scenario.

- Consumer to Consumer (C2C) where both participants are end-users. Auction sites where users sell second hand or new products directly to other users, would be a C2C scenario.

An alternative classification for e-commerce defines a subgroup named Machine to Machine (M2M) where e-commerce transactions are automated and executed by machines. But as technology advances, e-commerce also evolves and new scenarios and applications appear, like mobile commerce (m-commerce) where business transactions are executed from mobile devices (smartphones, tablets, etc.), or social commerce (s-commerce) where e-commerce interacts with social networks providing customers with tools to help them through the decision process (e.g., customer ratings, likes, reviews, etc.).

E-commerce offers many benefits to consumers and providers. Through Internet, providers can have immediate access to millions of potential consumers. And at the same time, consumers are just a few clicks away from any service or provider they could need. Providers can reduce or even eliminate their products' stock, enhance the processes and communication flow with consumers (customer loyalty programs, behaviour analysis, etc.) and providers, etc. Consumers can compare products and services from many different providers and choose the one that suits them better, etc.

We cannot deny that e-commerce has become an important sales and business channel, even now when we are still suffering an economic crisis. According to the last report from the ONTSI[1] on B2C e-commerce, in 2011 the Spanish B2C market generated a turnover of 10.917 M €, an increase of 19,8% over 2010, and close to 100% over 2007, when the B2C market generated a turnover of 5.911 M €. The study also reflects the behaviour of Spanish customers, highlighting price and comfort as the main drivers behind their decision to buy on Internet. Another interesting result is the classification of the online purchases in business sectors, being the touristy (travel tickets, accommodation services, etc..) related services the most demanded.

But the study also reflects the obstacles that customers find when they have to decide whether to make an online purchase or not: the lack of trust is the main

---

[1]http://www.ontsi.red.es/ontsi/ Observatorio Nacional de las Telecomunicaciones y de la Sociedad de la Información
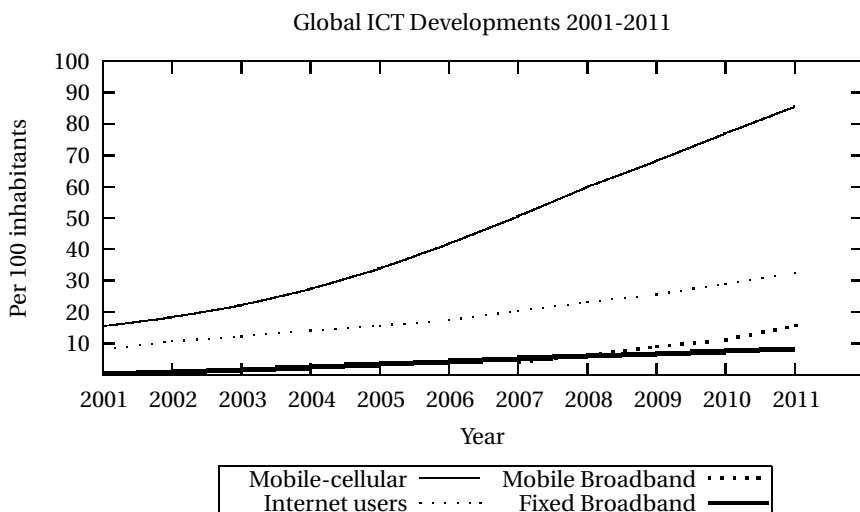
Figure 1.1: Global ICT Developments, 2001-2011 (Source: ITU-T)

obstacle to prevent it. More than 50% of customers (people who has purchased something within 2011) are concerned about the use of their personal information. Among ex-customers (people who bought something in 2010 or previous years, but not anymore), 22% think the system is not secure. This percentage increases to more than 50% among non-customers (people who has never bought through Internet), who also shows their distrust (45%) regarding the payment systems. This leads us to think that there are still security problems related to e-commerce that are not well solved.

In the physical world there are many situations in which we give the result for granted. When we sit in a terrace and order breakfast, we expect to receive food in exchange for money, when we sign a job offer, we assume we will get a copy of the signed contract (by all parties), and if we send a certified mail, we expect to receive evidence of reception. But it is not just some food and some money, or some contract, or some reception ticket. We want a particular kind of food in exchange for a particular amount of money, we want a copy of the same contract we have signed (which we agree on terms), and the evidence of receipt of the message we have sent. In any situation we assume we will receive some particular item in exchange of some other item we have. In other words, we expect the exchange to be fair. This fairness is based in the fact that we trust the other part will fulfil his duty, and in case they do not do it, we know who is to blame and we can report it to the pertinent authority.

But these kind of trusted relations do not happen in the electronic world. The problem is the ubiquity of the entities participating in the transaction and the na-

3

ture of the digital world. We do not really know who the other parties are, we are not even sure if they are a "who" or a "what", e.g., the other parties can be a software application. Moreover, digital resources can be forged or copied. Therefore we need mechanisms to ensure fairness in electronic transactions, and those are the protocols for Fair Exchange of Values.

1.1 E-commerce and Fair Exchange

Fair Exchange of Values is a problem that aims to solve a scenario where two parties, Alice and Bob, want to exchange some item between them, but Alice does not want to give her item to Bob unless she has an assurance that Bob will give her the item she is waiting for. When we have more than two parties involved, we talk about Multi-Party Fair Exchange. It has many applications, such as digital signature of contracts, where the items to be exchanged are signatures on contracts, certified delivery of messages, where we exchange a message for evidence of receipt, or a payment process, where we exchange a payment (e-cash, visa, e-check, etc.) for digital goods or a receipt.

The scientific community is working since late 1980's on secure protocols for the Fair Exchange of Values. Initially, two different research lines appeared to address fairness, protocols relying on a Trusted Third Party (TTP) and protocols without TTP. The TTP is an external entity who guarantees the fairness in a transaction. The solutions without TTP were based on the gradual exchange of secrets [4, 5, 6] or on probabilistic approaches [7, 8]. In these protocols the probability of achieving fairness increases along with the number of messages exchanged among the participants, the more messages exchanged the higher probability of achieving fairness. The main drawbacks in those kind of proposals are the assumption of equal computational power (in the case of gradual exchange of secrets), which is unlikely to happen in the real world (e.g., an end-user vs. a large corporation), and the great number of transmissions needed to complete the exchange in both cases. Therefore the majority of scientific proposals tend to use TTPs. Fair exchange protocols are usually classified in three groups, according to their use of the TTP:

Inline TTP In these protocols the TTP acts as a proxy between the participants, intervening in each protocol step (the participants do not have direct communication). It means that the TTP has to process each message sent during a protocol execution, with the consequent computational cost, and the risk of becoming a bottleneck.

Online TTP In an online TTP protocol the TTP participates in every protocol execution but not in every step. The participants communicate directly with each other, until they reach a point in the protocol execution where they require something from the TTP, typically at the beginning or end of the protocol. Online TTPs are subjected to a lower computational load that inline TTPs, and the risk of becoming a bottleneck is much lower.

Offline TTP Also called optimistic protocols. In these protocols the TTP only intervene in case of dispute, which is expected to be an exceptional case. If all participants behave according to the protocol rules and there is no network failure, the protocol will end without contacting the TTP. In case there is a dispute, any participant can contact the TTP and claim the missing evidence.

Despite their use of the TTP, there are other properties we can use to classify fair exchange protocols, like synchronous vs. asynchronous, two-party vs. multi-party, or their protocol architecture (ring, serial, star, etc.). Synchronous solutions use a time deadline to specify a limit for the duration of the protocol execution, while asynchronous do not have it, they rely on the protocol flow to end it. The main drawback to synchronous solutions is the clock synchronization among all the participants, but they are usually simpler than asynchronous solutions, which usually have a more complex execution flow. The difference between two-party (Alice and Bob) and multi-party (Alice, Bob, Chris,...) solutions is the number of participants, which will be determined by the application scenario. Finally, the protocol architecture is established by the communication flow between the participants, and in some cases, it can be restricted by the application.

1.2 Fair Exchange Requirements

Fair exchange and its applications is a problem that has been widely studied, resulting in a great number of heterogenous solutions. Even though we have a set of features we can use to classify fair exchange protocols (e.g., asynchronous, optimistic, number of participants), solutions within the same classification group can still differ on the set of security requirements the protocol meets, moreover, even the same security requirement can have different meanings depending on the author.

Nevertheless, most authors agree on a set of minimum requirements that fair exchange protocols must meet. In particular, one of the most cited references to define the requirements for optimistic fair exchange protocols is Asokan *et al.* [9]: *effectiveness, fairness (strong and weak), timeliness, non-repudiation and verifiability of the TTP*. Later, Zhou *et al.* [10], re-formulated these requirements for fair exchange.

Fair Exchange An exchange is considered to be fair if dishonest participants cannot gain advantage over the honest ones, neither during the exchange nor after it.

We will classify the security requirements for fair exchange protocols into core requirements (section 1.2.1) and additional requirements (section 1.2.2). The core requirements are the minimum set of requirements a protocol must meet to be fair. The additional requirements will provide the protocol with other desired properties, such as verifiability of the TTP. Requirements particular to fair exchange applications, like abuse-freeness for digital signature of contracts (chapter 2), non-selective receipt for certified e-mail (chapter 3) or unforgeability for electronic checks (chapter 5), will be addressed in the corresponding chapter of this dissertation.

1.2.1 Core Requirements

By definition, in an optimistic protocol the TTP only intervenes in case of a dispute arises. This is reflected in the effectiveness requirement, therefore it is a core requirement for optimistic fair exchange protocols.

Effectiveness If all participants in a fair exchange protocol behave correctly (and there are no network errors), the protocol will finish without the intervention of the TTP.

Fairness is a core property of fair exchange protocols, yet we can find different definitions in the scientific literature [9, 11, 12, 13]. The most referenced definition is the one by Asokan *et al.* [9], where two notions of fairness are provided: *strong* and *weak*. Intuitively, strong fairness means that either all the honest participants receive the item they expect from the other participants, or none receives it. While weak fairness means that whatever happens, the honest participants will be able to prove they behaved honestly (according to the protocol).

Strong fairness Upon finalization of a fair exchange protocol, either all honest participants have the item they expected from the other participants, or all of them have proof that the exchange has been canceled. None of the participants can receive evidence that contradicts the final state of the protocol execution.

Weak fairness Upon finalization of a fair exchange protocol, either strong fairness is met or all honest participants can prove they have behaved correctly.

Another core requirement for fair exchange protocols is non-repudiation. The non-repudiation requirement assures that the protocol generates enough evidence to allow the honest participants to maintain fairness, even when the other participants misbehave.

Non-repudiation Upon finalization of a fair exchange protocol, none of the participants can deny having participated. In particular, the participants cannot deny having originated (non-repudiation of origin) or received (non-repudiation of receipt) the items exchanged.

The timeliness property assures that any participant in a fair exchange protocol can finish its execution any time, maintaining fairness. And why is timeliness important? Imagine Alice, Bob and Charles initiate a fair exchange protocol. Once the protocol reaches the point where the only item to be exchanged left is Charles', he decides to delay his delivery, hoping to gain advantage on the situation. If Alice and Bob cannot contact the TTP to finish the protocol execution, they will be at Charles' will, they may have resources or goods committed to this fair exchange.

Timeliness Any participant in a fair exchange protocol can be sure that the duration of the protocol execution is finite. And once the protocol is finished, any honest participant will maintain the level of fairness obtained.

Lemma 1.1 *An asynchronous optimistic protocol meeting timeliness, can only meet the weak fairness requirement.*

Proof To meet the timeliness requirement any party must be able to contact the TTP to finish the protocol execution and obtain evidence of the result, either finished or canceled. Therefore, a dishonest party P_d can follow the protocol to the end (optimistic execution, without intervention of the TTP), obtaining the evidence of all the other participants. Later, P_d can contact the TTP and obtain a cancel evidence. The TTP only has the information received in P_d's request, thus if P_d forges a request complying the TTP's criterion for canceling the protocol, he will get a cancel token. In this situation, P_d will have evidence of finalization and evidence of cancelation. But the latter will not be useful in a dispute resolution because all other parties will have enough evidence to prove that P_d misbehaved, therefore the protocol can only meet weak fairness.

1.2.2 Additional Requirements

The TTP is an external entity that ensures fairness is maintained in fair exchange protocols: participating as a proxy between participants (inline TTP), taking part only in certain protocol steps (online TTP), or being involved only when a dispute arises (offline TTP). Usually, it is assumed to be trusted, i.e., it behaves according to the protocol rules, without favoring any user. But the usual assumption also implies that the TTP never fails. We cannot distinguish between a malfunction and a misbehaviour. Some authors adopt a more flexible approach, assuming it is semi-trusted. The term semi-trusted was first introduced by Franklin and Reiter [14], referring to a TTP that can misbehave (or malfunction) but not collude with other entities. In this case, when we assume a semi-trusted TTP, if the TTP malfunctions or misbehaves, an honest participant could lose fairness, therefore we need the verifiability of TTP requirement.

Verifiability of the TTP In a fair exchange protocol execution where the TTP intervenes, if the TTP malfunctions (either because of a system/network failure or a malicious act), the participants can prove it.

Transparent (or invisible) TTP is a requirement first introduced by Micali [15] in the field of certified e-mail. Informally, the objective of this requirement is to prevent the detection of the intervention of the TTP in a protocol execution. The rationale behind is related with trust. Neither the participants nor the TTP can always distinguish between a dispute related to a network or a system failure, from a dispute related to malicious users or a protocol misuse. Therefore to avoid the mistrust of protocol executions that required the intervention of the TTP to solve a dispute, a transparent TTP could be desirable. But designing a protocol where verifiability and transparency are compatible is very complex. In fact, to date, there is only one proposal where verifiability and transparency seems to coexist, Huang *et al.* [16].

Transparency of the TTP Upon finalization of a fair exchange protocol execution, the evidence generated by the TTP does not allow to detect whether the TTP has participated in the execution or not.

Finally, another additional requirement for fair exchange protocols is confidentiality. This requirement prevents non-authorized entities to have access to the items exchanged among the participants during the protocol execution.

Confidentiality In a fair exchange protocol, only the authorized participants must have access to the item exchanged. Not even the TTP must be able to access to its content.

1.3 Objectives of this Dissertation

A purchase of services (or products, digital goods, etc.) from an online provider is a contract signing process, a particular case of fair exchange where the items to be exchanged are signatures on contracts. And fair exchange is not a new problem, but as the number of Internet users grow and the Information Communications Technologies advance, the applications of fair exchange evolve and new scenarios appear. Nowadays it is very common for consumers (in B2C is the end-user while in the B2B is another company) to find different online providers offering the same or similar services: consumers and providers are closer than ever.

One of these new scenarios is the Atomic Multi-Two Party (AM2P) scenario. The direct access to providers becomes especially interesting for consumers when they decide to purchase a multi-product service, i.e., a package composed of several services (flights, hotels, cars, tours, etc.). The end-users can compare different offers for the same service from different providers and choose the most suitable one. In a similar way, companies can have online access to their providers. This real-time access to providers is especially beneficial for businesses which require a precise control of stocks and inventory. They can make real time orders of the many products needed, assuring they will get them on time and within budget. The problem arises at the moment of purchasing those products/services: for the consumer to obtain her desired objective, she needs to purchase X different services from Y different providers, therefore, she wants to have a commitment to all providers or none, otherwise her objective will not be complete.

In an AM2P scenario, we have N parties (1 consumer, C, and $N-1$ providers, P_i) distributed as a set of $(N-1)$ pairs of parties $\{C, P_1\}$, $\{C, P_2\}$...,$\{C, P_{(N-1)}\}$, that want to sign a set of $(N-1)$ contracts $\{M_1, M_2, ..., M_{(N-1)}\}$ pairwise, i.e., C and P_1 want to sign the contract M_1, C and P_2 the contract M_2, etc. But C does not want to send her signature unless she has assurance she will receive all signatures from $P_1, ..., P_{(N-1)}$, neither P_i wants to send his signature unless he receives C's signature on the contract M_i. Despite the great efforts dedicated to the study of Multi-Party Fair Exchange, there are no solutions addressing this problem: digital signature of contracts in AM2P scenarios. In fact, digital signature of contracts in AM2P scenarios was a problem yet to be solved.

If we switch the attention from the consumers to the intermediaries, we have another new scenario, the Agent Mediated Scenario (AMS). The proximity of consumers and providers could lead us to the conclusion that intermediaries are not needed anymore. However, the truth is that the existence of intermediaries in electronic commerce is not unusual, moreover, in some cases is even desirable, as in the online tourism industry [17]. Travel agencies (intermediaries) sell leisure activities to consumers, but at the same time they buy these leisure activities from other providers. The problem is that the intermediary must execute two transactions: one as provider with the consumer and another one as consumer with its own provider. But in fact, these two transactions are part of the same one, a transaction between a consumer C and some provider P, even though they do not have direct contact. In this situation the intermediary is in an unfair position, because it has no means to relate its transaction with the consumer, with its transaction with the provider, more specifically, it cannot directly relate the results of both transactions. Consider the situation where the intermediary signs a contract with a consumer to deliver some services, but the intermediary's provider does not sign the corresponding contract with the intermediary. He may be forced to fulfil a contract with a consumer without having the necessary services.

Many research efforts have been dedicated to the study of the fair exchange problem, including its application to Multi-Party Contract Signing (MPCS) protocols. As a result we can find numerous proposals for MPCS protocols in the scientific literature [18, 19, 20, 21, 22, 23]. Even though there is a consensus about some of the requirements all MPCS protocols must meet, like fairness, most authors impose a set of different requirements. Moreover, sometimes they do no agree on the definition of these requirements. Within these proposals, we can find some that claim to propose optimal solutions or define lower-bounds to design MPCS protocols [18, 22, 23], but the different criteria applied to define requirements like fairness, or terms like round, step, etc. make it difficult to assert that those solutions are optimal. Moreover, even though we can use different topologies (ring, star,...) to design MPCS protocols, none of these optimal solutions contemplates the influence of the topology on the overall result.

Nowadays almost anything can be stored and processed as digital data: documents, music, images, etc., without loosing any information and even increasing the possibilities of their applications. As the information is converted to bits (zeroes and ones) and electronic impulses, so are the processes involved: payment, digital signature, etc. But still, there are a few things that are only widespread in the conventional world, like sending certified mails. With certified mail, items are handed over to the recipient only in exchange for a receipt. Moreover, by signing this receipt, the recipient cannot deny having participated in this exchange. Certified mail can thus be considered as a postal security service, which ensures a fair exchange and non-repudiation. Standard communication systems like Internet e-mail, equivalent to postal mail, have no security provisions in terms of a fair and non-repudiable receipt of message exchange.

Certified mail is a postal security service, therefore, Certified Electronic Mail (CEM) should be an e-mail security service. The majority of CEM proposals we can

find in the scientific literature, require a direct interaction between the sender and the recipient, which does not occur in traditional e-mail interactions: messages are relayed through Mail Transfer Agents (MTAs). This circumstance has been taken into account by some authors, which have adopted the model of semi-trusted MTAs to decouple the sender from the recipient. Still, most of these approaches use only one MTA, which does not fully reflect the traditional e-mail architecture, where we can have many MTAs between the sender and the recipient. Moreover, in these protocols senders and recipients must be registered with the same transfer agent. Regarding the security properties these proposals offer, most of them lack of verifiability of TTP, which assures the participants that the TTP cannot be dishonest without them being able to prove it. As certified mail users, we are used to trust the postal services, i.e., we trust the postman will only deliver the package/message in exchange of a signed receipt. But this trust relationship between users and providers does not have a counterpart in the digital world.

Probably this disconnection between the scientific proposals and e-mail infrastructure has made that no real implementation of those technical solutions can be found in general use. But, in recent years, some organizations (especially public administrations) have demanded a CEM service. So, in different countries we can find CEM services, many times with a limited scope, and without a standard guaranteeing the interoperability among different CEM services in different countries. Obviously, it is not a desirable situation, and for this reason some additional work is necessary.

As we previously explained, the boom of Internet has boosted the growth of electronic commerce. Nowadays, the most used payment systems are credit card, money transactions in COD (Collect On Delivery) or bank transfers. Other payment systems such as electronic money or micro-payment systems have a token presence. However, mentioned systems are not suitable for all scenarios of electronic commerce. Credit card payments have limitations when dealing with large amounts to be paid, and they do not provide anonymity to the participants. Money transactions in cash, offer anonymity to the transaction parties, but require the participants to carry these large amounts in cash, and some consumers may not feel comfortable carrying large amounts of cash. Even though bank transfers allow the payment of large amounts and do not have credit limitations, they do not provide anonymity to the consumer, allowing the profiling of the consumer's habits.

Among the traditional payment systems, we have one that allow users to transfer large amounts of money while maintaining the anonymity of the participants, the checks. We can find different kinds of paper-based checks: order or bearer checks, open or crossed, bank or personal checks, certified or non-certified, etc. Each kind of check presents different features and different degree of security. The paper based check is still widely used in traditional commerce; it has features that differentiate it from other payment systems. The check allows for payments of large amounts (thereby overcomes the limitations of the credit card [24, 25]), can achieve a high degree of anonymity (as opposed to bank transfers or credit card) and can also be transferable (operation known as endorsement if a check is nominal, or delivery if it is bearer check). So, the first conclusion is that we need a functional equiv-

alent in the electronic world for paper-based checks: electronic checks (e-checks).

There are real experiences of e-checks, among which stands out, above all, the Financial Services Technology Consortium [26]. But these experiences are limited in scope and they do not retain all the desired properties of paper based checks. Obviously, an e-check system has to be practical, but the main concern when designing a system for e-check is security. And the first issue to be resolved is that in the electronic world there is no original and copy (unlike the paper world). Therefore, security measures should be taken that prevent the same e-check can be used to make various payments. A second important issue for users is to guarantee that e-checks will be paid (avoid the situation in which the check bounces and cannot be cashed). And finally, a third issue is that users are becoming increasingly concerned about the privacy of electronic transactions (especially when dealing with electronic payments).

1.3.1 Contributions

This dissertation presents contributions on three fields related to fair exchange: digital signature of contracts, certified e-mail and payment systems. In the following paragraphs these contributions are presented.

Digital Signature of Contracts

The contributions in the field of digital signature of contracts presented in this dissertation are manifold. First, it presents two new scenarios for the digital signature of contracts: the AM2P and AMS scenarios. Both scenarios are described and the security requirements for fairness are developed, adapting them when necessary. For each new scenario an optimistic protocol is presented, meeting all the core security requirements, and some additional ones. Finally, the protocols are accompanied with a revision of the security requirements, to prove they are met.

On the other hand, this dissertation presents an efficiency study of fair exchange protocols for Multi-Party Contract Signing (MPCS), from their architecture point of view, an approach that has not been previously used. A set of common topologies is presented and defined: ring, star sequential and mesh. Some common terms and notions, as the meaning of round and message, are defined according to the topology where they are applied. The suitability of such common terms to measure the efficiency of the protocols is discussed, and the measure of transmissions is presented as alternative. Finally, an optimistic MPCS protocol meeting all the core security requirements is presented for each topology, proving that in each case the solution is optimal, improving the existent proposals of lower-bounds for fair MPCS protocols.

Certified Electronic Mail

This dissertation presents two optimistic CEM protocols designed considering their deployment into the traditional e-mail infrastructure. Both proposals allow the use of multiple MTAs, avoiding inline and not-verifiable TTPs and letting both sender and recipient to choose their own MTA. This way, the CEM protocols can be more

easily deployed on the Internet, allowing the use of Simple Mail Transfer Protocol (SMTP) as underlying protocol for transferring the messages.

The first proposal is the first of its kind to consider an optimistic approach with multiple untrusted MTAs, i.e., they can cheat/fail and collude with other entities. This strong assumption increases the complexity of the protocol, therefore in the second proposal this assumption has been relaxed, allowing users to trust their own MTA, while the others remain untrusted. Both proposals meet all the core security requirements (effectiveness, fairness, timeliness and non-repudiation), some additional ones (verifiability of the TTP and confidentiality) and a specific one, non-selective receipt. Like many CEM optimistic protocols, both approaches provide a non-repudiation of origin as well as a non-repudiation of receipt service, attesting, the latter, that the recipient actually has received the message. Since both protocols support multiple MTAs, they further provide a non-repudiation of submission service, attesting that the sender has actually handed over a message to his MTA, and non-repudiation of delivery, attesting that the message has been delivered to the recipient's MTA.

Payment Schemes

This dissertation proposes an electronic bearer bank check scheme allowing the electronic checks to be transferred fairly and anonymously. In addition, the payment of this kind of electronic check is guaranteed because it is a bank check, and this feature can facilitate the expansion of such mean of payment. Moreover, the proposed scheme provides a lightweight solution for the client side because this is, normally, the more restrictive participant in terms of computational resources.

1.3.2   Outline of this Work

The solutions for digital signature of contracts are presented in chapter 2. In chapter 3 the solutions for certified e-mail are presented, and the payment protocol is presented in chapter 4. Finally, in chapter 5 the final conclusions are presented.

1.4   Publications and Funding

The work included in this thesis has been presented in national and international conferences and journals where experts have reviewed it and contributed to its improvement with their reviews. Following we have the list of accepted publications related to this dissertation:

1. G. Draper-Gil, J. Zhou, J.L. Ferrer-Gomila, An agent-mediated fair exchange protocol, 2010 International Conference on Information and Communications Security (ICICS'10), Vol. 6476 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 235-250.

2. M.F. Hinarejos, J.L. Ferrer-Gomila, G. Draper-Gil, Ll. Huguet-Rotger, Anonymity and Transferability for an Electronic Bank Check Scheme, IEEE 11th

International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2012, pp. 427-435.

3. G. Draper-Gil, J.L. Ferrer-Gomila, M.F. Hinarejos, J.A. Onieva, J. López., Un Protocolo para la Firma de Contratos en escenarios Multi-Two-Party con Atomicidad, XII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2012), 2012.

4. G. Draper-Gil, J.L. Ferrer-Gomila, M.F. Hinarejos, J. Zhou, An Asynchronous Optimistic Protocol for Atomic Multi-Two-Party Contract Signing, The Computer Journal, 2013, Oxford

5. G. Draper-Gil, J. Zhou, J. Ferrer-Gomila, M.F. Hinarejos, An Optimistic Fair Exchange Protocol With Active Intermediaries, International Journal of Information Security, 2013, Springer.

6. G. Draper-Gil, J.L. Ferrer-Gomila, M.F. Hinarejos, A. Tauber, Certified Electronic E-mail, Architectures and Protocols for Secure Information Technology, 2013, IGI Global.

2

DIGITAL SIGNATURE OF CONTRACTS

As we have seen in section 1.1, fair exchange is a well known problem that has been extensively studied. But as e-commerce evolved, fair exchange scenarios have also evolved. This chapter is dedicated to the application of fair exchange to two of these new scenarios: Atomic Multi-Two Party (AM2P) scenarios (section 2.1) and Agent Mediated Scenarios (AMSs) (section 2.2). In particular, this chapter is dedicated to the application of digital signature of contracts to these scenarios, presenting one optimistic contract signing protocol for each scenario. Additionally, section 2.3 studies the efficiency of asynchronous optimistic Multi-Party Contract Signing (MPCS) protocols in relation to their topology. As a result, one optimal solution meeting all the core security requirements is presented for each topology.

2.1 Atomic Multi-Two Party (AM2P) Scenarios

Nowadays it is very common for consumers (in B2C is the end-user while in the B2B is another company) to find different online providers offering the same or similar services/products. In some sectors, like leisure activities, this direct access to providers becomes especially interesting for consumers when they decide to purchase a multi-product service, i.e., a package composed of several services (flights, hotels, cars, tours, etc.). The consumers can compare different offers for the same service from different providers and choose the most suitable one. In a similar way, companies can have online access to their providers. This real-time access to providers is especially beneficial for businesses which require a precise control of stocks and inventory. They can make real time orders of the many products needed, assuring they will get them on time and within budget. The problem arises at the moment of purchasing those products/services: for the consumer to obtain her desired objec-

tive, she needs to purchase X different services from Y different providers, therefore, she wants to have a commitment to all providers or none, otherwise her objective will not be complete. Without loss of generality and for ease of explanation, we will consider that the consumer wants a product/service from each provider.

In this section we present a new contract signing scenario, the Atomic Multi-Two Party (AM2P) scenario, where we have one consumer and many providers willing to sign a contract pairwise (consumer-provider), with the particularity that the consumer needs all the providers' signatures. We also propose a solution, an asynchronous optimistic protocol for AM2P contract signing, which is the first solution to solve this problem.

2.1.1   Motivation

In this section we will provide 2 examples of use cases to show the possible benefits of having AM2P solutions. In particular, we will present a B2C scenario, where a consumer purchases a leisure trip, and a B2B scenario, where a company purchases the materials to produce goods.

Business to Consumer (B2C)

Let us consider that Alice, a consumer, is looking to have an opera weekend in Milan. After checking on Internet, she finds a suitable offer for each service needed, from different providers: a two-way plane ticket from Barcelona to Milan (transportation), two nights in a hotel (accommodation) and two tickets to an opera play in La Scala (leisure activities). But Alice wants all services or none. It does not make sense to purchase the transportation and accommodation if she is not able to get the opera tickets, or to have the opera tickets without the transportation (she does not know if she will be able to find another offer). Since the services are booked from different providers, Alice cannot be sure she will get all of them, unless she uses an AM2P protocol.

This AM2P protocol could be implemented as a plug-in, specific service from a website, etc., where Alice would add her choices (plane ticket, accommodation, etc.), and click a "book" button. As a result, Alice would receive the corresponding evidence of signature from all services chosen. The whole protocol execution would remain transparent to the end-user, in this case, Alice.

Business to Business (B2B)

Let us consider that Acme is a company that manufactures products under demand, i.e., they do not have a stock of manufactured products neither raw materials. To plan the next quarter's production, Acme needs to assure the raw materials' supply. Therefore, she will contact each provider to gather information on costs and delivery time. Once she has found a set of providers that can supply her with the raw materials needed, satisfying her needs in terms of costs and delivery time (and any other criterion), she will place an order. But to assure the production, Acme

needs to be sure that all providers will comply with their agreement, otherwise the production may be delayed or the costs increased (the provider substitute may not agree with Acme's conditions).

In this case, the AM2P protocol could be a B2B utility for contracting. Providers would offer their services (products, digital goods, etc.) along with its conditions, and the clients could use it to locate the services they need, add them to a sort of shopping chart and sign the contracts, atomically. Again, the protocol execution would be transparent to the user, in this case, Acme.

2.1.2 Security Requirements

The requirements (core and additional) for optimistic fair exchange are defined in section 1.2. The solution presented in this section meets all the core requirements (effectiveness, fairness, timeliness and non-repudiation) and two additional ones, verifiability of TTP and confidentiality. Following we will detail the requirements for optimistic AM2P contract signing protocols, starting with the core requirements:

Definition 2.1 (*AM2P Effectiveness*) *If every participant involved behaves correctly, the consumer will receive her expected items (signatures) from the providers, and the providers will receive their expected item (signature) from the consumer, without TTP intervention.*

Definition 2.2 (*AM2P Weak Fairness*) *Upon finalization of an AM2P signature, an honest consumer will have the signature of all the providers, and all honest providers will have their corresponding signature from the consumer, or all honest parties will have enough evidence to prove they have behaved correctly in front of an arbiter.*

Definition 2.3 (*AM2P Timeliness*) *Any party involved in an AM2P signature can be sure that the protocol execution will be finished at a certain finite point of time. Once the protocol execution is finished, the level of fairness achieved by all honest participants cannot be degraded whatever the other participants do.*

Definition 2.4 (*AM2P Non-repudiation*) *In an AM2P signature that has involved a consumer and $N-1$ providers, neither the consumer nor the providers will be able to deny their involvement. In particular, given a signed contract M_i, neither consumer C nor provider P_i can deny having signed it.*

Finally, our optimistic AM2P contract signing protocol also meets two additional requirements:

Definition 2.5 (*AM2P Verifiability of TTP*) *If the trusted third party (TTP) misbehaves resulting in the loss of fairness for an honest party (consumer or provider), this party can prove the fact in front of an external arbiter.*

Definition 2.6 (*AM2P Confidentiality*) *Only the parties involved in a signature, consumer C and provider P_i, are allowed to know the content of the contract M_i. Not even the TTP can have access to the plain contract.*

Therefore, our asynchronous optimistic protocol must meet the following security requirements: *AM2P Effectiveness, AM2P Weak Fairness, AM2P Timeliness, AM2P Non-Repudiation, AM2P Verifiability of the TTP* and *AM2P Confidentiality*.

2.1.3 Previous Work

In an AM2P scenario the consumer C wants to sign $N-1$ contracts with $N-1$ providers pairwise, but the providers do not know each other: only C has a relationship with each P_i. Therefore we cannot use a ring architecture, where each participant X_i knows at least his neighbours $X_{(i-1)}$ and $X_{(i+1)}$. Instead of a ring architecture, our protocol uses a star architecture (figure 2.1), where the consumer contacts directly to each provider.

We can find other proposals for asynchronous multi-party signature of contracts [18, 19, 27, 20, 28], but none of them is addressed to AM2P scenarios. The objective of these multi-party proposals is to allow a group of N ($N \geq 2$) parties to fairly exchange their signatures on a pre-agreed contract M, being the contract M the same for each participant; meanwhile we need to exchange a set of $(N-1)$ different contracts $\{M_1, M_2, ..., M_{(N-1)}\}$, pairwise, between a consumer C and a provider P_i, where $1 \leq i \leq (N-1)$. Moreover, the consumer needs to acquire all $N-1$ signatures for the exchange to be considered fair. Therefore, we cannot apply these solutions to our scenario: AM2P.

Despite the great efforts dedicated to the study of Fair Exchange and contract signing, there are no solutions addressing signature of contracts in AM2P scenarios. But we can find a few proposals [29, 30, 31] that try to address a similar problem (AM2P), but in other scenarios.

In [29] Liu proposes an AM2P optimistic protocol for the exchange of payment for digital goods, composed of an exchange sub-protocol with 2 phases: negotiation and payment, and 3 resolve sub-protocols. The negotiation phase is a 4 steps cycle where the consumer and the providers agree on the terms of the exchange (price and product). If consumer and providers behave correctly, the payment phase requires the exchange of 5 messages between the consumer and each provider without TTP intervention. Even though the protocol achieves atomicity in the multi-two party exchange of payment tokens for digital goods, the problem solved differs from ours. We aim to the AM2P signature of contracts, therefore we want to achieve atomicity in the multi-two party exchange of digital signatures. Moreover, the proposal meets the non-repudiation and fairness requirements, but fails to meet verifiability of TTP and timeliness.

In [31] Onieva *et al.* present an extension of a multi-party non-repudiation protocol with online Trusted Third Party (TTP), which allows one originator ("Simple Origin") to send different messages to multiple recipients. In their paper, Onieva *et*

	Our proposal	Liu [29]	Onieva [31]	Yanping [30]
Scenario	contract signing	payment for goods	non-repudiation	non-repudiation
Atomicity	✓	✓	–	–
Effectiveness	✓	✓	–	✓
Fairness	weak	weak	strong	strong
Timeliness	✓	–	✓	✓
Non-repudiation	✓	✓	✓	✓
Verifiability TTP	✓	–	–	–
Confidentiality	✓	–	✓	✓

✓YES   –NO

Table 2.1: Multi-Two Party Solutions - A Comparative Analysis

al. classify the multi-party scenarios into two types: Simple Origin with Many Recipients for the exchange of the same Message ($SOMR-M$), and Simple Origin with Many Recipients for different Messages ($SOMR-M_i$). This second scenario type, $SOMR-M_i$, is similar to our AM2P scenario, though they aim to the exchange of messages meanwhile we aim to the signature of contracts. Moreover, atomicity is not addressed in Onieva *et al.* paper. Therefore their proposal cannot be applied to AM2P scenarios for signature of contracts.

Yanping and Liaojun [30] propose an optimistic multi-party non-repudiation protocol, which allows an originator to send different messages to multiple recipients, i.e., a multi-two party scenario. But again, their solution does not address atomicity, and their objective, non-repudiation, differs from ours, signature of contracts. Therefore, we cannot apply their proposal to AM2P scenarios for signature of contracts.

To the best of our knowledge, these references [29, 30, 31] are the only ones we can find in the literature related to the AM2P scenario, and as we can see in table 2.1 they do not solve our problem: signature of contracts in AM2P scenarios.

2.1.4 Overview and Notation

In this section we present an asynchronous optimistic protocol for AM2P contract signing. The protocol follows a star architecture (see figure 2.1), where the consumer C contacts all the providers P_i (with $1 \le i \le (N-1)$) at the "same time", and awaits for their response to continue with the protocol, i.e., she sends all $N-1$ messages at once, and awaits for the $N-1$ responses before continuing with the protocol execution. If the consumer fails to receive one or more responses she will stop the protocol execution and she will contact the TTP. All participants must agree to use the same TTP, e.g., they may accept it implicitly when agreeing to participate in the signature.

The messages sent from the consumer C to the providers P_i ($1 \le i \le N-1$) are called *commitments* (COM), and the corresponding responses from the providers to the consumer, are called *acceptances* (ACC). A round k is the exchange of a set of ($N-1$) *commitments* for the corresponding set of ($N-1$) *acceptances*. The $COM_{(k,i)}$
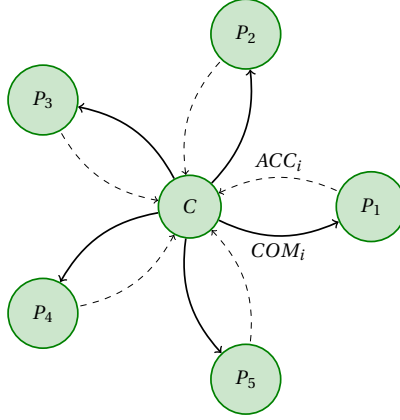
Figure 2.1: Star Architecture with $N = 6$, 1 Consumer and 5 Providers

and $ACC_{(k,i)}$ (k = round number, i = provider number) are the evidence that provider P_i and consumer C must receive, respectively. Commitments and acceptances are linked through the round number k.

Following we have the general notation we will use along the protocol description:

- **N** Number of participants: 1 Consumer and $N-1$ Providers.

- $\overline{\mathbf{X_{iZ}}} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_Z}\}$ Vector with Z elements, where $1 \leq i \leq Z$.

- **C** Consumer (in B2C is the end-user while in the B2B is another company).

- **P$_i$** Provider i, $1 \leq i \leq (N-1)$.

- **M$_i$** Message (contract) exchanged between the consumer C and the provider P_i. This contract must include the necessary information to execute the protocol. Some of these values are: the identities of participants C and P_i, the identity of the TTP, the number of rounds needed to finish the exchange without the intervention of the TTP, etc.

- **CID** unique Contract IDentifier. A random number used to uniquely identify a protocol execution. It is generated by C when initiates the protocol execution.

- **h(M$_i$)** Hash Function of message M_i.

- **S$_j$[M$_i$] = SK$_j$[h(M$_i$)]** j's Digital Signature on M_i (where SK_j is j's private key).

The TTP uses a set of rules to resolve the *resolution* sub-protocol requests fairly. These rules are based on a group of variables the TTP updates on every request received, indicating the state of a protocol execution identified by its CID. Following

we have these group of variables, its definition, and some notation used along the rules definition (see section 2.1.5).

- $\overline{\mathbf{X_N}}$ = $\{C, P_1, ..., P_{(N-1)}\}$ set of participants in an AM2P exchange.

- $\overline{\mathbf{XC}}$ set of participants who already requested resolution for the AM2P exchange with contract identifier "CID", and the contract M_i: $\{X_i, CID, h(M_i)\}$.

- $\overline{\mathbf{XA}}$ set of participants who have received a canceled token from the TTP, along with the CID and round value ($\{X_i, CID, k\}$).

- ***canceled*** boolean value stating that the exchange execution has been canceled if its value is true.

- ***signed*** boolean value stating that the contract has been signed if its value is true.

We can find different types of communication channels in the literature. Asokan *et al.* [32] defines three different channels: operational, reliable and resilient; and Ferrer-Gomila *et al.* [33] classifies them in operational, resilient and unreliable. In our protocol, we will assume that the communications between consumer C and providers P_i are executed through unreliable channels (messages can be lost), but messages between the TTP and the protocol participants are exchanged through a resilient channel (messages can be delayed but not lost).

2.1.5 Protocol for Fair Exchange in Atomic Multi-Two Party Scenarios

The asynchronous optimistic AM2P protocol is composed of two sub-protocols, *exchange* and *resolution*. If every party involved behaves honestly, the *exchange* sub-protocol will finish without TTP intervention.

Exchange sub-protocol

Table 2.2 shows the *exchange* sub-protocol execution flow and the corresponding values exchanged. Each sub-protocol complete execution is composed of N rounds, and each round requires the exchange of $N-1$ pairs of messages {COMmitment, ACCeptance} and thus $2N(N-1)$ messages are necessary. The reason why we need N rounds is explained later in this section. The evidence of signature is the corresponding to the round N, $COM_{(N,i)}$ and $ACC_{(N,i)}$ ($1 \le i \le N-1$), i.e., after the Nth round execution, every party involved will have the <u>corresponding signed</u> contract (the consumer requires the set of all $N-1$ contracts, $\overline{SignedContract_{i\,(N-1)}}$):

SignedContract$_i$ $\{CID, M_i, COM_{(N,i)}, ACC_{(N,i)}\}$

Each round starts when the consumer sends to each provider the corresponding commitment, $COM_{(k,i)}$. The commitments are the consumer signature on the contract identifier CID, the hash of the contract to be signed $h(M_i)$ (we do not want the TTP to have access to the contract data), and the round number k. This round

Exchange Sub-Protocol

Round		
1	$C \rightarrow P_i$	$CID, E_{k_i}(M_i), PK_{P_i}(k_i), 1, COM_{(1,i)}$
1	$C \leftarrow P_i$	$CID, 1, ACC_{(1,i)}$
$\vdots$	$\vdots$	$\vdots$
k	$C \rightarrow P_i$	$CID, k, COM_{(k,i)}$
k	$C \leftarrow P_i$	$CID, k, ACC_{(k,i)}$
$\vdots$	$\vdots$	$\vdots$
N	$C \rightarrow P_i$	$CID, N, COM_{(N,i)}$
N	$C \leftarrow P_i$	$CID, N, ACC_{(N,i)}$

N = number of participants and total rounds
k = round number
$E_{k_i}(M_i)$: M_i encrypted with symmetric key k_i
$PK_{P_i}(k_i)$: k_i encrypted with P_i's public key
$COM_{(k,i)}$: $S_C[CID, h(M_i), k]$
$ACC_{(k,i)}$: $S_{P_i}[CID, h(M_i), k]$

Table 2.2: Asynchronous Optimistic Atomic Multi-Two Party (AM2P) Exchange Sub-Protocol

number is the link between the commitment and the acceptance. Note that in the first message (see table 2.2), the consumer also sends the contract M_i encrypted with the symmetric key k_i, $E_{k_i}(M_i)$, and the key k_i encrypted with the provider P_i's public key, $PK_{P_i}(k_i)$, so P_i is the only one who can decrypt the content.

After sending the commitments, the consumer awaits to receive the answers, the acceptances, from the $N-1$ providers. The acceptance is equivalent to the commitment, but signed by the corresponding provider (see table 2.2). Once the consumer receives all $N-1$ acceptances, she will continue the execution starting the next round.

If the consumer fails to receive one, or more than one, of the provider's acceptances during round $k > 1$, she will execute the *resolution* sub-protocol before sending further commitments; otherwise she may lose atomicity on the entire AM2P signature.

Resolution sub-protocol

At any time, consumer (from round $k > 1$) and providers can contact the TTP to solve the protocol execution. The TTP uses a set of rules to solve the resolution requests fairly (table 2.3). During the first round ($k = 1$), any provider can contact the TTP to cancel the execution, while if $k > 1$ any party can contact the TTP to

Resolution Sub-Protocol (k = round number $1 \le k \le N$)

$X_i \longrightarrow TTP \ RES_{(k,i)}, EVRES_{(k,i)}$
If ($\{X_i, CID, h(M_i)\} \in \overline{XC}$ **)**
     reject request
else
     $\overline{XC} = \{X_i, CID, h(M_i)\} \cup \overline{XC}$
     If (k==1 and signed == false **)**
         canceled = true;
         $\overline{XA} = \{X_i, CID, 1\} \cup \overline{XA}$;
         $TTP \longrightarrow X_i \ Canceled_{TK}$
     else If($k == 1$ and signed == true **)**
         $TTP \longrightarrow X_i \ Signed_{TK}$
     else If($k > 1$ and canceled == false **)**
         signed = true;
         $TTP \longrightarrow X_i \ Signed_{TK}$
     else If($k > 1$ and signed == true **)**
         $TTP \longrightarrow X_i \ Signed_{TK}$
     else If($k > 1$ and canceled == true **)**
         If ($\forall \{X_i, r\} \in \overline{XA} \rightarrow r < k - 1$ **)**
             canceled = false; signed = true;
             $TTP \longrightarrow X_i \ Signed_{TK}$
         else
             $\overline{XA} = \{X_i, CID, k\} \cup \overline{XA}$;
             $TTP \longrightarrow X_i \ Canceled_{TK}$
         end If
     end If
end If

---

when X$_i$ = Consumer, C
$RES_{(k,i)} = CID, \overline{h(M_i)}_{(N-1)}, k$
$\overline{(COM_{(k,i)})}_{(N-1)}, \overline{(ACC_{((k-1),i)})}_{(N-1)}$
$EVRES_{(k,i)} = S_C[RES_{(k,i)}]$
$Signed_{TK} = \overline{S_{TTP}[CID, h(M_i), k, COM_{(k,i)}]}_{(N-1)}$
$Canceled_{TK} = \overline{S_{TTP}[CID, h(M_i), k, canceled]}_{(N-1)}$

When X$_i$ = Provider$_i$, P$_i$
$RES_{(k,i)} = CID, h(M_i), k$
         $COM_{((k-1),i)}, ACC_{((k-1),i)}$
$EVRES_{(k,i)} = S_{P_i}[RES_{(k,i)}]$
$Signed_{TK} = S_{TTP}[CID, h(M_i), k, ACC_{((k-1),i)}]$
$Canceled_{TK} = S_{TTP}[CID, h(M_i), k, canceled]$

Table 2.3: Asynchronous Optimistic Atomic Multi-Two Party (AM2P) Resolution Sub-Protocol

finish it. It does not make sense for the consumer to send a cancel request during round $k = 1$ because if she fails to receive the $ACC_{(1,i)}$ from any provider, she will just stop the protocol execution. During round $k = 1$ the providers can only cancel the signature, therefore the consumer cannot lose fairness. In table 2.3 we can see the *resolution* sub-protocol requests and execution flow.

The consumer can contact the TTP to claim she has not received the acceptance corresponding to round $k > 1$, from one or more providers. The *resolution* request message, $\{RES_{(k,i)}, EVRES_{(k,i)}\}$ (see table 2.3), has to include the contract identifier CID, a vector with the hash of the $(N-1)$ contracts $\overline{h(M_i)}_{(N-1)}$ (we do not want the TTP to have access to the contracts' content), the round number k, and the last commitments sent and acceptances received (from each provider i), i.e., $\{\overline{(COM_{(k,i)})}_{(N-1)}, \overline{(ACC_{((k-1),i)})}_{(N-1)}\}$. Along with these values C will send $EVRES_{(k,i)}$, the consumer's digital signature on the whole message $RES_{(k,i)}$, as evidence that C has contacted the TTP. After executing the *resolution* sub-protocol, the consumer will receive either a vector of $(N-1)$ cancel tokens or a vector of $(N-1)$ signed tokens, one for each contract. Therefore, a signed or canceled contract for a consumer will be as follows:

SignedContract_i CID, M_i, k, $COM_{(k,i)}$, $Signed_{TK}$

CanceledContract_i CID, M_i, k, $Canceled_{TK}$

The provider P_i can contact the TTP to claim he has not received the commitment corresponding to round k, $COM_{(k,i)}$. The *resolution* request message, $\{RES_{(k,i)}, EVRES_{(k,i)}\}$ (see table 2.3), has to include the contract identifier CID, the hash of the contract $h(M_i)$, the round number k, and the last commitment received and acceptance sent, i.e., $COM_{((k-1),i)}$ and $ACC_{((k-1),i)}$. Along with these values the provider will send $EVRES_{(k,i)}$, P_i's digital signature on the whole message $RES_{(k,i)}$, as evidence that P_i has contacted the TTP. After executing the *resolution* sub-protocol, a provider will either receive a cancel token or a signed token, therefore a signed or canceled contract for a provider will be as follows:

SignedContract_i CID, M_i, k, $ACC_{((k-1),i)}$, $Signed_{TK}$

CanceledContract_i CID, M_i, k, $Canceled_{TK}$

TTP rules

In order to solve the resolution requests fairly, the TTP has to apply the following rules:

RULE 0

The TTP will only accept one resolution request per participant and CID.

In order to provide timeliness, the participants can contact the TTP anytime (the consumer can contact anytime after round $k = 1$). If they are honest, the response given by the TTP (supposed that this one acts properly) will be maintained whatever

happens with any other participant of the contract signing protocol. So, if a participant decides that he wants to finish (either stop or sign the contract) the protocol run, he has to contact the TTP and stop sending any additional message to the other parties (even if he receives a message from them). This way, knowing that the decision of the TTP guarantees fairness for him, he cannot contact the TTP anymore (it could lead to confusing situations).

RULE 1

If the TTP receives a request from P_i during round $k = 1$, and the execution has not been previously finished (*signed=true*) by other party, the TTP will cancel it and send a canceled token to P_i.

   If an honest party contacts the TTP during the first round ($k = 1$) and nobody has contacted the TTP to finish the exchange previously, the TTP cannot know if all parties are compromised to sign the contract (perhaps $P_{i'}$ does not know that the contract signing protocol has been initiated). But we want an atomic solution, which means, everybody or nobody must have enough evidence of the signed or not signed contract. This way, and with only this information, the TTP has to provide a cancel token related to the signature of this contract. Perhaps this party is not honest, but following rules guarantee that honest parties will not be in an unfair situation (they can contact the TTP in order to solve possible unfair situations).

   On the other hand, it is possible that the protocol run has been finished (without TTP's intervention) and some party cancels with the TTP (this situation will always be possible in an asynchronous optimistic solution: parties can contact anytime with the TTP), but the evidence provided by the TTP will not be useful for him in order to prove the cancelation, because any other party has enough evidence (the signature of the contract provided directly by him) to prove that he is a cheating party.

RULE 2

If the TTP receives a request from X_i during round $k > 1$, and the execution has not been previously canceled by other party, the TTP will finish it (*signed=true*) and send a signed token to X_i.

   If some party (honest or not honest) contacts the TTP after the first round ($k > 1$), it means that this party provides to the TTP enough evidence that all parties know that the contract signing protocol among them has been initiated, and in fact all of them have compromised to sign the contract (all of them have sent, at least, the message of first round). If no party has contacted the TTP previously to cancel the exchange, the TTP has to send a signed token to that party: this evidence will prove that the contract is signed. Perhaps some other party has not received the necessary evidence to prove that the contract is signed (an unfair situation), but they can contact the TTP, and the TTP will apply this same Rule 2 (R2): it will give a signed token to that party. So, fairness is guaranteed in this situation.

RULE 3

If the TTP receives a request from X_i during round $k \geq 1$, and the execution has been

previously finished (*signed=true*) by other party, the TTP will send a signed token to X_i.

This rule is complementary to R1. If some party (honest or not honest) tries to cancel the exchange (it means that he provides evidence related only to first round), but the TTP has finished the exchange with some other party (which means that other party had provided enough evidence that all parties agreed to sign the contract, see R2), the TTP will send a signed token to the requesting party in order to maintain fairness for him and previous contacting parties. If some party lacks any evidence, he has to contact the TTP, with evidence of first round (TTP will apply Rule 3, R3) or a posterior round (TTP will apply R2).

RULE 4

If the TTP receives a request from X_i during round $k > 1$, and the execution has been previously canceled (*canceled=true*) by other party (or parties), the TTP will check the previously received requests to see if they have cheated. If $\forall \{X_i, r\} \in \overline{XA} \rightarrow r < k - 1$ (r = round number of previous requests, k = round number of current request), it means that all the previous requesters have cheated, therefore, the TTP will change the execution state to signed and send the corresponding token to X_i, otherwise the state will remain canceled and the TTP will send the corresponding canceled token.

This rule is related to the abort chaining problem explained in [28, 34] and it makes this rule a little bit more complex than the previous ones. The problem arises when some party, P_i, has canceled the protocol execution (he has contacted the TTP in first position and the TTP applying R1 has given a cancel token to that party), and later some other party, $P_{i'}$ or C (hence for $P_{i'}$), tries to finish the exchange with the TTP. If $P_{i'}$ provides evidence of round $k \geq 3$ ($\{RES_{(\geq 3, i')}, EVRES_{(\geq 3, i')}\}$) to the TTP then it will be obvious for the TTP that P_i has cheated: either the protocol execution was not at round $k = 1$ or he continued with the protocol after contacting with the TTP. But if $P_{i'}$ provides evidence of round $k = 2$, it can be coherent with the petition of P_i. In the next paragraph we present an example to clarify this situation.

Suppose P_i sends a request during round $k = 1$. He will send the commitment and acceptance from round $k = 1$, claiming he has not received the commitment from round $k = 2$: either the message has been lost or the consumer, C, has never sent it. If later $P_{i'}$ sends a request during round $k = 2$, within this request he will include the corresponding commitment, $COM_{(2, i')}$. Therefore, it is clear to the TTP that C started the round $k = 2$, but it cannot prove yet that P_i has misbehaved, the message ($COM_{(2, i)}$) can have been delayed or lost. But if $P_{i'}$ sends a request during round $k = 3$, it means that the consumer sent the commitments from round $k = 3$, and this can only happen if she received all the acceptances from round $k = 2$, including the acceptance from P_i. Therefore the TTP will be able to prove P_i's misbehaviour: either he has continued with the protocol execution after receiving a resolution response from the TTP, or the protocol execution was not at round $k = 1$.

About the need of N rounds

In this section we have stated that our protocol requires N rounds, but we have not explained why N in particular. In the following paragraphs we will prove it.

Theorem 1. For an asynchronous optimistic multi-two party signing protocol (one consumer and $N-1$ providers, where $N > 2$), meeting the timeliness and atomicity requirements, more than $N-1$ rounds are necessary to achieve fairness.

Proof. Assume that $N-1$ rounds are enough. Therefore, the evidence of signature are the $COM_{((N-1),i)}$ and $ACC_{((N-1),i)}$.

Let us imagine that the exchange has been finished for some honest P_i (let's call him $P_{(N-1)}$), and some dishonest party contacts the TTP but continues with the protocol execution, let's say P_1 sends a resolution request pretending to be at round $k = 1$. The TTP, without more information, will cancel the exchange.

Following, a second dishonest party, P_2, contacts the TTP pretending to be at round $k = 2$. This situation is compatible with the previous one, and therefore the TTP has to maintain the cancel state. In particular, the TTP will apply Rule 4.

Next, a third dishonest participant, P_3, contacts the TTP pretending to be at round $k = 3$. This situation is compatible with the previous one, but not with the first one (now it is clear that P_1 is a cheating party: Rule 4) and therefore, to be respectful with P_2 (maintain fairness) the TTP has to maintain the cancel state.

We can follow the same reasoning for P_i. P_i contacts the TTP to finish the exchange, pretending to be at round $k = i$. This situation is compatible with $P_{(i-1)}$ contacting the TTP pretending to be at round $k = (i-1)$, but not with the $P_{(i-2)}$ ($P_{(i-2)}$ is a cheating party) and therefore, to be respectful with $P_{(i-1)}$ the TTP has to maintain the cancel state. And so on, until we reach round $k = (N-2)$. After receiving all the acceptances, the consumer C sends her evidence of signature to each provider, $COM_{((N-1),i)}$, and awaits for the acceptances. But the $COM_{((N-1),(N-2))}$ never reaches its destination, thus $P_{(N-2)}$ contacts the TTP and obtains a cancel token. This kind of attack is known as *abort-chaining* [28, 34].

We have assumed that $P_{(N-1)}$ has evidence of the signed contract ($COM_{((N-1),i)}$), and the TTP has given a cancel token to an honest party $P_{(N-2)}$: he can be at round $k = (N-1)$. Therefore the protocol would not be fair. And this contradiction proves that more than $(N-1)$ rounds are necessary.

2.1.6 Security Analysis

In this section we will review if our protocol proposal (section 2.1.5) accomplishes the requirements we have established for an asynchronous optimistic AM2P contract signing protocol, presented in section 2.1.2: AM2P effectiveness, AM2P weak fairness, AM2P timeliness, AM2P non-repudiation, AM2P verifiability of TTP and AM2P confidentiality.

CLAIM 2.1: **AM2P Effectiveness**: *The TTP only intervenes in case of dispute.*

PROOF. The execution of the *exchange* sub-protocol (table 2.2) assures that, if every party involved behaves correctly (and there are no communication problems), the consumer will receive the signature of the $N-1$ providers, and each provider P_i will receive its corresponding signature from the consumer after N rounds and without TTP intervention. Therefore our protocol meets the effectiveness requirement.

CLAIM 2.2: **AM2P Weak Fairness**: *Upon a finalization of an AM2P protocol execution, either the consumer receives the signature of all providers and each provider receives the corresponding signature from the consumer, or none of the participants receive information that allow him to be at an advantage over the honest ones.*

PROOF. If we assume the consumer is honest, whatever the behaviour of the providers, we have to prove the consumer maintains fairness. There are two possibilities in which a provider P_i (with $1 \le i \le (N-1)$) can obtain proof of the consumer's signature:

- After receiving the consumer's N^{th} commitment $COM_{(N,i)}$, ($1 < i < (N-1)$) which means the consumer has $N-1$ provider's acceptances, thus she can contact the TTP and obtain a token of signature.

- After contacting the TTP, which means the TTP has the value *signed* set to true, therefore the consumer can obtain the signature evidence from the providers or from the TTP, if any provider decides to discontinue the sequence of N rounds.

In all situations, the consumer maintains fairness.

If we assume an honest provider P_i ($1 \le i \le (N-1)$), whatever the behaviour of the consumer, we have to prove that fairness is maintained for the provider. The consumer can obtain proof of the provider's signature by two means:

- After receiving the provider's N^{th} acceptance $ACC_{(N,i)}$, which means the provider already has the consumer's proof of signature $COM_{(N,i)}$.

- Contacting the TTP at round $k > 1$, which means the TTP has the value *signed* set to true, therefore, the provider can obtain the signature proof from the consumer, or from the TTP, if the consumer decides to discontinue the sequence of N rounds.

In all situations, the provider maintains fairness. Therefore our protocol proposal meets the AM2P weak fairness requirement.

CLAIM 2.3: **AM2P Timeliness**: *The duration of a protocol execution will always be finite, without any honest participant losing fairness.*

PROOF. At any time during the protocol execution, any party involved can execute the *resolution* sub-protocol and finish its execution obtaining a canceled or signed token from the TTP. If every party involved behaves correctly, our protocol proposal requires N rounds and $2N(N-1)$ messages, being N a finite and known

number. Therefore we can affirm that the protocol execution will end in a finite amount of time, whether by intervention of the TTP or the normal protocol flow. Moreover, once the protocol execution has ended, its final state cannot be changed. If the protocol ended by intervention of the TTP, this one will ensure coherence within the different requests received (following the TTP rules, section 2.1.5), meanwhile if the exchange ended with the Nth round, the evidence obtained by consumer and providers will prove its ending state. Therefore, we can affirm that our protocol proposal meets the timeliness requirement.

CLAIM 2.4: **AM2P Non-repudiation**: *Neither the consumer nor the providers can deny having participated in a contract signature, or being excluded from it.*

PROOF. In an AM2P signature, evidence of participation of consumer and providers are generated in every round. The $COM_{(k,i)}$ and $ACC_{(k,i)}$ relate the consumer and providers to the protocol execution. In particular, the Nth commitment and acceptance, are considered the signature of the contract itself. If a consumer tries to deny her involvement in the signature of a contract M_i, the provider P_i could prove it using the consumer's signature or the TTP's token of signed. In the same way, the consumer can prove the provider's involvement using his signature or the TTP's token of signed.

CLAIM 2.5: **AM2P Verifiability of TTP**: *If the TTP fails to generate a correct response (due to a misbehaviour or a system failure), the honest participants can prove it.*

PROOF. Assuming an honest consumer and a malicious TTP, the possible situations are described in the following paragraphs.

If the consumer receives a cancel token (a vector with $(N - 1)$ cancel tokens, one for each contract) without the TTP having received any cancel request, as the consumer is honest, she will stop the protocol execution. Therefore the providers will have to contact the TTP. If the TTP sends them a cancel token, fairness will be maintained (all participants will have evidence of cancelation). And if the TTP sends evidence of signature to any provider, the consumer and some providers will have different evidence, but when confronted, the consumer and/or provider will be able to prove the TTP's misbehaviour by presenting their evidence (the cancel token and the signed token contain the round number k) and demanding evidence of the decision taken, i.e., the previous resolution requests.

If the consumer receives a cancel token, but the provider P_i has already obtained a signed token, C and P_i will have contradictory evidence. But when confronted, they will be able to prove that the TTP misbehaved by presenting their corresponding evidence and demanding evidence of the decision taken.

If the consumer receives a signed token during round k, but the provider P_i has already canceled it, C and P_i will have contradictory evidence. Again, when confronted, they will be able to prove that the TTP misbehaved by presenting their corresponding evidence and demanding evidence of the decision taken. In particular, assuming that P_i contacted the TTP during round $k^{'}$ (notice that the honest party is

the consumer, therefore the provider may have misbehaved), we have the following situations:

- $k' \leq k$: There is only one possibility for this combination (C signed token, P_i cancel token) to happen without the TTP being dishonest. Therefore, unless the TTP can present the following evidence (proving P_i's misbehaviour), C will be able to prove its misbehaviour. In order to maintain fairness, if the TTP detects that a participant has misbehaved, it can change the state of the protocol execution. In particular, if $k > k' + 1$ and $k' = 1$, applying Rule 4, the TTP will change the state of the protocol execution from canceled to signed and will issue the corresponding signed evidence to the participant who sent the request.

- $k' > k$: In this situation, whatever evidence the TTP presents, the consumer will be able to prove it has misbehaved. If the TTP received P_i's request first, it should have issued him a signed token (Rule 2), instead of a cancel token. To prove the contrary, the TTP should provide evidence of a previous request sent during round $k'' = 1$, but then it would mean that the evidence issued to the consumer, a signed token, was wrong: it should be a cancel token (Rule 4). And if it was the consumer's request the first to arrive, her evidence would be coherent with the protocol situation (Rule 2). But the cancel token received by P_i would prove the TTP's misbehaviour, he should have received a signed token (Rule 3).

Assuming an honest provider P_i and a malicious TTP, we can have the following situation. The provider sends a resolution request during round k because he has failed to receive the commitment, and the TTP answers with a cancelation (after issuing a signed token or without previous cancelation when $k > 1$) or a finalization (after issuing a cancel token). Both results would be coherent with the TTP rules, but in any case, the provider and the consumer will have contradictory evidence sent from the TTP, which will prove its misbehaviour, as we have seen in the previous paragraphs.

As a conclusion, the protocol proposal meets the TTP's verifiability requirement.

CLAIM 2.6: **AM2P Confidentiality**: *The content of the contract M_i is only available to the consumer and the corresponding provider P_i.*

PROOF. The contract M_i is sent from the consumer C to the provider P_i encrypted with a symmetric key, and this symmetric key is encrypted with P_i's public key. Therefore, only P_i is able to read its content. Moreover, the *resolution* sub-protocol does not include the plain contract M_i, only its hash value, so the TTP does not have access to it. Therefore, we can affirm that only C and P_i have access to the plain contract M_i, and the protocol proposal meets the confidentiality requirement.

2.2 Agent Mediated Scenarios (AMS)

The Atomic Multi-Two Party (AM2P) scenario focus on the consumers, but if we switch the attention from consumers to providers, we find another new scenario, the Agent Mediated Scenario (AMS). Thanks to the development of communications, and in particular of Internet, consumers and providers are closer than ever. Consumers are few clicks away from any service or provider they could need. This ease of communication between consumers and providers could lead us to the conclusion that intermediaries are not needed anymore, however, the truth is that the existence of intermediary agents (intermediaries) in electronic commerce is not unusual, in fact, in some cases is even desirable [17]. These intermediaries act as independent business entities, not as mere transport agents, hence the name of *active intermediaries.*

In this section we present the first solution for fair exchange of values in a scenario with active intermediaries in between the consumer and the providers. The solution meets the core security requirements (section 1.2.1) for optimistic fair exchange protocols (effectiveness, fairness, non-repudiation and timeliness). It also achieves two additional requirements (section 1.2.2), verifiability of the TTP, and one requirement specific for optimistic fair exchange in scenarios with active intermediaries, traceability, that will be further defined. Finally, a performance analysis is sketched, and although it cannot be compared with previous solutions because we have not found any, this analysis suggests that it is a viable scheme.

2.2.1 Motivating Example

Online tourism industry is a good example where complex business transactions happen [17]. Travel agents are business entities that act as intermediaries between consumers and providers of services like transportation or accommodation. Moreover, those transportation or accommodation providers from which travel agencies obtain their offer of services (Tour Operators, Global Distribution Systems GDS,...), may also rely on other providers (Airlines, Hotel chains, primary suppliers like hotels, theme parks, etc.). In figure 2.2 we can see some different entities that can be involved in a transaction, and the relation between them.

The complexity of this transactions is not only due to the existence of intermediaries, it is also because of the bundled nature of the products offered. Tourism products are usually comprised by different kind of services: transportation, accommodation, leisure activities, etc., from different providers. Some of these services may only be available from one specific provider (either the service supplier itself or an intermediary), others may be available from more than one source (different Tour Operators may offer the same Hotel destination), but the travel agencies sell these multi-service products as one. Furthermore these services are dynamic (planes have finite number of seats, hotels have finite capacity, etc.), and have expiration date (events like concerts, flights, etc., have date and time), thus they are bought as they are needed. The next paragraphs will show this complexity with an example.
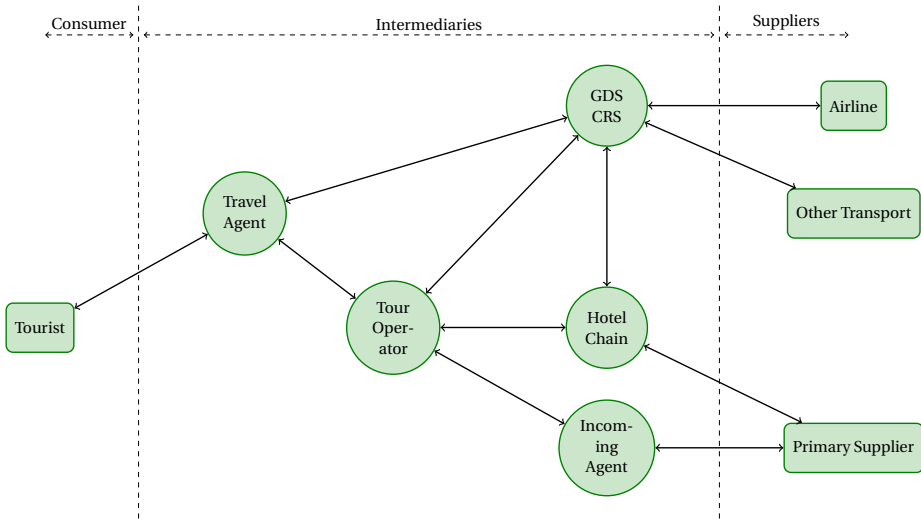
Figure 2.2: An AMS Application Example

Let us suppose Alice enters Bob's online travel agency and adds to her shopping-cart a two-way plane ticket from Barcelona to Milan, two nights in a hotel and two tickets for an opera in La Scala. From Alice's point of view she's buying one product: an opera weekend in Milan, thus she wants all services or none. But from Bob's point of view, Alice is purchasing 3 different services from 3 different providers: transportation, accommodation and leisure activities. When Alice sends the purchase request, a chain of exchanges is generated: {Alice ↔ Bob}, {Bob ↔ plane ticket's provider}, {Bob ↔ Hotel provider}, etc. These Exchanges are executed as independent transactions, leaving the intermediary (Bob) in an unfair situation: if during the execution of the online purchase Alice or any provider decides to quit the transaction, since Bob has executed the exchanges independently, he may be forced to fulfil a service agreement with Alice without having the resources, or a purchase agreement with the provider without having a client.

Bob will process Alice's request and will send requests to the transportation, accommodation and the leisure activities providers. Alice expects two possible responses: Ok (all services are booked) or nOk (one or more of the services are not available), meanwhile Bob may receive two responses from each provider, which leads to 8 different combinations ({Ok,Ok,Ok},..., {nOk, Ok, nOk}, {nOk, nOk, nOk}). The possible results are (ordered by time of arrival):

- (Ok,Ok,Ok): Bob will send an Ok response to Alice.

- (nOk,-,-): Bob will send a nOk response to Alice.

- (Ok,nOk,-) and (Ok,Ok,nOk): Unfair situation, disadvantage for Bob. He does not have confirmation for all services, thus it will answer with a nOk response. But the first and maybe second services are already closed/sold, that means that Bob will have to cancel them and bear the possible costs. Cancelations can bear a cancelation fee.

In a real scenario, travel agencies minimize the risk of cancelation fees by executing the requests sequentially, closing first the services that can be canceled without fee, which may decrease the system performance, and negotiating the "cancelation without fee" policy, which may require human intervention. In such a scenario a fair exchange protocol would eliminate the unfair situations. Moreover, allowing the parallel execution of requests may increase the system performance.

2.2.2 Model and Requirements

To begin with the model description we will define the participating entities: *consumer*, *providers* and *active intermediaries*; and the roles they can adopt: *originator* and *recipient*.

- **Originator**: An agent that initiates an exchange, by sending a request to n_K recipients ($n_K \geq 1$).

- **Recipient**: An agent that receives a request, processes it, and sends the corresponding response to the originator.

- **Consumer**: The entity that initiates a transaction. The consumer entity will act as an originator.

- **Provider**: The entity who hosts the service or product requested by the consumer. The provider entity will act as a recipient.

- **Active Intermediary**: As part of a chain, the active intermediary entity is connected to originators and recipients. To the originators, the active intermediary will act as a recipient, receiving requests. To the recipients, it will act as an originator sending the requests devised from the previous requests received as recipient. The objective of the so-called agent intermediaries is to provide a service to their clients, the originators.

  The main difference between a transport agent and an active intermediary is its objective. The transport agents deliver messages received from an originator to the recipient chosen by the originator. Whereas an active intermediary receives a service request from an originator, and to fulfil it, the active intermediary contacts to one or more recipients that it chooses. To simplify the nomenclature, we will refer to active intermediaries as **intermediaries**.

We will refer to these entities (consumer, intermediaries and providers) as nodes N_i, where i indicates its position within the transaction ($1 < i < N$, N = number of participants). A node N_i can be a consumer, an intermediary or a provider. If we

use N_i to refer to an originator, $N_{(i+1)_k}$ ($1 \leq k \leq n_K$) will be its recipient, with n_K = number of recipients contacted. And if N_i refers to a recipient, $N_{(i-1)}$ will be its originator. As example of the use of this nomenclature, suppose that N_i is N_2 (figure 2.3), then we have $N_{(i+1)_k}$, $1 < k < 2$. As originator, N_2 contacts 2 recipients, N_3 and N_4, therefore $N_{(i+1)_1} = N_3$ and $N_{(i+1)_2} = N_4$. And as a recipient, N_2 is contacted by the originator $N_{(i-1)} = N_1$.

In a transaction there are, at least, three entities involved: a consumer, an intermediary and a provider. A transaction is built from exchanges and an exchange is built on sub-exchanges (see figure 2.3).

- **Transaction**: The entire process in which a consumer and one or more providers, without knowing each other, exchange their parts using intermediaries.

- **Exchange**: A set of processes executed between one originator N_i and n_K recipients $\{N_{(i+1)_1}, ..., N_{(i+1)_{n_K}}\}$, ($n_K \geq 1$) in which originator and recipients exchange their parts.

- **Sub-Exchange**: A process where only one originator N_i and one recipient $N_{(i+1)_k}$ ($k \, \epsilon \, [1, .., n_K]$) exchange their items.

A Transaction starts when a consumer, acting as originator, sends a request to an intermediary, acting as recipient, and ends when the consumer and providers involved have received their desired items. When an intermediary receives a request, it may need to contact n_K different providers/intermediaries in order to acquire the resources needed to fulfil the request. This fulfilment may require the acquisition of all resources or just some of them, depending on the request received by the intermediary. Thus we will define two different situations:

- **Partial Response Scenario (PRS)**: The intermediary has to contact n_K different recipients to obtain n_R different resources ($n_K \leq n_R$, the intermediary may request more than one resource to the same provider), but the request can be considered fulfilled with the obtention of $n_R' \leq n_R$ resources.

  As an example of this scenario, imagine we are planning a holidays trip using an online travel agency. We choose a plane ticket, a hotel reservation, excursions, car rental, etc. Once we have chosen all we want, we proceed to the booking step (the purchase). When planning holidays, we usually have some flexibility on departure and arrival timing and dates, accommodation, etc. Thus if one or two of the products we have requested in the booking step are not available, we may still want to book the others and find an alternative.

- **Full Response Scenario (FRS)**: The intermediary has to contact n_K different recipients to obtain n_R different resources ($n_K \leq n_R$). To fulfil the request, the intermediary has to obtain all n_R resources.

  An example of this scenario could be a request to an online travel agency, where we ask for plane tickets and a hotel reservation for a business trip. When planning business trips, we do not usually have the flexibility we have
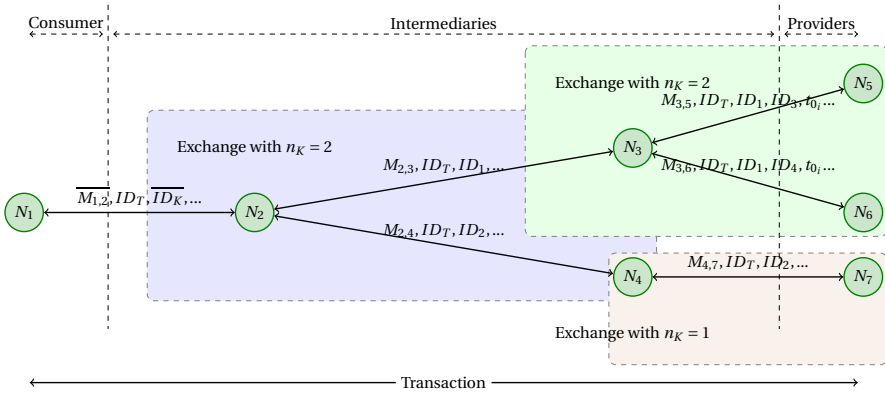
Figure 2.3: Agent Mediated Model

on holidays; we have to choose the flight departure times and dates according to our meeting schedule, even the accommodation can be restricted due to budget or agreements between companies. Therefore, in this case, when we send the booking request, we want to book all products or none.

From a security point of view, the transaction from a consumer (N_1) to a Provider (N_N) is a particular case of Fair Exchange even though they do not have direct contact. N_1 has an element "A" that wants to exchange for an element "B" that N_N has, but none of them is willing to deliver his element until have some kind of assurance that will receive the other expected element, furthermore, none of them knows each other. The only thing N_1 knows is that an intermediary N_2 can deliver "B" to her, either because he has it or because he knows where to find it. And N_N knows that $N_{(N-1)}$ can deliver "A" to him. At the same time, these intermediaries N_2 and $N_{(N-1)}$, may contact other intermediaries to obtain "A" and "B". Thus, we end up creating a chain where the links should be fair exchanges.

Requirements

In section 1.2 we have defined the core and additional requirements for fair exchange protocols. The proposal presented in this section meets all the core requirements (effectiveness, fairness, timeliness and non-repudiation) and two additional ones, verifiability of TTP and confidentiality. These requirements were defined for 2-party scenarios, where we only have one consumer and one provider, without intermediaries. But in our transaction model we can have many intermediaries and many providers, therefore we have to adapt the requirements to this new scenario. What is left of this section is to define the requirements according to our scenario, and to define a new requirement, specific to this scenario, traceability.

Definition 2.7 (*AMS Effectiveness*) *If every party involved in a transaction behaves correctly, the consumer will receive her expected item from the intermediary, the providers will receive their expected items from the intermediaries, and the intermediaries N_i will receive their expected items, as a recipients from $N_{(i-1)}$ and as originators from $N_{(i+1)_k}$, without TTP intervention. This is a requirement for optimistic protocols, where the Trusted Third Party (TTP) only intervenes in case of problems arise.*

Definition 2.8 (*AMS Fairness*) *Asokan et al. [9] defines two different kind of fairness (strong and weak), whereas Zhou et al. [10] presents only one definition, equivalent to the* strong *fairness definition from Asokan et al. [9]. Both fairness definitions where made in a scenario where we have only two participants, A and B. But in our scenario, a transaction has N participants, with $N \geq 3$ (at least 1 consumer, 1 intermediary and 1 provider); an exchange has 1 originator and n_K recipients, $n_K \geq 1$; and a subexchange has only two participants (originator and recipient). Therefore, we need to adapt the fairness definition to our scenario.*

*exchange fairness***:** *Upon finalization of an exchange execution, every honest party involved receives their expected item or none receive any valuable information enabling them to be at an advantage.*

*transaction fairness***:** *After a transaction execution every honest client and provider receives what they expected, either the agreement of his/her intermediary or an affidavit from the TTP, or either none of the participants (including intermediaries) receives any valuable information enabling them to be at an advantage, or all honest participants receive enough evidence to demonstrate who acted honestly and who did not.*

Definition 2.9 (*AMS Timeliness*) *Any party involved in a transaction can be sure that the protocol execution will be finished at a certain finite point of time, whether by contacting the TTP or following the protocol execution.*

Definition 2.10 (*AMS Non-Repudiation*) *In an AMS transaction that has involved a consumer and n_K providers, the consumer will not be able to deny the origin of it, and the providers will not be able to deny its involvement. Moreover, none of the intermediaries participants in the transaction will be able to deny their involvement.*

Besides the core fair exchange requirements, our protocol meets with two additional ones: verifiability of TTP and confidentiality.

Definition 2.11 (*AMS Verifiability of the TTP*) *If the Trusted Third Party (TTP) misbehaves resulting in the loss of fairness for a party, this party can prove the fact in a dispute resolution.*

In fair exchange protocols, confidentiality is usually applied between the participants and the TTP. The TTP must not have access to the plain message or messages that the participants are exchanging, even when resolving disputes. But in our scenario, we also apply confidentiality between participants.

Definition 2.12 (*AMS Confidentiality*) *The node N_i will only have information on the exchanges in which he participates, as a recipient $\{N_{i-1},N_i\}$ and/or originator $\{N_i,N_{i+1}\}$ (consumer and providers do not know each other).*

The rationale behind this requirement is the consideration of the intermediary as a business entity. In the intermediaries business model, the knowledge of which providers or intermediaries are available is an asset, thus, if consumers have access to this information, they may try to skip the intermediary and contact the provider directly. The same can happen with providers: if they know the consumer, they may contact directly and try to make a better offer than the intermediary.

Finally, our protocol meets a requirement specific for optimistic fair exchange protocols in scenarios with active intermediaries, the traceability requirement:

Definition 2.13 (*AMS Traceability*) *In our model, a transaction is a set of exchanges. In case of a problem arises, the TTP may need to be able to reconstruct the transaction in order to solve it, therefore the protocol must allow the TTP to reconstruct a transaction and identify all its participants, using as a source the information retrieved from an exchange.*

In [35] Garay *et al.* introduced a new property, *abuse-freeness*, which states that none of the participants is able to prove in front of an external party that he has the power to terminate (cancel the signature) or successfully complete the protocol (sign the contract). In our case, we have not considered it necessary, in fact, the lack of *abuse-freeness* could be interesting. An intermediary could use the protocol to improve its response time by sending the same request to many providers and accepting the first as response (canceling the others). On the other hand, it can be used to get the best fare, by sending a request to many providers, accepting as a response the one with better price or conditions. Therefore, even though *abuse-freeness* can be a desirable property for some contract signing scenarios, it is not included in this protocol.

Communication Channel

As we did in the AM2P protocol (see section 2.1.4), we will assume that the communication channels between participants, originators and recipients, are executed through unreliable channels (messages can be lost), and the messages exchanged between the TTP and the protocol participants use a resilient channel (messages can be delayed but not lost).

2.2.3 Previous Work

The vast scientific literature about the fair exchange fails to solve the problem of fair exchange in transactions with active intermediaries. The proposed solutions are focused on scenarios with 2 parties involved, or scenarios where N known parties have to exchange some items. However, in real electronic commerce transactions

we may have active intermediaries involved in a transaction, therefore we cannot apply any of the proposed solutions to ensure fairness from consumer to provider. An example where we have active intermediaries involved in business transactions are travel agencies [17]. Travel agencies typically rely on intermediaries to acquire the services that sell *on demand* to their customers, moreover, these external providers can also rely on other external sources to provide services to travel agencies (see example in section 2.2.1).

The use of agents or intermediaries in a transaction model is not entirely new. Franklin and Tsudik [36] proposed a Multi-party Fair Exchange protocol for e-barter, later reviewed by Mukhamedov *et al.* [37]. This proposal assumes that, at some point (after the preliminary phase), the identity of all participants is known, which we do not consider in our model. Moreover, in our model, the lack of knowledge among participants is a requirement (see Confidentiality in section 2.2.2). Another Multi-party Fair Exchange protocol, targeting e-commerce transactions, was proposed by Khill *et al.* [38]. These proposals use a ring architecture, which cannot be applied in our model as consumer and provider do not have direct contact (see example in section 2.2.1).

In [39] Yichun Liu proposes an optimistic Fair Exchange protocol for atomic exchanges between 1 to N parties, but without the presence of intermediaries. In [40] Onieva *et al.* presented a non-repudiation protocol for an AMS, where only one intermediary was involved in a transaction, communicating the consumer and providers; therefore it does not cover the problem outlined in this dissertation (section 2.2.2). We extend this scenario allowing the participation of more than 1 active intermediary and propose a Fair Exchange protocol for scenarios where intermediaries are independent business entities (see a comparison in section 2.2.4).

2.2.4   Protocol for Fair Exchange in Agent Mediated Scenarios

Our proposal uses as a building block the 2-party Fair Optimistic protocol proposed by Ferrer-Gomila *et al.* [41] (FPH2001, see figure 2.4). On the other hand the scenario considered in this dissertation is an extension of that presented by Onieva *et al.* [40], where only an intermediary intervenes and acting as a transport agent. The FPH2001 optimistic protocol is divided in 3 sub-protocols: *exchange*, *cancel* and *finish*. If both participants behave correctly, only the *exchange* sub-protocol will be executed and the Trusted Third Party (TTP) will not intervene.

Protocol for agent-mediated scenarios

The protocol uses a set of parameters and a specific protocol execution flow to provide the TTP with enough evidence to be able to correctly solve the problems that may arise during the execution of a transaction, allow traceability, fulfil the confidentiality requirement, and maintain fairness throughout the execution of a transaction. Next we describe these parameters and the protocol execution flow:

- To individually identify every sub-Exchange, the protocol uses random Universally Unique Identifiers (UUIDs) [42].
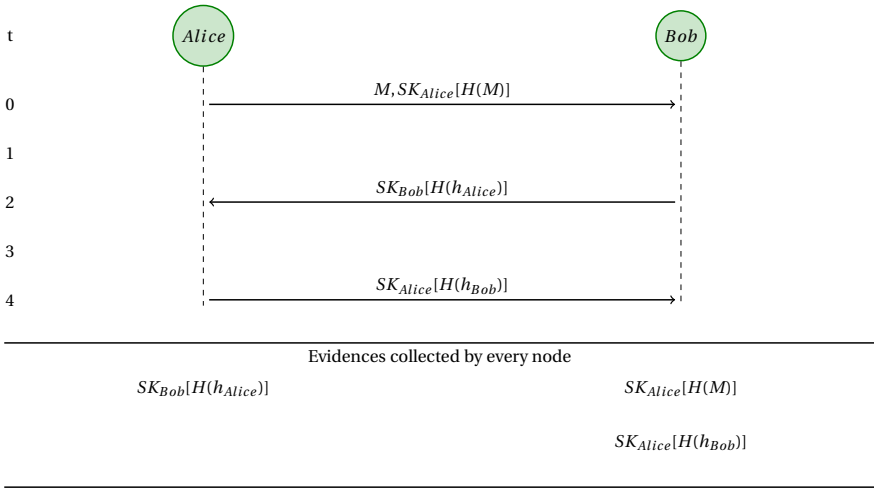
Figure 2.4: FPH2001 Protocol

- To maintain fairness, the protocol introduces a time restriction (t_{0_i}) in atomic 1 to N exchanges, in FRS scenarios.

- To allow traceability and maintain confidentiality, the protocol uses an encrypted parameter Ev_{i_k}, referred as *previous evidence*.

The protocol execution flow is as follows (see example in figure 2.5):

- As a recipient, an intermediary N_i receives a request from an originator $N_{(i-1)}$ (a consumer or other intermediary), and in order to be able to fulfil $N_{(i-1)}$'s requirements he sends a request to up to n_K recipients $N_{(i+1)_k}$ ($k \in [1, ..., n_K]$, providers or other intermediaries). Before answering the originator's request, the intermediary needs to be sure he can fulfil the originator's demands. Thus, the intermediary will wait until he gets the n_K recipients' ACCeptances (evidence of the recipients' agreements), then he will send his ACCeptance to the originator $N_{(i-1)}$.

- As an originator, the intermediary needs to ensure that $N_{(i-1)}$ will carry on with his commitment before sending the ACKnowledgement (evidence of the originator's agreement) to the recipients $N_{(i+1)_k}$. Therefore, the intermediary must wait until he receives the ACKnowledgement from the originator $N_{(i-1)}$, before sending his ACKnowledgements to the recipients $N_{(i+1)_k}$.

In figure 2.5 we have a simple example where we have: one consumer (N_1), two intermediaries (N_2 and N_3), and one provider (N_4). The protocol starts when the consumer, N_1 sends a request to his provider N_2, who is an intermediary. In other
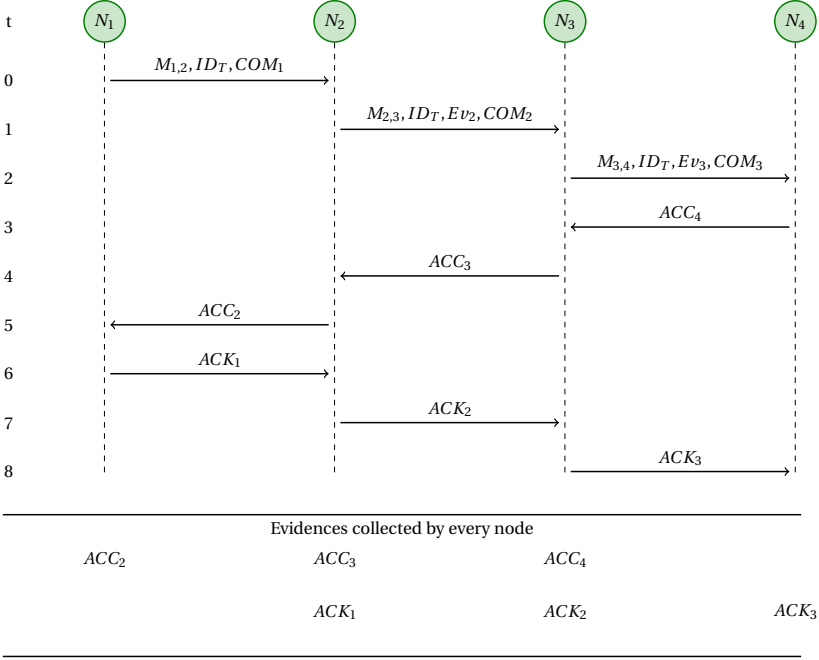
Figure 2.5: Execution Example with N=4

words, the intermediary N_i receives a request from an originator $N_{(i-1)}$, with $i = 2$. Then, the intermediary N_2 sends a request to his recipient N_3, and awaits his answer (ACCeptance) before sending a response to N_1's previous request. Using the nomenclature of our protocol, the intermediary N_i, acting as originator, sends a request to n_K recipients $N_{(i+1)_k}$, with $i = 2$ and $n_K = 1$. But since N_3 is also an intermediary, we can also say that, the intermediary N_i has received a request from an originator $N_{(i-1)}$, with $i = 3$.

In figure 2.3 we can see another example, where an intermediary has to contact more than one recipient. In particular ($1 \le k \le n_K$):

- N_2, acting as originator, sends a request to n_K recipients $N_{(i+1)_k}$ with $n_K = 2$.

- N_3, acting as originator, sends a request to n_K recipients $N_{(i+1)_k}$ with $n_K = 2$.

- N_4, acting as originator, sends a request to n_K recipients $N_{(i+1)_k}$ with $n_K = 1$.

In tables 2.4 and 2.5 we have the notation and the description of the 3 sub-protocols. The general nomenclature used along this section is:

- $\overline{V_{i_K}} = \{V_{i_1}, ..., V_{i_K}\}$ Vector with K elements.

$\mathbf{COM_{i_k}} = \mathbf{S_{N_{i_k}}}[\mathbf{H(M_{(i,(i+1)_k)})}, \overline{\mathbf{ID}}, \mathbf{t_{0_i}}, \mathbf{Ev_{i_k}}]$
Evidence of N_i's COMmitment on $M_{(i,(i+1)_k)}$ (as Originator).
$\mathbf{ACC_{(i+1)_k}} = \mathbf{S_{N_{(i+1)_k}}}[\mathbf{H(M_{(i,(i+1)_k)})}, \overline{\mathbf{ID}}, \mathbf{t_{0_i}}, \mathbf{Ev_{i_k}}]$
Evidence of node $N_{(i+1)_k}$'s ACCeptance on $M_{(i,(i+1)_k)}$ (as Recipient).
$\mathbf{ACK_{i_k}} = \mathbf{S_{N_{i_k}}}[\mathbf{ACC_{(i+1)_k}}]$
Evidence of node N_i's ACKnowledgement of receipt.
$\mathbf{Ev_{i_k}} = \mathbf{E_{TTP}}[\mathbf{H(M_{((i-1),i_k)})}, \overline{\mathbf{ID}}, \mathbf{t_{0_i}}, \mathbf{Ev_{(i-1)_k}}, \mathbf{COM_{(i-1)_k}}, \mathbf{ACC_{i_k}}]$
Chain of evidences of all Exchanges $(j, (j+1)) \mid (j+1) \le i$.
$\mathbf{ACK_{TTP_k}} = \mathbf{S_{TTP}}[\mathbf{ACC_{(i+1)_k}}]$
Recipient's evidence that the sub-exchange k has been finished.
$\mathbf{C_{i_k}} = \mathbf{S_{TTP}}(\mathbf{canceled}, \mathbf{COM_{i_k}})$
Originator's evidence that the sub-exchange k has been canceled.
$\mathbf{C_{(i+1)_k}} = \mathbf{S_{TTP}}(\mathbf{canceled}, \mathbf{ACC_{(i+1)_k}})$
Recipient's evidence that the sub-exchange k has been canceled.
$\mathbf{ECR_i} = \mathbf{S_{N_i}}[\mathbf{H(M_{(i,(i+1)_k)})}, \overline{\mathbf{ID}}, \mathbf{t_{0_i}}, \mathbf{COM_{i_k}}]$
Evidence of Cancel Request
Evidence that N_i has executed the cancel sub-protocol (as Originator).
$\mathbf{EFR_{(i+1)}} = \mathbf{S_{N_{(i+1)}}}[\mathbf{H(M_{(i,(i+1)_k)})}, \overline{\mathbf{ID}}, \mathbf{t_{0_i}}, \mathbf{Ev_{i_k}}, \mathbf{COM_{i_k}}, \mathbf{ACC_{(i+1)_k}}]$
Evidence of Finish Request
Evidence that $N_{(i+1)}$ has executed the finish sub-protocol (as Recipient).

Table 2.4: Notation and Elements Used in the Protocol

- **N** Number of participants.

- **H(M)** One way Hash function of message M.

- $\mathbf{S_z[M]} = \mathbf{SK_z[H(M)]}$ z's Digital Signature on M, where SK_z, is z's private key.

- $\mathbf{M_{(i,(i+1)_k)}}$ Message from the participant N_i to $N_{(i+1)_k}$, with $i \in (1, .., N]$ and $N > 2$ (there is, at least, one intermediary) and $k \in [1, .., n_K]$

- $\mathbf{E_z[M]} = \{\mathbf{PK_z(a)}, \mathbf{Enc_a(M)}\}$ Encryption of message M using the symmetric key a, which is sent encrypted using z's public key PK_z.

- $\mathbf{ID_x}$ Universally Unique Identifier (UUID) [42]. Identifies a transaction ID_T, and each sub-Exchange ID_k in it.

As we have seen in section 2.2.2, we have two different scenarios:

Partial Response Scenario (PRS): In a PRS the originator requests n_R different services (or products), and accepts as a valid response the ACCeptance of $n'_R \le n_R$ services from the recipient. Therefore, the originator will send a request vector, composed of the n_R different services $\overline{M_{n_R}} = \{M_1, ..., M_{n_R}\}$ and the evidence related

Exchange Sub-Protocol	
$N_i \rightarrow N_{(i+1)_k}$	Commitment: $M_{(i,(i+1)_k)}, \overline{ID}, t_{0_i}, Ev_{i_k}, COM_{i_k}$
$N_i \leftarrow N_{(i+1)_k}$	Acceptance: $ACC_{(i+1)_k}$
$N_i \rightarrow N_{(i+1)_k}$	Acknowledgement: ACK_{i_k}

Cancel Sub-Protocol	
$N_i \rightarrow$ TTP	$H(M_{(i,(i+1)_k)}), \overline{ID}, t_{0_i}, COM_{i_k}, ECR_i$
If (finished = true)	
$N_i \leftarrow$ TTP	$ACC_{(i+1)_k}, ACK_{TTP_k}$
Else	
$N_i \leftarrow$ TTP	C_{i_k}; TTP: stores cancelled = true

Finish Sub-Protocol	
$N_{(i+1)_k} \rightarrow$ TTP	$H(M_{(i,(i+1)_k)}), \overline{ID}, t_{0_i}, Ev_{i_k}, COM_{i_k}, ACC_{(i+1)_k}, EFR_{(i+1)}$
If (cancelled = true)	
$N_{(i+1)_k} \leftarrow$ TTP	$C_{(i+1)_k}$
Else	
$N_{(i+1)_k} \leftarrow$ TTP	ACK_{TTP_k}; TTP: stores finished = true

Table 2.5: Agent Mediated Scenarios (AMSs) Protocol

to each service $\overline{COM_{i_{n_R}}} = \{COM_{i_1}, ..., COM_{i_{n_R}}\}$, in order to receive proof of acceptance $\overline{ACC_{(i+1)n'_R}} = \{ACC_{(i+1)_1}, ..., ACC_{(i+1)_{n'_R}}\}$ for each service. In addition, the originator will generate a vector $\overline{ID_{n_R}} = \{\{ID_T, ID_1\}, ..., \{ID_T, ID_{n_R}\}\}$ of IDs that will be assigned to each service, thus the ID of the service r and its related exchanges along the transaction will be $\{ID_T, ID_r\}$ ($1 \le r \le n_R$).

Full Response Scenario (FRS): In a FRS the originator requests n_K different services (or products), and he only accepts as a valid response the acceptance of all n_K services. If the recipient is an intermediary, he will need to contact up to n_K different recipients (intermediaries or providers) to acquire all the resources. When $n_K > 1$, the intermediary will generate a vector $\overline{ID_{n_K}}$ of IDs that will be assigned to each sub-exchange k ($k \in [1, .., n_K]$). In addition to the new IDs the intermediary N_i will introduce another parameter, t_{0_i}, the resolution time (when $n_K > 1$). When the intermediary N_i sends t_{0_i} to the recipient $N_{(i+1)_k}$, he accepts not to enforce the agreement until $t > t_{0_i}$, and the recipient knows he cannot contact the TTP until $t > t_{0_i}$.

Transaction Identifiers

When a consumer initiates a transaction, she assigns a random UUID (ID_T) to it. In addition, originators initiating a PRS request must assign an extra ID_r to each

message r (with $r \in [1,..,n_R]$, number of services/products requested). In the same way, originators receiving an FRS request must assign an extra ID_k to each sub-exchange k (with $1 \le k \le n_K$, number of recipients contacted) they need to execute, so the TTP can identify the different sub-exchanges in order to be able to solve the potential cancel and finish requests. This sequence of IDs along with the hash $H(M_i)$, allows the TTP to determine if a cancel or finish request received is part of an already finished or canceled exchange or sub-exchange.

Note that we used two different indexes: r, n_R and k, n_K; when referring to the services and providers contacted in the PRS and FRS scenarios. This differentiation is done to stress the different way in which transaction identifers are created, depending on the scenario. In the PRS scenario, the number of identifiers created depends on which results are accepted by the originator (minimum number of services), independently of how many providers are contacted. Whereas in the FRS scenario, the number of identifiers depend on the number of providers contacted instead on the minimum number of services accepted. As example of how transaction identifiers are generated, following we will describe the example from figure 2.3:

- The consumer N_1 wants to acquire 3 different services: s_a, s_b and s_c. But she wants either s_a and s_b, or just s_c, or all of them. Therefore she initiates a PRS request, sending a vector comprised of 2 requests. She generates two additional identifiers (besides ID_T): ID_1 and ID_2. The identifier of request one (s_a and s_b) will be $\{ID_T, ID_1\}$ and the identifier of request two (s_c) $\{ID_T, ID_2\}$.

- The intermediary N_2 receives a PRS request vector of size 2, and in order to fulfil it, he contacts two providers N_3 (s_a and s_b) and N_4 (s_c). Both requests are FRS with $n_K = 1$ (independent one from each other), therefore N_2 does not need to generate extra identifiers neither to add t_{0_i}.

- The intermediary N_3 receives a request from N_2 requesting two services, s_a and s_b. In order to fulfill it, N_3 contacts two providers N_5 (s_a) and N_6 (s_b).We are in a FRS situation with $n_K = 2 > 1$. Therefore, N_3 will add a t_{0_i} and will create an extra identifier per request: the request to N_5 will have the identifier $\{ID_T, ID_1, ID_3\}$ and the request to N_6 the identifier $\{ID_T, ID_1, ID_4\}$.

- The intermediary N_4 receives an FRS request from N_2 and in order to fulfil it he contacts one provider, N_7. Since he only needs to contact one provider ($n_K = 1$), he does not need to generate extra identifiers nor to add a resolution time (t_{0_i}).

When the TTP receives a request with a sequence of identifiers $\overline{ID_{ID}}$ and the message hash $H(M_i)$, it checks this sequence with the values stored in its database. The TTP knows that all sequences $\overline{ID_a} \subset \overline{ID_{ID}}$ belong to the same transaction, and are part of a previous exchange, meanwhile all sequences $\overline{ID_b} \supset \overline{ID_{ID}}$ belong to the same transaction and are part of a posterior exchange.

Exchange sub-protocol

The exchange sub-protocol has 3 steps (see table 2.5): *COMmitment*, where the originator sends her commitment to fulfil the agreement she is sending; *ACCeptance*, where the recipient agrees to fulfil the agreement; and *ACKnowledgement*, where the originator sends proof that the agreement has been accepted by both parts. If it is completely executed, without incidences, there is no need to contact the TTP, originator (N_i) and recipients ($\overline{N_{(i+1)_{n_K}}}$) will hold non-repudiation evidence that the agreement has been accepted: the originator will have the recipients ACCeptances ($\overline{ACC_{(i+1)_{n_K}}}$), and each recipient will have the originator's ACKnowledgement (ACK_{i_k}). Even though the agreement may have been redirected to many intermediaries, until it reached its final destination, N_i must not have any knowledge about nodes N_n, with $n > (i+1)$ or $n < (i-1)$, therefore, evidence of previous exchanges (Ev_i) will be encrypted. Following we describe the exchange sub-protocol for each scenario, PRS and FRS

PRS scenario

The Exchange between N_i and $\overline{N_{(i+1)_{n_R}}}$ can be interpreted as n_R independent sub-Exchanges: the originator wants to obtain n_K services/products (resources) and contacts n_K recipients ($n_K \leq n_R$, N_i may request more than one resource to the same recipient).

Commitment:	$\overline{M_{(i,(i+1))_{n_R}}}, \overline{ID_{n_R}}, \overline{Ev_i}, \overline{COM_{i_{n_R}}}$
Acceptance:	$\overline{ACC_{(i+1)_{n_R}'}}$ $(n_R' \leq n_R)$
Acknowledgement:	$\overline{ACK_{i_{n_R}'}}$

FRS scenario

The intermediary N_i must send the *resolution time*, t_{0_i}, to maintain fairness, when $n_K > 1$. In the commitment step N_i will send the resolution time t_{0_i} to each $N_{(i+1)_k}$ ($1 \leq k \leq n_K$, $n_K > 1$). If by $t = t_{0_i}$ N_i has not received the acceptance ACC_{i_k} from all $\overline{N_{(i+1)_{n_K}}}$, she must execute the *cancel* sub-protocol. To execute the *exchange* sub-protocol correctly, the intermediary will also have to follow a specific execution flow: *The intermediary must cancel the exchange if after t_{0_i} he has not received a response from each intermediary/provider contacted.* Otherwise the intermediary may find himself in a disadvantage; any $N_{(i+1)_k}$ will be able to contact the TTP requesting finalization, which would also finish the exchange between $N_{(i-1)}$ and the intermediary N_i, forcing the intermediary to fulfil the $N_{(i-1)}$'s request. It is a situation of disadvantage, the intermediary may find himself in a situation where the $\{N_{(i-1)}, N_i\}$'s exchange has been finished, but he lacks all the necessary resources to fulfil it.

Commitment:	$M_{(i,(i+1)_k)}, \{ID_T, ID_k^*\}, t_{0_i}^*, Ev_i, COM_{i_k}$
Acceptance:	$ACC_{(i+1)_k}$
Acknowledgement:	ACK_{i_k}
* ID_k and t_{0_i} are only mandatory when $n_K > 1$	

Cancel sub-protocol

If an originator node N_i claims she has not received the evidence $(ACC_{(i+1)_k})$ from the recipient $N_{(i+1)_k}$ ($1 \leq k \leq n_K$), she may initiate the cancel sub-protocol, sending N_i's evidence of commitment. The TTP will check the status of the sub-Exchange and will send the corresponding response:

- If the sub-Exchange has been finished, the TTP will send to N_i the evidence of finalization: $ACC_{(i+1)_k}$, ACK_{TTP_k}.

- If no other node N_j with $j > (i+1)$ has contacted the TTP or the sub-Exchange has been canceled, the TTP will send the cancelation evidence C_{i_k}.

Finish sub-protocol

If a recipient $N_{(i+1)_k}$ claims he has not received the ACKnowledgement from the originator, he may initiate the finish sub-protocol sending N_i's evidence of commitment, $N_{(i+1)_k}$'s evidence of signature, and Ev_i evidence for all exchanges $M_{(j,(j+1))}$ with $(j+1) \leq i$ (note that if N_2 is initiating the finish sub-protocol, he does not need to send Ev_i, since there are no previous exchanges). $N_{(i+1)_k}$ can contact the TTP and request the finalization of the sub-exchange $\{M_{(i,(i+1)_k)}, ID_T, ID_k\}$ at any time (assuming he has received N_i's commitment). The TTP may respond in 2 different ways:

- If the sub-exchange has been canceled, the TTP will send the corresponding evidence of cancelation: $C_{(i+1)_k}$.

- If no other node N_j with $j < i$ has contacted the TTP previously, the TTP will send to $N_{(i+1)_k}$ the finalization evidence: ACK_{TTP_k}.

TTP resolution

In our proposal, even though not all parties involved in a protocol execution know each other, all of them must trust the same third party, the TTP. In this dissertation we will assume that all parties agree to use the same TTP, e.g., the consumer and first intermediary agree on using a TTP during the negotiation phase and as the transaction is being executed, all parties involved agree on using it, otherwise they reject the transaction (the TTP's identity may be included in the contract). Moreover, we assume that the TTP will always send a valid response to any request received. Even though it may seem a strong assumption, when we think on real scenarios it is feasible. By definition a TTP is an external entity on which users rely to solve their disputes, if the TTP begins to fail to answer the requests received, it will result in an immediate loss of trust.
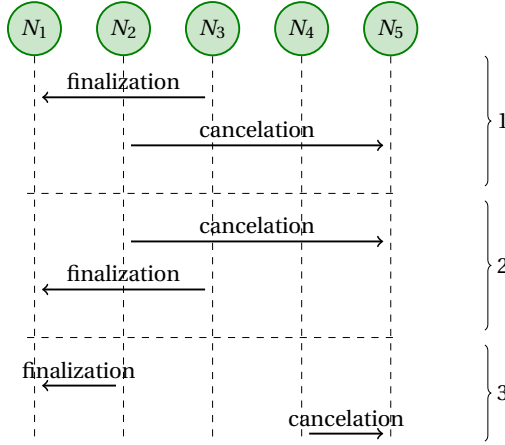
Figure 2.6: Cancelation and Finalization Propagation

Achieving timeliness in an asynchronous protocol may require a stateful TTP, i.e., the TTP needs to keep information on the transaction for an undefined amount of time. In a real scenario, the TTP may be compelled to keep the evidence acquired due to the legal framework related to the application of the protocol (even in synchronous scenarios).

In our model, a transaction from node N_1 to node N_N generates a chain of requests/responses. The optimistic fair exchange protocol proposed generates a chain of 2-party fair exchanges following the requests, propagating from node N_i to $N_{(i+1)}$. The whole chain of fair exchanges is linked, from N_1 to N_N, meaning that the cancelation or finalization of an exchange by a node N_i will have effects on the other exchanges (see figure 2.6).

The propagation of a cancelation or finalization is done at a TTP level. This means that the TTP knows that a chain of exchanges identified by its ID vector $\{ID_T, ID_{K_1}, ..., ID_{K_K}\}$ has been canceled or finalized, but it does not contact the nodes involved. The TTP plays a passive role, the nodes involved in transactions and exchanges are the ones who contact the TTP when needed. If node N_i cancels an exchange with $N_{(i+1)}$, she will not send the ACK_i finishing the exchange sub-protocol. Thus, $N_{(i+1)}$ will be forced to execute the finish sub-protocol to know whether N_i has canceled the exchange or the ACK_i message has been delayed or lost.

The cancelation of an exchange by node N_i will affect the nodes N_j with $j \geq i$ (see figure 2.6). When a node N_i executes the cancel sub-protocol on a sub-Exchange with a set of identifiers $\overline{ID_i}$, all the exchanges with $j \geq i$ and identifier set $\overline{ID_j} \mid \overline{ID_j} \supseteq \overline{ID_i}$ will be canceled. Thus, if a node N_j executes the finish sub-protocol on a sub-Exchange with $j \geq i$ and identifier $\overline{ID_j} \supseteq \overline{ID_i}$, the TTP will answer

that the exchange has already been canceled. But if the node N_j executes the finish sub-protocol on a sub-Exchange with $j < i$, or $\overline{ID_j} \not\supseteq \overline{ID_i}$ the TTP will answer with the corresponding ACK_{TTP}, meaning that the exchange has been finished.

The finalization of an exchange by node N_i will affect the nodes N_j with $j \leq i$ (see figure 2.6). This means that if a node N_i executes the finish sub-protocol on the sub-Exchange with a set of identifiers $\overline{ID_i}$, all the exchanges with $j \leq i$ and identifier $\overline{ID_j} \mid \overline{ID_j} \subseteq \overline{ID_i}$ will be finished. Therefore, if a node N_j tries to execute the cancel sub-protocol on a sub-Exchange with $j \leq i$ and identifier $\overline{ID_j} \mid \overline{ID_j} \subseteq \overline{ID_i}$, the TTP will answer with a proof of finalization. But if the node N_j executes the cancel sub-protocol on a sub-Exchange with $j > i$, or $\overline{ID_j} \not\subseteq \overline{ID_i}$ the TTP will cancel the exchange and will send the corresponding cancelation evidence.

In figure 2.6 we have three simple examples of TTP intervention and cancelation/finalization propagation:

1. In the first example, N_3 executes the finish sub-protocol. Later, N_2 executes the cancel sub-protocol. Due to the propagation, we can see how the exchange $M_{(2,3)}$ becomes affected by both sub-protocols. But, since N_3 has contacted the TTP first, requesting finalization, when N_2 executes the cancel sub-protocol, the TTP will detect that the exchange $M_{(2,3)}$ has been finished and will answer with the corresponding finalization evidence.

2. In the second example, we have the opposite case, the first to contact the TTP is N_2 requesting cancelation. After that, N_3 contacts the TTP to execute the finish sub-protocol. In this situation, the TTP will detect that the exchange $M_{(2,3)}$ has been canceled, therefore it will send the corresponding cancelation evidence to N_3.

3. In the third example, N_2 executes the finish sub-protocol on $M_{(1,2)}$ and N_4 the cancel sub-protocol on $M_{(4,5)}$. Since there is no intersection in the propagation, the order in which the sub-protocols are executed is irrelevant. Both nodes will receive their corresponding evidence from the TTP: N_2 will have its finalization evidence and N_4 will receive the cancelation evidence. As a result of these executions, the exchange $\{N_1, N_2\}$ will be finished, while the exchange $\{N_4, N_5\}$ will be canceled.

Comparison with previous solutions

In section 2.2.3 we have explained the lack of fair exchange solutions for scenarios with active intermediaries, reviewing the existent proposals. However, one of these proposals can be compared with our solution when we have only one intermediary and we are in a PRS scenario: [40] Onieva *et al.*'s proposal.

These authors present a non-repudiation protocol with online TTP for scenarios with one intermediary agent. It is a synchronous protocol, meeting the fairness, non-repudiation, timeliness, and confidentiality requirements, but not addressing the verifiability of TTP. Our solution for PRS scenarios is asynchronous, optimistic

and meets the verifiability of TTP requirement. In addition, our proposal allows originators to cancel and recipients to finish a transaction, without losing fairness.

Moreover, the objective of their proposal differs from ours: they focus on delivering messages from an originator to a set of recipients, meanwhile we intend to sign contracts between a consumer and the intermediary, and between the intermediary and many providers. In practice, in Onieva *et al.*'s proposal, when the recipient generates the non-repudiation of receipt evidence he is acknowledging the reception of the message encrypted. But he did not have access to its content yet, therefore we cannot consider it a contract signature. And we can apply the same reasoning with the intermediary. On the other hand, in our protocol proposal, the consumer signs a contract with the intermediary, and the intermediary signs a contract with the provider. Therefore, even though both proposals present some similarities, Onieva *et al.*'s proposal cannot be applied to solve the problem outlined in this section.

2.2.5 Security Analysis

In this section we will review if our protocol proposal for AMS (section 2.2) meets the requirements we have presented in section 2.2.2: effectiveness, fairness, timeliness, non-repudiation, verifiability of the TTP, confidentiality and traceability.

CLAIM 2.7: **AMS Effectiveness**: *If every party involved behaves correctly, they will receive their expected items without participation of the TTP.*

PROOF. In the case of our protocol, it means that originators N_i will receive the acceptances $ACC_{(i+1)_k}$ and recipients $N_{(i+1)_k}$ will receive the corresponding acknowledgement ACK_{i_k}. The execution of the exchange sub-protocol (table 2.5) ensures the reception of the expected evidence, therefore we can assure that the protocol meets the effectiveness requirement at a sub-exchange (or exchange with $n_K=1$) level where only one originator and one recipient are involved.

In exchanges with $n_K > 1$ (PRS and FRS situations) if every party involved behaves correctly, and follows the protocol execution flow (section 2.5), the exchange sub-protocol, again, assures the reception of the expected evidence without TTP intervention. Thus, the protocol meets the effectiveness requirement.

CLAIM 2.8: **AMS Fairness**: *After an AMS protocol execution, the consumer, providers and intermediaries receive the corresponding signatures from the other participants, or none of them receives any information to allow him to be at an advantage over an honest participant.*

PROOF. To illustrate that our protocol proposal meets the fairness requirement, in the following paragraphs we explain the possible outcomes of the protocol, according to each participant's behaviour, and for each scenario (PRS and FRS, when necessary we will particularly refer to one scenario). Each part is devoted to one of the participant's behaviour, and the adjacent nodes are described as neighbours:

- To the nodes N_j with $j < i$ within N_i's branch, we will refer as previous participant.

- To the nodes N_j with $j > i$ within N_i's branch, we will refer as posterior participant.

- To the nodes $N_{(i-1)}$ and $N_{(i+1)}$ we will refer as neighbours of N_i.

CONSUMER

Before sending her *COM*, the consumer can cancel the transaction, but it is a nonsense because she is not achieving any advantage.

After sending her *COM*, and before receiving the *ACC*, the following can happen:

- If the consumer does not receive the *ACC* and does not contact the TTP, she is assuming the risk that some posterior participant finishes the exchange, leaving her in an unfair situation (she does not have proof of the ending state, her fault).

- If the consumer contacts the TTP (to cancel the exchange, she cannot finish) and nobody has contacted the TTP before her, the TTP will cancel the exchange and she will be in a fair condition: she has no *ACC* and the exchange is canceled.

- If the consumer contacts the TTP but somebody has finished the exchange (her neighbour or some posterior participant), the TTP will give her an affidavit, ACK_{TTP}, and again she is in a fair condition.

After receiving her neighbour's *ACC* and before sending her *ACK*, she has the evidence that this particular exchange has been finished. Now, if she is honest she should not contact the TTP, but she can try to cheat:

- If somebody has finished the exchange (her neighbour or some posterior participant), the TTP will give her an affidavit, ACK_{TTP}. She has not achieved an unfair condition for her neighbour: the affidavit is equivalent to the *ACC* that she already had.

- If nobody has contacted the TTP before her, the TTP will cancel the exchange and initially it seems that she has cheated: she has an *ACC* and the exchange is canceled (she has a double evidence). If she does not send the *ACK*, her neighbour will contact the TTP to finish the exchange and he will receive a cancel message from the TTP, therefore both honest participants are in a fair situation. If, later, the consumer misbehaves and tries to use the *ACC*, her neighbour can use the cancel token received from the TTP to prove it.

After sending her *ACK*, it is a nonsense to contact the TTP (no one except the consumer can cancel her exchange). The TTP will act as in the previous case, but in

this case it can be even worse for her because she has sent her ACK, and if the recipient has received it, this will clearly prove that she was cheating when she contacted the TTP.

Notice that these cases can be applied to both PRS and FRS (section 2.2.2). When a consumer initiates a transaction she only contacts with one intermediary, this initial transaction determines if we work in a PRS or in a FRS scenario. In a FRS scenario the consumer will send one COM to his intermediary, meanwhile in the PRS she will send a vector of COMs.

INTERMEDIARY

Before receiving a COM from a neighbour (the consumer or another intermediary), the intermediary can do nothing; he is not even aware that a signature protocol is being executed.

After receiving a COM from a neighbour (and before sending his COMs), the intermediary can do the following:

- Stop, assuming no risk (his neighbour can only cancel the exchange with him), and the intermediary has not sent any compromising evidence.

- If the intermediary contacts the TTP (to finish the exchange with his previous neighbour) and nobody has previously contacted the TTP, he will receive a finish token from the TTP. If afterwards a previous party in his branch contacts the TTP, he will receive a finish token. Observe that the intermediary will remain in an unfair situation (his fault): the previous parties have a finish token (and so do him with them), but he has no agreement with posterior parties.

- If the intermediary contacts the TTP, but some previous party (in his branches) has canceled his exchange, the TTP will send a cancel token for him.

After receiving a COM from a neighbour, sending his COMs, and before receiving the corresponding ACCs, the intermediary can do the following:

- If the intermediary does not receive the ACCs from his neighbours (recipients), neither contacts the TTP, he is assuming the risk that some posterior party had finished the exchange, leaving him in an unfair situation: he does not have proof of the ending state, his fault. Moreover, in a FRS scenario the intermediary may have lost atomicity.

- In a FRS scenario, if the intermediary contacts the TTP before t_{0_i} to cancel the exchange with his posterior neighbours, the TTP will cancel the exchange and send him a cancel token. Notice that it is not possible that any party had finished the exchange; posterior parties can only contact the TTP after t_{0_i}.

- If the intermediary contacts the TTP to cancel the exchange with posterior participants (PRS or FRS with $t > t_{0_i}$) we can have the following situations:

– no posterior party has contacted the TTP and finished the exchange, the TTP will cancel it. In a PRS scenario the intermediary will be in a fair condition: he has not sent his *ACC* and the exchange is canceled. Previous participants cannot finish the exchange with him, neither they do have his *ACC*. But in a FRS scenario the intermediary will need to cancel the exchanges with all his neighbours to maintain fairness, and since $t > t_{0_i}$, any of them may have finished his exchange, therefore he may be in an unfair situation (his fault).

– if some posterior party has finished the exchange, the TTP will give him an affidavit, ACK_{TTP}. If we are in a PRS scenario, the intermediary will be in a fair situation. But in a FRS scenario the intermediary will need to finish the exchanges with all his neighbours to maintain fairness, and since $t > t_{0_i}$, any of them may have canceled his exchange (by discontinuing the protocol), therefore he may be in an unfair situation (his fault).

– if some previous party has canceled the exchange, the TTP will send him a cancel token, and again he is in a fair situation. The cancelation of $\{N_{(i-1)}, N_i\}$ will cancel all $\{N_j, N_{(j+1)}\}$ ($j \geq i$) within the same transaction.

• The intermediary can contact the TTP to finish the exchange (with his previous neighbour):

– if some previous party (in his branches) has canceled the exchange, the TTP will send him a cancel token.

– if some posterior party (in his branches) has finished the exchange, the TTP will send him a finish token. In a PRS scenario the intermediary will be in a fair condition. But in a FRS scenario he may be in an unfair situation (his fault), he does not have evidence of all his posterior neighbour's signature (he has not received their *ACC*s), and $t > t_{0_i}$ (otherwise the posterior parties cannot finish), therefore it is possible that some posterior party had decided not to sign his contract (he discontinues the protocol execution).

– if nobody has contacted the TTP previously, the TTP will send a finish token to the intermediary. If afterwards a previous party in his branch contacts the TTP, she will receive a finish token. Observe that the intermediary can remain in a unfair situation (his fault): the previous parties can obtain evidence of finalization with him, but he has not necessarily an agreement with posterior parties.

After receiving the *ACC*s from his neighbours (posterior) and before sending his *ACC* (this means that he has proof with posterior neighbours but not with his previous one), the intermediary can do the following:

• If the intermediary does not receive the *ACK* from his neighbour (previous), neither contacts the TTP, he is assuming the risk that some posterior party

finishes the exchange (in a FRS scenario $t > t_{0_i}$), leaving him in an unfair situation: he does not have evidence of the ending state regarding his previous neighbour, but it is his fault.

- The intermediary can contact the TTP to cancel the exchange (with the posterior neighbours):

    - if a posterior participant has contacted the TTP and finished the exchange, the TTP will send an affidavit, ACK_{TTP}, to the intermediary as evidence of contract signed. In a FRS scenario this situation is possible only if $t > t_{0_i}$. In both scenarios, PRS and FRS, the intermediary is in a fair situation (he has all his neighbour's ACCs).

    - if some previous party has canceled the exchange, the TTP will send him a cancel token, and again the situation is fair. The intermediary does not have the ACK but the exchange is canceled.

    - if nobody has contacted the TTP before him (PRS and FRS), or $t < t_{0_i}$ in a FRS scenario, the TTP will cancel the exchange and the intermediary may have cheated. He has the ACCs and the exchange is canceled. The correct course of action after receiving the ACCs would be to send the corresponding ACC, but the intermediary may decide to dismiss the ACC and request cancelation. As long as he does not use the ACCs, he will remain honest, and in a fair situation; and whatever his decision, his posterior neighbours will be in a fair situation, they can obtain evidence of the intermediary's cancelation by contacting the TTP. The intermediary cannot use the ACCs and the cancel tokens. If posterior parties do not try to finish it would be their fault. So the situation is fair. This situation is equivalent to the fact that the ACC has been lost or not sent.

- The intermediary can contact the TTP to finish the exchange (with the previous parties):

    - if some previous party (in his branches) has canceled the exchange, the TTP will send a cancel message for him.

    - if some posterior party (in his branches) has finished the exchange, the TTP will send a finish message for him. Again, this is only possible if $t > t_{0_i}$ if we are in a FRS scenario, but in this case we maintain fairness (the intermediary already has all his neighbour's ACCs).

    - if nobody has contacted previously with the TTP, this one will send a finish message to the intermediary. If afterwards a previous party in his branch contacts the TTP, he will receive a finish message. Now the exchange is completed for intermediary, he has the finish token regarding his previous neighbour, and the ACCs from posterior neighbours.

After his neighbour (previous) receives his ACC and before he receives his neighbour's (posterior) ACK (this means that he has evidence with posterior neighbours

but not with previous neighbours), the outcome is analogous to the previous case, except for one situation:

- The intermediary contacts the TTP to cancel the exchange (with the posterior neighbours). If nobody has contacted the TTP before him (or $t < t_{0_i}$ in a FRS scenario), the TTP will cancel the exchange and the intermediary may have cheated: he has the ACCs and the exchange is canceled. The correct course of action after sending the ACC would be to wait for the corresponding ACK or requesting finalization, but the intermediary can decide to dismiss the ACC and request cancelation. This will leave him in an unfair situation (his fault), he has compromised himself with his previous neighbour by sending his ACC, and he has canceled the agreement with the posterior neighbours.

After receiving the ACK from his neighbour (previous), and before sending his ACKs (this means that he has proof with posterior neighbours and with his previous neighbour), the intermediary can do the following:

- The natural situation is that the intermediary does not contact the TTP: he has all the necessary evidence (from previous and posterior neighbours).

- The intermediary can contact the TTP to cancel the exchange (with the posterior neighbours):

  - if some posterior party has finished the exchange, the TTP will give him an affidavit, ACK_{TTP}, and again he has tried to cheat but he has not achieved it. In a FRS scenario this situation is possible only if $t > t_{0_i}$.

  - if nobody has contacted the TTP before him (or $t < t_{0_i}$ in a FRS scenario), the TTP will cancel the exchange and the intermediary may have cheated: he has the ACK from the previous neighbour, the ACCs from the posterior neighbours, but the exchange is canceled. The correct course of action after receiving the ACK would be to send the corresponding ACKs, but the intermediary can decide to dismiss the ACCs and request cancelation. This will leave him in an unfair situation (his fault), he has compromised himself with his previous neighbours by sending the ACC, and he has canceled the agreement with the posterior neighbours. If any posterior party contacts the TTP, they will receive evidence of cancelation.

- The intermediary can contact the TTP to finish the exchange, which is a nonsense because he has the ACK (the exchange is finished between him and his previous neighbour):

  - if some previous party (in his branches) has canceled the exchange, the TTP will send a cancel message for him. In a dispute resolution it would be clear that a previous party has cheated: he has an ACK and a cancel token.

- if some posterior party (in his branches) has finished the exchange, the TTP will send a finish message for him (nothing new for him). In a FRS scenario this situation is possible only if $t > t_{0_i}$.

- if nobody has contacted previously with the TTP, he will receive a signed token (the same that he has). If afterwards a previous party in his branch contacts the TTP, she will receive a signed token. The exchange is completed for the intermediary, he has the evidence of signature from both previous and posterior neighbours.

After his neighbours receive his ACKs (it is a nonsense to contact the TTP: previous and posterior neighbours have their evidence), the outcome is analogous to the previous case, except for one situation:

- The intermediary can contact the TTP to cancel the exchange. If nobody has previously contacted the TTP (or $t < t_{0_i}$ in a FRS scenario), the intermediary will receive a cancel token. It is a clear situation where the intermediary is cheating, he has evidence of signature with his previous neighbour, and both proof of cancel and finish with posterior neighbours. The posterior neighbours do not need to contact the TTP, because they have the intermediary's evidence of signature, thus, if the intermediary tries to use the cancel evidence, they can proof he is cheating by using his ACK. And if a posterior party contacts the TTP, he will receive a cancel message from the TTP, but the intermediary's ACK will prove he cheated.

PROVIDER

Before receiving a COM from his neighbour, the provider can do nothing, he is not even aware of the transaction.

After receiving a COM from his neighbour, and before sending his ACC, the provider can do the following:

- He can stop, assuming no risk. His neighbour can only cancel the exchange with him, and the provider has not sent any compromising evidence.

- He can contact the TTP (to finish the exchange) and if nobody has previously contacted, the TTP will send a finish token to the provider. If afterwards a previous party in his branch contacts the TTP, this party will receive a finish token.

- He can contact the TTP, but if some previous participant (within his branch) has canceled the exchange, the TTP will send a cancel token to him.

After sending his ACC and before receiving his neighbour's ACK, the following may occur:

- If the provider does not receive the ACK neither contacts the TTP, he is assuming a risk: his neighbour has evidence of his signature but he does not have evidence of the ending state. He is in an unfair situation (his fault).

- He can contact the TTP to finish the exchange and if nobody has contacted previously with the TTP, this one will send a finish token to the provider. If afterwards a previous party in his branch contacts the TTP, he will receive a finish token. Now the exchange is completed for the provider: he has the finish token of his neighbour.

- If he contacts the TTP but some previous party (in his branches) has canceled the exchange, the TTP will send a cancel token for him.

After receiving the *ACK* from his neighbour, these are the possible outcomes:

- The natural situation is that the provider does not contact the TTP: he has all the necessary evidence (from his neighbour).

- The provider can contact the TTP to finish the exchange, which is a nonsense because he has the *ACK* (the exchange is finished between him and his neighbour):

  - if some previous party (in his branches) has canceled the exchange, the TTP will send a cancel message for him. In a dispute resolution it would be clear that a previous party has cheated (he has an *ACK*).
  - if nobody has contacted previously with the TTP, this one will send a finish token to the provider. If afterwards a previous party in his branch contacts the TTP, she will receive a finish message. The exchange is completed for the provider: he has his neighbour's *ACK*.

Notice that these cases can be applied to both PRS and FRS (section 2.2.2). The only difference between the two scenarios is that in a FRS the provider cannot contact the TTP until $t > t_{0_i}$.

Evidence Forgery to Break Fairness

The TTP relies on the evidence sent by the parties involved in a transaction to resolve the problems arisen. During the execution of a transaction, originators N_i have to send the evidence of the previous exchanges, Ev_{i_k}, to the recipients $N_{(i+1)_k}$. These evidence are encrypted using the TTP's public key (see section 2.2.4), thus, recipients cannot verify its value. Therefore, a malicious originator can take advantage of that situation and send forged or incorrect evidence, trying to break fairness.

When the TTP receives a finish or cancel request, the first thing it does is to validate the evidence Ev using its signatures. Then it has to ensure that all the evidence received are part of the same transaction, checking the identifiers received (see section 2.2.4). If the TTP finds that the IDs of the evidence received do not match, it has to find out who has sent the wrong evidence. Since all evidence messages are digitally signed by their generators, the TTP will be able to identify them. The difficult task is to tell who among all the parties involved is cheating and who is not. We will show it with an example.

Figure 2.7 depicts a transaction configuration with 1 consumer, 4 intermediaries and 4 providers. Suppose the consumer sends a FRS commitment requesting 4 different resources to the intermediary I_1.

Client $\longrightarrow$ I₁

$M_{(client,I_1)}, \{ID_T\}, COM_{client}$

The intermediary I_1 decides he needs to contact 2 different intermediaries/providers (he does not know, necessarily, if he's contacting an intermediary or a final provider).

I₁ $\longrightarrow$ I₂

$M_{(I_1,I_2)}, \{ID_T, ID_a\}, t_{0_{I_1}}, Ev_{I_1}, COM_{(I_1,I_2)}$

I₁ $\longrightarrow$ I₃

$M_{(I_1,I_3)}, \{ID_T, ID_b\}, t_{0_{I_1}}, Ev_{I_1}, COM_{(I_1,I_3)}$

$$Ev_{I_1} = E_{TTP}[H(M_{client,I_1}), \{ID_T\}, COM_{client}, ACC_{I_1}]$$

At this point, a malicious Intermediary I_3 decides to forge the evidence he sends to his intermediaries/providers ($t_{0_{I_3}} \leq t_{0_{I_1}}$).

I₃ $\longrightarrow$ P₂

$M_{(I_3,P_2)}, \{ID_T, ID_b, ID_c\}, t_{0_{I_3}}, Ev_{I_3}^*, COM_{(I_3,P_2)}$

I₃ $\longrightarrow$ I₄

$M_{(I_3,I_4)}, \{ID_T, ID_b, ID_d\}, t_{0_{I_3}}, Ev_{I_3}^*, COM_{(I_3,I_4)}$

I₃ $\longrightarrow$ P₄

$M_{(I_3,P_4)}, \{ID_T, ID_b, ID_e\}, t_{0_{I_3}}, Ev_{I_3}^*, COM_{(I_3,P_4)}$

$$Ev_{I_3}^* = E_{TTP}^*[H(M_{I_1,I_3}), \{ID_T^*, ID_b^*\}, t_{0_{I_3}}^*, Ev_{I_1}^*, COM_{(I_1,I_3)}^*, ACC_{I_3}^*]$$

There are 3 ways in which I_3 can forge the evidence:

1. Sending a random value: none of the recipients contacted by I_3 will be able to detect the forgery, but if any of them contact the TTP, this one will detect it (the TTP will not be able to decrypt $Ev_{I_3}^*$). To determine who is responsible for the forgery, the TTP will contact I_3 and the entities involved. Those entities will be able to provide proof that they received Ev_i from the intermediary I_i using the evidence received COM_i, therefore, the TTP will conclude that I_3 generated the forgery.

2. Using previous values from other transactions: I_3 cannot create the $COM_{(I_1,I_3)}$ value, since he does not have I_1's private key. But he can use a COM value from a previous transaction (COM'). Now I_3 has 2 possibilities, either use only the COM' or use the COM' and its associated values ($H(M)'$, $\overline{ID'}$, t'_{0_i}, Ev'). In the first case, the TTP will detect the forgery because it will not be able to validate the signature. In the second situation, the TTP will be able to validate the signature, but the ID sequence will be different than the rest (see section 2.2.4). The problem to detect who introduced the forged values can be resolved as in the previous situation.

3. Using previous values from the same transaction: if I_3 has access to the message sent between I_1 and I_2, he can use the information to forge the $Ev_{I_3}^*$. If I_3 copies the values from the message sent from I_1 to I_2, the TTP will be able to validate the signature, but as in the previous case the IDs sequence will not match (see section 2.2.4). The problem to detect who introduced the forged values can be resolved as in the first situation.

CLAIM 2.9: **AMS Timeliness**: *Each protocol execution has a finite duration, and once it finishes, all honest participants maintain the level of fairness acquired.*

PROOF. When only one originator and one recipient are involved (sub-exchange or exchange with $n_K = 1$), the originator is able to cancel the transaction any time after sending the COMmitment (executing the cancel sub-protocol), or to finish it after receiving the ACCeptance (sending the ACKnowledgement message). The recipient is able to finish the transaction (executing the finish sub-protocol) or to cancel it (not sending the ACCeptance message nor contacting the TTP) at any time after receiving the originator's COMmitment. Therefore the timeliness requirement is met at sub-exchange or exchange with $n_K = 1$.

At an exchange level with $n_K > 1$ we have two possible scenarios, PRS and FRS. In a PRS situation, an exchange between an originator N_i and n_K different recipients $\overline{N_{(i+1)_k}}$ ($1 \le k \le n_K$) is a set of n_R independent sub-exchanges, and every sub-exchange meets timeliness, then timeliness is met. In a FRS situation we have a time restriction, originators cannot enforce the contract and recipients cannot contact the TTP to finish the transaction before t_{0_i}. But since fairness is maintained before t_{0_i}, even though recipients cannot execute the finish sub-protocol, t_{0_i} is a finite value and after t_{0_i} both originators and providers are allowed to terminate the execution without losing fairness, therefore we can affirm that our protocol meets the timeliness requirement at exchange level.

At transaction level, consumers act as originators and providers as recipients. Thus, as originators, consumers can cancel the transaction by contacting the TTP, and finish it by sending the ACKnowledgement. The providers, as recipients, can finish the transaction by sending the ACCeptance or contacting the TTP, therefore, we can affirm that the protocol meets the *timeliness* requirement.

CLAIM 2.10: **AMS Non-repudiation**: *Originators cannot deny having sent a COMmitment, ACKnowledgement or a request to the TTP, and recipients cannot deny having send an ACCeptance or a request to the TTP. Moreover, none of them can be excluded from having sent one of these messages.*

PROOF. Non-repudiation evidence are generated during the execution of a transaction, linking originators and recipients through digital signatures on the messages exchanged (see table 2.4). Originators and recipients can try to deny their involvement in a transaction, to avoid fulfilling an agreement or to change the conditions in which a transaction has been executed (agreement's conditions, timing, price, etc.), but the evidence they generate along the transaction can prove their involvement.

An originator N_i can prove the involvement of a recipient $N_{(i+1)}$ using the acceptance $ACC_{(i+1)_k}$ or the TTP's proof of finalization ACK_{TTP}. In the same way, a

recipient $N_{(i+1)}$ can prove the involvement of an originator N_i using her acknowledgement of receipt ACK_i (if the exchange sub-protocol has been finished) or her commitment COM_i and the TTP's proof of finalization ACK_{TTP} (see table 2.5). Moreover, using the transaction identifiers $\overline{ID}$ an intermediary N_i is able to prove that $N_{(i-1)}$ is involved as originator and $N_{(i+1)}$ as recipient of the same transaction. Therefore, *non-repudiation* is achieved.

CLAIM 2.11: **AMS Verifiability of the TTP**: *All users involved in a transaction are able to prove a possible TTP's misbehaviour.*

PROOF. The TTP is contacted to solve possible conflicts, but, the TTP could send wrong answers to the cancel and finish sub-protocols requests. In particular, the TTP has two possibilities of misbehave:

1. In a response to a cancel request:

    - Sending cancelation evidence, C_i, when a transaction has been previously finished by another party.

    - Sending finalization evidence, ACK_{TTP}, when a transaction has not been previously finished by another party.

    In the first case, originator and recipient will have contradictory evidence from the TTP. If the recipient tries to enforce the agreement using the TTP's evidence ACK_{TTP}, the originator will be able to prove the TTP's misbehaviour with C_i, evidence of cancelation from the TTP. The same goes if the originator tries to prove the cancelation using C_i, the recipient will be able to prove the TTP's misbehaviour using ACK_{TTP}.

    In the second case, if the originator tries to enforce the agreement using the TTP's evidence ACK_{TTP}, the recipient will be able to prove the TTP's misbehaviour by asking the TTP to present evidence that the recipient has executed the finish sub-protocol, i.e., P_i's signature on the request, $EFR_{(i+1)}$ (see tables 2.4 and 2.5).

2. In a response to a finish request:

    - Sending finalization evidence, ACK_{TTP}, when a transaction has been previously canceled by another party.

    - Sending cancelation evidence, $C_{(i+1)}$, when the transaction has not been previously canceled by another party.

    Both cases are analogous to the previous examples of misbehaviour, as a response of a cancel request, where we have seen that the verifiability property is maintained. In particular in the second case, the originator will be able to prove the TTP's misbehaviour asking evidence of her execution of the cancel sub-protocol.

Therefore, the verifiability requirement is met.

CLAIM 2.12: **Confidentiality**: *A user N_i only has access to the content of the exchanges in which he is directly involved: $\{N_{(i-1)}, N_i\}$ as a recipient, and $\{N_i, N_{(i+1)}\}$ as originator.*

PROOF. Following the sub-exchange protocol (see table 2.5), the information exchanged between originator and recipient belongs to both of them, except for the evidence of previous exchanges Ev_i. But as we show in table 2.4, these evidence are encrypted with a symmetric key, and the key value is encrypted using TTP's public key (only the TTP can decrypt it). Moreover, not even the TTP has information on the messages exchanged between originators and recipients. The evidence are signatures over the hash of this message (H(M), see section 2.2.4), therefore the TTP only needs this H(M) to validate the evidence. Then, we can assure that *confidentiality* is maintained along the transaction.

CLAIM 2.13: **AMS Traceability**: *The TTP is able to trace a transaction to its origin, the consumer, and its ending, the providers.*

PROOF. Starting from an intermediary N_i, the TTP can trace the previous and posterior nodes using the evidence received $Ev_{(i-1)}$ from $N_{(i-1)}$, and the ACCeptance $ACC_{(i+1)}$ from $N_{(i+1)}$.

Backwards traceability, i.e., finding the parties involved from the intermediary N_i to the consumer N_1 is immediate, using the previous evidence $Ev_{(i-1)}$, which includes:

$H(M_{(i-2),(i-1)})$, $\overline{ID}$, $Ev_{(i-2)}$, $COM_{(i-2)}$, ACC_{i-1}
$H(M_{(i-3),(i-2)})$, $\overline{ID}$, $Ev_{(i-3)}$, $COM_{(i-3)}$, ACC_{i-2}
...
$H(M_{(1,2)})$, $\overline{ID}$, COM_1, ACC_2

Forward traceability, i.e., finding the parties involved from the intermediaries N_i to the providers $\overline{N_{(N)_{n_K}}}$ ($n_K \geq 1$), requires the cooperation of the N_j with $j > i$ parties involved. Using $ACC_{(i+1)}$ the TTP can identify the participant $N_{(i+1)}$ and ask this $N_{(i+1)}$ to provide the evidence to identify $N_{(i+2)}$, the $ACC_{(i+2)}$. If every party acts to its best benefit, we can affirm that the protocol meets the traceability requirement.

Evidence Forgery to Avoid Traceability

Backwards traceability relies on the evidence within the value $Ev_{(i-1)}$ to trace a transaction from N_i to N_1, thus, a malicious user could try to tamper this value in order to prevent a transaction to be traced. But as we have seen in section 2.2.5, the TTP would be able to detect this malicious user.

Forward traceability relies on the user cooperation and the evidence of acceptance, ACC. Again, a user may try to tamper the acceptance value in order to avoid the traceability of the transaction. The participant N_i can try to forge $ACC_{(i+1)}$, evidence of acceptance, in two ways:
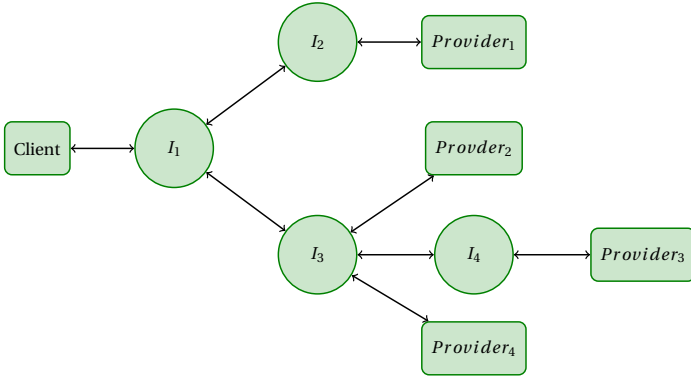
Figure 2.7: Complex Transaction Example

1. Providing a valid evidence $ACC'_{(i+1)}$ from a previous transaction or recipient. In this case, the TTP will be able to validate the signature, but the set of $\overline{ID}$ will not match the current transaction. Thus, the TTP will detect the attempt of forgery.

2. Providing an invalid value. In this case the TTP will not be able to validate the signature, detecting the forgery attempt.

In any case if the user tries to forge evidence to avoid traceability or denies to cooperate, the TTP will be able to detect it. Moreover, the TTP will be able to identify the malicious user. In these cases, the resolution of the dispute can be brought to the pertinent authority.

2.2.6 Protocol Efficiency Analysis

In section 2.2.1 we have presented a use case that could benefit from our protocol proposal, the online tourism industry, where communications among agents are usually carried out using XML (eXtensive Markup Language [43]) messages. Therefore we will use XML and its associated security standards, XMLdsig for digital signature [44], and XMLenc for encryption [45], to review the protocol performance in terms of message and computational overhead, added by the extra message information and cryptographic operations needed. The performance review has been done on the optimistic scenario, i.e., the case in which all the participants behave correctly; because it is the most probable and desirable scenario. Even so, in table 2.6 we have the TTP's computational overhead for each sub-protocol resolution.

Due to the lack of equivalent proposals in the literature, to the best of our knowledge, we cannot provide a comparison with other solutions. However, in the case of message overhead, we will compare our solution with a transaction executed without any security protocol, where the originator sends a request and the recipient

answers with a response. We will refer to this kind of transaction as *plain transaction*.

Message Overhead

A plain transaction exchanges 2 messages between originator and recipient, request and response. But in our protocol proposal we need to exchange 3 messages: COMmitment, ACCeptance, and ACKnowledgement (section 2.2.4 and table 2.5). Moreover, the messages exchanged in plain transactions are composed only of data related to the purpose of the transaction, e.g. when we want to purchase a plane ticket, the request (*plain request*) message contains the information related to the plane ticket, consumer, payment method, etc. Meanwhile messages exchanged during an execution of the exchange sub-protocol include the *plain request*, and other data like digital signatures, evidence of previous exchanges, identifiers, etc. Therefore, the protocol adds message overhead in terms of number of messages exchanged and message size.

To measure the overhead in terms of number of messages, we will calculate the number of messages exchanged during the execution of a transaction. The total number of messages exchanged during a transaction is $3(n_I + n_P)$, with $n_I \geq 1$ and $n_P \geq 1$ number of intermediaries and providers respectively. But, our transaction model has a tree structure where the branches can be executed in parallel, thus, another interesting value is the number of messages exchanged by the largest branch, which will give us an idea of the protocol response time, i.e., the number of messages we need to exchange before the client receives a response. This value is $3(n'_I + 1)$, where $n'_I \geq 1$ is the number of intermediaries in the largest branch.

In table 2.5 we have the composition of the messages exchanged during the execution of the exchange sub-protocol: plain message, identifiers, digital signatures and encrypted values.

1. Plain Message/s: $\overline{M_{(i,(i+1))}}$ the request (plain message), which size is variable.

2. Identifiers: $\{\overline{ID}, t_{0_i}\}$, a set of random UUIDs and an optional time value. Each UUID is 16 byte long [42], and the time value is represented as a dateTime value (see [46], 3.2.7), e.g., 2011-04-26T11:38:57.106+02:00, is 29 bytes long.

3. Digital Signatures: COM_i, ACC_i, ACK_i, etc. which have a size that will depend on the XMLdsig configuration [44] ($\simeq$ 1,8KB using DSA with SHA1, detached mode).

4. Encrypted Data: Ev_i the evidence of previous exchanges. Its size will depend on the number of them.

To measure the overhead, in terms of message size, we will calculate the relation between the plain message, the request message, and the commitment message generated by the exchange sub-protocol. To build the protocol messages, we have created an XML data structure following the protocol specifications (see table 2.4).
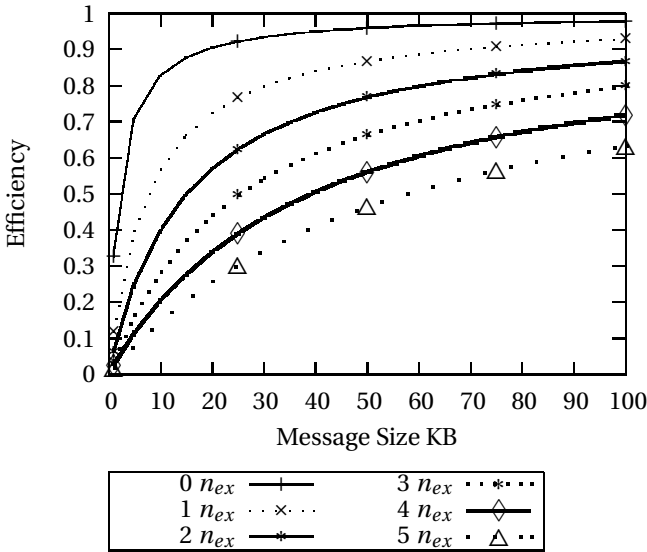
Figure 2.8: Plain Request vs. Commitment Message Protocol (n_{ex} number of previous exchanges)

The plain message will be represented as a string with random values and variable size (1KB, 5KB, 10KB, ..., 100KB). The results can be seen in figure 2.8.

As we can see in figure 2.8, the efficiency increases with the *plain message* size, but decreases quickly with the number of previous exchanges. This decrease of performance is due to the previous evidence Ev_i element, which includes the information of the previous exchanges. The increase of performance with the *plain message* size seems obvious; the extra elements needed to build a commitment message have a fixed size ($\overline{ID}$, digital signatures, etc...). The client and providers will not be affected by the decrease of performance, because the client commitment is the first message and does not have the Ev_i element, and the provider acts only as a recipient, receiving the commitment message and sending the acceptance. The intermediaries are the most affected by this performance decrease, however, they are also the most benefited from the use of the protocol (see example in section 2.2.1).

Computational Overhead

In the following paragraphs we will analyze the overhead added by the cryptographic operations, i.e. digital signatures (signature and validation) and encryption (encryption and decryption) operations. In table 2.6 we have the number of cryptographic operations that each party executes during the exchange, cancel and finish sub-protocols, for each situation (PRS and FRS).

The total number of operations executed during a transaction is the *TOTAL* value,

PRS	Signature	Validation	Encryption	Decryption
Client	$2n_R$	n_R	0	0
Provider	n_{Preq}	$2n_{Preq}$	0	0
Intermediary	$3n_{Ireq}$	$3n_{Ireq}$	n_{Ireq}	0
TOTAL	$3n_R(1+n_I)$	$3n_R(1+n_I)$	$\sum_{i=1}^{n_I} n_{Ireq_i}$	0

TTP Cancel	n_{TTPreq}	n_{TTPreq}	0	0
TTP Finish	n_{TTPreq}	$2(n_{TTPreq}+n_{ex})$	0	$n_{TTPreq} * n_{ex}$

FRS	Signature	Validation	Encryption	Decryption
Client	2	1	0	0
Provider	1	2	0	0
Intermediary	$1+2n_K$	$2+n_K$	1	0
TOTAL	n_P+3n_I	$3n_P+3n_I$	n_I	0

TTP Cancel	1	1	0	0
TTP Finish	1	$2(n_{ex}+1)$	0	n_{ex}

n_R	services/goods requested
n_P	number of providers
n_I	number of intermediaries
n_K	recipients contacted by the intermediary
n_{Preq}	requests received by the provider
n_{Ireq}	requests received by the intermediary
n_{TTPreq}	requests received by the TTP
n_{ex}	number of previous exchanges

Table 2.6: Number of Cryptographic Operations for Each Kind of Participant

the system load. In table 2.7 we have calculated the total number of operations, an optimistic approximation, and a more realistic approximation, using as example the figure 2.7. The approximations assume that the different branches of our tree structure are executed in parallel. In our configuration example (figure 2.7), the longest branch is the one formed by the nodes: client, I_1, I_3, I_4 and $Provider_3$.

The optimistic approximation assumes that, within a node, the operations related to different branches are executed in parallel. If an intermediary receives a PRS with n_{Ireq}, when calculating the number of operations as response time, instead of counting $3n_{Ireq}$ validations (see table 2.6), we count 3. On the other hand, a more realistic approximation assumes that operations within a node are executed sequentially, therefore we have to count $3n_{Ireq}$, to get an idea of the response time.

PRS	Signature	Validation	Encryption	Decryption
TOTAL	75	75	12	0
Optimistic approx.	12	12	3	0
Realistic approx.	45	42	11	0

FRS	Signature	Validation	Encryption	Decryption
TOTAL	24	24	4	0
Optimistic approx.	12	12	3	0
Realistic approx.	18	15	3	0

configuration example figure 2.7
$n_R = 5, n_I = 4, n_P = 4$
$n_{Ireq_1} = 5, n_{Ireq_2} = 1, n_{Ireq_3} = 4, n_{Ireq_4} = 2$
$n_{K_1} = 2, n_{K_2} = 1, n_{K_3} = 3, n_{K_4} = 1$

Table 2.7: Example of Total Number of Operations (related to figure 2.7)

2.3 Optimal Solutions for Asynchronous Multi-Party Contract Signing in Different Topologies

The objective of a Multi-Party Contract Signing (MPCS) protocol is to allow a set of participants P_i ($1 < i < N$) to exchange a valid signature on a contract C, without any of them gaining advantage over the others. We can describe the protocol as an application of fair exchange as: N parties want to sign a contract C, but none of the participants is willing to give his signature away unless he has an assurance that he will receive the other participant's signature.

We can find some proposals for MPCS in the scientific literature [18, 19, 20, 21, 22, 23]. Even though there is a consensus about some of the requirements all MPCS must meet, like fairness, authors impose a set of of different requirements. More-over, sometimes they do no agree on the definition of these requirements. Within these proposals, we can find some that claim to propose optimal solutions or de-fine lower-bounds to design MPCS protocols [18, 22, 23], but the different criteria applied to define requirements like fairness, or terms like round, step, etc., makes it difficult to assert the validity of such optimal solutions. Moreover, even though we can use different topologies to design MPCS protocols, none of these solutions contemplates the influence of the topology on the overall result.

The contribution of this section is twofold. Firstly, we review some common terms and notions used in previous works as the meaning of round or message. We define these terms according to the topology they are applied, and we discuss their suitability, proposing the use of the concept of transmissions to measure the effi-ciency of protocols. Secondly, we present asynchronous MPCS protocols for a set of common topologies (ring, star, sequential and mesh), meeting all the core secu-rity requirements, and we prove our solutions are optimal, improving the existent

proposals of lower-bounds for fair protocols.

2.3.1 Related Work

Garay *et al.* present in [18] the first abuse-free MPCS protocol, with $O(N^3)$ messages and $O(N^2)$ rounds, using private contract signatures (PCS). Unfortunately, it has been proved that the protocol is flawed. A very important conclusion of this paper is their theorem 3: "any complete and fair optimistic contract-signing protocol with N participants requires at least N rounds in an optimistic run".

Baum-Waidner *et al.* propose in [19] a protocol with $N + 3$ rounds in the worst case and $N + 1$ in the optimistic case, and a variant with $2(N - 1) + 6$ rounds. They are the first to compare two topologies: mesh and star topologies. They present the number of rounds and messages required for each protocol as a function of the number of dishonest parties t. But we cannot know the number of dishonest participants beforehand, therefore we will assume $t = (N - 1)$. In this case, a mesh topology requires $(N + 1)$ rounds and $(N + 1)N(N - 1)$ messages in the optimistic case. In the star topology, the protocol requires $2(N - 1) + 6$ rounds, and $(2(N - 1) + 6)2(N-1)$ messages. In [47] Baum-Waidner presents an optimization of the previous proposal [19] where they assume a number of dishonest participants $t < (N - 1)$, but it is difficult to see the utility of this enhancement because, as we have said, we cannot predict the number of dishonest parties.

Khill *et al.* [38] propose a protocol for multi-party fair exchange using a ring topology. They affirm (without any proof) that the ring model is more efficient than the full mesh model (asserting that the latter is very complex). Their protocol consists on 3 rounds and $3N$ messages (in the optimistic case; $7N$ messages in the worst case). A serious drawback of the protocol is that it is supposed that the TTP broadcasts its decision to all parties. This assumption is dangerous, because the channels can be resilient, but some party can be unreachable for other reasons.

Chadha *et al.* [20] analyze formally two previous works: Garay *et al.* [18] and Baum *et al.* [19]. They focus on three properties: fairness, timeliness and abuse-freeness. They conclude that the proposal of Baum-Waidner *et al.* [19], a protocol with $N+1$ rounds, has no security problems. On the other hand, they prove that the proposal of Garay *et al.* [18] presents a security flaw when $N = 4$: it is not fair. They try o fix the problem, but a posterior security analysis reveals that the proposal fix is also flawed.

Mukhamedov *et al.* [48] prove that Chadha *et al.*'s [20] proposal (a fix to [18]), is also flawed. They assume that parties are ordered and cite the following requirements: fairness, timeliness and abuse-freeness. An interesting issue discussed in this paper is the abort chaining problem (or resolve impossibility): non-honest parties can group together to propagate a TTP's abort decision. In fact, this is a way to prove the necessity of more than $(N - 1)$ rounds for N users (in order to avoid the abort chaining attack). The abort chaining attack is a sequence of requests made by dishonest participants to force the TTP to deliver cancel evidence, even though some other honest participant may have already signed the contract.

Ferrer *et al.* present in [49] an optimal solution for asynchronous optimistic MPCS in a ring topology, with quasi N rounds (more than $N-1$ but less than N) for N parties. Their solution meets the following requirements: effectiveness, weak fairness, timeliness, non-repudiation and verifiability of TTP. The proposal takes into account the abort chaining problem.

Onieva *et al.* [50] summarizes some properties of previous works: effectiveness, fairness, timeliness, non-repudiation, verifiability of TTP, transparency of TTP and abuse-freeness. In their paper they analyze a previous work of Ferrer *et al.* [41], proving that the protocol is flawed; it does not meet fairness requirement.

In [21] Mukhamedov and Ryan criticize the work of Baum *et al.* [19] alleging that they use a non-standard notion of signed contract and they need $(N+1)N(N-1)$ messages, more than in the solution provided in [21], $N(N-1)(\lceil N/2 \rceil + 1)$. In [21] fairness, abuse-freeness and timeliness are considered. They assume that parties are ordered and a sequential topology is used. The protocol needs $(\lceil N/2 \rceil + 1)$ rounds, and authors observe that it is not coherent with Garay's Theorem of [35], but they argue that the concept of round, used in different papers, is not clear. Later, in [28], Mukhamedov *et al.* analyze the protocol using the NuSMV model checker for the cases where $N = 4$ and $N = 5$. Even though the analysis does not find any flaw in the protocol, their definition of fairness corresponds to Asokan *et al.*'s strong fairness, but only weak fairness can be achieved in an asynchronous optimistic MPCS protocol (see section 1.2.1).

Onieva *et al.* present an interesting survey on non-repudiation in [51]. Although most of the paper is devoted to non-repudiation and certified electronic mail, a small part is devoted to MPCS. They present a solution for the MPCS problem, but they assume a synchronous model.

Mauw *et al.* [22] use the concept of abort chaining of Mukhamedov *et al.* [48] to derive a lower bound on the number of messages in MPCS protocols. The authors model contract signing protocols as sequences of numbers. They consider three security requirements: fairness, timeliness and abuse-freeness (but they affirm that the latter "will not play a role in our observations on message minimality"). Their definition of fairness matches that of Asokan *et al.*'s [9] strong fairness. They conjecture that for $N \geq 3$, all MPCS protocols with fewer than $N^2 + 1$ messages are unfair. They provide a protocol for $N = 3$ with 10 messages. In fact, instead of messages we should talk about transmissions, because some transmissions contain more than one message (see definitions in section 2.3.2). In section 2.3.4 we will prove, depending on the topology, that $(N+1)(N-1) < (N^2+1)$ transmissions are enough to achieve fairness.

Zhang *et al.* propose in [52] a game-based verification of MPCS protocols. They assume that MPCS protocols have to satisfy three properties: fairness, timeliness and abuse-freeness. They analyze the protocols provided in [28] and [22], proving the latter to be flawed for 3 signers and proposing a fix. Authors assume that "once having contacted TTP by initiating a sub-protocol, the signers would never be allowed to proceed the main protocol any more", but we cannot forbid a dishonest party to contact the TTP and proceed with the main protocol.

Following a similar reasoning than [22], Kordy *et al.* [23] propose protocols de-

rived from sequences of numbers. They consider the following requirements: fairness, timeliness and abuse-freeness. An example with $N = 3$ results in a protocol (sequential topology) with 18 messages, that can be converted to 12 messages. They talk about message complexity (the total number of messages sent in the optimistic case), bandwidth complexity (the total number of send instructions produced by their protocol compiler) and round complexity. The latter is not considered because it does not fit well in their topology ("that measure assumes the existence of a repeating structure in the protocols"). They cannot provide closed expressions for all values of N, and only provide upper bounds.

2.3.2 Topologies

Throughout the solutions found in the literature, 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.

In our opinion, the term round should not be used for measure the efficiency of a protocol, but to help in its description. The problem is that rounds in different topologies are not equal, e.g., in a ring topology a round requires N transmissions, while in a mesh topology a round requires $N(N-1)$ transmissions. Moreover, in a ring topology the protocol execution must follow a certain order, and this information can be used by the TTP to detect malicious users (see TTP rules for ring topology, in section 2.3.4), meanwhile in a mesh topology there is no execution order among participants.

Another measure typically provided is the number of messages required to complete a protocol execution, but instead of messages we should talk about transmissions, because some transmissions contain more than one message. Therefore, to measure the efficiency of a protocol we will use the number of transmissions required. Following, we will give a definition of message, transmission, and a definition of round for each topology, that we will use later to describe each protocol.

Definition 2.14 (*Message*) *A "logical" set of information sent from an originator A to B, where B can be a set of recipients $\{B_1, ..., B_N\}$.*

Definition 2.15 (*Transmission*) *The action of transmitting a set of information M from an originator A to B_i, where M can be a set of messages $\{m_1, ..., m_{M'}\}$.*

Ring

In a ring topology the message transmissions occur between two adjacent nodes P_i and $P_{(i+1)}$, until the execution flow reaches P_N, whose transmission recipient is P_1, the initiator node. The ring architecture executes the transmissions on a serial basis. In figure 2.9a we have depicted a complete round of a ring topology.
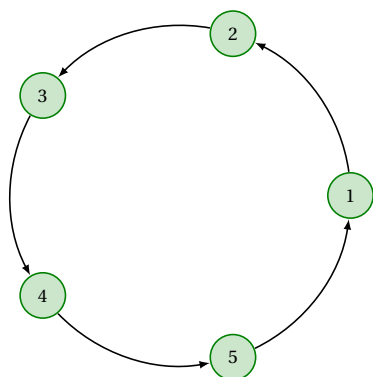
(a) Ring Topology with $N = 5$

(b) Sequential Topology with $N = 4$
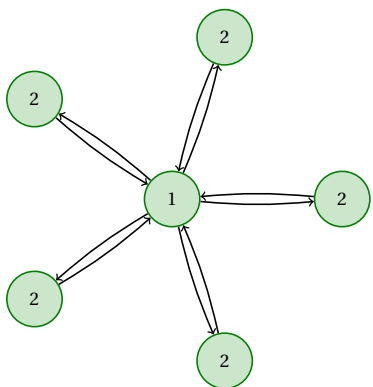
(c) Star Topology with $N = 6$
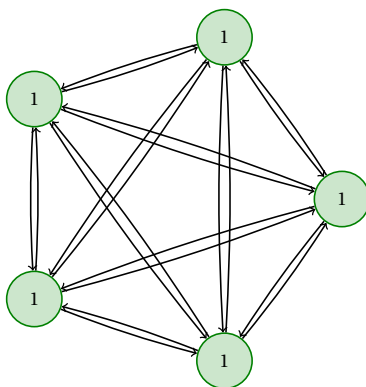
(d) Full Mesh Topology with $N = 5$

Figure 2.9: Multi-Party Contract Signature Topologies

Definition 2.16 (*Ring-Round.*) *A round begins when P_i (lets say P_1) sends a transmission to P_2 ($P_{(i+1)}$), then P_2 transmits to P_3, ..., and ends when P_1 receives the transmission from P_N, closing the ring.*

A complete *ring-round* requires N transmissions, and generates information on the execution order that can be used by the TTP to detect attempts of misbehaviour.

Sequential

In a sequential topology the transmissions are executed on a serial basis. The protocol execution flows from P_1 to P_N, and back to P_1, going through all the partici-

pants in between. In figure 2.9b we have a complete round execution of a sequential topology depicted.

Definition 2.17 (*Sequential-Round.*) *A round begins when the participant P_i (typically P_1) transmits one or more messages to the participant $P_{(i+1)}$, and ends when the same P_i who initiated the round receives a transmission from $P_{(i+1)}$.*

A complete *sequential-round* requires $2(N-1)$ transmissions. It also generates information of execution order.

Star

In a star topology the transmissions between participants are routed through a central node/participant. The central node P_i receives all the transmissions from the participants P_j ($j \in [1..N]$, $j \neq i$), and then returns to each P_j the corresponding messages. In figure 2.9c we have a complete round execution of star topology depicted.

Definition 2.18 (*Star-Round.*) *A star-round begins when the initiator P_i (lets say P_1) transmits some message or messages to all P_j ($j \in [2..N]$), and ends when P_i has received the corresponding transmission from each P_j. Alternatively, the round can be started by P_j ($j \in [2..N]$) transmitting to P_1, and finish when each P_j has received P_1's transmission.*

The star topology only generates information about who initiated the *star-round*. It has a cost of $2(N-1)$ transmissions.

Mesh

In a mesh topology the transmissions are executed on a parallel basis. Each P_i, with $1 \leq i \leq N$ will send a transmission to each P_j, with $j \in [1..N]$, $j \neq i$. In figure 2.9d we have a complete round execution of mesh topology.

Definition 2.19 (*Mesh-Round.*) *A round begins when, each P_i, with $1 \leq i \leq N$ sends a transmission to each P_j, with $j \in [1..N]$, $j \neq i$. The round will end when every participant has received a transmission from the other $N-1$ participants.*

A complete *mesh-round* requires $N(N-1)$ transmissions, and it does not generate additional information: the participants are not ordered.

2.3.3 Overview of the Protocols

In this section we will give an overall description of the protocols we will define in section 2.3.4. The objective is to design asynchronous optimistic MPCS protocols in which N participants sign the same contract C. All protocols follow the same principle: in turns, the participants exchange a series of commitments to sign the

contract *C*, until they have enough evidence to consider the contract signed. What is a "turn" or what is "enough evidence" will be determined by the topology.

Each protocol is composed of two sub-protocols: the exchange sub-protocol and the resolution sub-protocol. All protocols are optimistic, therefore if every participant behaves correctly and there are no network errors, only the exchange sub-protocol will be executed and the TTP will not intervene. Each of the proposals comply with the core security requirements: effectiveness, weak fairness, non-repudiation and timeliness (see section 1.2.1). We will use the following notation:

- **N** Number of participants.

- $\overline{\mathbf{X_{iZ}}} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_Z}\}$ Vector with Z elements.

- $\mathbf{P_i}$ Participant i, $1 \leq i \leq N$.

- $\mathbf{M_{(k,i)}}$ A set of messages transmitted during round k, generated by the participant i (see definition in section 2.3.2).

- **C** contract to be signed.

- **CID** unique Contract IDentifier. A random number used to uniquely identify a protocol execution.

- $\mathbf{h(M_i)}$ Hash Function of message M_i.

- $\mathbf{S_j[M_i]} = \mathbf{SK_j[h(M_i)]}$ j's Digital Signature on M_i (where SK_j is j's private key).

- $\mathbf{m_{(k,i)}} = S_i[CID, C, k]$.

We assume that the contract *C* includes the necessary information, as the identity of the participants, the TTP, the number *N* of participants, etc.

As regards the communications channels we make some usual assumptions:

- channels among participants P_i are unreliable, the messages can be delayed or lost.

- channels among participants P_i and the TTP are resilient, the messages can be delayed but not lost.

The TTP

To solve the resolution requests (table 2.8) the TTP follows a set of rules. These rules are based on a group of variables the TTP updates on every request received, indicating the state of a protocol execution. Following we have these group of variables, their definition, and some notation used along the rules definition.

- $\overline{X}_N = \{P_1, ..., P_N\}$ set of participants in a MPCS.

- $\overline{XR}$ set of participants who already requested resolution.

MPCS resolution sub-protocol

$P_i \rightarrow$ TTP:	$CID, C, k, M_{(k,i)}, S_{P_i}[CID, C, k, M_{(k,i)}]$

if the TTP decides canceled

$P_i \leftarrow$ TTP:	Cancel Token

else

$P_i \leftarrow$ TTP:	Signed Token

Cancel Token: $S_{TTP}[CID, C, k,' CANCELED']$
Signed Token: $S_{TTP}[CID, C, \overline{m_{(k,i)}}_N]$
$\forall \, m_{(k',i)} \in \overline{MR} \, / \, k \geq k'$, i.e., the last signature available from each participant

Table 2.8: Resolution Sub-Protocol for All Topologies

- $\overline{XC}$ set of participants who have received a canceled token from the TTP.

- $\overline{XS}$ set of participants who have received a signature token from the TTP.

- $\overline{MR}$ set of messages $M_{(k',i')}$ received by the TTP.

- $\overline{PC}$ set of participants that are allowed to cancel the contract signature.

- *canceled* boolean value stating that the contracting protocol has been canceled if its value is true.

- *signed* boolean value stating that the contracting protocol has been finished (signed) if its value is true.

The rules are the same for all the protocols we present, but there are some particularizations depending on the topology, that will be explained along each protocol. Following we have the common set of rules that the TTP will follow to solve the resolution requests received (the term *x-round* refers to the particular round of each topology):

RULE 0
The TTP will only accept one resolution request per participant P_i: if $P_i \in \overline{XR}$, the TTP will dismiss the request.

RULE 1
If the TTP receives a request from $P_i \in \overline{PC}$ during *x-round* $k = 1$, and the execution has not been previously finished (*signed=true*) by other party, the TTP will cancel it and send a canceled token to P_i.

RULE 2
If the TTP receives a request from P_i during *x-round* $k > 1$, and the execution has

not been previously canceled by other party, the TTP will finish it (*signed=true*) and send a signed token to P_i.

RULE 3

If the TTP receives a request from P_i during *x-round* $k \geq 1$, and the execution has been previously finished (*signed=true*) by other party, the TTP will send a signed token to P_i.

RULE 4

If the TTP receives a request from P_i during *x-round* $k > 1$, and the execution has been previously canceled (*canceled=true*) by other party, the TTP will check the previously received requests to see if they have cheated. If the TTP can prove that all previous requestors cheated, it will change the protocol status from canceled to finished, and deliver the signed token to P_i. Otherwise the TTP will send a cancel token to P_i. In the following sections, we will explain the process of detecting previous cheating requests, adapting it to each topology.

2.3.4 Asynchronous Optimistic MPCS Protocols

In this section we present an asynchronous optimistic MPCS protocol for each topology (ring, sequential, star and mesh). All protocols meet the core requirements defined in section 1.2.1: effectiveness, weak fairness, timeliness and non-repudiation.

An Asynchronous Optimistic MPCS protocol using Ring Topology

This protocol can be found in Ferrer-Gomila *et al.* [49], table 2.9 shows the exchange sub-protocol.

In this protocol all participants except P_N can cancel the protocol, therefore we have that in TTP's R1 $\overline{PC} = \{P_1, ..., P_{(N-1)}\}$. When P_N receives the first message he already has evidence that proves that all other participants are willing to sign the contract. Therefore, if he does not want to sign the contract, he only needs to discontinue the protocol execution. In a protocol with ring topology, TTP's R4 states:

- if $\exists M_{(k',i')} \in \overline{MR} / (k' = k)$ or $(k' = k-1$ and $i' > i)$, the TTP will send a cancel token to P_i to maintain fairness for the previous honest requests.

- if $\forall M_{(k',i')} \in \overline{MR} / k' < k-1$, then $P_{i'}$ cheated.

- if $\forall M_{(k',i')} \in \overline{MR} / k' = k-1$ and $i' < i$, then $P_{i'}$ cheated.

Notice that the TTP's rule R4 for a ring topology uses the information generated by the protocol flow (when comparing the index i with i'), the execution order to detect cheating participants.

Lemma 2.1 *An asynchronous optimistic MPCS protocol with ring topology, meeting timeliness, requires $(N+1)(N-1)$ transmissions to be fair.*

MPCS protocol with Ring topology

1.1	P_1	$\rightarrow P_2$:	$m_{(1,1)}$	$M_{(1,1)}$
1.2	P_2	$\rightarrow P_3$:	$m_{(1,1)}, m_{(1,2)}$	$M_{(1,2)}$
...				
1.$(N-1)$	$P_{(N-1)}$	$\rightarrow P_N$:	$m_{(1,1)}, m_{(1,2)}, ..., m_{(1,(N-1))}$	$M_{(1,(N-1))}$
1.N	P_N	$\rightarrow P_1$:	$m_{(1,2)}, m_{(1,3)}, ..., m_{(1,N)}$	$M_{(1,N)}$
...				
$(N-1).1$	P_1	$\rightarrow P_2$:	$m_{((N-1),1)}$	$M_{((N-1),1)}$
$(N-1).2$	P_2	$\rightarrow P_3$:	$m_{((N-1),1)}, m_{((N-1),2)}$	$M_{((N-1),2)}$
...				
$(N-1).(N-1)$	$P_{(N-1)}$	$\rightarrow P_N$:	$m_{((N-1),1)}, m_{((N-1),2)}, ...$ $..., m_{((N-1),(N-1))}$	$M_{((N-1),(N-1))}$
$(N-1).N$	P_N	$\rightarrow P_1$:	$m_{((N-1),2)}, m_{((N-1),3)}, ...$ $..., m_{((N-1),N)}$	$M_{((N-1),N)}$
$N.1$	P_1	$\rightarrow P_2$:	$m_{(N,1)}$	$M_{(N,1)}$
$N.2$	P_2	$\rightarrow P_3$:	$m_{(N,1)}, m_{(N,2)}$	$M_{(N,2)}$
...				
$N.(N-1)$	$P_{(N-1)}$	$\rightarrow P_N$:	$m_{(N,1)}, m_{(N,2)}, ..., m_{(N,(N-1))}$	$M_{(N,(N-1))}$

N = number of participants
$i \neq j$; $i, j \in [1..N]$)

Table 2.9: Asynchronous Optimistic MPCS Protocol with Ring Topology

Proof We will prove it by contradiction. Assume that $(N+1)(N-1)-1$ transmissions are enough (we eliminate the last transmission: $M_{(N,(N-1))}$). It means that P_N has all the evidence when he receives $M_{(N-1),(N-1)}$. Now we will construct the abort-chaining attack.

$P_{(N-1)}$ sends a resolution request claiming he has sent $M_{(1,(N-1))}$ but he has not received $M_{(2,(N-2))}$. He is the first to contact the TTP, therefore the TTP will apply rule R1: cancel the protocol and deliver a cancel token to $P_{(N-1)}$.

Next, $P_{(N-2)}$ sends a resolution request claiming he has sent $M_{(2,(N-2))}$ but he has not received $M_{(3,(N-3))}$. The TTP will apply R4 ($k' = k-1$ and $i' > i$) and it will send a cancel token to $P_{(N-2)}$.

Following, $P_{(N-3)}$ sends a resolution request claiming he has sent $M_{(3,(N-3))}$ but he has not received $M_{(4,(N-4))}$. The TTP will apply R4. This time, the TTP will detect that $P_{(N-1)}$ (($k' = 1$) < ($k-1 = 2$)) cheated, but to maintain fairness for $P_{(N-2)}$ ($1 = 2-1$ and $(N-1) > (N-2)$), it will send a cancel token to $P_{(N-3)}$.

We can continue this abort-chaining attack, until P_2 sends a resolution request claiming he has sent $M_{((N-2),2)}$ but he has not received $M_{((N-1),1)}$. Applying R4, the TTP detects that P_4 cheated, but to maintain fairness for P_3 it sends a cancel token to P_2.

Finally, P_1 sends a resolution request claiming he has sent $M_{((N-1),1)}$ but he has not received $M_{((N-1),N)}$. Again, the TTP will apply R4, and deliver a cancel token to P_1 to maintain fairness for P_2 (the TTP can prove that $\{P_3, P_4, ..., P_{(N-2)}, P_{(N-1)}\}$ have cheated). In this scenario, an honest P_N may have received all evidence, but an honest P_1 has a cancel token from the TTP ($M_{((N-1),N)}$ may be lost due to a network error), therefore fairness is broken. But if we add another transmission: $M_{(N,(N-1))}$, we can avoid the abort chaining attack.

Continuing the previous execution, with the additional transmissions, we have two possibilities:

- If P_1 is honest, he will not continue with the protocol execution, therefore P_N will send a resolution request to the TTP claiming the missing evidence. The TTP will be able to prove that P_2 cheated, but again, to maintain fairness for the honest participants it will send a cancel token to P_N.

- If P_1 is dishonest, and all other dishonest participants continue with the protocol execution, P_N will receive $M_{(N,(N-1))}$, therefore he will have evidence the contract has been signed.

In both cases weak fairness is met. Therefore we can affirm that, the minimum number of transmissions that an asynchronous optimistic MPCS protocol with ring topology needs to be fair is $(N+1)(N-1)$.

An Asynchronous Optimistic MPCS protocol using Star Topology

In table 2.10 we have the exchange sub-protocol. Note that the protocol initiates when all P_i ($i \in [2..N]$) send to P_1 the first commitment message $M_{(1,i)}$, and ends when each P_i sends their N^{th} commitment, the signature. Therefore, to P_1, evidence of signature is the Nth commitment, while for P_i ($i \in [2..N]$), the evidence of signature is the $(N-1)^{th}$ commitment. If the protocol was initiated by P_1 broadcasting his first commitment $m_{(1,1)}$, the protocol would need $(N-1)$ transmissions more.

In our solution with star topology, all participants except P_1 can cancel the protocol: $\overline{PC} = \{P_2, ..., P_N\}$, TTP's R1. When P_1 receives $M_{(1,i)}$ ($i \in [2..N]$) he has enough evidence to prove that all participants are committed to the signature of the contract. If he does not want to sign the contract, he only need to discontinue the protocol execution. To apply R4 the TTP follows the next assertions:

- if $\exists M_{(k',i')} \in \overline{MR} / k' \geq k-1$, the TTP will send a cancel token to P_i to maintain fairness for the previous honest requesters.

- if $\forall M_{(k',i')} \in \overline{MR} / k' < k-1$, then $P_{i'}$ cheated.

Notice that since the protocol flow of a star topology does not follow a sequence, the TTP's rule R4 can only use information on the *star-round* to detect cheating participants.

MPCS protocol with Star topology

1.1	$P_i \to P_1$:	$m_{(1,i)}$	$M_{(1,i)}$
1.2	$P_i \leftarrow P_1$:	$m_{(1,1)}, \overline{m_{(1,j)}}$	$M_{(1,1)}$
2.1	$P_i \to P_1$:	$m_{(2,i)}$	$M_{(2,i)}$
2.2	$P_i \leftarrow P_1$:	$m_{(2,1)}, \overline{m_{(2,j)}}$	$M_{(2,1)}$
...			
$(N-1).1$	$P_i \to P_1$:	$m_{((N-1),i)}$	$M_{((N-1),i)}$
$(N-1).2$	$P_i \leftarrow P_1$:	$m_{((N-1),1)}, \overline{m_{((N-1),j)}}$	$M_{((N-1),1)}$
$N.1$	$P_i \to P_1$:	$m_{(N,i)}$,	$M_{(N,i)}$

N = number of participants; $i, j \in [2..N] \diagup j \neq i$

Table 2.10: Asynchronous Optimistic MPCS Protocol with Star Topology

Lemma 2.2 *An asynchronous optimistic MPCS protocol with star topology, meeting the timeliness requirement, requires $(2N-1)(N-1)$ transmissions to be fair.*

Proof We will prove it by contradiction. Assume that $(2N-1)(N-1) - 1$ transmissions are enough, we eliminate $M_{(N,N)}$. It means that P_1 has all the evidence when he receives $\{m_{(N,2)}, m_{(N,3)}, ..., m_{(N,(N-1))}, m_{((N-1),N)}\}$. Now we will construct the abort-chaining attack.

P_2 sends a resolution request claiming he has sent $M_{(1,2)}$ but he has not received $M_{(1,1)}$. He is the first to contact the TTP, therefore applying R1, the TTP will cancel the protocol and deliver a cancel token to P_2.

Next, P_3 sends a resolution request claiming he has sent $M_{(2,3)}$ but he has not received $M_{(2,1)}$. The TTP will apply R4, $k' \geq k-1$, therefore the TTP will send a cancel token.

Following, P_4 sends a resolution request claiming he has sent $M_{(3,4)}$ but he has not received $M_{(3,1)}$. Again, the TTP will apply R4. This time, the TTP will detect that P_2 cheated $((k' = 1) < (k-1 = 2))$, but to maintain fairness for P_3, it will send a cancel token to P_4.

We continue the abort-chaining attack until, $P_{(N-1)}$ contacts the TTP claiming he has sent $M_{((N-2),(N-1))}$ but he has not received $M_{((N-2),1)}$. Applying R4, the TTP will detect the malicious behaviour of $\{P_2, P_3, ..., P_{(N-3)}\}$, but to maintain fairness for $P_{(N-2)}$ it will deliver a cancel token to $P_{(N-1)}$. Now, P_N, who is honest, contacts the TTP claiming he has sent $M_{((N-1),N)}$ (his evidence of signature) but he has not received $M_{((N-1),1)}$ (the message maybe lost due to a network error). Again, applying R4, the TTP will detect that $P_{(N-2)}$ cheated, but the TTP cannot prove $P_{(N-1)}$ cheated, therefore it will send a cancel token to P_N. In this situation we can have two honest participants with contradictory evidence, P_N has a cancel token from the TTP, and P_1 has the signature from all participants, therefore the protocol is

not fair. But we can avoid the abort chaining attack if we add another transmission $M_{(N,N)}$ to the protocol.

Continuing the abort chaining attack, with the additional transmissions, we have two possibilities:

- P_N is honest, therefore he will not send $M_{(N,N)}$ because he already has evidence the protocol is canceled. Therefore P_1 will contact the TTP claiming he has not received $M_{(N,N)}$ and the TTP will send him a cancel token.

- P_N is dishonest and delivers $M_{(N,N)}$ to P_1. P_1 will have evidence of signature from all the participants. If any of them tries to deny having signed the contract, P_1 can prove it showing the corresponding $M_{(N,i)}$.

In both situations fairness is maintained, therefore we can affirm that the minimum number of transmissions required in a star topology is $(2N-1)(N-1)$.

An Asynchronous Optimistic MPCS protocol using Sequential Topology

The table 2.11 shows the exchange sub-protocol. References to k or k' in the sequential topology rules for the TTP are not references to a *sequential-round*, but to half-*sequential-round*. Each block in table 2.11 corresponds to a *sequential-round* ($P_1 \rightarrow P_N \rightarrow P_1$), and is composed of two half-*sequential-rounds* ($P_1 \rightarrow P_N$, $P_N \rightarrow P_1$). The sub-indexes k of messages $M_{(k,i)}$, indicate the number of half-*sequential-round*.

In a protocol with sequential topology we have: $\overline{PC} = \{P_1, ..., P_{(N-1)}\}$, i.e. all participants except P_N can cancel the protocol. When P_N receives $M_{(1,(N-1))}$ he has enough evidence to prove that all participants are willing to sign the contract C. If he does not want to sign the contract, he only has to discontinue the protocol execution. To apply R4, the TTP follows these statements:

- if $\exists M_{(k',i')} \in \overline{MR}$ / ($k' = k$) or ($k' = k-1$ and $i' < i$), the TTP will send a cancel token to P_i to maintain fairness for the previous honest requesters.

- if $\forall M_{(k',i')} \in \overline{MR}$ / $k' < k-1$, then $P_{i'}$ cheated.

- if $\forall M_{(k',i')} \in \overline{MR}$ / $k' = k-1$ and $i' > i$, then $P_{i'}$ cheated.

Lemma 2.3 *An asynchronous optimistic MPCS protocol, meeting the timeliness requirement and using sequential topology, requires $(N+1)(N-1)$ transmissions to be fair.*

Proof We have to distinguish situations with N even and N odd. We will start with N even. Assume that $(N+1)(N-1) - 1$ messages are enough. We eliminate the last transmission: $M_{((N+1),(N-1))}$, which means that P_N has all evidence when he receives $M_{((N-1),(N-1))}$. With this assumption we will start the abort chaining attack.

MPCS protocol with Sequential topology

1.1	P_1	$\rightarrow P_2$:	$m_{(1,1)}$	$M_{(1,1)}$
1.2	P_2	$\rightarrow P_3$:	$m_{(1,1)}, m_{(1,2)}$	$M_{(1,2)}$
…				
1.$(N-1)$	$P_{(N-1)}$	$\rightarrow P_N$:	$m_{(1,1)}, m_{(1,2)}, ..., m_{(1,(N-1))}$	$M_{(1,(N-1))}$
1.$(N-1)+1$	P_N	$\rightarrow P_{N-1}$:	$m_{(1,N)}$	$M_{(2,N)}$
1.$(N-1)+2$	P_{N-1}	$\rightarrow P_{N-2}$:	$m_{(1,N)}, m_{(1,(N-1))}$	$M_{(2,(N-1))}$
…				
1.$2(N-1)$	P_2	$\rightarrow P_1$:	$m_{(1,2)}, m_{(1,3)}, ..., m_{(1,N)}$	$M_{(2,2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

If N is EVEN

K.1	P_1	$\rightarrow P_2$:	$m_{(K,1)}$	$M_{((N+1),1)}$
K.2	P_2	$\rightarrow P_3$:	$m_{(K,1)}, m_{(K,2)}$	$M_{((N+1),2)}$
…				
K.$(N-1)$	$P_{(N-1)}$	$\rightarrow P_N$:	$m_{(K,1)}, m_{(K,2)}, ..., m_{(K,(N-1))}$	$M_{((N+1),(N-1))}$

If N is ODD

K.1	P_1	$\rightarrow P_2$:	$m_{(K,1)}$	$M_{((N),1)}$
K.2	P_2	$\rightarrow P_3$:	$m_{(K,1)}, m_{(K,2)}$	$M_{((N),2)}$
…				
K.$(N-1)$	$P_{(N-1)}$	$\rightarrow P_N$:	$m_{(K,1)}, m_{(K,2)}, ..., m_{(K,(N-1))}$	$M_{((N),(N-1))}$
K.$(N-1)+1$	P_N	$\rightarrow P_{N-1}$:	$m_{(K,N)}$	$M_{((N+1),N)}$
K.$(N-1)+2$	P_{N-1}	$\rightarrow P_{N-2}$:	$m_{(K,N)}, m_{(K,(N-1))}$	$M_{((N+1),(N-1))}$
…				
K.$2(N-1)$	P_2	$\rightarrow P_1$:	$m_{(K,2)}, m_{(K,3)}, ..., m_{(K,N)}$	$M_{((N+1),2)}$

N = number of participants
$K = \lceil (N+1)/2 \rceil$

Table 2.11: Asynchronous Optimistic MPCS Protocol with Sequential Topology

P_2 sends a resolution request claiming he has sent $M_{(1,2)}$ but he has not received $M_{(2,3)}$. He is the first to contact the TTP, and therefore the TTP will apply R1: it will cancel the protocol and deliver a cancel token to P_2.

Next, $P_{(N-1)}$ sends a resolution request claiming he has sent $M_{(2,(N-1))}$ but he has not received $M_{(3,(N-2))}$. The TTP will apply R4 ($k' = k-1$ and $i' < i$), therefore the TTP will send a cancel token.

Following, P_3 sends a resolution request claiming he has sent $M_{(3,3)}$ but he has not received $M_{(4,4)}$. Again, the TTP will apply R4. This time, the TTP will detect that P_2 cheated ($k' < k-1$), but to maintain fairness for $P_{(N-1)}$, it will send a cancel token to P_3.

To build an abort-chaining attack, the cheating parties must contact the TTP in a certain order. In particular: $P_2, P_{(N-1)}, P_3, P_{(N-2)}, ..., P_{(N/2)}, P_{((N/2)+1)}$.

Finally, P_1 (honest participant) contacts the TTP claiming he has sent $M_{((N-1),1)}$ but he has not received $M_{(N,2)}$. Applying R4, the TTP will detect $P_{(N/2)}$ has cheated, but to maintain fairness for $P_{((N/2)+1)}$ it will send a cancel token to P_1. This situation is consistent with the fact that P_N and P_1 are honest, if we assume all other participants cheated and continued the protocol execution, we have that an honest participant, P_1 has received cancel evidence from the TTP, and another honest participant P_N has received evidence that the contract has been signed by all parties $M_{((N-1),(N-1))}$. The protocol is not fair. But if we add the last transmission, $M_{((N+1),(N-1))}$, we can avoid the abort-chaining attack.

Continuing the previous execution, with the additional transmissions, we have two possibilities:

- If P_1 is honest, he will not send $M_{((N+1),1)}$, therefore P_N will send a resolution request to the TTP claiming the missing evidence. The TTP will be able to prove that $P_{((N/2)+1)}$ cheated, but again, to maintain fairness for the honest participants (P_1) it will send a cancel token to P_N.

- If P_1 is dishonest, and all other dishonest participants continue with the protocol execution, P_N will receive $M_{((N+1),(N-1))}$, therefore he will have evidence the contract has been signed.

Now we will prove the case with N odd. Assume that $(N+1)(N-1)-1$ messages are enough, we eliminate the last transmission: $M_{((N+1),2)}$, which means that P_1 has all evidence when he receives $M_{((N-1),2)}$. With this assumption we will start the abort chaining attack.

$P_{(N-1)}$ sends a resolution request claiming he has sent $M_{(1,(N-1))}$ but he has not received $M_{(2,N)}$. He is the first to contact the TTP, therefore the TTP will apply R1: it will cancel the protocol and deliver a cancel token to $P_{(N-1)}$.

Next, P_N sends a resolution request claiming he has sent $M_{(2,N)}$ but he has not received $M_{(3,(N-1))}$. The TTP will apply R4, ($k' = k-1$ and $i' < i$), therefore the TTP will send a cancel token.

Following, $P_{(N-3)}$ sends a resolution request claiming he has sent $M_{(3,(N-3))}$ but he has not received $M_{(4,(N-2))}$. Again, the TTP will apply R4. This time, the TTP will

detect that $P_{(N-1)}$ cheated ($k' < k-1$), but to maintain fairness for P_N, it will send a cancel token to $P_{(N-3)}$.

We continue the abort chaining attack in the following order: $\{P_{(N-1)}, P_N, P_{(N-3)}, P_{(N-2)}, ..., P_2, P_3\}$.

Finally, P_3 contacts the TTP claiming he has sent $M_{((N-1),3)}$ but he has not received $M_{(N,2)}$. Applying R4, the TTP will detect P_4 has cheated, but to maintain fairness for P_2 it will send a cancel token to P_3. This situation is consistent with the fact that P_3 and P_1 are honest, we have that an honest participant, P_3 has received cancel evidence from the TTP, and another honest participant P_1 has received evidence that the contract has been signed by all parties, $M_{((N-1),2)}$. The protocol is not fair. But if we add the last transmission, $M_{((N+1),1)}$, we can avoid the abort-chaining attack.

Continuing the previous execution, with the additional transmissions, we have two possibilities:

- If P_3 is honest, he will not send $M_{((N+1),3)}$, therefore P_1 will send a resolution request to the TTP claiming the missing evidence. The TTP will be able to prove that P_2 cheated, but again, to maintain fairness for the honest participants (P_3) it will send a cancel token to P_1.

- If P_3 is dishonest, and all other dishonest participants continue with the protocol execution, P_1 will receive $M_{((N+1),2)}$, therefore he will have evidence the contract has been signed.

In both situations, with N even and N odd, the protocol meets the weak fairness requirement. Therefore we can affirm that a sequential protocol requires $(2N-1)(N-1)$ transmissions to be fair.

An Asynchronous Optimistic MPCS protocol using Mesh Topology

The table 2.13 shows the exchange sub-protocol. In the mesh topology, all participants can cancel the protocol: $\overline{PC} = \{P_1, ..., P_N\}$, in TTP's R1. Regarding the detection of cheating users, R4 for a mesh topology states:

- if $\exists\, M_{(k',i')} \in \overline{MR} \,/\, k' \geq k-1$, the TTP will send a cancel token to P_i to maintain fairness for the previous honest requests.

- if $\forall\, M_{(k',i')} \in \overline{MR} \,/\, k' < k-1$, then $P_{i'}$ cheated.

Lemma 2.4 *An asynchronous optimistic MPCS protocol with mesh topology, meeting the timeliness requirement, requires $N^2(N-1)$ transmissions to be fair.*

Proof We will prove it by contradiction. Assume that $N^2(N-1)-1$ transmissions are enough, P_N does not send $m_{(N,N)}$ to $P_{(N-1)}$. Which means that $P_{(N-1)}$ has all the evidence when he receives $\{m_{(N,1)}, m_{(N,2)}, ..., m_{(N,(N-1))}, m_{((N-1),N)}\}$. Now we will construct the abort-chaining attack.

Protocol	Topology	Transmissions*	N=3	4	5	6	7	8
★	ring	$(N+1)(N-1)$	8	15	24	35	48	63
★	star	$(2N-1)(N-1)$	10	21	36	55	78	105
[19]	star	$4(N-1)(N+2)$	40	72	112	160	216	280
★	sequential	$(N+1)(N-1)$	8	15	24	35	48	63
[21]	sequential	$(\lceil N/2 \rceil + 1)2(N-1)$	12	18	32	40	60	70
★	mesh	$N^2(N-1)$	18	48	100	180	294	448
[19]	mesh	$N(N+1)(N-1)$	24	60	120	210	336	504

★ Our proposal.
N, number of participants.
∗ optimistic case, the TTP does not intervene in the protocol execution, and
$N-1$ malicious participants assumed.

Table 2.12: Efficiency of Asynchronous Optimistic MPCS Protocols

MPCS protocol with Mesh topology		
1	$P_i \rightarrow \overline{P_j}_{(N-1)}:$	$m_{(1,i)}$
2	$P_i \rightarrow \overline{P_j}_{(N-1)}:$	$m_{(2,i)}$
…		
$(N-1)$	$P_i \rightarrow \overline{P_j}_{(N-1)}:$	$m_{((N-1),i)}$
N	$P_i \rightarrow \overline{P_j}_{(N-1)}:$	$m_{(N,i)}$

N = number of participants and rounds; $i, j \in [1..N]$; $j \neq i$

Table 2.13: Asynchronous Optimistic MPCS Protocol with Mesh Topology

P_1 sends a resolution request to the TTP, claiming he has sent $m_{(1,1)}$ to all parties but he is missing some $m_{(1,i)}$ ($i \in [1..N], i \neq 1$). He is the first to contact the TTP, therefore according to rule R1, the TTP will send a cancel token to P_1.

Next, P_2 sends a resolution request to the TTP, claiming he has sent $m_{(2,2)}$ to all parties but he is missing some $m_{(2,i)}$ ($i \in [1..N], i \neq 2$). The protocol execution has already been canceled, therefore the TTP applies R4, $k' \geq k-1$, and it will send a cancel token to P_2.

Following, P_3 sends a resolution request to the TTP, claiming he has sent $m_{(3,3)}$ to all parties but he is missing some $m_{(3,i)}$ ($i \in [1..N], i \neq 3$). Applying R4, the TTP detects that P_1 cheated: $(k' = 1) < (k-1 = 2)$. But P_3's claim is coherent with the one from P_2. Therefore, to guarantee fairness for P_2 the TTP will deliver a cancel token to P_3. We can continue the abort-chaining until $P_{(N-2)}$ sends a resolution request to the

TTP, claiming he has sent $m_{((N-2),(N-2))}$ to all parties but he is missing some $m_{((N-2),i)}$ ($i \in [1..N]$, $i \neq (N-2)$). Applying rule R4, the TTP detects that P_{N-4} cheated, but to maintain fairness for P_{N-3}, the TTP sends a cancel token to $P_{(N-2)}$.

Finally, P_N sends a resolution request to the TTP, claiming he has sent $m_{((N-1),N)}$ (evidence of P_N's signature for $P_{(N-1)}$) to all parties but he is missing some $m_{((N-1),i)}$ ($i \in [1..N]$, $i \neq (N-1), N$). Again, the TTP applies R4 and detects that $P_{(N-3)}$ cheated, but to maintain fairness for $P_{(N-2)}$ it will send a cancel token to P_N. If the protocol ends here, we have that P_N, honest participant, has evidence that the contract signature has been canceled, but $P_{(N-1)}$, also honest, has evidence proving the contract has been signed, $\{m_{(N,1)}, m_{(N,2)}, ..., m_{(N,(N-1))}, m_{((N-1),N)}\}$ (the cheating participants continue the protocol execution), therefore the protocol is not fair. But if we restore P_N's last transmission to $P_{(N-1)}$, we can avoid the abort-chaining attack. Continuing the previous chain, with the additional transmissions, we have two possibilities:

- If P_N is honest, he will not send $m_{(N,N)}$, therefore $P_{(N-1)}$ will send a resolution request to the TTP claiming the missing evidence. The TTP will be able to prove that $P_{(N-2)}$ cheated, but again, to maintain fairness for the honest participants (P_N) it will send a cancel token to $P_{(N-1)}$.

- If P_N is dishonest, and all other dishonest participants continue with the protocol execution, $P_{(N-1)}$ will receive all $m_{(N,i)}$, therefore he will have evidence the contract has been signed.

In both cases the weak fairness is met. Therefore we can affirm that the minimum number of transmissions a mesh protocol needs to be fair is $N^2(N-1)$.

2.3.5 Protocol Comparison

In table 2.12 we can compare the efficiency of our proposals and some of the most relevant presented in the related work (we eluded proposals that have been proved flawed). In section 2.3.2 we have given a definition of message and transmission to avoid confusions (a transmission can include several messages). From the results we can affirm that, regarding the number of transmissions needed, the ring and sequential topologies are more efficient than star and mesh topologies. Within each topology, our proposals are the most efficient, in particular, the solutions with ring and sequential topology. They only require $(N+1)(N-1)$ transmissions.

Following we will explain how we have calculated the number of transmissions for the proposals from Baum-Waidner *et al.*'s [19] and Mukhamedov *et al.* [21]. Baum-Waidner *et al.*'s [19] describes the cost of their proposal in number of messages. Their use of the term message matches our definition of transmission (see section 2.3.2). We calculated the number of transmissions from their description of the cost of each protocol:

- mesh topology: $(N-1)+2$ rounds, $N(N-1)$ transmissions each round $\rightarrow$ $N(N-1)(N+1)$

- star topology: $2(N-1)+6$ rounds, $2(N-1)$ transmissions each round $\rightarrow 4(N-1)(N+2)$

In its paper, Mukhamedov *et al.* [21] present a formula to calculate the number of messages. They use the term message as we have defined it in section 2.3.2. Since we are comparing the number of transmissions, we have calculated this number from their description of the protocol: $(\lceil N/2 \rceil + 1)$ rounds, $2(N-1)$ transmissions each round.

2.4 Conclusions

In this chapter of the dissertation, some contributions related to digital signature of contracts as a fair exchange application have been presented. On one hand this chapter presents two asynchronous optimistic protocols for new digital signature scenarios, the Atomic Multi-Two Party (AM2P) scenario (section 2.1) and the Agent Mediated Scenario (AMS) (section 2.2). In an AM2P scenario, we have N parties (1 consumer, C, and $N-1$ providers, P_i) distributed as a set of $(N-1)$ pairs of parties $\{C, P_1\}, \{C, P_2\}..., \{C, P_{(N-1)}\}$, that want to sign a set of $(N-1)$ contracts $\{M_1, M_2,...,M_{(N-1)}\}$ pairwise, i.e., C and P_1 want to sign the contract M_1, C and P_2 the contract M_2, etc. But C does not want to send her signature unless she has assurance she will receive all signatures from $P_1,...,P_{(N-1)}$, neither P_i wants to send his signature unless he receives C's signature on the contract M_i. Whereas in an AMS scenario we have a consumer that initiates a transaction contacting a provider, but to fulfil the consumer's request, this provider may need contact n_K different providers, and in turn, each of these providers may need to contact another set of providers. Thus, we end up creating a chain of exchanges where the links should be fair exchanges. Despite there are many research works related to fair protocols for digital signature of contracts, these two scenarios were not solved yet. The presented AM2P and AMS protocols are the first of their kind to meet all the core requirements defined in section 1.2 (effectiveness, weak fairness, timeliness, and non-repudiation) and two additional requirements, confidentiality and verifiability of the TTP. Moreover, the AMS protocol also meets a specific security requirement; traceability.

On the other hand, this chapter of the dissertation reviews some common terms and definitions used in previous works about MPCS protocols and defines a common ground, clarifying some differences. In particular, it describes the four most commonly used topologies (ring, star, sequential and mesh), and for each one it clearly defines the terms message and round, typically used to measure efficiency, and introduces the term transmission, which we suggest should be used for efficiency measures instead of the term round or message. Using a common approach for the design of optimistic asynchronous MPCS protocols (the exchange of several commitments/promises of signature among the participants in order to generate enough evidence to ensure fairness), we propose an optimistic asynchronous MPCS protocol for each topology. Each proposal meets with the core security requirements for optimistic asynchronous MPCS protocols (as defined in section 1.2).

Moreover, we prove that these proposals are optimal: the number of transmissions required by each protocol is the minimum needed to maintain fairness, therefore we also define a new set of lower-bounds, minimum number of transmissions, for each topology.

CERTIFIED ELECTRONIC MAIL (CEM)

This chapter is dedicated to the application of fair exchange to Certified Electronic Mail (CEM) protocols. In particular, the work presented here focuses on the integration of CEM protocols into the existent Internet e-mail infrastructure. Section 3.1 presents the security requirements for optimistic CEM protocols with multiple MTAs, which includes the core fair exchange requirements, some additional ones and an specific one for CEM protocols. In Section 3.2 we review the previous work on CEM protocols. Following, two optimistic protocols with multiple MTAs are presented, sections 3.4 and 3.5. In one case, it is assumed that all MTAs are untrusted, the first proposal of this kind, while in the second protocol, it is assumed that each user trusts his own MTA. Moreover, the protocol flow follows a similar approach than traditional e-mail when delivering (SMTP) and retrieving (Post Office Protocol - Version 3 (POP3) or Internet Message Access Protocol (IMAP)) messages.

3.1 Requirements

CEM services and security requirements found in the literature are reviewed by Ferrer-Gomila *et al.* [33]. The list of these requirements depends on the definition of the CEM service, which is clear, from Ferrer-Gomila *et al.* [33] and Tauber [53], that there is no general consensus on this definition. However, a CEM protocol is a subset of the fair exchange protocols family. Therefore, an optimistic CEM protocol must comply with the fair exchange requirements, defined in section 1.2. Moreover, as in our protocol architecture we have MTAs between a sender and a recipient, we require the following *non-repudiation* services: *origin, submission, delivery* and *receipt* (as stated by Onieva *et al.* [54]).

Kremer and Markowitch [55] added a new requirement specific to CEM protocols, *no author based selective receipt*, which states that once the recipient knows the identity of the sender, he cannot prevent the delivery of the CEM, i.e., the sender will receive the non-repudiation of receipt evidence. This requirement is an extension from the *no message based selective receipt*, which states that once the recipient knows the message content, he cannot prevent its delivery. Consider the situation where a person is in debt and receives a certified mail. He may guess the content of the delivery beforehand and thus refuse its acceptance. In our case, we will take a more *certified-mail-like* approach, adding the *non-selective receipt* as requirement. The certified mail focus on the *no message based selective receipt* requirement, if a recipient asks the identity of the sender, the postman will tell him who is the sender, but not the content of the message. Considering the particularization of core and additional requirements, and the necessity of an specific one, next we list and define the security requirements for optimistic CEM protocols with multiple MTAs:

Definition 3.1 (CEM *Effectiveness*) *If every party behaves correctly, the CEM will be delivered without TTP's intervention: the recipient will be able to read the message received, and the sender will have proof of it.*

Definition 3.2 (CEM *Fairness*) *At the end of a CEM exchange, every honest party will have their expected items. The honest recipient will have the means to obtain the plain message, and proof of its origin, while the honest sender will have proof of having submitted the message, and proof that the recipient has received it. Otherwise no one will have advantage over the other party, i.e., the sender cannot prove the recipient has received the CEM, and the recipient cannot access the plain message.*

Definition 3.3 (CEM *Timeliness*) *Every party involved in a CEM exchange can be sure that this exchange will end in a finite amount of time and, once the protocol has finished, fairness will be maintained.*

Definition 3.4 (CEM *Non-Repudiation*) *In a CEM delivery involving a sender, her mail transfer agent, a recipient and his mail transfer agent, no one can deny being involved or can be excluded from having participated. In particular, we have the following non-repudiation services:*

Origin (NRO) *The sender cannot deny having originated a CEM.*

Submission (NRS) *The sender's MTA cannot deny having received a CEM from the sender.*

Delivery (NRD) *The recipient's MTA cannot deny having received a CEM.*

Receipt (NRR) *A recipient cannot deny having received a CEM.*

Additional requirements met by our CEM protocols:

Definition 3.5 (CEM *Verifiability of TTP*) *If the TTP misbehaves resulting in a possible loss of fairness for an honest participant, this participant can prove the TTP's misbehaviour.*

Definition 3.6 (CEM *Confidentiality*) *The content of a CEM is available only to the sender and the recipient.*

Finally, we have one specific requirement for our CEM protocols:

Definition 3.7 (*Non-Selective Receipt*) *Once the recipient knows he has received a CEM, he cannot prevent its delivery.*

3.2 Related Work

Even though OpenPGP (Pretty Good Privacy [56]) and S/MIME (Secure Multipurpose Internet Mail Extensions [57]) provide confidentiality, integrity and authenticity to Internet e-mail, the lack of a fair exchange standard proposal still remains. The Internet community has tried to fill this gap with the receipting mechanisms of Delivery Status Notifications (DSN [58]), Message Disposition Notifications (MDN [59, 60]), SMTP service extensions [61] as well as S/MIME receipts [62]. However, none of these mechanisms makes use of a TTP and thus each one relies on fairly acting recipients. Therefore, dispute resolutions are hard to carry out since each party can deny having participated in the communication.

Since a decade, governments, postal services and private companies are thus putting custom certified mail systems into operation on the Internet. Tauber [53] has assessed and evaluated dominant certified mail systems and standards with respect to their security properties. Interestingly, all systems in operation are using inline TTPs, whereas the research community is focusing on optimistic approaches. There is a simple reason for that: even though optimistic approaches are more efficient and require less trust in TTPs, high interactivity between the sender and recipient, as well as changes in the sending and receiving environment (e.g. e-mail client plug-ins) make them harder to deploy in practice.

The scientific community is working since the late 1980s on secure protocols for the Fair Exchange family, including protocols for CEM applications. The vast majority of proposals for CEM protocols assume that senders and recipients can communicate directly, without intermediaries. Among them we find proposals with online TTP [63, 64, 65, 66, 67, 68], and optimistic ones [69, 70, 71, 72, 55, 68, 73, 74, 40, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86]. But most of them do not meet the TTP's verifiability requirement. Moreover, the assumption of a direct communication between sender and receiver does not fit with the general e-mail architecture, where senders and recipients communicate through intermediaries asynchronously. Therefore, we cannot consider them as useful solutions for CEM. Following, we briefly review some of these protocols.

Schenier and Riordan [64] proposed a protocol where the keys to decipher the messages are published in publicly available forums, which requires them to be totally trusted. Abadi *et al.* [65] approach does not require a Public Key Infrastructure (PKI), and no software installation is required on the recipient's side. Permpoontanalarp and Kanokkanjanapong [66] solution is based on Abadi *et al.* [65] protocol, and they further enhance it adding protection to Denial of Service attacks, and measures to penalize the selective receipt of messages, to the point where recipients cannot receive messages anymore. The authors refer to this property as undeniable fairness.

Ferrer-Gomila *et al.* [69] presented an efficient certified e-mail protocol requiring only three steps to deliver a certified e-mail. Their proposal was reviewed by Monteiro and Dahab [70], who proposed a fix. But in a later paper, Wang *et al.* [71] exposed some weaknesses and security flaws that could affect both proposals ([69] and [70]). Mut and Ferrer-Gomila [72] presented an optimistic protocol which uses a group of TTPs, where the resolution decisions are made by majority vote of the TTPs. Kremer and Markowitch [55] major contribution is the definition of a new property for certified e-mail: no-author based selective receipt. In order to meet this new property, their proposal requires a TTP to provide anonymity to the sender, thus it is not entirely optimistic. Imamoto an Sakurai [68] solution with online TTP has the particularity that is the receiver who chooses the TTP.

Ateniese and Nita-Rotaru [74] and Nenadic *et al.* [75] proposed optimistic protocols with transparent TTP. Their solutions are based on different forms of Verifiable and Recoverable Encrypted Signatures (VRES) [87, 88]. Ma *et al.* [77] showed that Nenadic *et al.* [75] protocol does not meet fairness, and proposed a fix. But later, Hwang and Lai [82] reviewed Ma *et al.* [77] protocol, exposing some weaknesses, and presented a solution. Wang *et al.* [79] use Convertible Signatures to obtain TTP transparency in their optimistic certified e-mail protocol.

Ferrer-Gomila *et al.* [73], Onieva *et al.* [89] and Zhou [76] proposed optimistic multi-party protocols. In a multi-party CEM protocol, we have a sender and a set of recipients. Ferrer-Gomila *et al.* [73] proposed a protocol where the sender could notify a set of recipients (the same message to each recipient), but only those who acknowledged the reception of the message could have access to its content. Onieva *et al.* [89] proposal is an extension of Micali's [90] two-party fair exchange protocol, allowing a sender to deliver the same certified message to several recipients. Zhou [76] exposed some security flaws in Ferrer-Gomila *et al.*'s [73] multi-party certified e-mail protocol, and proposed an improved version fixing the weaknesses detected.

Oppliger and Stadlin [78] proposed a certified mail system for the Internet based on an online TTP, which uses dual signatures to cryptographically link keys to the corresponding certified mail messages. Cao *et al.* [80] based their optimistic protocol in the use of bilinear pairing [91]. Cederquist *et al.* [81] proposed a certified e-mail protocol using key-chains [92]. A key-chain is a sequence of keys, where these keys are the result of executing a hash function on the previous key. Later, Liu *et al.* [67] presented an enhanced version of Cederquist *et al.* [81] protocol, which solves a security flaw and adds TTP transparency using a VRES scheme [93]. Wang *et al.* [83, 85] proposals are based on signcryption [94].

Blundo *et al.* [86] present a certified e-mail protocol with an offline trusted TTP and a Time Stamp Server (TSS). Their proposal meets the fairness, non-repudiation and timeliness requirements, but fails to address the verifiability of the TTP and the non-selective receipt (a description of the message content is included in the initial protocol steps).

There are, though, some proposals for CEM that assume sender and recipient communicate through some kind of intermediary [95, 96, 97, 98, 99], either a TTP, a trusted agent or a semi-trusted one. The term semi-trusted was first introduced by Franklin and Reiter [100], referring to a TTP that can misbehave but not collude with other entities. These solutions [95, 96, 97, 98, 99] are more interesting from a practical point of view because their architecture resembles to the general e-mail one. Following we will describe these proposals and we will explain why they are not a proper solution for the CEM service.

Bahreman and Tygar [95] proposed 2 different CEM protocols, one with inline TTP (B-CEM) and the other one without TTP (S-CEM). In the B-CEM proposal the sender uses an inline TTP, the Postmaster, to deliver CEMs. It provides non-repudiation of origin, submission and receipt evidence, but it fails to meet the non-selective receipt, verifiability of the TTP, and confidentiality requirements. Moreover, fairness depends on the assumption that the TTP cannot misbehave and there are no network failures. Even though the authors recommend that messages exchanged should be MIME compliant, it cannot be easily deployed on the existent Internet e-mail architecture where senders and recipients can be registered with different mail providers, and these mail providers should not be considered TTPs. The S-CEM proposal assumes that senders and recipients have equal computational power, which is unrealistic.

Zhou and Gollman [96] proposed a protocol where a CEM is delivered through a set of delivery agents. The protocol provides evidence of submission, delivery and receipt, but fairness is not addressed. Moreover, it relies on the honest behaviour of users and delivery agents.

With their TRICERT protocol, Ateniese *et al.* [97] proposed an optimistic protocol, which makes use of an offline TTP and a unique semi-trusted postal agent, which acts as a proxy between the sender and the recipient to ensure a fair exchange. However, from a trust point of view, the protocol is more beneficial for the sender since she can choose the postal agent. Further, by looking at the decentralized Internet e-mail system, this approach cannot be easily deployed on top of the existing architecture for e-mail. Senders and recipients can be registered with different mail providers, thus a protocol with at least two mail transfer agents is required. Park and Cho [98] enhanced the TRICERT protocol adding a delivery deadline to avoid indefinite delays on the e-mail delivery. Neither Ateniese *et al.*'s [97] nor Park and Cho's [98] solutions meet the verifiability of TTP requirement, thus in case of dispute, fairness relies on the TTP's good will.

Liu *et al.* [99] presented an optimistic CEM protocol based on the current e-mail infrastructure. Their approach is the most practical, using Mail User Agents (MUA) as client applications and Mail Transfer Agents (MTA) as intermediaries (each participant chooses his own). However, to solve recovery protocol requests, the TTP

requires contacting the end-users, which is not practical. As Ateniese *et al.* [97], Liu *et al.* [99] assume that the TTP and the sender's MTA are trusted entities, i.e., they cannot cheat. Moreover, the recipient's MTA is also trusted, none of the MTAs (sender and recipient) is verifiable, and the protocol does not provide the necessary non-repudiation services.

3.3 Certified Electronic Mail Overview

This section presents an overview of the CEM proposals. The proposal with untrusted MTAs is described in section 3.4 and the proposal with trusted MTA in section 3.5. Figure 3.1 sketches the entities and sub-protocols involved in the proposals. As we can see, a CEM protocol consists of a *delivery* sub-protocol and a set of *resolution* sub-protocols. If all the participants behave honestly the TTP will not intervene, and only the *delivery* sub-protocol will be executed. To solve the possible claims devised from a problematic delivery of a CEM, we use the *resolution* sub-protocols. In the CEM architecture three different entities can be distinguished (figure 3.1):

- UA_X The User Agent is the end user's application. A user agent can behave as sender (UA_S) and/or recipient (UA_R).

- MTA_Y The Mail Transfer Agent is an entity who offers e-mail services, and in particular CEM services. We can distinguish the sender's MTA, the MTA_S, and the recipient's MTA, the MTA_R.

- TTP The Trusted Third Party is an external entity to the conventional transferring e-mail system that guarantees fairness for the CEM service.

Typically when describing protocols where two participants are involved, we refer to them as Alice and Bob (A and B). Hence, even though the user agents (UA_X) and mail transfer agents (MTA_Y) are software entities, to ease the protocol description we will assign different gender to sender (she) and recipient (he) entities, while the TTP will remain neutral (it).

Following there is a list of assumptions common to both CEM protocol proposals:

- The TTP is a semi-trusted entity, as defined by Franklin and Reiter [100]: it can misbehave but it cannot collude with other parties.

- The communication channel between user agents and mail transfer agents, and between mail transfer agents are *unreliable* (messages can be lost), and they are *resilient* between the TTP and the participants (messages can be delayed, but they will arrive).

- The UA_S has knowledge of the recipient's e-mail address, and his public key PK_{UA_R} (key management is not addressed).
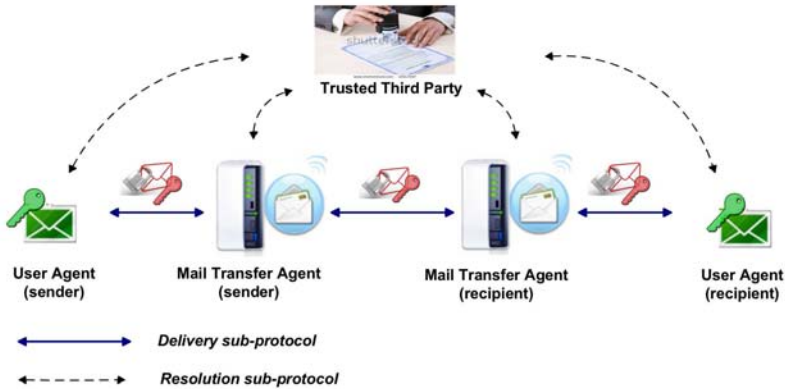
Figure 3.1: CEM Architecture

To simplify the protocol description, we have explicitly omitted the references to some values as the e-mail recipient's address, the identity of entities (we can assume that it is embedded in the signature), a delivery identifier, etc. In some messages, though, we have added a string to help identify the type of message (see table 3.4, e.g., "$LIST$", "$CANCEL$", etc.).

3.3.1 Evidence Generation

During a CEM protocol execution several evidence will be generated. The aim of a CEM protocol is to exchange an encrypted message and its corresponding decryption key, for an evidence proving the recipient is able to read the message. Table 3.1 describes the general nomenclature used along the protocols, and the composition of the evidence and the affidavit (evidence generated by the TTP). Following we list and describe these evidence (see its use and time of generation in the corresponding protocol description, sections 3.4 and 3.5):

K − NRO *Non-Repudiation of Origin* (NRO) plus *Key* ($PK_{UA_S}(k)$). Both values are generated by the UA_S. The NRO is an evidence proving that the UA_S has generated the CEM. The *Key* is not an evidence by itself; it is the symmetric key used to encrypt the CEM ($E_k(C)$). To maintain confidentiality, this value k is encrypted with the UA_R's public key, therefore only the UA_R will be able to open it. It is the only evidence that the TTP is not able to generate.

NRS *Non-Repudiation of Submission.* This evidence is created by the MTA_S when she receives a new CEM submission from the UA_S. It proves that the UA_S has sent a new CEM, and that her MTA has received it.

NRK *Non-Repudiation of Knowledge.* This evidence is generated by the UA_R when he receives a new CEM. It proves that the UA_R knows he has a new CEM.

CEM Protocols Overview

t_D	Deadline for the intervention of the TTP.
$E_k(C)$	The encryption of message C using symmetric key k.
$PK_{UA_R}(k)$	The encryption of symmetric key k using UA_R's public key.
$S_X[m]$	The signature of X on the hash of message m.
NRO	$S_{UA_S}[E_k(C), PK_{UA_R}(k), LTS_{NRO}, t_D]$
NRS	$S_{MTA_S}[E_k(C), LTS_{NRO}, LTS_{NRS}, t_D]$
NRK	$S_{UA_R}[E_k(C), LTS_{NRO}, LTS_{NRK}, t_D]$
NRS'	$S_{MTA_S}[PK_{UA_R}(k), NRO, LTS_{NRS'}]$
NRD	$S_{MTA_R}[E_k(C), PK_{UA_R}(k), LTS_{NRO}, LTS_{NRD}, t_D]$
NRR	$S_{UA_R}[E_k(C), PK_{UA_R}(k), LTS_{NRO}, LTS_{NRR}, t_D]$
$K - NRO$	$S_{UA_S}[PK_{UA_R}(k), NRO]$
NRS_{TTP}	$S_{TTP}[E_k(C), LTS_{NRO}, LTS_{NRS}, t_D]$
NRK_{TTP}	$S_{TTP}[E_k(C), LTS_{NRO}, LTS_{NRK}, t_D]$
NRS'_{TTP}	$S_{TTP}[PK_{UA_R}(k), NRO, LTS_{NRS'}]$
NRD_{TTP}	$S_{TTP}[E_k(C), PK_{UA_R}(k), LTS_{(NRO,NRD)}, t_D]$
NRR_{TTP}	$S_{TTP}[E_k(C), PK_{UA_R}(k), LTS_{(NRO,NRR)}, t_D]$
cem_A	$E_k(C), LTS_{NRO}, t_D$
cem_B	$PK_{UA_R}(k), NRO$
LTS_{EV}	Local TimeStamp for evidence EV $(NRO, NRK, etc.)$.
$N - NRK$	$S_{TTP}[E_k(C), LTS_{NRO}, t_D, "CANCEL"]$ evidence of delivery canceled
$NACK$	$S_X[request, "NACK"]$ signature of X on the $request$ received with a not acknowledgement
$SYNCH_{RS}[X]$	$SYNCH_{RS}$ containing message X

Table 3.1: Evidence and General Notation used along the CEM Protocols

NRS' *Non-Repudiation of Key Submission.* The MTA_S generates this evidence when she receives the $K - NRO$ from the UA_S. It proves the UA_S has delivered the $K - NRO$ to the MTA_S, and that the MTA_S has received it.

NRD *Non-Repudiation of Delivery.* Evidence generated by the MTA_R when he receives the $K - NRO$ from the MTA_S. It proves the $K - NRO$ has arrived to the recipient's inbox (MTA_R).

NRR *Non-Repudiation of Receipt.* Evidence generated by the UA_R, stating that he has received the non-repudiation of origin NRO, and the key to decipher the encrypted CEM, i.e., the $K - NRO$.

SYNCH_RS *Synchronization Response.* It is generated by the MTAs as response to a synchronization request $SYNCH_{RQ}$. It states the time at which users checked

their CEM inbox, and the content of the response received.

Affidavit If the TTP receives enough information it can generate affidavits (evidences signed by the TTP, see table 3.1) of all evidence except the $K - NRO$.

A CEM is represented as a set of information. It is composed of the content message C encrypted with the symmetric key k, a timestamp, the delivery deadline, the key k encrypted with the UA_R's public key and the non-repudiation of origin:

$$\text{CEM} = \{E_k(C), LTS_{NRO}, t_D, PK_{UA_R}(k), NRO\} = cem_A + cem_B$$

3.3.2 Some Remarks on the Delivery of a CEM

Following a similar approach than traditional e-mail, the User Agent (UA) receives the messages and evidence from their MTA by executing a synchronization operation: *LIST* and *FETCH* in IMAP [101], *LIST* and *RETR* in POP3 [102]. This operation is executed on a regular basis (asynchronously), and its objective is to fetch new CEM and evidence from the MTA. These operations will be referred as synchronization requests ($SYNCH_{RQ}$) and synchronization responses ($SYNCH_{RS}$). Even though in figures 3.2 and 3.3 the messages follow a sequence from *message 1* to *message N*, the messages related to synchronization operations are not part of a sequence, because they are executed at the UA's will (as in traditional e-mail). It means that the UA_S may execute the $SYNCH_{RQ}$ many times before getting any evidence.

In a same way, an extension of the Simple Mail Transfer Protocol (SMTP [103]) can be defined to submit and forward CEMs. In order to deliver a CEM, sender and receiver must be registered to enhanced MTAs (offering CEM services), and use enhanced UAs (supporting CEM services). The possible intermediary MTAs (between the MTA_S and MTA_R) can be regular e-mail servers, because they will just forward the messages received.

3.3.3 Behaviour of the TTP

The *resolution* sub-protocols are a set of claims (the participants claim they have not received some evidence) that all participants can make (UAs and MTAs), directed to the TTP, and its corresponding responses. To solve the claims, the TTP is able to contact the mail transfer agents (MTA_S and/or MTA_R), but not the user agents (UA_S and/or UA_R). This decision increases the complexity of the protocol, but from a practical point of view it increases its feasibility. The MTAs are server entities (they offer CEM services), while the UAs are client entities, therefore we can assume that the MTAs will be online always, while the UAs will only be online when they are executing an operation (sending or retrieving messages): they are not servers.

3.3.4 Some Remarks on the Mail Transfer Agent (MTA)

Notice that even the MTAs are untrusted, they are allowed to contact the TTP and request evidence. This may seem a contradiction: if we do not trust the MTAs, then why should we "trust" they are going to contact the TTP when the *delivery* sub-protocol is interrupted? Moreover, from a security point of view, the MTAs are not needed. We can take a protocol designed with direct communication between sender and recipient and encapsulate the information they exchange within an e-mail message, as Blundo *et al.* [86] did. The rationale behind this decision is the same that we followed when we decided which participants can be contacted by the TTP (section 3.3.3): increase the protocol's feasibility, by integrating it with the e-mail service and by protecting the honest MTAs. On one hand, the traditional e-mail service is well-known, users are familiar with it. On the other hand, the objective of an MTA is to deliver messages, therefore when the *delivery* sub-protocol is interrupted, an honest MTA will be able to contact the TTP and finish the delivery (fulfilling its duties).

3.3.5 Delivery Deadline t_D and Local TimeStamps

We establish a delivery deadline t_D that is a value indicating a time limit for the delivery of a CEM. In particular, it states the limit for the TTP's intervention, generating *new evidence*. When the UA_S initiates the *delivery* sub-protocol she chooses a t_D, according to the message content or some other criteria. Therefore, at any time $t < t_D$ during the *delivery* sub-protocol execution, any party can contact the TTP to ask for resolution (sections 3.4.2 and 3.5.3). After $t \geq t_D$ the TTP will only issue evidence coherent with the decisions it took during $t < t_D$, i.e, the TTP will not contact any MTA to solve the requests, it will only use the evidence collected and/or generated before t_D. Knowing t_D, the UA_S and UA_R are responsible of contacting the TTP at $t < t_D$ in case they do not have all their evidence. Otherwise they risk losing fairness.

The Local TimeStamps (LTS_{NRO}, LTS_{NRS}, ...) are values stating the time at which the evidence were created (stated by its generator). In case we need stronger assertions, a TimeStamp Authority should be used to sign the timestamp evidence. Regarding clock synchronization, we will assume that the TTP's clock is the reference.

3.4 Optimistic CEM Protocol with Untrusted Mail Transfer Agents

In the next sections (3.4.1 and 3.4.2) we present the first proposal of an optimistic CEM protocol to consider multiple and untrusted MTAs. As it is described in figure 3.1, the protocol consists of a *delivery* sub-protocol (section 3.4.1), and a set of *resolution* sub-protocols (section 3.4.2). Finally, in section 3.4.3 we review the security requirements presented in section 3.2. Besides the common assumptions explained in section 3.3, the CEM protocol with untrusted MTAs assumes the following:

$M_x\{Y\}$: Message number x delivers evidence Y
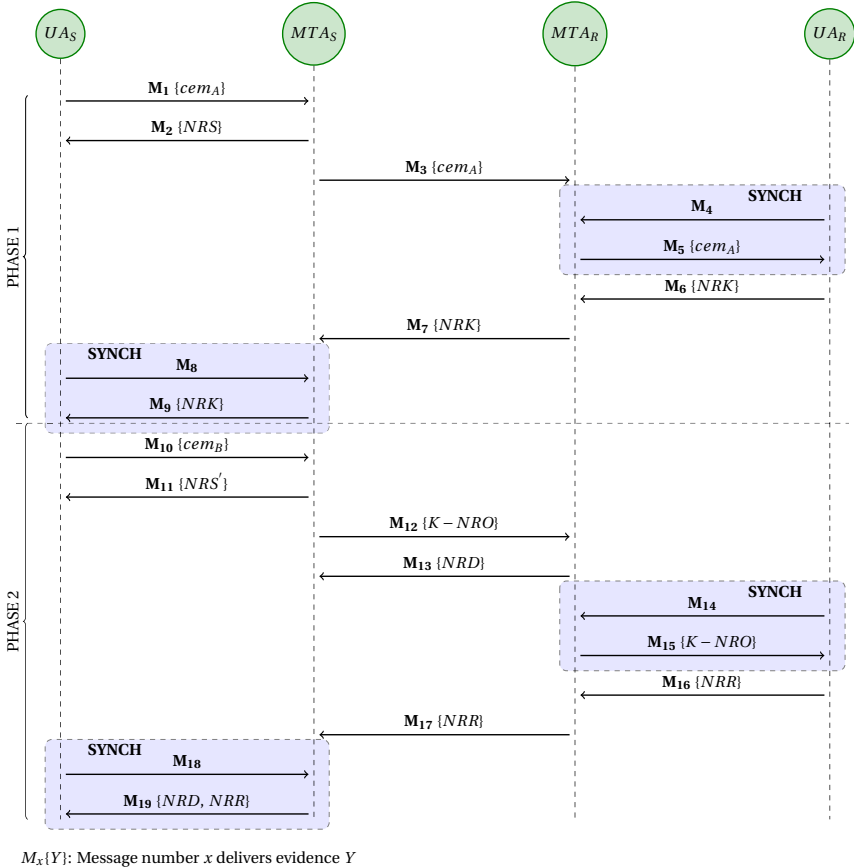
Figure 3.2: CEM Delivery Sub-Protocol (Untrusted Mail Transfer Agent (MTA))

- The MTA_S, MTA_R and the possible MTAs intermediaries are untrusted entities. They can misbehave and collude with other parties.

3.4.1 Delivery Sub-Protocol

In figure 3.2 and table 3.2 we describe the protocol execution flow and the content of the messages. A CEM is delivered in two phases. In the first phase (messages 1 to 9), the UA_S and the UA_R exchange the encrypted CEM for evidence proving the UA_R has received it, i.e., he knows he has received a new CEM. In the second phase (messages 10 to 19), they exchange the key to decrypt the CEM content and the non-repudiation of origin for evidence proving the UA_R has received it, i.e., he is able to open the CEM and knows who has sent it. The rationale behind these two phases is to allow the UA_S to send a CEM using untrusted intermediaries, the MTAs, and to

Delivery Sub-Protocol

$M_1 = m_1, S_{UA_S}[m_1]$ — where $m_1 = E_k(C), LTS_{NRO}, t_D$

$M_2 = m_2, S_{MTA_S}[m_2]$ — where $m_2 = LTS_{NRS}, NRS$

$M_3 = m_3, S_{MTA_S}[m_3]$ — where $m_3 = E_k(C), LTS_{NRO}, t_D$

$M_4 = m_4, S_{UA_R}[m_4]$ — where $m_4 = $ "LIST", $LTS_{RQ} = SYNCH_{RQ}$

$M_5 = m_5, S_{MTA_R}[m_5]$ — where $m_5 = LTS_{(RQ,RS,NRO)}, E_k(C), t_D$

$M_6 = m_6, S_{UA_R}[m_6]$ — where $m_6 = LTS_{NRK}, NRK$

$M_7 = m_7, S_{MTA_R}[m_7]$ — where $m_7 = LTS_{NRK}, NRK$

$M_8 = m_8, S_{UA_S}[m_8]$ — where $m_8 = SYNCH_{RQ}$

$M_9 = m_9, S_{MTA_S}[m_9]$ — where $m_9 = LTS_{NRK}, NRK$

$M_{10} = m_{10}, S_{UA_S}[m_{10}]$ — where $m_{10} = PK_{UA_R}(k), NRO$

$M_{11} = m_{11}, S_{MTA_S}[m_{11}]$ — where $m_{11} = LTS_{NRS'}, NRS'$

$M_{12} = m_{12}, S_{MTA_S}[m_{12}]$ — where $m_{12} = m_{10}, S_{UA_S}[m_{10}]$

$M_{13} = m_{13}, S_{MTA_R}[m_{13}]$ — where $m_{13} = LTS_{NRD}, NRD$

$M_{14} = m_{14}, S_{UA_R}[m_{14}]$ — where $m_{14} = SYNCH_{RQ}$

$M_{15} = m_{15}, S_{MTA_R}[m_{15}]$ — where $m_{15} = LTS_{(RQ,RS)}, m_{10}, S_{UA_S}[m_{10}]$

$M_{16} = m_{16}, S_{UA_R}[m_{16}]$ — where $m_{16} = LTS_{NRR}, NRR$

$M_{17} = m_{17}, S_{MTA_R}[m_{17}]$ — where $m_{17} = LTS_{NRR}, NRR$

$M_{18} = m_{18}, S_{UA_S}[m_{18}]$ — where $m_{18} = SYNCH_{RQ}$

$M_{19} = m_{19}, S_{MTA_S}[m_{19}]$ — where $m_{19} = LTS_{(RQ,RS,NRD,NRR)}, NRD, NRR$

Table 3.2: Protocol Notation (Untrusted Mail Transfer Agents (MTAs))

fulfil the non-selective receipt requirement: once a recipient knows he has received a CEM (phase 1), he cannot prevent its delivery.

Delivery Execution

The *delivery* sub-protocol starts when the UA_S sends the first part of the CEM, cem_A, to the MTA_S, in exchange for the NRS (M_2). In the next steps the MTA_S and the MTA_R exchange cem_A, for the NRK. But before the MTA_R can deliver the NRK to the MTA_S, the UA_R must execute a $SYNCH_{RQ}$ (M_4) operation and receive a new CEM as a response (M_5). After receiving the response, the UA_R will send the NRK (M_6) to the MTA_R. Now the MTA_R can deliver the NRK to the MTA_S (M_7). Before continuing, the UA_S has to execute a $SYNCH_{RQ}$ operation to download the NRK (M_8 and M_9).

The second phase of the protocol starts when the UA_S sends the second part of the CEM, cem_B, to the MTA_S in exchange for the NRS' (M_{10} and M_{11}). In the next step, the MTA_S and MTA_R exchange the $K - NRO$ for the NRD (M_{12} and M_{13}). To continue with the sub-protocol execution, the UA_R has to execute a $SYNCH_{RQ}$ (M_{14}) operation to obtain the $K - NRO$ as a response (M_{15}). Finally, the UA_R delivers the NRR to the MTA_R (M_{16}), the MTA_R sends it to the MTA_S (M_{17}) and the UA_S obtains it as a response (M_{18}) to a $SYNCH_{RQ}$ message (M_{19}) from the UA_S.

Delivery Cancelation

As we have explained, the delivery is done in two phases: first we exchange the cem_A for the NRK, and later we exchange the cem_B and $K - NRO$ for the NRR. During the first phase of the protocol execution and before the NRK (or NRK_{TTP}) is generated, the TTP can cancel the delivery by issuing an $N - NRK$ (Negative Non-Repudiation of Knowledge, evidence of cancelation, table 3.1). Once the TTP has issued an $N - NRK$, it will answer with it to any claim regarding the same protocol execution. The NRK (or NRK_{TTP}) proves that the UA_R is aware he has a pending delivery, therefore after it has been issued, the protocol cannot be canceled, except when the UA_S misbehaves, i.e., she does not deliver the $K - NRO$ (see $K - NRO$ resolution in section 3.4.2).

3.4.2 Resolution Sub-Protocol

The communication between UAs and MTAs, and between MTAs is done through Internet, an unreliable channel (i.e., messages can be lost or delayed). Moreover, the MTAs are untrusted, meaning that they can misbehave. Therefore, the *delivery* sub-protocol execution may be interrupted by a communication problem or the intervention of a malicious entity. In this situation, either sender or recipient can find themselves in a disadvantageous position: the UA_S may deliver the $K - NRO$ and not receive the NRR, or the UA_R may not receive the $K - NRO$. To avoid these unfair situations, we need the *resolution* sub-protocols.

The *resolution* sub-protocols are a set of claims (they claim they have not received some evidence) each participant can make, and the corresponding TTP's response. In table 3.3 we have the set of different requests the TTP admits. As we can see, each claim is related to one kind of evidence (NRS, NRK, etc.), and each participant is only allowed to request evidence related to his role in the CEM delivery, e.g., the UA_S is the only one allowed to demand the NRS and the NRS'. A special case are the NRR and NRK resolution requests, which have different content depending on the participant who demands it.

NRS Resolution

Only the UA_S can send the NRS resolution request. If the TTP has previously received a request of later evidence (NRK, NRS', NRD, NRR, $K - NRO$), it will issue an NRS_{TTP} (or an NRS, if available), and it will deliver it to the UA_S, along with the evidence obtained on previous requests. Otherwise, the TTP will contact the MTA_S sending a message equivalent in content to M_1 (table 3.2): m_1, $S_{TTP}[m_1]$. If the MTA_S answers with the NRS, the TTP will deliver it to the UA_S. In case the MTA_S does not answer, the TTP will contact the MTA_R sending the same message (notice $m_1 = m_3$ in table 3.2). If the MTA_R has some delivery evidence (NRK, NRD, NRR), the TTP will deliver it to the UA_S, along with an affidavit NRS_{TTP}. If the MTA_R answers with a $NACK$, it means he does not have any information on the CEM delivery or he has received the CEM but he does not have the NRK yet. Therefore, the TTP will send a $N - NRK$ to the MTA_S, MTA_R (if available) and the UA_S, canceling

Claim Request		Who can claim it
$\mathbf{M_{NRS}}$	$= m_{NRS}, S_X[m_{NRS}]$	UA_S
$\mathbf{M_{NRK}}$	$= m_{NRK}, S_X[m_{NRK}]$	UA_S, MTA_S
$\star$	$= m_{NRK'}, S_X[m_{NRK'}]$	MTA_R
$\mathbf{M_{NRS'}}$	$= m_{NRS'}, S_X[m_{NRS'}]$	UA_S
$\mathbf{M_{NRD}}$	$= m_{NRD}, S_X[m_{NRD}]$	UA_S, MTA_S
$\mathbf{M_{NRR}}$	$= m_{NRR}, S_X[m_{NRR}]$	UA_S, MTA_S
$\star$	$= m_{NRR'}, S_X[m_{NRR'}]$	MTA_R
$\mathbf{M_{K-NRO}}$	$= m_{KNRO}, S_X[m_{KNRO}]$	MTA_R, UA_R
$\mathbf{M_{SYNCH_{RS}}}$	$= m_{SRS}, S_X[m_{SRS}]$	UA_R

m_{NRS}	$= cem_A$
m_{NRK}	$= cem_A, NRS$
$m_{NRK'}$	$= SYNCH_{RQ}, SYNCH_{RS}[cem_A]$
$m_{NRS'}$	$= cem_A, cem_B, NRK, K - NRO$
m_{NRD}	$= cem_A, cem_B, NRK, NRS', K - NRO$
m_{NRR}	$= cem_A, cem_B, NRK, NRD, K - NRO$
$m_{NRR'}$	$= SYNCH_{RQ}, SYNCH_{RS}[cem_B, K - NRO]$
m_{KNRO}	$= cem_A, NRK$
m_{SRS}	$=$ "LIST", LTS_{RQ}

$\star$ The M_{NRK} and M_{NRR} depend on who sends it.

Table 3.3: Resolution Sub-Protocol Requests (Untrusted Mail Transfer Agents (MTAs))

the delivery. In case the MTA_R does not answer, the TTP will also cancel the delivery issuing the $N-NRK$. From now on, the TTP will answer with a $N-NRK$ to any request regarding this CEM delivery.

NRK Resolution

If the UA_S sends the NRK resolution request, the following can happen. If any party has previously contacted requesting the NRK, the TTP will deliver the same evidence. If a previous request has demanded later evidence (NRS', NRD, NRR or $K-NRO$), the TTP will already have the NRK (see table 3.3), thus, it will deliver it along with other later evidence obtained during previous requests. Otherwise, the TTP will contact the MTA_S sending a message equivalent in content to M_1 plus the NRS: m_1, NRS, $S_{TTP}[m_1, NRS]$. If the MTA_S answers with some evidence (NRK and/or NRD and/or NRR), the TTP will deliver it to the UA_S, otherwise ($NACK$

or no answer) the TTP will contact the MTA_R sending the same message it sent to the MTA_S. If the MTA_R answers with some evidence (NRK and/or NRD and/or NRR), the TTP will deliver it to the UA_S, otherwise ($NACK$ or no answer) the TTP will cancel the execution and deliver an $N - NRK$ to the UA_S, and both MTA (if available). Notice that if the MTA_S is the one who requests the NRK, the TTP can skip contacting her, and go directly to the MTA_R.

When the MTA_R is the one who sends the request (table 3.3), the following can happen. If the UA_R has previously sent a $SYNCH_{RS}$ request including a $SYNCH_{RQ}$ with the same timestamp than the one included in the MTA_R's NRK request, and the TTP has sent a $NACK$ as response (the TTP has not been able to obtain the $SYNCH_{RS}$), the TTP will also send a $NACK$ to the MTA_R (to maintain coherence). Otherwise, the TTP will answer by issuing a non repudiation of knowledge affidavit, NRK_{TTP}. Note that $SYNCH_{RQ}$ and $SYNCH_{RS}$ are linked by their timestamp values.

NRS$'$ Resolution

As with the NRS, only the UA_S can send the NRS' resolution request. If the TTP has received a previous request demanding later evidence (NRD, NRR), it will issue a NRS'_{TTP} (or a NRS', if available), and deliver it to the UA_S, along with the evidence obtained on previous requests. Otherwise, the TTP will contact the MTA_S sending her a message equivalent to M_{10}: $M_{10}, S_{TTP}[M_{10}]$ (M_{10} contains the $K - NRO$). If the MTA_S answers with the NRS', the TTP will deliver it to the UA_S. In case the MTA_S does not answer, the TTP will deliver an affidavit to the UA_S, NRS'_{TTP}.

NRD Resolution

The UA_S and the MTA_S can send the NRD request. If any party has previously contacted the TTP and demanded the NRD, the TTP will deliver the same evidence, either the NRD or NRD_{TTP}. Otherwise, the TTP will send a message equivalent to M_{10} plus m_{11} (which contains the NRS') to the MTA_S: $M_{10}, m_{11}, S_{TTP}[M_{10}, m_{11}]$. If the MTA_S answers with the NRD, the TTP will deliver it as a response to the UA_S. But if the MTA_S sends a $NACK$ or fails to answer, the TTP will contact the MTA_R sending a message equivalent to M_{12}: $m_{12}, S_{TTP}[m_{12}]$. If the MTA_R answers with the NRD, the TTP will deliver it as a response (either to the UA_S or the MTA_S). But if the MTA_R sends a $NACK$ or fails to answer, the TTP will deliver an affidavit, the NRD_{TTP}. If the MTA_S is the requesting party, the TTP will follow the same execution flow but without contacting the MTA_S.

NRR Resolution

If the UA_S sends the NRR resolution request, the following can happen. If some other party has previously contacted the TTP and claimed the NRR, the TTP will send the same evidence to the requesting party. Otherwise, the TTP will send a message equivalent in content to the MTA_S, M_{13}: $m_{13}, S_{TTP}[m_{13}]$. If the MTA_S answers with the NRR, the TTP will deliver it to the UA_S. But if the MTA_S sends

a $NACK$ or fails to answer, the TTP will contact the MTA_R sending the same information it sent to the MTA_S. If the MTA_R answers with the NRR, the TTP will deliver it to the UA_S. But if the MTA_R sends a $NACK$ or fails to answer, the TTP will deliver an affidavit: the NRR_{TTP} (either to the UA_S or the MTA_S). If it is the MTA_S who claims the NRR, the TTP will skip contacting the MTA_S and it will contact the MTA_R directly.

When the MTA_R is the one who sends the NRR resolution request (table 3.3), the following can happen. If the UA_R has previously claimed he has not received the $SYNCH_{RS}$, and the TTP has sent a $NACK$ as response (the TTP has not been able to obtain the $SYNCH_{RS}$), the TTP will also send a $NACK$ to the MTA_R (to maintain coherence). Otherwise, the TTP will answer by issuing an affidavit, NRR_{TTP}. Remember that $SYNCH_{RQ}$ and $SYNCH_{RS}$ are linked by their timestamp values.

K − NRO Resolution

The MTA_R and the UA_R can claim the $K - NRO$. If the UA_S or MTA_S have previously contacted the TTP requesting the NRS', NRD or NRR, the TTP will already have the $K - NRO$ (see table 3.3), thus, the TTP will deliver it to the requesting party (either the MTA_R or the UA_R). Otherwise, the TTP will send a message equivalent in content to M_7 to the MTA_R: m_7, $S_{TTP}[m_7]$. If the MTA_R answers providing the $K - NRO$, the TTP will deliver it. In case the MTA_R answers with a $NACK$ or does not answer (the server is down or does not want to answer), the TTP will follow the same procedure with the MTA_S: it will send m_7, $S_{TTP}[m_7]$. If the MTA_S answers with the $K - NRO$, the TTP will deliver it (either to the UA_R or the MTA_R), otherwise, the TTP will deliver a $NACK$.

Note that following the $K - NRO$ claim, the UA_R can find himself in a situation where he does not receive the key neither from the MTA_R nor from the TTP (it answers with a $NACK$). But since t_D states the time limit for the protocol execution, if the UA_R receives a $NACK$ from the TTP after t_D, and neither the MTA_S nor MTA_R do have the $K - NRO$ (or they claim not having it), the UA_R can assume the delivery has been canceled. The UA_S has no means to obtain the NRR, neither from the TTP (it will not issue an affidavit after t_D), nor from the MTA_R (the UA_R will not deliver the NRR before receiving the $K - NRO$).

SYNCH_RS Resolution

Only the UA_R can request this evidence. If the MTA_R has previously contacted the TTP claiming he has not received the NRK, the TTP will already have the corresponding $SYNCH_{RS}$ ($SYNCH_{RQ}$ and $SYNCH_{RS}$ are linked through the timestamp), thus, the TTP will deliver it to the UA_R. Otherwise, the TTP will resend the $SYNCH_{RQ}$ (signed by the TTP) to the MTA_R. If the MTA_R answers with M_5, the $SYNCH_{RS}$, the TTP will deliver it to the UA_R. In case the MTA_R does not answer (the server is down or does not want to answer), the TTP will deliver a $NACK$ to the UA_R. The UA_R will try again sending a new $SYNCH_{RQ}$ to the MTA_R, and/or the TTP.

The UA is a software entity that can have two roles, sender and recipient. The objective of the $SYNCH_{RS}$ claim is to protect the UA_R (UA acting as recipient) against possible protocol malfunctions due to network errors or malicious users, when receiving a CEM. With the $SYNCH_{RS}$ claim, the UA_R can obtain evidence from the TTP that he contacted the MTA_R but did not receive any evidence of a new CEM (the MTA_R does not answer or the message is lost). And in case the MTA_R answers with an empty $SYNCH_{RS}$, the UA_R can provide this evidence to prove the MTA_R misbehaviour. Imagine the situation where the UA_R sends a $SYNCH_{RQ}$ and the MTA_R answers with an empty $SYNCH_{RS}$, meaning there are no new CEMs, but in fact, there is a new CEM. Then, the MTA_R contacts the TTP asking for the NRK (section 3.4.2), obtaining an affidavit. From here, the MTA_R continues the *delivery* sub-protocol execution without relaying the messages to the UA_R. Therefore, the UA_S will receive the NRK_{TTP} and will continue with the second phase. Once the UA_S receives the NRD, she will be able to claim the NRR and obtain it from the TTP (section 3.4.2). Thus, at the end of the protocol execution, the UA_S will have evidence that the UA_R has received the CEM and the corresponding key to decipher it, the NRR, but the UA_R will not have any evidence of the existence of such a CEM: he may lose fairness. With this claim he will be able to prove the MTA_R's misbehaviour.

3.4.3 Security Analysis

In this section we will review the accomplishment of the requirements we presented in section 3.1, to prove that our protocol meets them.

CLAIM 3.1: **CEM Effectiveness**: *If sender and recipient behave according to the protocol and there are no communication errors, the TTP will not intervene.*

PROOF. Following the *delivery* sub-protocol execution (see figure 3.2 and section 3.4.1), we can see that the TTP does not intervene. Therefore we can affirm that if every party involved behaves correctly (and there are no network errors), the UA_S will obtain her evidence (NRS, NRK, NRS', NRD and NRR) and the UA_R will obtain his (cem_A, cem_B and $K-NRO$) without TTP's intervention, meeting the effectiveness requirement.

CLAIM 3.2: **CEM Fairness**: *The UA_S will receive her evidence if the UA_R can obtain his. And the UA_R will receive his evidence if the UA_S can obtain hers.*

PROOF. Assuming an honest sender UA_S, the recipient UA_R can obtain the $K-NRO$ from the MTA_R or the TTP. Following the *delivery* sub-protocol (see figure 3.2), the UA_S must have received the NRK or NRK_{TTP} in order to deliver the $K-NRO$ (M_{10}). Therefore, if the UA_S does not receive the NRR (either because of some party misbehaviour or a lost message), she can contact the TTP and obtain it. In a "worst case scenario", where both MTAs are misbehaving or having communication problems (not answering), the UA_S can contact the TTP and obtain the NRS'_{TTP} first, then the NRD_{TTP} and finally the NRR_{TTP} (see section 3.4.2).

Assuming an honest recipient UA_R, the sender UA_S can obtain evidence of receipt (NRR or NRR_{TTP}), from the MTA_S or the TTP. If the UA_S obtains the NRR, following the protocol (see figure 3.2) it means the UA_R has already received the $K - NRO$ (M_{15} and M_{16}), even when the TTP intervenes (the TTP obtains the NRR from the MTA_R, see NRR resolution in section 3.4.2). And if the UA_S obtains the NRR_{TTP}, it means the TTP already has the $K - NRO$ (see table 3.3), therefore the UA_R can contact the TTP and obtain it.

As a conclusion, we can affirm that our proposal meets the fairness requirement.

CLAIM 3.3: **CEM Timeliness**: *The delivery of a CEM will finish in a finite amount of time, maintaining fairness for all honest participants.*

PROOF. The delivery deadline t_D establishes a time limit for the TTP's intervention. After t_D, the TTP will not generate new evidence (affidavit) neither contact the MTAs to solve a claim, it will answer according to the evidence received and generated before t_D. Therefore, at any time $t < t_D$ any party can contact the TTP and ask for resolution (section 3.4.2). In particular, before the NRK is generated, the delivery can be canceled (the TTP issues a $CANCEL$ token, an $N - NRK$). After the NRK is generated, the TTP can provide the evidence needed by the UA_S and the UA_R to consider the message delivered (obtaining it through the MTAs or generating an affidavit).

Therefore we can affirm that, either by TTP's intervention, the correct execution of the *delivery* sub-protocol or because of the deadline t_D, the delivery can finish in a finite amount of time $t \leq t_D$, maintaining fairness for all parties: the protocol meets the timeliness requirement.

CLAIM 3.4: **CEM Verifiability of TTP**: *Whatever the TTP's behaviour (honest or dishonest), both UAs maintain fairness.*

PROOF. The TTP is a semi-trusted entity (as defined by Franklin and Reiter [100]), i.e., it can misbehave but not collude with other entities. To lose fairness, one participant (either UA_S or the UA_R) should have evidence of delivery and the other one evidence of delivery canceled or failed. In particular we can have two situations.

In the first situation, the UA_S sends the $K - NRO$ to the MTA_S (M_{10}, figure 3.2) but it never reaches its destination (network failure), and therefore the MTA_S will not send the NRS'. Then the UA_S contacts the TTP claiming she has not received the NRS' and the TTP answers with a NRS'_{TTP} without contacting the MTA_S. Later, the UA_S contacts the TTP and claims the NRD and the NRR, and again, the TTP issues an affidavit (NRD_{TTP} and NRR_{TTP}) without contacting the MTAs (the $K - NRO$ has never reached the MTA_S). At this point the TTP has the $K - NRO$, therefore the UA_R should be able to receive it after contacting the TTP. But every time the UA_R contacts the TTP before t_D, it sends a $NACK$, and when the UA_R contacts again, after t_D it delivers a $NACK$ too. Which leaves us in a situation where the UA_S has evidence of delivery and the UA_R evidence of failed delivery. But confronting the evidence, the UAs will be able to prove the TTP's misbehaviour. The NRR_{TTP} proves that the TTP had the $K - NRO$, and therefore the TTP should not have sent an $NACK$ after t_D.

In the second situation, due to a network failure, the message M_1 (figure 3.2) never reaches the MTA_S. The UA_S sends an NRS request to the TTP and it issues an affidavit without contacting the MTA_S. Since the MTA_S has not received the CEM (cem_A), the UA_S will not receive the NRK and she will claim it to the TTP, and, again, it will issue an affidavit without contacting the MTAs. From here, the execution follows the previous paragraph (the UA_S claims the NRS', NRD, NRR and receives an affidavit from the TTP), with the exception that the UA_R does not know he has received a CEM, therefore he will not contact the TTP. At the end of the protocol execution, the UA_S will have proof of delivery while the UA_R has no proof at all. When confronted, the UA_S will be able to prove the TTP's misbehaviour by asking evidence of its resolutions, in particular the NRK resolution. To deliver a NRK_{TTP}, the TTP must have previously received a resolution request from the MTA_R, claiming the NRK.

The TTP's misbehaviour can be proved, therefore we can affirm that our protocol meets the verifiability of TTP requirement.

CLAIM 3.5: **CEM Confidentiality**: *The content of a CEM is only available to the UA_S and the UA_R.*

PROOF. The UA_S sends the content of the certified message C encrypted with a symmetric key k, and this key encrypted with PK_{UA_R}, the recipient's public key. Therefore, the content of the message is only known to the sender UA_S and the recipient UA_R, who is the only capable of obtaining the symmetric key k. Thus we can affirm that the protocol meets the confidentiality requirement.

CLAIM 3.6: **CEM Non-Selective Receipt**: *Once the NRK is generated, either by the UA_R or the TTP (NRK_{TTP}), the delivery cannot be canceled by the recipient.*

PROOF. The UA_R sends the NRK after receiving a $SYNCH_{RS}$, stating that he knows he has a pending CEM delivery. If by any chance the UA_R tries to reject the CEM, by not sending the NRK, the MTA_R can contact the TTP and obtain an affidavit from the TTP, the NRK_{TTP}. Thus, the UA_R has no chance to reject an incoming CEM. Therefore, we can affirm that our protocol meets the Non-Selective Receipt requirement.

CLAIM 3.7: **CEM Non-Repudiation**: *None of the participants in the delivery of a CEM can deny having participated, neither can they be excluded from it.*

PROOF. Along the protocol execution, evidence is generated as non-repudiation proof for various services. In particular, the protocol offers:

- *NRO*, Non-Repudiation of Origin. When the UA_S issues a message, she includes her signature on the message as proof she is the originator. If the UA_S tries to deny having originated it, the UA_R can prove it using the UA_S's signature on the message (this evidence is part of the $K - NRO$).

- *NRS*, Non-Repudiation of Submission. When the UA_S delivers a CEM to her MTA_S, she receives the MTA_S signature on the message, as proof she has sent it. Thus, the MTA_S cannot deny having received the CEM.

- NRK, Non-Repudiation of Knowledge. When the UA_R receives a synchronization response, $SYNCH_{RS}$, including new CEM, he sends his signature to the MTA_R as proof that he knows there is a pending delivery. This avoids the UA_R to deny knowing he had received a CEM.

- NRS', Non-Repudiation of Key Submission. When the UA_S delivers the $K - NRO$ to her MTA_S, she receives the MTA_S signature on the message, as proof she has sent the key and NRO. Thus, the MTA_S cannot deny having received the $K - NRO$ evidence.

- NRD, Non-Repudiation of Delivery. When the MTA_R receives the key from the MTA_S, he sends his signature on the message, as proof the $K - NRO$ is in the recipient's inbox. Thus, the MTA_R cannot deny having received it.

- NRR, Non-Repudiation of Receipt. When the UA_R receives a $SYNCH_{RS}$ including the $K - NRO$, he sends his signature to the MTA_R as proof of receipt. This prevents the UA_R to deny having received a CEM and its corresponding key to decipher it.

3.5 Optimistic CEM Protocol with Trusted Mail Transfer Agent

In order to reduce the complexity of the protocol with untrusted MTAs, and following our goal of designing feasible protocols, in this section we present another CEM protocol following the next assumption (besides the common assumptions, section 3.3):

- Each user trusts his own MTA (the UA_S trusts the MTA_S, and the UA_R trusts the MTA_R). The other MTAs participating in the protocol are considered to be untrusted for that user.

It may seem a strong assumption, but as Ateniese *et al.* [97] explain in their paper, either the UA_S chooses his own MTA or there is a business relation between them, in which the MTA agrees to provide CEM services to the UA_S or both. The UA_R, on the other hand, has the same relation with his own MTA and trusts the TTP (which is semi-trusted and verifiable) to solve any problem related to the delivery.

Figure 3.1 sketches the entities and sub-protocols involved in the proposal. As we can see, the CEM protocol consists of a *delivery* sub-protocol (explained in section 3.5.2) and a set of *resolution* sub-protocols (explained in section 3.5.3). Finally, in section 3.5.4 a review of the security requirements is presented.

3.5.1 Motivating Example

In our optimistic proposal, the MTA where a user is registered with is trusted by this user, while the other possible participant MTAs are untrusted entities. To illustrate the practical use of our proposal, we present two example scenarios where the protocol could be applied.

The first scenario that we present is a B2B environment. Alice, who works as sales representative at ACME, must notify Bob, Bit inc.'s Chief Executive Officer, that there will be a modification in the Service Level Agreement (SLA) both companies signed. The notification gives Bit inc. 3 months to decide whether to agree or not with the new SLA terms. If there is no communication from Bit inc. in this period of time, ACME would take for granted that Bit inc. accepted the new terms and the new SLA will take effect automatically. To notify Bob, Alice sends a CEM using ACME's certified e-mail service. Since Alice works at ACME, she can trust the MTA_{ACME}, but not the $MTA_{Bitinc.}$. It will make no sense for the MTA_{ACME} to act against her, it would be against its own interests. In a same way, Bob can trust the $MTA_{Bitinc.}$ when sending or receiving certified e-mails.

Our second example of scenario where it could be applied is within the public administration. Arya, who works in the public administration (PA), has to notify Bronn that he has to pay his taxes before the end of the year or he will receive a fine. To send the notification, Arya sends a CEM using the PA's CEM services. Since she is sending a notification on behalf of the public administration, she can trust that the MTA_{PA} will not act against her, but she cannot trust the possible intermediary MTAs nor Bronn's MTA. In a same way, Bronn who has chosen his own MTA, can trust his MTA will not act against him. The objective/interest of Bronn's MTA is to provide users with CEM services, but to do so it needs the user's trust. Therefore, acting again its users could jeopardize its own interests, losing its users trust.

In both examples, the users and their MTA (UA_S and MTA_S, UA_R and MTA_R) share a common interest/objective: either they belong to the same company or administration or there is a commercial contract binding both entities. We can generalize these examples and affirm that the CEM protocol we present in this section can be applied to any scenario where the users and their MTAs share a common interest.

3.5.2 Delivery Sub-Protocol

In figure 3.3 we can see the *delivery* sub-protocol flow, the messages exchanged and the order in which they are exchanged. In tables 3.1 and 3.4 we can see the general nomenclature and the composition of these messages and evidence. As shown in figure 3.3, if every party involved behaves correctly, the TTP will not intervene in the message delivery (in fact, it does not appear in this figure).

Notice that there is no reference to a non-repudiation of receipt (NRR) evidence neither in figure 3.3 nor in table 3.4, but we listed it as a requirement (section 3.1). Instead, we can see an evidence named Non-Repudiation of Knowledge, NRK, which proves that the recipient knows he has received a CEM, even though he does not have access to a necessary key yet. But if we can also prove that he has access to this key, we can consider it delivered. Therefore, our evidence of receipt will be the combination of two different evidence: evidence of knowledge, proof that the recipient knows he has a CEM in his inbox, and evidence of delivery, proof that the message decryption key is in the recipient's inbox, i.e., the recipient has access to it ($NRR \equiv NRK + NRD$).
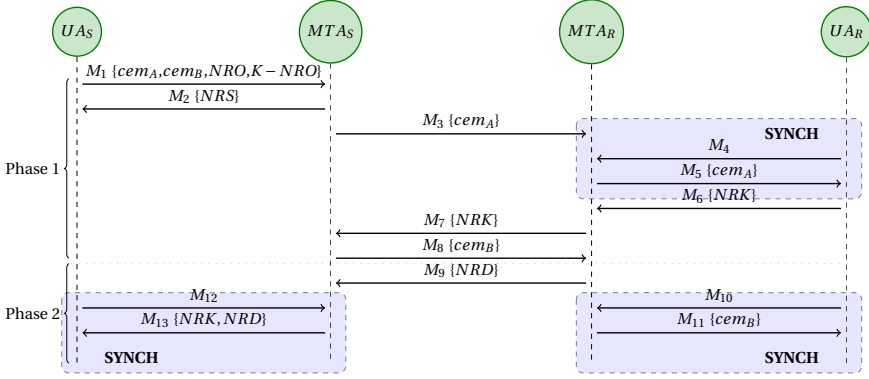
Figure 3.3: CEM Delivery Sub-Protocol (Trusted Mail Transfer Agent (MTA))

Delivery Sub-Protocol

$M_1 = m_1, S_{UA_S}[m_1]$	where $m_1 = E_k(C), PK_{UA_R}(k), LTS_{NRO}, t_D, NRO, K-NRO$
$M_2 = m_2, S_{MTA_S}[m_2]$	where $m_2 = LTS_{NRS}, NRS$
$M_3 = m_3, S_{MTA_S}[m_3]$	where $m_3 = E_k(C), LTS_{NRO}, t_D$
$M_4 = m_4, S_{UA_R}[m_4]$	where $m_4 =$ "LIST", LTS_{RQ}
$M_5 = m_5, S_{MTA_R}[m_5]$	where $m_5 = LTS_{RQ}, LTS_{RS}, E_k(C), LTS_{NRO}, t_D$
$M_6 = m_6, S_{UA_R}[m_6]$	where $m_6 = LTS_{NRK}, NRK$
$M_7 = m_7, S_{MTA_R}[m_7]$	where $m_7 = LTS_{NRK}, NRK$
$M_8 = m_8, S_{MTA_S}[m_8]$	where $m_8 = PK_{UA_R}(k), NRO, K-NRO$
$M_9 = m_9, S_{MTA_R}[m_9]$	where $m_9 = LTS_{NRD}, NRD$
$M_{10}= m_{10}, S_{UA_R}[m_{10}]$	where $m_{10} =$ "LIST", LTS_{RQ}
$M_{11}= m_{11}, S_{MTA_R}[m_{11}]$	where $m_{11} = LTS_{RQ}, LTS_{RS}, PK_{UA_R}(k), NRO, K-NRO$
$M_{12}= m_{12}, S_{UA_S}[m_{12}]$	where $m_{12} =$ "LIST", LTS_{RQ}
$M_{13}= m_{13}, S_{MTA_S}[m_{13}]$	where $m_{13} = LTS_{RQ}, LTS_{RS},$
	$\qquad LTS_{NRK}, LTS_{NRD}, NRK, NRD$

Table 3.4: Protocol Notation (Trusted Mail Transfer Agent (MTA))

The CEM delivery is done in two phases: Firstly, the MTA_S exchanges the cem_A (table 3.1) for the non-repudiation of knowledge, NRK. Secondly, the MTA_S exchanges the cem_B for the non-repudiation of delivery NRD. The first phase corresponds to the *messages 1* to *7* (M_1 to M_7), and the second phase to the *messages 8* to *13* (M_8 to M_{13}). In the following paragraphs we describe the execution of the *delivery* sub-protocol, where *message x* is a reference to M_x from table 3.4.

The *delivery* sub-protocol starts when the UA_R sends *message 1* (M_1 from table 3.4) to the MTA_S including the encrypted message $E_k(C)$, the key to decrypt it (the key is encrypted with the UA_R public key), a local timestamp value (LTS_{NRO}) indicating the time of creation, the deadline t_D, and the non-repudiation of origin NRO. The MTA_S answers with *message 2*: a local timestamp, stating the time of submission LTS_{NRS}, and the non-repudiation of submission NRS. Then the MTA_S sends *message 3*: the encrypted message $E_k(C)$, the timestamp received LTS_{NRO} and the deadline t_D to the MTA_R, and waits for the NRK and LTS_{NRK}, *message 7*, which has to be generated by the UA_R. The UA_R receives information on new e-mails and/or evidence received by synchronizing with his MTA_R. The UA_R will send a $SYNCH_{RQ}$ command to the MTA_R in order to receive information of new deliveries and evidence of pending ones, *message 4*. As a response, the MTA_R will deliver the new CEM received and/or new evidence of pending deliveries. If the response contains new deliveries, the UA_R will send the NRK to the MTA_R, *message 6*, and the MTA_R will forward it to the MTA_S, *message 7*.

After receiving the NRK, *message 7*, the MTA_S will start the second phase of the protocol, exchanging the cem_B, *message 8*, for the NRD, *message 9*. The second phase will end after the UA_S has received both the NRK and NRD evidence, *messages 12* and *13*, and the UA_R receives the cem_B, *messages 10 and 11*.

3.5.3 Resolution Sub-Protocols

To solve the resolution requests the TTP may need to contact the mail transfer agents (MTA_S and/or MTA_R) to obtain the evidence claimed. In some occasions, the TTP will not be able to obtain such evidence, but it will have enough information to generate an affidavit (table 3.1). It may also occur that the TTP cannot obtain the evidence neither issue an affidavit, then the TTP will send a $NACK$ (table 3.1) as a response. Similarly, the MTAs will answer with a $NACK$ when they do not have the evidence the TTP is requesting (or they claim they do not have it).

Before the NRK (or NRK_{TTP}) is generated (*message 6*), the TTP can cancel the delivery by issuing a $N - NRK$ (table 3.4). After the NRK is generated, the TTP will assure the execution is carried out, by delivering the necessary evidence, the NRD or NRD_{TTP}, and the $K - NRO$ (the key and the NRO).

The *resolution* sub-protocols are a set of requests where the participants claim they did not receive some evidence. In table 3.5 we have the set of different requests the TTP admits. As we can see, each claim is related to one kind of evidence (NRS, NRK, etc.), and each participant is only allowed to request evidence related to his role in the CEM delivery, e.g., the UA_S is the only one allowed to demand the NRS. A

Claim Request		Who can claim it
M$_{NRS}$	$= m_{NRS}, S_X[m_{NRS}]$	UA_S
M$_{NRK}$	$= m_{NRK}, S_X[m_{NRK}]$	UA_S, MTA_S
$\star$	$= m_{NRK'}, S_X[m_{NRK'}]$	MTA_R
M$_{NRD}$	$= m_{NRD}, S_X[m_{NRD}]$	UA_S, MTA_S
M$_{K-NRO}$	$= m_{KNRO}, S_X[m_{KNRO}]$	MTA_R, UA_R
M$_{SYNCH_{RS}}$	$= m_{SRS}, S_X[m_{SRS}]$	UA_R

m_{NRS}	$= cem_A$
m_{NRK}	$= cem_A, NRS$
$m_{NRK'}$	$= SYNCH_{RQ}, SYNCH_{RS}[cem_A]$
m_{NRD}	$= cem_A, cem_B, NRK, K - NRO$
m_{KNRO}	$= cem_A, NRK$
m_{SRS}	$=$ "LIST", LTS_{RQ}

$\star$ The M_{NRK} depends on who sends it.

Table 3.5: Resolution Sub-Protocol Requests (Trusted Mail Transfer Agent (MTA))

special case is the NRK resolution request, which has different content depending on the participant who demands it.

NRS Request

Only the UA_S can send the NRS claim request. If the TTP has received a previous request claiming later evidence (NRK, NRD or the key), it will issue a NRS_{TTP} (or a NRS, if available), and it will deliver it to the UA_S, along with the evidence obtained on previous requests. Otherwise, the TTP will contact the MTA_S sending a message equivalent in content to M_3 (table 3.4): $m_3, S_{TTP}[m_3]$. If the MTA_S answers, either a $NACK$ or the NRS, the TTP will deliver the evidence to the UA_S. In case the MTA_S does not answer, the TTP will contact the MTA_R sending the same message. If the MTA_R has some delivery evidence, either the NRK or both NRK and NRD, the TTP will deliver it to the UA_S, along with an affidavit NRS_{TTP}. If the MTA_R answers with a $NACK$, it means he does not have any information on the CEM delivery or he has received the CEM but he does not have the NRK yet. Therefore, the TTP will send a $N - NRK$ to the MTA_S, MTA_R (if available) and the UA_S, canceling the delivery. In case the MTA_R does not answer, the TTP will also cancel the delivery issuing the $N - NRK$. From now on, the TTP will answer with a $N - NRK$ to any request regarding this CEM delivery.

NRK Request

If the UA_S or the MTA_S request the NRK, the following can happen. If any party has previously contacted claiming the NRK, the TTP will deliver the same evidence. If a previous request has claimed later evidence (NRD or the $K-NRO$), the TTP will already have the NRK, thus, it will deliver it along with other evidence obtained during previous requests (NRD or NRD_{TTP}). Otherwise, the TTP will contact the MTA_R sending a message equivalent in content to M_3: m_3, $S_{TTP}[m_3]$. If the MTA_R answers with some evidence (NRK and/or NRD), the TTP will deliver it to the UA_S, otherwise ($NACK$ or no answer) the TTP will cancel the execution and deliver a $N-NRK$ to the UA_S, and both MTA (if available).

When the MTA_R is who sends the request (table 3.5), the following can happen. If the UA_R has previously sent a $M_{SYNCH_{RQ}}$ (table 3.5) request including the same LTS_{RQ} than the one included in the MTA_R's NRK request (M_{NRK}), and the TTP has sent a $NACK$ as response (the TTP has not been able to obtain the $SYNCH_{RS}$), the TTP will also send a $NACK$ to the MTA_R (to maintain coherence). Otherwise, the TTP will answer by issuing a non repudiation of knowledge affidavit, NRK_{TTP}. Note that $SYNCH_{RQ}$ and $SYNCH_{RS}$ are linked by their timestamp values.

NRD Request

The UA_S and the MTA_S can claim the NRD. If any party has previously contacted the TTP and claimed the NRD, the TTP will deliver the same evidence, either the NRD or NRD_{TTP}. Otherwise, the TTP will send to the MTA_R a message equivalent in content to M_8: m_8, $S_{TTP}[m_8]$. If the MTA_R answers with the NRD, the TTP will deliver it as a response. But if the MTA_R sends a $NACK$ or fails to answer, the TTP will deliver an affidavit, the NRD_{TTP}.

K − NRO Request

The MTA_R or the UA_R can claim the $K-NRO$. If the UA_S or MTA_S have previously contacted the TTP requesting the NRD, the TTP will already have the $K-NRO$, thus, the TTP will deliver it. Otherwise, the TTP will send to the MTA_S a message equivalent in content to M_7: m_7, $S_{TTP}[m_7]$. If the MTA_S answers providing the $K-NRO$, the TTP will deliver it. In case the MTA_S answers with a $NACK$ or does not answer (the server is down or does not want to answer), the TTP will deliver a $NACK$.

Note that following the latter claim, the UA_R can find himself in a situation where he does not receive the key neither from the MTA_R nor from the TTP (it answers with a $NACK$). But since t_D states the time limit for the protocol execution, if the UA_R receives a $NACK$ from the TTP after t_D, and the MTA_R does not have the $K-NRO$, the UA_R can assume the delivery has been canceled. The UA_S has no means to obtain the NRD, neither from the TTP (it will not issue an affidavit after t_D), nor from the MTA_R (she will not send M_9 after t_D).

SYNCH$_{RS}$ Request

Only the UA_R can send this request. If the MTA_R has previously contacted the TTP, claiming he has not received the NRK, the TTP will already have the corresponding $SYNCH_{RS}$, thus, the TTP will deliver it to the UA_R. Otherwise, the TTP will send to the MTA_R the M_4 signed: M_4, $S_{TTP}[M_4]$. If the MTA_R answers with the $SYNCH_{RS}$, the TTP will deliver it to the UA_R. In case the MTA_R does not answer (the server is down or does not want to answer), the TTP will deliver a $NACK$ to the UA_R.

3.5.4 Security Analysis

In this section we will review the accomplishment of the requirements we have presented in section 3.1, to prove that our protocol meets them.

CLAIM 3.8: **CEM Effectiveness**: *If all participants in a CEM delivery behave according to the protocol and there are no communication errors, the TTP will not intervene.*

PROOF. Following the *delivery* sub-protocol execution (see figure 3.3 and section 3.5.2), we can see that the TTP does not intervene. Therefore we can affirm that if every party involved behaves correctly and there are no network failures, the UA_S will obtain her evidence (NRS, NRK and NRD) and the UA_R will obtain his evidence ($E_k(C)$, $PK_{UA_R}(k)$ and NRO) without TTP's intervention, meeting the effectiveness requirement.

CLAIM 3.9: **CEM Fairness**: *The UA_S will receive her evidence (NRS, NRK and NRD) if an only if the UA_R obtains his (the message, the key and the NRO), and the UA_R will receive his evidence if and only if the UA_S obtains hers.*

PROOF. Assuming an honest sender UA_S, the recipient UA_R can obtain the message $E_k(C)$ and the $K - NRO$ from the MTA_R or from the TTP. If the UA_R obtains the $K - NRO$ from the MTA_R, it means the MTA_R has already delivered the NRK to the MTA_S. Thus she can contact the TTP and obtain the NRD or an affidavit (NRD_{TTP}). Moreover, the sender UA_S can also contact the TTP and obtain the NRK and NRD (or NRD_{TTP}). In the case the UA_R obtains the $K - NRO$ from the TTP, the UA_S or the MTA_S have contacted previously claiming the NRD, therefore they already have all the evidence they need. In any case, the UA_S can obtain her evidence, thus fairness is maintained.

Assuming an honest recipient UA_R, the sender UA_S can obtain her evidence (NRK and NRD) from the MTA_S or from the TTP. These evidence can be the original value (signed by the recipient) or an affidavit from the TTP. If the UA_S obtains her evidence from the MTA_S and it is signed by the recipient, it means the recipient has already received the $K - NRO$: the MTA_R sends the NRD after receiving the $K - NRO$. If the evidence contains an affidavit (NRD_{TTP}), it means the MTA_S contacted the TTP claiming the NRD, therefore the TTP has the $K - NRO$ and the UA_R can contact the TTP and claim it. In the case the UA_S obtains the evidence from the TTP (the original or an affidavit), it means the TTP already has the $N - NRO$ (see

M_{NRD} in table 3.5), therefore the UA_R can contact the TTP and claim it. In both situations, the UA_R can obtain his evidence, thus, fairness is maintained.

As a conclusion, our proposal meets the fairness requirement.

CLAIM 3.10: **CEM Timeliness**: *The delivery of a CEM will end in a finite amount of time, maintaining fairness for all honest participants.*

PROOF. Along with the parameters sent within the message, there is the delivery deadline t_D. After t_D, the TTP will not generate new evidence, i.e, the TTP will not contact any MTA to solve the claim, and it will answer according to the evidence received before t_D. If before t_D any user has not received his evidence, he must contact the TTP and ask for resolution. In the case of the UA_S, she may claim the NRS, the NRK or the NRD. Meanwhile the UA_R will claim the $K - NRO$. Moreover, at any time $t < t_D$ during the protocol execution, any party can contact the TTP and claim resolution. In particular, before the NRK is generated (M_6 in table 3.4), the delivery can be canceled (the TTP issues a $N - NRK$). After the NRK is generated, the TTP can deliver the NRD or NRD_{TTP} and the $K - NRO$, evidence needed by the UA_S and the UA_R to consider the message delivered. Therefore we can affirm that, either by the TTP's intervention, the correct execution of the *delivery* sub-protocol or because of the deadline t_D, the delivery can finish in a finite amount of time $t \le t_D$, maintaining fairness for all honest parties.

CLAIM 3.11: **CEM Verifiability of TTP**: *If the TTP issues wrong evidence, the participants can prove it.*

PROOF. The TTP could misbehave and, as a result, an honest party could lose fairness. The TTP could try to influence the delivery result, so that one party, either UA_S or the UA_R, has evidence of delivery, and the other one, evidence of delivery canceled. In particular we can have three different situations.

In the first situation, the UA_S contacts the TTP claiming she has not received the NRS or the NRK. The TTP can answer with a $N - NRK$, but do not deliver it to the MTA_S. Therefore the delivery will continue, the MTA_S will receive the NRK and NRD; and the UA_R the message $E_k[C]$, and the $K - NRO$. This could be an unfair situation, but the UA_S will be able to detect it when she executes a $SYNCH_{RQ}$ message. She has a $N - NRK$, but the MTA_S holds the evidence of delivery NRK and NRD.

A second case can happen when the UA_S contacts the TTP claiming she has not received the NRS, the message has not reached the MTA_S, therefore it has not been sent. But the TTP can answer with the NRS_{TTP} and NRK_{TTP}, without contacting any MTA. Later, the UA_S claims the NRD, and receives a NRD_{TTP} from the TTP, again without contacting any MTA. The UA_S has all delivery evidence and the UA_R does not have anything. When the UA_S tries to "enforce" the delivery, the UA_R will be able to prove the TTP's misbehaviour by requesting evidence:

- to generate the NRS_{TTP}, the TTP must have previously received a request (signed) either by the MTA_R or the UA_R.

- to generate the NRK_{TTP}, the TTP must have the corresponding $SYNCH_{RQ}$ and $SYNCH_{RS}$ (M_4 and M_5 in figure 3.3 and table 3.4).

Finally it can happen that the UA_R and/or MTA_R contact the TTP claiming they have not received the key. The TTP can answer with a $NACK$, until t_D expires, even when it already has the key. Therefore, the honest UA_S will have the evidence of delivery and the UA_R will have nothing. But if the UA_S tries to "enforce" the delivery, the UA_R and/or the MTA_R will be able to prove they contacted the TTP, and got a $NACK$, proving the TTP's misbehaviour.

In all situations the TTP's misbehaviour can be proved, therefore we can affirm that our protocol meets the verifiability of TTP requirement.

CLAIM 3.12: **CEM Confidentiality**: *Only the UA_S and the UA_R have access to the content C of a CEM.*

PROOF. The UA_S sends the content of the certified message C encrypted with the symmetric key k, and this key k encrypted with PK_{UA_R}, the recipient's public key. Therefore, the content of the message is only known by the sender UA_S and the recipient UA_R who is the only capable of obtaining the symmetric key k. Thus we can affirm that the protocol meets the confidentiality requirement.

CLAIM 3.13: **CEM Non-Selective Receipt**: *Once the NRK is generated, either by the UA_R or the TTP (NRK_{TTP}), the delivery cannot be canceled.*

PROOF. The UA_R sends the NRK after receiving a $SYNCH_{RS}$, stating that he knows he has a pending CEM delivery. If by any chance the UA_R tries to reject the CEM, by not sending the NRK, the MTA_R can contact the TTP and obtain an affidavit from the TTP, the NRK_{TTP}. Thus, the UA_R has no chance to reject an incoming CEM. Therefore, we can affirm that our protocol meets the Non-Selective Receipt requirement.

CLAIM 3.14: **CEM Non-Repudiation**: *None of the participants in the delivery of a CEM can deny having participated, neither can they be excluded from it.*

PROOF. Along the protocol execution, evidence is generated as non-repudiation proof for various services. In particular, the protocol offers:

- *NRO*, Non-Repudiation of Origin. When the UA_S issues a message, she includes her signature on the message as proof she has created it. If the UA_S tries to deny having originated it, the UA_R can prove it using the UA_S's signature on the message.

- *NRS*, Non-Repudiation of Submission. When the UA_S delivers a CEM to her MTA_S, she receives the MTA_S signature on the message, as proof she has sent it. Thus, the MTA_S cannot deny having received the CEM.

- *NRD*, Non-Repudiation of Delivery. When the MTA_R receives the key from the MTA_S, he sends his signature on the message, as proof the CEM is in the recipient's inbox. Thus, the MTA_R cannot deny having received it.

- NRR, Non-Repudiation of Receipt. The UA_R cannot deny having received the CEM. To provide NRR, we introduce the NRK, Non-Repudiation of Knowledge. When the UA_R receives a $SYNCH_{RS}$ including new CEM, he sends his signature to the MTA_R as proof that he knows there is a pending delivery (NRK). This avoids the UA_R to deny knowing he had received a CEM. Therefore, if we combine the NRK and NRD, we have that: the MTA_R cannot deny having received the $K - NRO$, and the UA_R cannot deny knowing he has a message. Therefore we can affirm that the UA_R knows he has a CEM, and that he has the means to fetch it and open it, so, we can consider the message received, $NRK + NRD \equiv NRR$. Thus, we can affirm that our protocol meets the Non-Repudiation of Receipt requirement.

3.6 Conclusions

In this chapter of the dissertation two contributions related to Certified Electronic Mail (CEM) as a fair exchange application have been presented. The contributions are two optimistic CEM protocols designed following the current e-mail architecture. Most of the previous proposals presented protocols where the sender and recipient have direct interaction, or they communicate through only one intermediary, which does not reflect the current e-mail infrastructure where each user chooses his own Mail Transfer Agent (MTA_S and MTA_R for sender and recipient, respectively). Moreover, we can have many MTAs between the MTA_S and MTA_R. In addition, the protocol flow of each proposal has been designed to mimic the behaviour of POP3 or IMAP, used in the traditional e-mail by the users to fetch the messages from their MTA. These features, the architecture and protocol flow, facilitate the deployment of the proposals on the existent Internet e-mail infrastructure, reducing costs for mail providers in terms of infrastructural investments, and allowing senders and recipients to use standard Internet e-mail clients, with minimum modifications.

The proposal in section 3.4 is the first of its kind to assume all MTAs are untrusted entities, they can cheat and collude, whereas the second proposal (section 3.5) assumes that each participant trusts his own MTA (the sender trusts the MTA_S, and the recipient trusts the MTA_R) while consider the other ones untrusted. Both proposals meet all the core security requirements (effectiveness, fairness, timeliness and non-repudiation), two additional ones, verifiability of the TTP and confidentiality, and one specific to CEM protocols, non-selective receipt. In the case of the non-repudiation requirement, the proposals provide evidence of the whole process: origin, submission, delivery and receipt. Thus, in a CEM delivery involving a sender, a sender's transfer agent, a recipient's transfer agent, and a recipient, no one can deny having been involved.

4

ELECTRONIC PAYMENT SCHEMES

In this chapter of the dissertation a fair scheme for electronic bearer bank checks is presented. The scheme allows payments of large amounts, with anonymity in payment, with anonymous transferability and guaranteeing that all valid checks will be paid (in cash or depositing in bank accounts). It meets all the core fair exchange requirements (section 1.2.1) and the three main properties of the paper-based bearer bank check. First, this scheme allows payments of arbitrary amounts (those agreed between the bank and the client). Moreover, the scheme allows e-checks to be transferred, with no need for the receiver to deposit the e-check at the bank. Finally, the proposed scheme can maintain the anonymity of all users involved in a chain of transfers of an electronic check, so-called bearer check.

4.1 Legal Issues

Definition and types of checks can vary from country to country, but it is commonly accepted that a check is a negotiable instrument in writing containing an unconditional order, signed by the maker, directing a certain person to pay a certain sum of money only to the order of a certain person, or to the bearer of the instrument, and a bank is always the drawee.

According to the "Convention Providing a Uniform Law for checks" [104], a paper-based check is a payment instrument that should contain: a) The term "check" inserted in the body of the instrument and expressed in the language employed in drawing up the instrument; b) An unconditional order to pay a determinate sum of money; c) The name of the person who is to pay (drawee); d) A statement of the date when and the place where the check is drawn; e) The signature of the person

who draws the check (drawer); and optionally, f) A statement of the place where payment is to be made.

Depending of the content of a check, and without being exhaustive, checks can be classified as:

- *Open check* or *uncrossed check*: A check is call "Open" when it is possible to get cash over the counter at the bank. The holder of an Open check can do the following: a) Receiving the money over the counter at the bank; b) Depositing the check in his account; c) Passing it to someone else by signing on the back of the check (process called endorsement).

- *Crossed check*: The payment of crossed check is not made over the counter at the bank. This check can only be paid into the bank account of the payee. A check can be crossed by drawing two transverse parallel lines across the check, with or without the writing "account payee" or "not negotiable". This is an interesting option because open checks are subject to risk of theft, and so it is dangerous to issue such checks (in paper). This risk can be avoided by issuing crossed checks.

- *Bearer check*: A check which is payable to any person who presents it for payment at the bank counter. A bearer check can be transferred by mere delivery and requires no endorsement.

- *Order check*: An order check is one which is payable to a particular person. In such a check the word "bearer" may be cut out or cancelled and the word "order" may be written. The payee can transfer an order check to someone by signing his name on the back of it (an endorsement). If the words "not to order" are written in the check, it means that the check cannot be endorsed.

- *Banker's draft* or *bank check*: This is a check issued or drawn by a bank (the bank is the drawer and the drawee). It can be usually obtained from a bank for a small fee. This is useful for making large value payments where cleared funds are required and where it may be risky using cash.

- *Guaranteed* or *certified check*: In this kind of check the bank (drawee) guarantees that it will pay the amount indicated in the check.

This dissertation proposes a solution for a specific kind of check: an open bearer bank check. An open bearer bank check should meet the following generic requirements: a) It is payable to any person who presents it for payment at the bank counter, and transferable by mere delivery; b) It is issued or drawn by a bank (the bank is the drawer and the drawee), so, the requester does not sign the check nor any personal information is contained in the check; c) It is possible to get cash over the counter at the bank; d) The requester can be identified by the issuing bank, but he can remain anonymous in front of other users; e) The payee can remain anonymous, and transfer the check anonymously; f) Payment is guaranteed.

From now on, we will refer only to this kind of check, *electronic bearer bank check*.

4.2   Bearer Check Scenario

This section defines the role of the different entities involved in our electronic check scheme. Most entities involved in the scheme are the same as in the paper world:

- *Payer* (P_r): an entity registered with the issuer (a bank) who wishes to obtain an e-check so as to make a payment to another entity.

- *Payee* (P_e): an entity who receives an e-check from a payer. In our scheme, a payee, having received an e-check, can become a payer for another entity and with the "same" e-check.

- *Issuer* (I): the bank responsible for issuing e-checks to payers.

- *Acquirer* (A): the bank to which the payee is a registered account holder. The payee could also withdraw/cash the e-check value over the counter at the payer's bank I.

Additionally, in our scheme, a new entity is necessary:

- Trusted Third Party (TTP): this entity will guarantee the fairness of exchanges among all parties involved in an e-check transaction.

Legal terminology and technical terminology does not always coincide, therefore it is necessary to clarify the meaning of the terms payer, payee, drawer and drawee. The latter two are standard legal terms in the law relating to checks (see section 4.1), while the first two are terms used in e-checks papers. The drawer is the (individual or legal) person issuing a check, signing it, and therefore he is liable for the payment of the check. The drawee is the entity that will pay the amount indicated in the check, if there is enough money in the drawer's account. The drawee has always to be a bank. The payer is the person using a check for making a payment (he can be a drawer or not). The payee is the entity accepting a check as a method of payment in exchange of goods or services.

Perhaps the more confusing situation is when the payer is not the drawer. It is the case of bank checks: the bank is the drawer and gives the check to the requester client that will act as a payer in a payment to a payee. Another case is when a payee has received a bearer check and decides to pay with it (without depositing it): in the new payment he will be the payer and he will not be the drawer (who is the bank signing the check).

4.3   Requirements for Electronic Bearer Bank Checks

We can classify e-check requirements into three categories: security requirements, functional requirements, and finally, legal requirements. Despite the previous classification, some requirements are difficult to classify because they can have an impact on more than one category.

4.3.1 Security Requirements

We are in front of a kind of fair exchange of values (e.g., an e-check for an account decrement or increment), and so, fair exchange requirements as defined in section 1.2 must be met. In particular, the e-check proposal meets all the core requirements: effectiveness, fairness, timeliness and non-repudiation. But as application of fair exchange, the e-check proposal also requires some specific security requirements, related to electronic bearer bank checks. For other kinds of e-checks, different requirements can be necessary. For instance, an electronic order check requires the payee's identity can be specified on the e-check at the issuing phase [25, 105]. Thus, payee's anonymity is not a requirement for that kind of e-check, but it is required for electronic bearer bank checks.

Core Fair Exchange Security Requirements

Following we describe the core fair exchange security requirements applied to electronic bearer bank checks.

Definition 4.1 (*e-check Effectiveness*) *If every participant in an exchange (payment, deposit/cashing or transferring with/of an e-check) behaves according to the protocol, the TTP will not intervene.*

Definition 4.2 (*e-check Fairness*) *Every honest participant (A) in an exchange (payment, deposit/cashing or transferring with/of an e-check) will receive what is waiting for from the other participant (B), or the TTP will provide him (A) with evidence (or proof) that the other party (B) is not behaving as expected.*
     This requirement ensures that at the end of an exchange between two or more entities, or everybody achieve the expected items, or nobody can stand in a privileged situation. This property can be useful for multiple processes related to e-checks management:

- *obtaining/paying: if the payer's account is decremented for a specific amount then he should receive a valid e-check from the issuer.*

- *transferring: if the payer delivers a valid e-check to the payee, the payee must provide the service or good, or a proof that she has received the e-check.*

- *depositing/cashing: if the payee sends a valid e-check to the bank for depositing or cashing, then his account must be increased by the corresponding amount or he has to receive the amount in cash.*

Definition 4.3 (*e-check Timeliness*) *Every participant in an exchange (payment, deposit/cashing or transferring with/of an e-check) can be sure this exchange will finish in a finite amount of time and, once the exchange is finished, fairness will be maintained.*

Definition 4.4 (*e-check Non-repudiation*) *None of the participants in an exchange can deny having participated nor be excluded of it. In particular:*

- *obtaining/paying: the payer (P_r) cannot deny having requested an e-check $\mathbb{EC}$ with a face value Q, and the Issuer (I) cannot deny having issued it.*

- *transferring: in this exchange the payer (P_r) remains anonymous, therefore the issuer (I) cannot prove that a particular P_r requested to change the ownership of an e-check $\mathbb{EC}$, but it can prove that whoever demanded it, was the owner. And I cannot deny having changed the ownership of $\mathbb{EC}$.*

- *depositing/cashing: the bearer (P) cannot deny having requested depositing the e-check $\mathbb{EC}$, and the issuer (I) cannot deny having deposited it. If P decides to cash $\mathbb{EC}$, his identity will remain anonymous. Therefore as in the case of transferring an e-check, I can prove that whoever cashed it, was the owner. And I cannot deny having cashed it.*

Specific Electronic Bearer Bank Checks Security Requirements

Here we list security requirements specific to e-checks that we are considering.

Definition 4.5 (*Unforgeability*) *Only authorized entities can issue valid e-checks.*

In other words, it cannot be possible to forge e-checks to be given in good, as if they were issued by an authorized issuer. This property is directly related with non-repudiation of origin property, and therefore with integrity and authenticity properties. In our scenario only banks can issue electronic bearer bank checks.

Definition 4.6 (*Non-overspending*) *An e-check only belongs to one user at a given time.*

A payer cannot be able to pay twice with the same e-check. He can make copies of one e-check, but only one of them will be given as good. The use of the same e-check has to be prevented or, at least, detected.

Definition 4.7 (*Anonymity*) *Users of e-checks (payers/payees) should be able to remain anonymous in non-identified processes related to the e-check.*

Paper-based bearer bank checks allow that payers and payees remain anonymous in front of the bank when they pay with, transfer or cash the check. Obviously, depositing is a process without anonymity: the user has to provide his bank account information.

Definition 4.8 (*Transferability*) *A payee can transfer an e-check (previously received as payment) to another payee without depositing or cashing it, and without losing anonymity.*

Payees receiving an e-check in a transfer (not directly from an authorized issuer; the bank) must be able to verify if the e-check is valid (it will be easy if unforgeability is met) and it has not been already spent by the transferor entity (or previous transferors). Anonymity must not be lost in the transfer process. An e-check can be transferred indefinitely without being necessary to deposit it. Observe that a transfer is not an endorsement; the latter is a non-anonymous process.

Definition 4.9 (*Untraceability*) *Together with anonymity, privacy is also related to the impossibility to track the entities involved in a chain of transfers by the bank with the same or a different e-check.*

A bank can identify two of the entities involved in the chain of transfers: the initial payer (who is an identified client of the bank and requests the issue of an e-check) and the last payee (who deposits the e-check in her bank account). But the intermediaries payees of that chain cannot be identified by the bank. Moreover, a specified entity cannot be linked to a particular e-check because neither the identity of the owner of the e-check is included in the content of the e-check nor the intermediaries are identified by the bank.

4.3.2 Functional Requirements

Now we are listing some functional requirements. These requirements are not so directly related to security, but they can be as important as those explained previously.

Definition 4.10 (*Online operations*) *Paying with an e-check or transferring it requires a persistent connection with an entity different from the payer and the payee (e.g., a trusted centralized system).*

Typically offline options are preferred, alleging costs, possible bottleneck, etc.; but in a e-world where millions of transactions with credit card are made online, and with companies working with great computational power (Google, Facebook, etc.), it seems that this argument is no longer valid. In terms of security, online operations for payment and transfer are better in order to carry out double-spending checking.

Definition 4.11 (*Efficiency*) *Processing an e-check has not to be resource consuming, mainly, on the client side.*

We have to bear in mind that e-checks should be held in portable devices (PDAs, smart-phones, etc.). Mobile terminals can be limited in terms of computational power, and so protocol operations and especially cryptographic operations have to be reduced only to necessary ones.

4.3.3 Juridical Requirements

In this section we will explain the legal requirements to be met by electronic bearer bank checks.

Definition 4.12 (*Content*) *A bearer check must contain the following information: the words "bearer check", the sum of money to be paid, the name of the entity who is to pay (drawee), the signature of the person who draws the check (drawer) and, optionally, date when and place where the check is drawn, and a statement of the place where payment is to be made.*

The check can contain more information (a serial number for every check, etc.), but the data specified above is the content according to the Geneve "Convention Providing a Uniform Law for Checks" [104].

Definition 4.13 (*Drawee and drawer*) *On all checks the drawee must be a bank, and for bank checks the drawer must also be a bank.*

This means that the signature contained in a e-check must be performed by a bank.

4.4 Related Work

There are some proposals to provide security to electronic checks [106, 105, 107, 108, 24, 109, 25, 110, 111, 112, 113, 114, 115]. However, the vast majority of the proposed solutions do not fit the definition of check, or do not accomplish security requirements for electronic bearer bank checks [106, 105, 107, 108, 24, 109, 25] (see section 4.2).

The first solutions for e-checks appear in the literature in the late 80's [110, 111, 112, 113], but it is from 2002 when it receives more attention [114, 106, 115, 105, 107, 108, 24, 109, 25]. However, not all previous solutions we find in the literature using the word e-check can be considered as a proper e-check. This is the case of some works such as [110, 111, 112, 113, 114, 115]. It is easy to detect this kind of solutions because all of them include a refund process that it is not allowed in the paper-based checks. In fact, these schemes are more similar to cash-like payment systems, as stated by [114]: "We must not confuse the checks explained here with conventional paper-based checks". Therefore, these solutions are not useful as substitutive of paper-based checks.

This way, we can consider that there are, mainly, a few solutions which fit the legal check definition [106, 105, 107, 108, 24, 109, 25]. Among these references, we can differentiate two main types of checks based on its use or their definition: order checks and bearer checks. Despite the efforts dedicated to the study of electronic checks in general, to the best of our knowledge, there is only one proposal that address the problem of electronic bearer checks. But we review the order checks because some of them mention that their solutions could work as another type of e-check.

4.4.1 Order e-checks

In the order e-check classification, we find some solutions [105, 107, 24, 116] whose main objective is providing an scheme where specific information, such as payee identity, face value and date can be specified when the payer make effective the payment to the payee. That is, they want to provide flexibility to the e-check scheme. But, those schemes have a main problem; the availability of the funds is not guaranteed. So, if there are insufficient funds available for the payment, the corresponding e-check is a loss for the payee. To solve this problem Hsin *et al.* [105] propose to balance between flexibility and warranty. To do so, the authors consider two modes of operation. In the first one, it works as the above solutions, all the information is attached to the e-check at the time of the transaction between the payer and the payee. In the second one, the bank transfers the money from the payer's account to a special holding account, providing a certification from the bank that guarantees the funds. The face value must be fixed at issuing phase, but the other information (payee's identification and pay date) can be defined later, preserving some flexibility. Besides, Hsin *et al.* [105] state their scheme provides privacy. This is not completely achieved, at least from the payment and deposit points of view. The payee must be identified by the bank at depositing stage because the e-check includes the identity of the payee fixed by the payer at paying phase.

Chang *et al.* [109] highlight as an important requirement the mutual authentication between a payer and a payee to enhance the system security. It can be useful in some applications for e-check, but it is not an intrinsic requirement for checks (look at paper-based checks), in the same way that fair exchange of an e-check for a product or service can be very important, but not an intrinsic requirement of e-checks. In this vein, Chen *et al.* [25] and Pasupathinathan *et al.* [108] provide solutions with anonymity for the payer. This anonymity is based on pseudonyms provided or managed by the bank; therefore strictly speaking we cannot affirm that it is anonymity. Whatever case, we have to remember that from a juridical point of view, one of the compulsory fields of a paper-based check is the signature of the entity who draws the check, and the aim of this signature is to be able to identify the payer. Therefore, the e-check must be paid to the entity to whom the e-check is issued, making the e-check non-transferable and non-anonymous.

Another issue considered by some authors is the performance efficiency. It has been considered from two points of view. On one hand, improving the use of every individual e-check. For example, [107] tries to improve the efficiency of Chaum's scheme [111] from the computational point of view using hash based computation and avoiding the use of public key cryptography. On the other hand, we find the use of booklet checks to avoid contacting the bank for every required e-check [109, 24].

However, the main problem from a security point of view, is the way in which the payment/transfer is performed. The authors focus on providing proposals for working offline [107, 108, 24], that is, the transaction between the payer and the payee does not require the bank intervention. The objective is improving the availability and reducing the burden on the solution. In this way, when the payee receives the e-check, she can verify its authenticity but not its double-spending be-

cause double-spending check is made at deposit phase (after payment), but a large amount e-check should not be accepted without prior double-spending check. To solve that, in [109, 25] the double-spending is prevented with an online verification with the bank, avoiding drawbacks of prior solutions.

The FSTC's e-check system [26] requires a special mention since it is an electronic payment instrument specifically developed for the Internet and designed to work like paper-based checks. The FSTC's e-check is based on the same legal framework that applies to paper-based checks and thus all existing legislations and account agreements that apply to paper-based checks still hold for electronic checks. However, as indicated in the analysis done by [108], there are various security issues concerning confidentiality, privacy and traceability that are yet to be addressed. For example, the payee has unrestricted access to the payer's account information. In addition, the FSTC's system leaves the smart card as the only source of proof for transactions, so the loss of the smart card by a payer also implies the loss of proofs for all transactions. A final concern with the system is traceability of payment instructions. The smart card stores all e-check transaction details. Because of limited memory in the card, they have to be returned to either the issuer or to a trusted third party (TTP). This raises the issue of payer privacy for transaction details because the issuer or TTP will be able to obtain all transaction details including the purchasing details of the payer, opening payer's privacy issues.

4.4.2 Bearer e-checks

Despite the efforts dedicated to the study of electronic checks in general, to the best of our knowledge, there is only one proposal that addresses the problem of electronic bearer checks [106]. In this scheme [106], an e-check is payable to any person who presents it for payment at the bank, but the amount of money is transferred from the payer's account to the payee's account. So, the scheme inherits the same problem that the vast majority of order e-checks, where the bank transfers the authority to draw e-checks to the payer (signer). That is, the payment of the e-check to the bearer is not guaranteed. Thus, it is not useful for payments of large amounts of money.

The authors mention that their solution is offline, however, the scheme requires the intervention of a new entity, a third party (TTS - Time-Stamp Service), for double-spending detection support. The TTS is contacted by the payee at the transfer protocol. At this stage, the entity receiving the e-check verifies two things: the authenticity of the e-check (non-forgery), and the proof of possession of the entity trying to transfer the e-check. However, the way for proving whether the e-check has already been used is contacting the bank, and that is not considered by the authors. Of course, offline schemes can decrease the overhead of transactions but they can also reduce the level of security. Moreover, in the scheme proposed in [106], different e-checks from a same customer can be linked because a consumer's pseudonym is included in each e-check, thus, it detracts from payer privacy.

Finally, a very important issue is that the proposal in [106] does not allow a real transferability, in the sense that an e-check cannot be transferred indefinitely from

a payer to a payee without being necessary a deposit to a bank, and without losing payee's anonymity.

We proposed an scheme for electronic bearer bank checks [117] based on hash operations. In this scheme, when the bearer of the e-check must prove he is the legitimate owner of the e-check, he must send a secret value. This secret value is linked to a public value embedded in the e-check, and these values are related through a hash operation. Thus, if problems arise, and the e-check cannot be deposited or transferred, the secret value is revealed to the bank or the TTP (used for providing fairness). In this case, the bearer of the e-check can lose the fairness in the e-check process, and even worst, the bank or the TTP could be in an advantageous position. That is, a fraudulent bank or TTP may try to cash the e-check, or they could provide another entity with the secret value linked to the e-check. So, the e-check could be cashed or transferred by an entity other than the legitimate owner of the e-check.

	Our proposal	[117]	[106]	[105]	[107]	[24]	[25]	[118]
Unforgeability	✓	✓	✓	✓	✓	✓	✓	✓
Non-overspending	✓	✓	✓	✓	✓	✓	✓	✓
Anonymity	✓	✓	*	✓	*	–	–	*
Transferability	✓	*	–	–	–	–	–	–
Untraceability	✓	✓	–	–	–	–	–	–
Fairness	✓	*	–	–	–	–	–	–
Operations								
Payment	*ON*	*ON*	*OFF*	ON/OFF	*ON*	*OFF*	*ON*	ON/OFF
Transfer	*ON*	*ON*	*ON*	–	–	–	–	–
Drawer	B	B	P_r	P_r	P_r	P_r	P_r	P_r

✓YES –NO * partially *ON* online *OFF* offline

Table 4.1: E-Check Solutions - A Comparative Analysis

4.4.3 Conclusions

The Table 4.1 shows a comparison of the requirements met by the proposals presented in section 4.4 and our proposal. As a conclusion, we want to emphasize that although there are a few solutions accomplishing e-check definition, their main drawback is that they are designed for order or nominative e-check scenarios where payee's identity is provided in the e-check. Although authors such as Hsin *et al.* [105] point out their schemes can be adapted to support another type of checks, the fact is that they are mainly designed to work as non-anonymous checks where payer and payee are identified in both payment and deposit phase. In fact, the solutions are not real bank e-checks where the amount is fixed and guaranteed by the bank at the issuing phase. Most of the authors have focused their efforts on allowing to fix payer's information and the face-value at the payment phase, but if the funds are not guaranteed, the corresponding check translates into a loss for the payee. More-

over, no solution allows a real transferability. Therefore, new solutions are needed to accomplish with the requirements of anonymous, transferable and guaranteed e-checks, in addition to maintain fairness in all processes involving an e-check as defined in sections 4.2 and 4.3.

4.5 Cryptographic Background: Zero-Knowledge Protocol

In this section, we briefly review a zero-knowledge protocol which will be used in our proposal as a building block. The aim of a zero-knowledge protocol is to allow a user A (the prover) to prove to another user B (the verifier) that A knows a secret without revealing the secret to B (the verifier). This kind of protocol must be complete and sound. Completeness means that an honest prover succeeds in convincing an honest verifier, and soundness means that a dishonest prover does not succeed in convincing the verifier of a false statement. In our proposal we use the protocol defined by [119] because it was designed to be very fast and efficient, mainly on the amount of computation performed by the prover (probably a mobile device in our scheme).

In [119], the protocol is defined for a cyclic group $\mathcal{G}_q$ of order q with generator g, more formally: $\mathcal{G}_q = \langle g \rangle = \{g^q : q \in \mathbb{Z}_n^*\}$.

In order to prove the knowledge of a secret value $x = \log_g y$, the prover interacts with the verifier as follows:

(1) $PoKCommitment$: the prover generates a random value $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and sends $pok_{Co} = g^r$ to the verifier;

(2) $PoKChallenge$: the verifier replies with a challenge pok_{Ch} chosen at random $pok_{Ch} \overset{R}{\leftarrow} \mathbb{Z}_q$;

(3) $PoKResponse$: after receiving pok_{Ch}, the prover sends the third and last message (the response) $pok_R = (r + pok_{Ch} \cdot x) \mod q$.

The verifier accepts, if: $g^{pok_R} \equiv pok_{Co} \cdot y^{pok_{Ch}}$

For clarity, we use the notation introduced by [120] for proof of knowledge. Thus, the scheme introduced above is expressed as follows: $PK\{(x) : y = g^x\}$. In our proposal, a payer or payee act as prover for convincing a verifier (payee or bank, respectively) that he is the legitimate owner of an e-check using the PK scheme.

4.6 An Electronic Bearer Bank Check Scheme

In this section we present an electronic bearer bank check scheme that accomplishes all the requirements defined in section 4.3. Table 4.2 shows the notation used along with the protocols description.

4.6.1 Overview and Data Structure of the e-check

Remember that we can differentiate five participants in our scheme (see 4.2): the payer (P_r), the payee (P_e), the issuer (I), the acquirer (A) and a trusted third party (TTP). In order to simplify the explanation we assume, as the vast majority of proposals, that the payer and the payee are working with the same bank (I and A are the same entity). In a real situation they may be different, and the current system of clearing can be used.

Protocol Entities	
P_r	Entity who requests an e-check to the bank
P_e	Entity who receives an e-check from P_r
I	Bank responsible for managing e-checks
TTP	Trusted third party involved in dispute resolution
General Used Notation	
$H(x)$	One-way collision resistant hash function over x
sK_X and pK_X	X's key pair from a public key cryptosystem
$Cert_X$	Public key certificate of entity X
$S_X(m)$	Digital signature of m made by entity X
$x \xleftarrow{R} Z_n^*$	x randomly chosen from Z_n^* set
$E_X(m)$ - $D_X(m)$	Asymmetric encryption and decryption of m using X's key
s_X	X's secret value from the PK scheme
w_X	X's Bank Account Number
τ_u	Time mark indicating when an evidence was generated by the TTP
$\mathbb{EC}$	Electronic Check
Γ	Content of an electronic check
Content of an electronic bearer bank check Γ	
c_{id}	Text indicating the type of the e-check
I_{id}	I's identification
Q	Face value of the e-check
p_X	X's public value associated to s_X
t_{cd}	Optional field about the way the e-check can be cashed
τ_{exp}	Information about until when the e-check can be deposited
l_{cd}	Information about where the e-check can be deposited

Table 4.2: Notation Used in the Protocols Description

Figure 4.1 shows the protocol flow and the interactions among all involved participants. Our scheme consists of three main protocols and three dispute resolution protocols whose objective is explained as follows:

- *Payment protocol*: this is the protocol by means of a user (the payer) obtains an e-check with the agreement of his bank (issuer), and gives it to the payee. The e-check is signed by the bank.

- *Deposit or cashing protocol*: this is the protocol by means of the holder of the e-check (a payee) can deposit or obtain cash for an e-check at the bank.

- *Transfer protocol*: this is the protocol by means of a user (a payee than now acts as a payer) can transfer an e-check to another user (a new payee).

- *Payment resolution protocol*: this protocol will be invoked by the payer of an e-check in case the bank does not follow the steps of the payment protocol.

- *Deposit/Cashing resolution protocol*: this protocol will be invoked by the bank or the bearer of an e-check in case the other part does not follow the steps of the deposit protocol.

- *Transfer resolution protocol*: this protocol will be invoked by the bank or the bearer of an e-check in case the other part does not follow the steps of the transfer protocol.

The dispute resolution protocols are necessary to ensure that no party involved in an e-check payment, transfer or deposit, gains an unfair advantage over the other party by misbehaving, misrepresenting or by prematurely aborting the protocol execution.
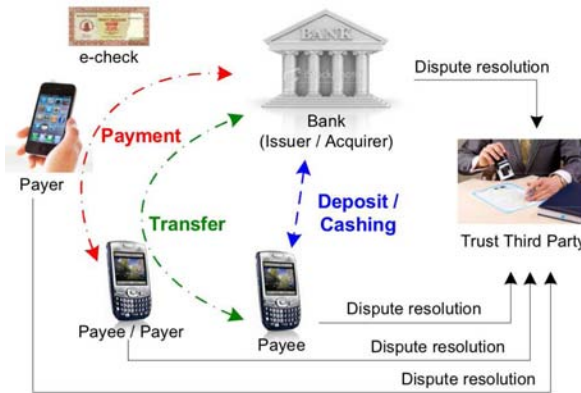
Figure 4.1: Protocol Flow and Involved Entities

Next, we define the format of an electronic bearer bank check in our scheme as follows:

$$\mathbb{EC} = [\Gamma, S_I(\Gamma)] \text{ where } \Gamma = (c_{id}, I_{id}, Q, p_X, t_{cd}, \tau_{exp}, l_{cd})$$

The type of e-check determines its content as explained in section 4.1. The e-check we are considering is an electronic bearer bank check, so the term "bearer check" is contained on the e-check in order to fulfil legal requirements (c_{id} = "bearer check"). I_{id} is the identity of the bank issuing the e-check. Q is the amount to be paid to the bearer of the e-check. p_X is the public value of the PK scheme (explained in section 4.5). The secret value of the PK scheme s_X must be only known by the owner of the e-check. The pair of values s_X, p_X will be used to verify if the entity

trying to use the e-check is the legitimate and unique owner of the e-check executing the three step protocol defined in section 4.5. Note that p_X is a fully anonymous parameter: nobody can infer from this parameter the real identity of the owner of an e-check. Optionally, the e-check contains a date (τ_{exp}) and a place (l_{cd}) showing when and where the e-check can be cashed or deposited. t_{cd} is also an optional parameter indicating if the e-check can be paid over the counter at the bank (open check), or must be deposited at the bearer's bank account (crossed check). Finally we find the signature of the bank (I) on the previous information ($S_I(\Gamma)$).

4.6.2   Payment Protocol

Let us suppose P_r has to pay an amount Q to P_e. The payment protocol must follow the next steps (see figure 4.2):

Step 1. First of all P_e initiates the PK scheme. She obtains a secret value s_{P_e}, and a public one p_{P_e}. This public value (p_{P_e}) will be used as a proof of possession of the e-check by P_e, and it will be necessary for transferring, depositing and cashing operations in order the bank can verify that the bearer of the e-check is its legitimate holder. This validation is carried out using the interactive proof of knowledge protocol defined in section 4.5. Moreover, P_e sends Q and $\langle p, g, q \rangle_{P_e}$ to P_r. This information is signed by P_e to prevent impersonation attacks [121].

Step 2. P_r verifies the signed data received from P_e. If it passes validations, P_r can request I to generate an e-check. P_r provides I all information required to draw the e-check. We call this information pre-check (Δ), which contains the face value (Q), the public values provided by P_e ($\langle p, g, q \rangle_{P_e}$), the type of e-check to be issued (c_{id} = "bearer check") and his bank account number (w_{P_r}). The previous information is signed and sent along with his digital certificate ($Cert_{P_r}$).

Step 3. I has to verify the identity of P_r and the pre-check's signature. I has also to check if P_r is a legitimate client of I and he has funds, or even without funds, if he is authorized to be issued an e-check for that amount ($\Delta.Q$). We assume that this is the case (in other case an error message has to be sent). Then I diminishes the amount $\Delta.Q$ from the payer's account and issues a valid e-check, with the amount $\Delta.Q$ and the public value provided by P_e ($\langle p_{P_e} \rangle$). The identity of the bank (I_{id}) has to be embedded in the e-check because of legal requirements. I stores ($\langle \mathbb{EC}, \langle p, g, q \rangle_{P_e}, w_I \rangle$) for e-check validation purposes when transferring and depositing. w_I is the I's bank account from which the e-check must be cashed. This bank transfer guarantees the payment of the e-check to its bearer. Finally, I sends the e-check to P_r. Note that P_e remains anonymous in front of I.

Step 4. P_r verifies the received $\mathbb{EC}$. Now P_r can change this e-check with P_e, typically for a service or a product, and involving a protocol for fair exchange of

$$m_1 = Q, \langle p, g, q \rangle_{P_e}$$
$$m_2 = \Delta = c_{id}, Q, \langle p, g, q \rangle_{P_e}, w_{P_r}$$
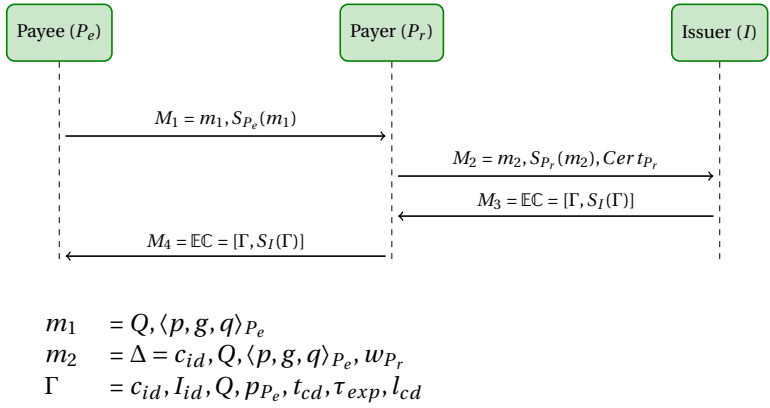$$\Gamma = c_{id}, I_{id}, Q, p_{P_e}, t_{cd}, \tau_{exp}, l_{cd}$$

Figure 4.2: Payment Protocol Flow

values. This kind of protocols can be found in the literature [122, 123, 124], and it is out of the scope of part of this dissertation.

Step 5. P_e can verify that the $\mathbb{EC}$ received from P_r is a valid e-check, issued by I and that she is the owner of the $\mathbb{EC}$ using her secret value $\langle s_{P_e} \rangle$.

This first protocol could be divided into two protocols: one to obtain the e-check and the other one to pay with it. But in fact, it would be a waste of time and resources, as explained next. Let us suppose we use two different protocols: one to obtain an e-check and the other one to pay with it. In this case, the payer requests the issue of an e-check to the bank. The payer provides his public value from the PK scheme, along with the required information. Then the bank issues the e-check embedding that data. Later, the payer can run the payment protocol; the payer must contact the bank again, and must prove he is the legitimate owner of the e-check executing the interactive proof of knowledge protocol defined in section 4.5. Finally, the payee must provide her public value from the PK scheme and the bank must sign a new e-check embedding the new public value.

As we can see, the above process requires that some of the protocols steps have to be run several times. For example, the bank must sign the e-check twice. In our proposal, the payer can directly request an e-check and pay with it in the same process. This way, we reduce the computational cost on the participants, as we will show in section 4.7.4.

Payment resolution protocol

If the bank I does not send the message of *Step* 3 in the payment protocol (see figure 4.3), I has the consent of P_r to transfer an amount Q from P_r's bank account to I's bank account, but P_r has not received the e-check corresponding to that transfer.
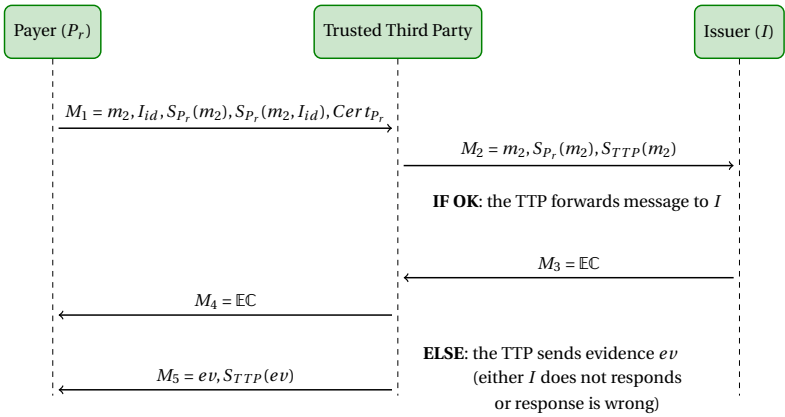
ev problem with I and P_r's consent: Δ, w_{P_r}, τ_u
m_2 $= \Delta = c_{id}, Q, \langle p, g, q \rangle_{P_e}, w_{P_r}$

Figure 4.3: Payment Resolution Protocol Flow

So, P_r has to contact the TTP to resolve the conflict, following the next steps (see figure 4.3):

Step 1. P_r sends the TTP the information he sent to I at *Step* 2 in the payment proto-col ($m_2 = \Delta$ and $Cert_{P_r}$), along with the bank's identity (I_{id}). That informa-tion is the P_r's consent to issue an e-check for an amount Q and to withdraw an amount Q from the P_r's bank account.

Step 2. The TTP checks the information provided by P_r. If it passes validations, the TTP will try to contact I; if I replies according to the protocol, i.e., it sends the issued $\mathbb{EC}$ to the TTP m_3, the TTP will retransmit $\mathbb{EC}$ to P_r and the protocol will finalize successfully. If I does not reply according to the protocol, the TTP will send an evidence signed message to P_r ($ev, S_{TTP}(ev)$), indicating a problem with I and P_r's consent (Δ) at time τ_u, in order P_r can use it in front of a court. If necessary, this evidence allows P_r to recover the money transferred by I from P_r's bank account.

4.6.3   Deposit/Cashing Protocol

An electronic bearer bank check can be deposited or cashed (as explained in section 4.6.1). The main difference is that, in the first case, the bearer (payer or payee) of the e-check loses his anonymity (but he can transfer it anonymously, see section 4.6.4), while in the second one the bearer can remain anonymous. We assume that the bearer of an e-check (P) wants to deposit it at his bank (I), and remember that,

130

for clarity, the payer and the payee are working with the same bank. If it is not the case, the bank of the payee (A) should have to contact with the bank of the payer I (a clearance process) in order to verify that the e-check is valid: it has not been deposited or transferred previously. The deposit protocol is as follows (see figure 4.4):

Step 1. When depositing, P has to send a valid $\mathbb{EC}$ to I. Additionally P has to provide his identity ($Cert_P$) in order to deposit the amount Q in the correct bank account w_P (in a cashing procedure P is not compelled to send neither his identity nor w_P). Moreover, P must generate and send a commitment value (pok_{Co}) following the PK scheme. Similarly to the payment protocol, P signs that information to be used with two purposes: as the P's consent to deposit $\mathbb{EC}$ and as a mechanism to prevent impersonation attacks.

Step 2. I verifies: w_P is a valid bank account at I, $\mathbb{EC}$ is a valid e-check (valid signature and non-expired) and it has not been deposited previously. If the e-check is in the I's database of used e-checks, it means that it is a trial of overspending the e-check. In this case I signs a proof of this fact, evidence that can be used if disputes arise. If the e-check has not been used, I must generate a challenge value (pok_{Ch}) following the PK scheme. I sends the PK along with a message indicating that the e-check is blocked (pending of finishing the deposit procedure).

Step 3. P checks if the status of $\mathbb{EC}$ is blocked, and uses his public value (p_P) and the challenge value (pok_{Ch}) sent by I to calculate the response value (pok_R). This value is encrypted with I's public key and sent to I.

Step 4. I decrypts the value pok_R. Now the bank verifies if P is the owner of $\mathbb{EC}$ using the equation (1), section 4.5. If it passes validation, it proves P is the legitimate owner of the e-check. Finally, I registers $\mathbb{EC}$ as an used e-check for future trials of overspending, and increases P's account by $\Gamma.Q$. If the verification fails, I sends an error message to P.

In case of cashing, I pays, over the counter, the face value of the e-check $\Gamma.Q$ to P. Obviously, in this case P has to physically be at the bank for the cashing of the e-check.

Step 5. P verifies the message from I to know if the $\mathbb{EC}$ has been deposited or not.

Deposit/cashing resolution protocol

We can differentiate three cases depending on the entity who is not acting properly and the step at which the protocol was interrupted. All resolution protocols have a similar structure as the explained in Figure 4.3, but there are two main differences: the message sent by the entity who requests the conflict resolution, and the evidence generated by the trusted third party. Thus, for simplicity, we will only explain the main goal of each resolution protocol.
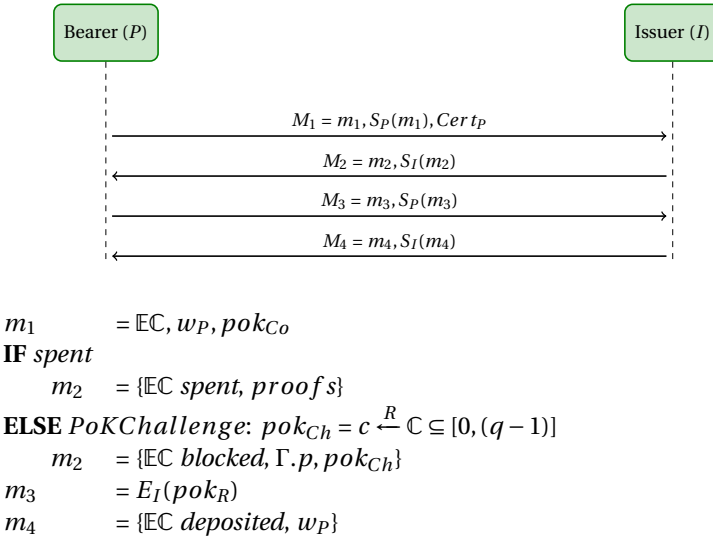
Figure 4.4: Deposit Protocol Flow

m_1 $= \mathbb{EC}, w_P, pok_{Co}$

IF *spent*

     m_2 $= \{\mathbb{EC} \; spent, proofs\}$

ELSE *PoKChallenge*: $pok_{Ch} = c \xleftarrow{R} \mathbb{C} \subseteq [0, (q-1))]$

     m_2 $= \{\mathbb{EC} \; blocked, \Gamma.p, pok_{Ch}\}$

m_3 $= E_I(pok_R)$

m_4 $= \{\mathbb{EC} \; deposited, w_P\}$

- In the first case, if I does not proceed with the message of *Step* 2 (m_2), unintentionally or intentionally[1], P could not deposit the e-check before its expiration date ($\Gamma.\tau_{exp}$), causing a loss for P. Thus, P has to contact the TTP sending the deposit request previously sent to I (m_1). The TTP checks the information provided by P. If it passes validations, the TTP will try to contact I; if I replies according to the protocol (m_2), the TTP will retransmit the information to P, and the protocol will finalize successfully. If I does not reply according to the protocol, the TTP will send a signed evidence to P ($ev, S_{TTP}(ev)$), indicating a problem with I and the attempt of deposit at time (τ_u) ($ev =$ problem with I and EC P's deposit, w_P, τ_u).

- In the second case, if I does not proceed with the message of *Step* 4 (m_4), P has not received a deposit confirmation. Thus, P has to contact the TTP sending the e-check blocked message received from I ($m_2, S_I(m_2)$) along with the response value previously sent to I ($m_3, S_P(m_3)$). The TTP checks the information provided by P. If it passes validations, the TTP will try to contact I. If I replies according to the protocol (m_4), the TTP will retransmit the information to P, and the protocol will finalize successfully because the e-check was deposited and P has a deposit confirmation from I. If I does not reply according to the protocol, the TTP will send a signed evidence message to P ($ev, S_{TTP}(ev)$), indicating a problem with I and the attempt of deposit at time τ_u ($ev =$ problem with I and EC P's deposit, w_P, τ_u).

---

[1] For example, P could be under a DoS attack [125, 126].

- In the third case, if P does not proceed with the message of *Step* 3 (m_3), it means that the e-check will remain blocked and perhaps it is an attempt of trap (the real owner of the e-check can be another user). I has to contact the TTP to resolve the conflict. To do this, I sends the TTP the deposit request received from P ($m_1, S_P(m_1)$) along with the e-check blocked message ($m_2, S_I(m_2)$) . The TTP will try to contact P; if P replies with the response value ($m_3 = E_I(pok_R)$), the TTP will retransmit the information to I, and the deposit protocol will continue its execution. If P does not reply according to the protocol, the TTP will send a signed evidence to I ($ev, S_{TTP}(ev)$), authorizing I to unblock the e-check at time τ_u ($ev = $ the TTP authorizes I to unblock EC, τ_u).

Note that the e-check is anonymous, and the owner of the e-check P has not been identified by the bank during the payment or the transfer process. As the TTP must contact P, P should provide to the bank an anonymous e-mail address at the deposit stage. This way, the TTP can contact P, and in turn P can maintain his anonymity.

4.6.4 Transfer Protocol

In this case, the owner (P_r) of an e-check wants to transfer it to another user (P_e), and the latter wants to remain anonymous, i.e. P_r probably knows P_e, but P_e wants nobody else to know she will be the owner of the e-check (even the bank). The transfer protocol is as follows (see figure 4.5):

Step 1. In the first step P_r sends to P_e the e-check, in order P_e can verify P_r has a valid e-check.

Step 2. P_e verifies that the e-check is valid (valid signature, non-expired) and setups PK scheme. From this setup, P_e obtains a secret value ($\langle s_{P_e} \rangle$), and a public one ($\langle p_{P_e} \rangle$). These values are analogous to those generated by P_e at the payment phase (see section 4.6.2). The set of public values ($\langle p, g, q \rangle_{P_e}$) is signed by P_e (to prevent impersonation attacks [121]), and sent to P_r.

Step 3. P_r verifies P_e's digital signature of $\langle p, g, q \rangle_{P_e}$. Next, P_r demands to his bank to change the public value linked to $\mathbb{EC}$. P_r has to send to I the e-check and the public values $\langle p, g, q \rangle_{P_e}$ protected in order an impostor could not change them. Along with these values, P_r must generate and send a commitment value pok_{Co} starting the interactive three steps PK protocol. This way, P_r will prove he is the legitimate owner of $\mathbb{EC}$ and in turn he is authorized to transfer it.

Step 4. I verifies that the e-check is not spent yet, as in *Step* 2 of the deposit protocol. If the e-check has not been previously used, the status of the e-check changes to blocked waiting for a future action from P_r (or the TTP if conflicts arise). It means that neither I nor other users can do nothing with that

e-check. I generates a challenge value (pok_{Ch}) using the commitment value (pok_{Co}) sent by P_r and following the PK scheme. After that, I sends a message containing pok_{Ch} and indicating that the e-check is blocked (pending of finishing the transfer procedure) for the values $\Gamma.p_{P_r} - p_{P_e}$.

Step 5. Similarly to *Step* 3 in the deposit protocol, P_r checks if the status of $\mathbb{EC}$ is blocked, and uses his public p_{P_r} value and the challenge value pok_{Ch} sent by I to calculate the response value pok_R. This value is encrypted with I's public key and sent to I.

Step 6. I decrypts the value pok_R, and uses the equation (1) (see section 4.5) to verify if P_r is the owner of $\mathbb{EC}$: $g^{pok_R} \stackrel{?}{\equiv} pok_{Co} \cdot (\Gamma.p_{P_r})^{pok_{Ch}}$. If it passes validation, I changes the $\mathbb{EC}$'s owner adding P_e's public value (p_{P_e}) provided by P_r in *Step* 3 of this protocol. I stores the collected proofs during the transfer process linked to the e-check, and the modified $\mathbb{EC}'$ is sent to P_r.

Step 7. P_r can send the modified e-check $\mathbb{EC}'$ to P_e. Probably this e-check will be the item exchanged for another item using a protocol for fair exchange of values (this part is out of scope of this protocol).

Step 8. Finally, P_e can verify she has a valid e-check, and more important, that she is the only entity that can deposit or transfer this e-check, because she is the owner of the secret value ($\langle s_{P_e} \rangle$) linked to $\mathbb{EC}'$ via $\Gamma.p_{P_e}$.

Now P_e can deposit or transfer $\mathbb{EC}'$. We want to remark that the protocol is anonymous for originator (P_r) and recipient (P_e) in front of the bank. I will only know that this e-check was issued some day to somebody, but now it can be transferred anonymously to other entities. Moreover, and a key issue, the secret value is never disclosed to any party other than the e-check owner.

Transfer resolution protocol

The entity transferring the e-check or the bank could not fulfil the flow of the transfer protocol. We can differentiate three cases depending on the entity who is not acting properly and the step at which the protocol was interrupted, similarly to the deposit resolution protocol.

- In the first case, if I does not proceed with message of *Step* 4 (m_4), P_r cannot transfer the e-check to P_e. Thus, P_r has to contact the TTP sending the transfer request previously sent to I (m_3). The TTP checks the information provided by P_r. If it is correct, the TTP will try to contact I; if I replies with the e-check blocked message, waiting transfer completion ($m_4, S_I(m_4)$), the TTP will retransmit the information to P_r, and the protocol will continue its execution. If I does not reply according to the protocol, the TTP will send a signed evidence to P ($ev, S_{TTP}(ev)$), indicating a problem with I and the attempt of transfer at time τ_u ($ev = $ problem with EC's transfer, $\Gamma.p_{P_r} - p_{P_e}, \tau_u$).

$$m_2 = \langle p, g, q \rangle_{P_e}$$
$$m_4 = \{\mathbb{EC}\; blocked, \Gamma.p_{P_r} - p_{P_e}, pok_{Ch}\}$$
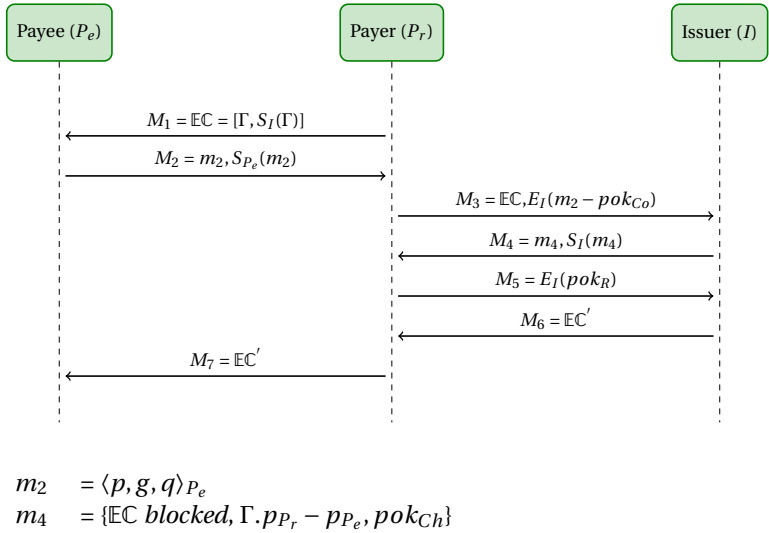
Figure 4.5: Transfer Protocol Flow

- In the second case, if I does not proceed with message of *Step* 6 (m_6), P_r has not received the modified e-check. This case is similar to the explained above. The main difference is the proof provided by P_r to the TTP (m_4, m_5). But it has the same resolution flow, with the same evidence provided by the TTP in case I does not respond.

- In the third case, if P_r does not proceed with message of *Step* 5 (m_5), the e-check will remain blocked. This case matches up with the resolution case explained in section 4.6.3. I must send the TTP the transfer request received from P_r (m_3) along with the e-check blocked message ($m_4, S_I(m_4)$). The TTP will try to contact P_r and if P_r replies with the response value ($m_5 = E_I(pok_R)$), the TTP will retransmit the information to I, and the transfer protocol will continue its execution. If P_r does not reply according to the protocol, the TTP will send a signed evidence to I ($ev, S_{TTP}(ev)$), authorizing I to unblock the e-check at time τ_u ($ev =$ the TTP authorizes I to unblock EC, τ_u).

In order to allow the TTP to contact P_r anonymously, the same process explained in section 4.6.3 (anonymous e-mail) can be used.

4.7 Discussion of Requirements Fulfilment

In this section we present an analysis of our scheme in order to prove that it fulfils the security, functional and juridical requirements defined in section 4.3.

4.7.1 Security Requirements

First we will review the accomplishment of the core fair exchange requirements.

CLAIM 4.1: **e-check Effectiveness**: *If all participants in a transaction follow the protocol rules, the TTP will not intervene.*

PROOF. In section 4.6.2 we have the description of the payment protocol, and as explained, only the payee P_e, the payer P_r and the issuer I are involved in a correct execution of the protocol (see figure 4.2). In section 4.6.3 we have the description of the deposit/cashing protocol, and as defined by the protocol, only the bearer B and the issuer I are involved in an execution (see figure 4.4). In section 4.6.4 we have the definition of the transfer protocol flow, and as it is explained, only the payee P_e, the payer P_r and the issuer I are involved in a correct protocol execution. Therefore we can affirm that in any transaction defined in the e-check scheme presented in this section: payment, deposit and transfer, the TTP will only intervene in case of dispute.

CLAIM 4.2: **e-check Fairness**: *At the end of an exchange between two or more entities, either everybody achieve the expected items, or nobody can stand in a privileged situation.*

PROOF. If we assume P_r and P_e are honest, whatever the behaviour of I, P_r and P_e maintain fairness. There are three possibilities in which I can try to obtain advantage in front of clients:

1. At the payment phase I can obtain the consent of P_r to transfer the money from P_r's bank account to a guaranteed I's bank account, but P_r does not receive the corresponding $\mathbb{EC}$ from I. P_r will contact the TTP and if I does not respond, P_r will receive proof to cancel the transfer from his account to I's bank account. If I responds, P_r will obtain the corresponding $\mathbb{EC}$.

2. At the deposit phase, two possibilities can arise:

   a) I ignores the request from P_r to deposit the $\mathbb{EC}$. Due to the $\mathbb{EC}$ has an expiration time, P_r could lose the money. P_r will contact the TTP and if I does not respond, P_r will receive proof with a time mark indicating I is misbehaving at the deposit phase and before $\mathbb{EC}$ has expired.

   b) After *Step* 3 I has the secret value linked to the $\mathbb{EC}$, so I has all the required information about the $\mathbb{EC}$. If the bearer of the $\mathbb{EC}$ does not receive confirmation of deposit from I, the bearer will contact the TTP and if I does not respond, he will receive proof demonstrating I has not made the deposit.

3. At the transfer phase, I could prevent transferring $\mathbb{EC}$, similar to the deposit phase explained above. Thus, P_r will contact the TTP. If I does not respond, P_r will receive proof to cancel the transfer. If I responds, P_r will obtain the new $\mathbb{EC}$.

If we assume I is honest, whatever the behaviour of P_r and P_e, I maintains fairness. There are two possibilities in which participants can try to obtain advantage in front of I: when transferring and when depositing. In both cases, the same problem appears: the $\mathbb{EC}$ could remain blocked if P does not follow *Step* 3 when depositing or *Step* 5 when transferring. In order to unblock the $\mathbb{EC}$, I contacts and sends to the TTP the evidence proving P's intention to transfer or to deposit the $\mathbb{EC}$ ($S_P(\mathbb{EC}, w_P)$). The TTP provides I with the authorization to unblock the $\mathbb{EC}$ if the bearer of the $\mathbb{EC}$ does not respond, allowing I to accept new operations with the $\mathbb{EC}$.

Fairness of the protocols for paying, depositing and transferring is guaranteed, if necessary with the involvement of the TTP. If parties behave according to the protocols, they will have at the end of the execution what they want: a payment, a deposit or a transfer for an e-check. If some party misbehaves (or even if an unfair situation arises due to communication problems), the TTP will solve the unfair situation or will provide proof of misbehaviour in order to repair it in front of a court.

CLAIM 4.3: **e-check Timeliness**: *Every protocol execution will finish in a finite amount of time, without any participant losing fairness.*

PROOF. In the case of the payment protocol, the role of the issuer I is to answer the payer P_r request of an e-check, therefore as soon as it answers it can consider the protocol finished. The resolution protocol allows the payer P_r to contact the TTP and claim the check if he does not receive it from the issuer I. Therefore we can affirm that the issuer I and the payer P_r can be sure the protocol finishes in a finite amount of time, either because they follow the protocol flow or they contact the TTP. And in any situation fairness is maintained. In the case of the payee P_e, the exchange of messages M_1 and M_4 (figure 4.2) can be part of another fair exchange (another payment protocol), which is out of the scope, e.g., M_1 can be considered as the end of a negotiation phase, and M_4 the first message of a fair exchange of values protocol, where they exchange an e-check for a service/receipt or digital goods. Therefore, as a fair exchange protocol it meets the timeliness requirement.

In the case of the deposit protocol (figure 4.4), if the bearer P fails to receive M_2 or M_4 he can contact the TTP to resolve the protocol execution, whereas the issuer I can do the same if it fails to receive M_3, therefore we can affirm that both bearer P and issuer I can be sure that the protocol execution will end in a finite amount of time, and that they will maintain fairness.

In the transfer protocol (figure 4.5), the payer P_r can contact the TTP if he fails to receive M_4 or M_6 from the issuer I, whereas the issuer can contact the TTP if it fails to receive M_5 from the payer P_r. In any case, both (P_r and I) can be sure the protocol execution will finish either because they follow the protocol flow or because they contact the TTP, and that they will maintain fairness. The payee P_e is in the same situation that in the payment protocol, his exchange of messages with the payer P_r is considered a fair exchange of values, out of the scope of the e-check scheme. But as fair exchange protocol it must meet the timeliness requirement.

Therefore, we can affirm that any transaction in the e-check scheme will finish in a finite amount of time either because the participants follow the corresponding

protocol or because they contact the TTP, timeliness is met.

CLAIM 4.4: **e-check Non-repudiation**: *None of the participants in a transaction can deny having participated nor be excluded of it.*

PROOF. In the payment protocol (figure 4.2), the payer P_r requests an e-check sending M_2, a message that includes his signature, therefore he cannot deny having requested it. The issuer I answers with the e-check $\mathbb{EC}$ which includes its identity, therefore he cannot deny having issued it.

In the deposit protocol (figure 4.4), both bearer P and issuer I sign their corresponding messages, therefore they cannot deny having requested the deposit of an e-check or having deposited it. In the case of cashing the e-check, P does not sign any message, but uses his public value p_P and the challenge value pok_{Ch} sent by I to generate the response value pok_R, proving he is the owner. Therefore I can use this pok_R to prove the owner cashed the e-check.

In the transferring protocol (figure 4.5), the payer P_r does not sign the messages he sends to the issuer I, but he proves he is the owner following the PK scheme (as in the cashing protocol), whereas the issuer signs all its messages. Therefore the payer P_r as owner of the e-check cannot deny having requested the transfer of it (he cannot claim ownership of an e-check and deny having transferred it), and the issuer cannot deny having changed the e-check's ownership.

Thus, we can conclude that the e-check scheme meets the non-repudiation requirement.

Following we review the specific security requirements.

CLAIM 4.5: **Unforgeability**: *Only authorized entities can issue valid e-checks.*

PROOF. Only I can issue a valid $\mathbb{EC}$, and only an $\mathbb{EC}$ signed by I can be valid. Therefore forgery is not possible. If an entity tries to modify information contained in an $\mathbb{EC}$, the validation of the signature of that $\mathbb{EC}$ allows detecting whether its content has been modified. The forgery of a complete $\mathbb{EC}$ is not feasible because the payment and the transfer procedures require the knowledge of I's private key in order to create a valid $\mathbb{EC}$. Therefore nobody but I can create a valid $\mathbb{EC}$ because only I knows the required private key.

CLAIM 4.6: **Non-overspending**: *An e-check only belongs to one user at a given time.*

PROOF. When transferring or depositing an $\mathbb{EC}$, I must verify if the presented $\mathbb{EC}$ is already spent, i.e., if the status of the $\mathbb{EC}$ in I's database is spent. This is possible because the scheme is online, thus, I is involved in each procedure, and he must verify the status of the $\mathbb{EC}$.

The $\mathbb{EC}$'s status changes to spent when depositing, so an $\mathbb{EC}$ can be spent only once since the bank maintains a database of $\mathbb{EC}$s that have been used, and therefore it is impossible the overspending of an $\mathbb{EC}$.

CLAIM 4.7: **Anonymity**: *Payers and payees remain anonymous in non-identified processes related to the e-check.*

PROOF. At the payment phase, we can observe that the receiver (P_e) remains anonymous in front of the bank (I). This entity does not know who will hold the e-check rights. The information which I can access is P_r's identity and the set of public values generated by P_e ($\langle p, g, q \rangle_{P_e}$). But I cannot infer P_e's identity from that set of public values. It is even possible to pay with the e-check maintaining the anonymity of the sender (perhaps other problems can arise related to the purchase if the client is not identified, but not for the payment itself). The same happens at the transfer phase: the receiver (P_e) is not identified to carry out the operation. P_r can transfer the e-check and only the receiver will be able to identify who P_r is. With respect the $\mathbb{EC}$, it conveys information about I's identity but not about the holder of the $\mathbb{EC}$. The only data generated by the holder and included in the $\mathbb{EC}$ is p_{P_e}, but it does not reveal any data about P_e's identity. Therefore, any entity other than a bank's client requesting the issue of an e-check or depositing the e-check in his bank account, remains anonymous.

CLAIM 4.8: **Transferability**: *A payee can transfer a received e-check to another payee without depositing or cashing the e-check, and without losing anonymity.*

PROOF. It is not necessary to deposit a received e-check: an e-check can be transferred. When P_r transfers an e-check to P_e, P_r must prove he is the actual owner of this e-check and request I to change the public value of the PK (p_{P_r}) embedded in the $\mathbb{EC}$ to a new one, p_{P_e}. At the end of the process, the new $\mathbb{EC}$ owner (P_e) has an electronic check that contains her p_{P_e}. So, she is the only party that knows the secret value linked to that $\mathbb{EC}$. In addition, I has marked as used the public value p_{P_r}. Thus, P_r can no longer prove he is the owner of the $\mathbb{EC}$. Now, nobody but P_e can transfer or deposit the $\mathbb{EC}$.

Moreover, the requirements of unforgeability, non-overspending and anonymity are met when transferring, as proved in claims 4.5, 4.6 and 4.7, respectively. Therefore, the transferability requirement is met.

CLAIM 4.9: **Untraceability**: *An e-check cannot be linked to a particular payee in a chain of transfers in front of the bank.*

PROOF. A payee does not need to reveal her identity neither in the payment protocol nor in the transfer protocol to I. As explained in claims 4.7 and 4.8, the anonymity of a payee is guaranteed during the payment and transfer protocol. Of course, the first payer (client of the bank who requests the issue of an e-check) and the last payee (who deposits the e-check in her bank account) are the two entities that can be traced. In fact, the e-check does not contain personal or related information of involved entities. Therefore, the intermediaries entities in the chain of transfers cannot be identified by the bank.

4.7.2 Functional Requirements

CLAIM 4.10: **Online Operations**: *Payment with an e-check, and transferring or depositing an e-check requires a persistent connection with the bank.*

PROOF. Deposit operations have to be online in order to verify the e-check is valid and avoid overspending. It may be questioned whether the transfer operation should be online based on service availability issues, but we insist that nowadays it is not a serious drawback (what about credit card operations in present millions of operations). However, this must be online for proving $\mathbb{EC}$ is not spent before transferring. As explained in section 4.6 all the protocols require the online involvement of the bank.

CLAIM 4.11: **Efficiency**: *Processing an e-check is not computer consuming, mainly, on the client side.*

PROOF. We proof this claim through a performance comparison conducted in section 4.7.4.

4.7.3 Juridical Requirements

CLAIM 4.12: **Content**: *The content of an e-check includes all the required data.*

PROOF. $\mathbb{EC} = [\Gamma, S_I(\Gamma)]$ where $\Gamma = (c_{id}, I_{id}, Q, p_X, t_{cd}, \tau_{exp}, l_{cd})$ depicts the content of the e-check. As explained in section 4.6.1, $\mathbb{EC}$ contains both mandatory and non mandatory data required for an electronic bearer bank check as specified in section 4.1, where the identity of the drawer is I.

CLAIM 4.13: **Drawee and drawer**: *The bank is the drawer and the drawee of the e-checks.*

PROOF. $\mathbb{EC} = [\Gamma, S_I(\Gamma)]$ where $S_I(\Gamma)$ depicts the digital signature of the entity who signs the e-check. In this case, the drawer is I, because I issues the e-check. Moreover, the content of the e-check must include the drawee (the entity that must make the payment), and we can observe the drawee is also I ($\Gamma.I_{id}$). Therefore, I is drawer and drawee of the e-check.

4.7.4 Performance Comparison

We conduct a performance comparison among our scheme and Tsao's scheme [106], the only previous solution designed for electronic bearer check scenario, as explained in section 4.4. For the comparison, we consider the computational cost measured as the operations complexity performed in each protocol run by each involved entity. For simplicity and without loss of generality, we have obtained the computational cost as the number of long and short exponentiations. The rest of operations have been considered not significant compared to exponentiation operations. A hash function cost (e.g. SHA2) can be considered equivalent to two mod-

ular multiplications. Modular multiplications are negligible respect to exponentiations because an exponentiation with a l-bit exponent is the order of l modular multiplications. Thus, considering that a short exponentiation is about 160 bits, the number of modular multiplications required in all proposals is very low with regard to a modular exponentiation.

	Payment			Transfer			Deposit	
	P_r	P_e	I	P_r	P_e	I - TTS	P	I
[106]	$4E_l + 2E_s$	$2E_l + 6E_s$	$5E_s$	E_l	$3E_l + 4E_s$	E_l	0	$3E_l + 4E_s$
★	$E_l + 2E_s$	$E_l + 2E_s$	$E_l + E_s$	$6E_s$	$E_l + 3E_s$	$5E_l + 2E_s$	$E_l + 4E_s$	$4E_l + 2E_s$

E_s: short exponentiation, E_l: long exponentiation, ★ our proposal

Table 4.3: Performance Comparison

Table 4.3 presents an analytical evaluation of the above parameters. In the proposal of [106], the authors do not include certain operations required for validation purposes. The most obvious example is the deposit protocol, which requires only one step. It is clear that, at least, one more step is required, where the bank should send a signed confirmation of the result of the deposit, and the bearer should check the validity of that response. So, it is unrealistic that the bearer should not perform any operation, as reflected in table 4.3. In contrast, our scheme includes all the messages needed to provide confirmation of the state in each protocol run. Still, our approach improves the performance on the payer/payee side. For example, regarding to a transfer process, our protocol reduces the number of required operation by 1.5 at the payer side and by 2.5 at the payee side with regard to the transfer process proposed in [106]. This outcome is similar in the payment protocol, where the payer reduces by 3.5 the number of required operations and the payee requires around less than half of the operations of Tsao's scheme [106].

Finally, the results support our proposal to perform the protocols of issuing and payment in the same process. Our payment protocol is equivalent to the delegation and payment protocol in [106], so, as shown in table 4.3, the total cost on the payer and payee is reduced in our scheme, by 3.5 at the payer side, and by 2 at the payee side.

4.8 Conclusions

In this chapter of the dissertation a fair and efficient electronic bearer bank check scheme has been presented. As bearer check scheme, the identity of the owner is not included in the e-check, which offers him anonymity, and as a bank check, the e-check funds are assured. The solution proposed meets the core requirements for fair exchange protocols (effectiveness, fairness, timeliness and non-repudiation), and five specific security requirements for e-checks. Two of them are generic to e-check solutions, unforgeability and non-overspending, and the others are achieved with new mechanisms that improve the security and functionality of the scheme,

anonymous transferability and untraceability. Moreover, the proposed scheme accomplishes a security level similar to crossed checks because the holder of an e-check must prove he is its legitimate owner, while maintaining his anonymity. Finally, the electronic bearer bank check scheme provides a lightweight solution for the client side because this is, normally, the more restrictive participant in terms of computational resources.

5

CONCLUSIONS

This dissertation has presented contributions on fair exchange applied to e-commerce (digital signature of contracts and payment systems), and certified e-mail. Following we present these contributions, as well as the future work.

Digital Signature of Contracts.
This dissertation has presented digital Multi-Party Contract Signing (MPCS) solutions for two new scenarios, previously not addressed in the fair exchange literature, the Atomic Multi-Two Party (AM2P) and the Agent Mediated Scenario (AMS). In the AM2P scenario a consumer wants to sign a contract with a set of providers, with two particularities: she wants to sign a different contract with each provider, and she wants either all contract signatures or none. Whereas in the AMS, consumers access to the providers through intermediaries. These intermediaries are the focus of the scenario. On one side, an intermediary has to provide a client with some services, while on the other side, he has to acquire those services from another provider, which, in turn, may have to contact other providers to acquire the services requested. Both proposals are optimistic, i.e., if every participant behaves according to the protocol rules, and there are no network errors, the TTP will not intervene in the execution. Regarding the security requirements, both protocols meet all the core security requirements: effectiveness, fairness, timeliness and non-repudiation, and two additional ones: verifiability of the TTP and confidentiality. Additionally, the AMS protocol meets a specific requirement, the traceability, which allows the TTP to identify all the participants in a transaction from a single request.

On the other hand, this dissertation presents a study of the efficiency of MPCS protocols from their architecture point of view. Four of the most used architectures (ring, sequential, star and mesh) are described, and with them the terms typically

used to define the efficiency of a protocol, such as message and round. In addition, we define the term transmission and instead of using the typical messages and round, we use this transmission term to measure the efficiency of a protocol. Finally, we present an optimal MPCS solution for each of the architectures defined, and we prove that those solutions are the most efficient, in terms of number of transmissions, setting up a new lower bound for the efficiency of MPCS protocols.

As future work in the line of digital signature of contracts, we can define two main lines of work. On one hand, we have the AMS protocol, which is the first to address the digital signature of contracts in presence of agent intermediaries. Further study on this field should address the efficiency of the protocols, and improve the security requirements such as traceability, allowing a TTP to automatically trace a transaction either backwards or forwards, without requiring to contact any participant. Even though the solution proposed is optimistic, further work should not discard online solutions, and must also consider other cryptographic primitives such as hash chains or verifiable encryption. On the other hand we have the study on the efficiency of MPCS protocols. Future work on this field should include the formalization of the base definitions, such as the security requirements and the terms message, transaction etc. This formalization would facilitate the study through simulations and implementation of the current proposals described and further ones. Moreover, the results may also be useful in other lines of work, such as formal verification of protocols.

Certified E-mail.

This dissertation has presented two optimistic Certified Electronic Mail (CEM) protocols designed with the goal of achieve feasible solutions easy to implement, use and deploy. Both protocols assume the participation of multiple Mail Transfer Agents (MTAs) in the delivery of a certified message, following the current e-mail architecture, therefore facilitating its integration in the e-mail infrastructure. The proposals mimic the asynchronous nature of e-mail clients, where the User Agents (UAs) synchronize with their MTA at their own will, which facilitates its assimilation by current e-mail users. The first proposal is the first CEM protocol that assumes that all the MTAs participating in an execution are untrusted, i.e., they can misbehave and collude. Whereas the second proposal takes a different approach, assuming that each UA trusts his own MTA, reducing the complexity of the first proposal. Both protocols meet all the core fair exchange requirements (effectiveness, fairness, timeliness and non-repudiation) and two additional ones (verifiability of TTP and confidentiality). In addition, the protocols also meet a specific requirement for CEM protocols, the non-selective receipt, which does not allow the recipient to decide whether to receive a CEM or not, depending on its content. Finally, since both protocols assume the participation of multiple MTAs, the proposed protocols provide all the required non-repudiation evidence: origin, submission, delivery and receipt.

Following the results presented in this dissertation, a natural extension of the work presented on CEM protocols would be to develop both proposals into real

applications. This goal implies the study of the current e-mail protocol standards (SMTP, POP3, etc.) to develop the necessary modifications and/or extensions to deploy both CEM proposals. The final goal of this line of work would be to make a proposal of standard for CEM services using the CEM protocols presented in this dissertation.

Payment Systems.

This dissertation presented a new e-check scheme as payment system. The proposed solution allows the users to maintain their anonymity when transferring and cashing the e-check, as paper bearer checks, while assuring the payment of the funds, as in a paper bank check. The proposal meets all the core security requirements for fair exchange protocols (effectiveness, fairness, timeliness and non-repudiation) and a set of specific ones for e-checks: unforgeability, non-overspending, anonymity, transferability and untraceability. Additionally, the e-check scheme proposed meets other requirements that have been defined as functional and juridical requirements. The proposed e-check scheme describes three protocols: a payment protocol where the issuer generates an e-check, the deposit/cashing protocol, where the bearer can decide whether to deposit the e-check funds in a bank account or to cash them, and a transfer protocol, where a payer decides to pay with an e-check, i.e., he transfers the ownership of the e-check.

Further work in the line of payment systems should expand the e-check scheme presented in this dissertation and include a proposal of payment by digital goods or receipt, to complete the payment and transfer protocols proposed. Since there are many kinds of paper based checks, further work should also consider the inclusion of these other kind of checks (e.g. nominal checks) into the e-check scheme, therefore obtaining an e-check scheme that would allow users to work with any kind of e-check.

BIBLIOGRAPHY

[1] C. European Parliament, "Directive 2002/22/ec of the european parliament and of the council of 7 march 2002 on universal service and users' rights relating to electronic communications networks and services (universal service directive)," Official Journal L 108 of April 24 2002, 2002.

[2] ——, "Directive 2009/136/ec of the european parliament and of the council of 25 november 2009 amending directive 2002/22/ec on universal service and users' rights relating to electronic communications networks and services, directive 2002/58/ec concerning the processing of personal data and the protection of privacy in the electronic communications sector and regulation (ec) no 2006/2004 on cooperation between national authorities responsible for the enforcement of consumer protection laws (text with eea relevance)," Official Journal I 337 of December 18 2009, 2009.

[3] J. del Estado, "Ley 2/2011, de 4 de marzo, de economia sostenible," BOE num. 55, de 5 de marzo de 2011, pp. 25033 - 25235, 2011.

[4] M. Blum, "How to exchange (secret) keys," *ACM Transactions on Computer Systems*, vol. 1, no. 2, pp. 175 – 193, 1983.

[5] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, pp. 637 – 647, 1985.

[6] T. Okamoto and K. Ohta, "How to simultaneously exchange secrets by general assumptions," in *Proceedings of the 2nd ACM Conference on Computer and communications security*, ser. CCS '94. New York, NY, USA: ACM, 1994, pp. 184 – 192.

[7] O. Markowitch and Y. Roggeman, "Probabilistic non-repudiation without trusted third party," in *2nd Conference on Security in Communication Networks*, ser. SCN'99, 1999.

[8] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest, "A fair protocol for signing contracts," *IEEE Transactions on Information Theory*, vol. 36, no. 1, pp. 40 – 46, 1990.

[9]   N. Asokan, V. Shoup, and M. Waidner, "Asynchronous protocols for optimistic fair exchange," in *IEEE Symposium on Security and Privacy*, ser. S&P'98.   Los Alamitos, CA, USA: IEEE Computer Society, 1998, pp. 86–99.

[10]   J. Zhou, R. Deng, and F. Bao, "Some remarks on a fair exchange protocol," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science.   Springer Berlin / Heidelberg, 2000, vol. 1751, pp. 46–57.

[11]   F. C. Gartner, H. Pagnia, and H. Vogt, "Approaching a formal definition of fairness in electronic commerce," in *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, ser. SRDS '99.   Washington, DC, USA: IEEE Computer Society, 1999, pp. 354–359.

[12]   H. Pagnia, H. Vogt, and F. Gärtner, "Fair exchange," *The Computer Journal*, vol. 46, no. 1, pp. 55–75, 2003.

[13]   S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Elsevier Computer Communications*, vol. 25, pp. 1606–1621, 2002.

[14]   M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party (extended abstract)," in *Proceedings of the 4th ACM conference on Computer and Communications Security*, ser. CCS'97.   New York, NY, USA: ACM, 1997, pp. 1–5.

[15]   S. Micali, "Certified e-mail with invisible post offices," in *An invited presentation at the 6th annual RSA Data Security conference*.   San Francisco, CA, USA: Available from author, 28-31 Jan. 1997.

[16]   X. Huang, Y. Mu, W. Susilo, W. Wu, J. Zhou, and R. H. Deng, "Preserving transparency and accountability in optimistic fair exchange of digital signatures," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 2, pp. 498–512, 2011.

[17]   H. Werthner and F. Ricci, "E-commerce and tourism," *Communications of the ACM*, vol. 47, pp. 101–105, 2004.

[18]   J. A. Garay and P. D. MacKenzie, "Abuse-free multi-party contract signing," in *Proceedings of the 13th International Symposium on Distributed Computing*. Bratislava, Slovak Republic: Springer-Verlag, London, UK, 27-29 Sep. 1999, pp. 151–165.

[19]   B. Baum-Waidner and M. Waidner, "Round-optimal and abuse-free optimistic multi-party contract signing," in *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*.   Geneva, Switzerland: Springer-Verlag, London, UK, 9-15 July 2000, pp. 524–535.

[20] R. Chadha, S. Kramer, and A. Scedrov, "Formal analysis of multi-party contract signing," in *Proceedings of the 17th IEEE workshop on Computer Security Foundations.* Pacific Grove, CA, USA: IEEE Computer Society, Washington, DC, USA, 28-30 June 2004, pp. 266–279.

[21] A. Mukhamedov and M. Ryan, "Improved multi-party contract signing," in *Proceedings of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, ser. FC'07/USEC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 179–191.

[22] S. Mauw, S. Radomirovic, and M. T. Dashti, "Minimal message complexity of asynchronous multi-party contract signing," in *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, ser. CSF '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 13–25.

[23] B. Kordy and S. Radomirovic, "Constructing optimistic multi-party contract signing protocols," in *Proceedings of the 25th Computer Security Foundations Symposium*, ser. CSF 2012. Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 215–229.

[24] T.-H. Chen, S.-C. Yeh, K.-C. Liao, and W.-B. Lee, "A practical and efficient electronic checkbook," *Journal of Organizational Computing and Electronic Commerce*, vol. 19, no. 4, pp. 285–293, 2009.

[25] C. Chin-Ling, W. Cheng-Hsiung, and L. Wei-Chech, "Improving an on-line electronic check system with mutual authentication," in *International Conference on Advanced Information Technologies (AIT)*, 2010.

[26] FSCT, "FSTC echeck initiative," Financial Services Technology Consortium (URL: http://www.echeck.org), 2005.

[27] B. Baum-Waidner, "Optimistic asynchronous multi-party contract signing with reduced number of rounds," in *Proceedings of the 28th International Colloquium on Automata, Languages and Programming.* Crete, Greece: Springer-Verlag, London, UK, 8-12 July 2001, pp. 898–911.

[28] A. Mukhamedov and M. D. Ryan, "Fair multi-party contract signing using private contract signatures," *Elsevier Information and Computation*, vol. 206, pp. 272–290, 2008.

[29] Y. Liu, "An optimistic fair protocol for aggregate exchange," in *Proceedings of the 2009 Second International Conference on Future Information Technology and Management Engineering.* Sanya, China: IEEE Computer Society, Washington, DC, USA, 13-14 Dec. 2009, pp. 564–567.

[30] L. Yanping and P. Liaojun, "Multi-party non-repudiation protocol with different message exchanged," in *Proceedings of the 2009 Fifth International Conference on Information Assurance and Security - Volume 01.* Xi'an, China: IEEE Computer Society, Washington, DC, USA, 18-20 Aug. 2009, pp. 491–494.

[31] J. A. Onieva, J. Zhou, M. Carbonell, and J. Lopez, "A multi-party non-repudiation protocol for exchange of different messages," in *18th IFIP International Information Security Conference. Security and Privacy in the Age of Uncertainty (IFIP SEC'03).* Athens, Greece: Kluwer Academic Publishers, Dordrecht, the Netherlands, 26-28 May 2003, pp. 37–48.

[32] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous protocols for optimistic fair exchange," in *Proceedings of the IEEE Symposium on Security and Privacy, 1998.*, ser. SP '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 86–99.

[33] J. L. Ferrer-Gomilla, J. A. Onieva, M. Payeras, and J. Lopez, "Certified electronic mail: Properties revisited," *Elsevier Computers & Security*, vol. 29, pp. 167–179, 2010.

[34] S. Mauw, S. Radomirovic, and M. T. Dashti, "Minimal message complexity of asynchronous multi-party contract signing," in *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium.* Port Jefferson, NY, USA: IEEE Computer Society, Washington, DC, USA, 8-10 July 2009, pp. 13–25.

[35] J. A. Garay, M. Jakobsson, and P. D. MacKenzie, "Abuse-free optimistic contract signing," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '99. London, UK: Springer-Verlag, 1999, pp. 449–466.

[36] M. Franklin and G. Tsudik, "Secure group barter: Multi-party fair exchange with semi-trusted neutral parties," in *Financial Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998, vol. 1465, pp. 90–102.

[37] A. Mukhamedov, S. Kremer, and E. Ritter, "Analysis of a multi-party fair exchange protocol and formal proof of correctness in the strand space model," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3570, pp. 578–578.

[38] I. Khill, J. Kim, I. Han, and J. Ryou, "Multi-party fair exchange protocol using ring architecture model," *Elsevier Computers & Security*, vol. 20, pp. 422 – 439, 2001.

[39] Y. Liu, "An optimistic fair protocol for aggregate exchange," in *Proceedings of the 2009 Second International Conference on Future Information Technology and Management Engineering*, ser. FITME'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 564 – 567.

[40] J. A. Onieva, J. Zhou, J. Lopez, and M. Carbonell, "Agent-mediated non-repudiation protocols," *Elsevier Electronic Commerce Research and Applications*, vol. 3, pp. 152–162, 2004.

150

[41] J. Ferrer-Gomila, M. Payeras-Capellà, and L. Huguet-Rotger, "Efficient optimistic n-party contract signing protocol," in *Information Security*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001, vol. 2200, pp. 394–407.

[42] *X.667 : Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 object identifier components*, ITU, August 2008.

[43] X. C. W. Group, *Extensible Markup Language (XML)*, 5th ed., World Wide Web Consortium (W3C), November 2008, w3C Recommendation.

[44] X. S. S. M. W. Group, *XML digital signature*, 2nd ed., World Wide Web Consortium (W3C), June 2008, w3C Recommendation.

[45] X. E. W. Group, *XML Encryption*, 1st ed., World Wide Web Consortium (W3C), November 2002, w3C Recommendation.

[46] W. X. S. W. Group, *XML schema*, 2nd ed., World Wide Web Consortium (W3C), October 2004, w3C Recommendation.

[47] B. Baum-Waidner, "Optimistic asynchronous multi-party contract signing with reduced number of rounds," in *Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*, ser. ICALP '01. London, UK: Springer-Verlag, 2001, pp. 898–911.

[48] A. Mukhamedov and M. D. Ryan, "Resolve-impossibility for a contract-signing protocol," in *Proceedings of the 19th IEEE workshop on Computer Security Foundations*, ser. CSFW '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 167–176.

[49] J. Ferrer-Gomila, M. Payeras-Capellà, and L. Huguet-Rotger, "Optimality in asynchronous contract signing protocols," in *Trust and Privacy in Digital Business*, ser. Lecture Notes in Computer Science. Berlin/Heidelberg: Springer, 2004, vol. 3184, pp. 200–208.

[50] J. A. Onieva, J. Zhou, and J. Lopez, "Attacking an asynchronous multi-party contract signing protocol," in *Proceedings of the 6th international conference on Cryptology in India*, ser. INDOCRYPT'05, vol. 3797. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 311–321.

[51] ——, "Multiparty nonrepudiation: A survey," *ACM Computer Surveys*, vol. 41, no. 1, pp. 5:1–5:43, 2009.

[52] Y. Zhang, C. Zhang, J. Pang, and S. Mauw, "Game-based verification of multi-party contract signing protocols," in *Proceedings of the 6th international conference on Formal Aspects in Security and Trust*, ser. FAST'09. Berlin/ Heidelberg: Springer-Verlag, 2009, pp. 186–200.

[53]   A. Tauber, "A survey of certified mail systems provided on the internet," *Elsevier Computers & Security*, vol. 30, no. 6-7, pp. 464–485, 2011.

[54]   J. A. Onieva, J. Lopez, and J. Zhou, *Secure Multi-Party Non-Repudiation Protocols and Applications*, 1st ed., ser. Advances in Information Security.   Springer Publishing Company, Incorporated, 2009, vol. 43.

[55]   S. Kremer and O. Markowitch, "Selective receipt in certified e-mail," in *Progress in Cryptology -INDOCRYPT 2001*, ser. Lecture Notes in Computer Science.   Springer Berlin / Heidelberg, 2001, vol. 2247, pp. 136–148.

[56]   J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," RFC 4880 (Proposed Standard), Internet Engineering Task Force, nov 2007, updated by RFC 5581. [Online]. Available: http://www.ietf.org/rfc/rfc4880.txt

[57]   B. Ramsdell and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification," RFC 5751 (Proposed Standard), Internet Engineering Task Force, jan 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5751.txt

[58]   K. Moore, "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)," RFC 3461 (Draft Standard), Internet Engineering Task Force, jan 2003, updated by RFCs 3798, 3885, 5337, 6533. [Online]. Available: http://www.ietf.org/rfc/rfc3461.txt

[59]   T. Hansen and G. Vaudreuil, "Message Disposition Notification," RFC 3798 (Draft Standard), Internet Engineering Task Force, may 2004, updated by RFCs 5337, 6533. [Online]. Available: http://www.ietf.org/rfc/rfc3798.txt

[60]   D. Katz, R. Saluja, and D. Eastlake 3rd, "Three-Way Handshake for IS-IS Point-to-Point Adjacencies," RFC 5303 (Proposed Standard), Internet Engineering Task Force, Oct. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5303.txt

[61]   T. Hansen, "Message tracking model and requirements," RFC 3888 (Informational), Internet Engineering Task Force, sep 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3888.txt

[62]   P. Hoffman, "Enhanced Security Services for S/MIME," RFC 2634 (Proposed Standard), Internet Engineering Task Force, jun 1999, updated by RFC 5035. [Online]. Available: http://www.ietf.org/rfc/rfc2634.txt

[63]   R. Deng, L. Gong, A. Lazar, and W. Wang, "Practical protocols for certified electronic mail," *Journal of Network and Systems Management*, vol. 4, no. 3, pp. 279–297, 1996.

[64]   B. Schneier and J. Riordan, "A certified e-mail protocol," in *Proceedings of the 14th Annual Computer Security Applications Conference*, ser. ACSAC'98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 347–352.

[65] M. Abadi, N. Glew, B. Horne, and B. Pinkas, "Certified Email with a Light On-line Trusted Third Party : Design and Implementation," in *Proceedings of the 11th International World Wide Web Conference.* ACM Press, 2002, pp. 387–395.

[66] Y. Permpoontanalarp and J. Kanokkanjanapong, "Dynamic undeniable fair certified email with ddos protection," in *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications,* ser. AINA'08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 763–770.

[67] Z. Liu, J. Pang, and C. Zhang, "Extending a key-chain based certified email protocol with transparent ttp," in *Proceedings of the 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing,* ser. EUC'10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 630–636.

[68] K. Imamoto and K. Sakurai, "A certified e-mail system with receiver's selective usage of delivery authority," in *Proceedings of the Third International Conference on Cryptology: Progress in Cryptology,* ser. INDOCRYPT'02, vol. 2551. London, UK: Springer-Verlag, 2002, pp. 326–338.

[69] J. L. Ferrer-Gomila, M. Payeras-Capellà, and L. Huguet i Rotger, "An efficient protocol for certified electronic mail," in *Proceedings of the Third International Workshop on Information Security,* ser. ISW'00, vol. 1975. London, UK: Springer-Verlag, 2000, pp. 237–248.

[70] J. R. M. Monteiro and R. Dahab, "An attack on a protocol for certified delivery," in *Proceedings of the 5th International Conference on Information Security,* ser. ISC'02, vol. 2433. London, UK: Springer-Verlag, 2002, pp. 428–436.

[71] G. Wang, F. Bao, and J. Zhou, "On the security of a certified e-mail scheme," in *Progress in Cryptology - INDOCRYPT 2004,* ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2004, vol. 3348, pp. 1335–1346.

[72] M. Mut Puigserver, J. Ferrer Gomila, and L. Huguet i Rotger, "Certified electronic mail protocol resistant to a minority of malicious third parties," in *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies,* ser. INFOCOM'00. IEEE, 2000, pp. 1401–1405.

[73] J. L. Ferrer-Gomila, M. Payeras-Capellà, and L. Huguet i Rotger, "A realistic protocol for multi-party certified electronic mail," in *Proceedings of the 5th International Conference on Information Security,* ser. ISC'02, vol. 2433. London, UK: Springer-Verlag, 2002, pp. 210–219.

[74] G. Ateniese and C. Nita-Rotaru, "Stateless-recipient certified e-mail system based on verifiable encryption," in *Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology,* ser. CT-RSA'02, vol. 2271. London, UK: Springer-Verlag, 2002, pp. 182–199.

[75] A. Nenadić, N. Zhang, and S. Barton, "Fair certified e-mail delivery," in *Proceedings of the 2004 ACM symposium on Applied computing*, ser. SAC '04. New York, NY, USA: ACM, 2004, pp. 391 – 396.

[76] J. Zhou, "On the security of a multi-party certified email protocol," in *Information and Communications Security*, ser. Lecture Notes in Computer Science.   Springer Berlin/Heidelberg, 2004, vol. 3269, pp. 277–280.

[77] C. Ma, S. Li, K. Chen, and S. Liu, "Analysis and improvement of fair certified e-mail delivery protocol," *Computer Standards & Interfaces*, vol. 28, no. 4, pp. 467–474, 2006.

[78] R. Oppliger and P. Stadlin, "A certified mail system (cms) for the internet," *Computer Communications*, vol. 27, no. 13, pp. 1229–1235, 2004.

[79] H. Wang, Y. Ou, J. Ling, X. Xu, and H. Guo, "A new certified email protocol," in *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, ser. DEXA'07.   Washington, DC, USA: IEEE Computer Society, 2007, pp. 683–687.

[80] X. Liang, Z. Cao, R. Lu, and L. Qin, "Efficient and secure protocol in fair document exchange," *Computer Standards & Interfaces*, vol. 30, no. 3, pp. 167–176, 2008.

[81] J. Cederquist, M. T. Dashti, and S. Mauw, "A certified email protocol using key chains," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 01*, ser. AINAW'07.   Washington, DC, USA: IEEE Computer Society, 2007, pp. 525–530.

[82] R.-J. Hwang and C.-H. Lai, "Efficient and secure protocol in fair certified e-mail delivery," *WSEAS Transactions on Information Science and Applications*, vol. 5, no. 9, pp. 1385–1394, 2008.

[83] C. Wang, X. Yang, C. Lan, and X. Cao, "An efficient identity-based certified e-mail protocol," in *Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, ser. IIH-MSP'09.   Washington, DC, USA: IEEE Computer Society, 2009, pp. 1197–1200.

[84] Y.-X. Gao, D.-Y. Peng, and L.-L. Yan, "Design and formal analysis of a new fair multi-party certified mail protocol," in *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, ser. ICMLC'2010.   IEEE Computer Society, 2010, pp. 3101–3106.

[85] C. Wang, C. Lan, S. Niu, and X. Cao, "An id-based certified e-mail protocol suitable for wireless mobile environments," in *Proceedings of the 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, ser. PAAP'11.   Washington, DC, USA: IEEE Computer Society, 2011, pp. 146–150.

[86] C. Blundo, S. Cimato, and R. D. Prisco, "Certified email: Design and implementation of a new optimistic protocol," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, ser. ISCC '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 828–.

[87] G. Ateniese, "Efficient verifiable encryption (and fair exchange) of digital signatures," in *Proceedings of the 6th ACM conference on Computer and communications security*, ser. CCS '99, New York, NY, USA, 1999, pp. 138–146.

[88] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Advances in Cryptology - EUROCRYPT'98*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998, vol. 1403, pp. 591–606.

[89] J. López, J. A. Onieva, and J. Zhou, "Enhancing certified email service for timeliness and multicast," in *Proceedings of 2004 International Network Conference*, ser. INC'04. Plymouth, UK: University of Plymouth, 2004, pp. 327–335.

[90] S. Micali, "Simple and fast optimistic protocols for fair electronic exchange," in *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. Boston, MA, USA: ACM, New York, NY, USA, 13-16 July 2003, pp. 12–19.

[91] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, ser. EUROCRYPT'03, vol. 2656. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 416–432.

[92] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[93] F. Zhang, R. Safavi-Naini, and W. Susilo, "Efficient verifiably encrypted signature and partially blind signature from bilinear pairings," in *Progress in Cryptology - INDOCRYPT 2003*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2904, pp. 191–204.

[94] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption)," in *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO'97. London, UK, UK: Springer-Verlag, 1997, pp. 165–179.

[95] A. Bahreman and J. Tygar, "Certified electronic mail," in *Proceedings of Network and Distributed System Security Conference*. Pittsburgh, Carnegie Mellon University. Information Networking Institute, 1994, pp. 3–19.

[96] J. Zhou and D. Gollman, "A fair non-repudiation protocol," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, ser. SP '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 55 – 61.

[97]   B. d. M. Giuseppe Ateniese and M. T. Goodrich, "Tricert: A distributed certi-
        fied e-mail scheme," in *ISOC 2001 Network and Distributed System Security
        Symposium (NDSS'01)*.   San Diego, CA, USA: Internet Society, Reston, VA,
        USA, 7-9 February 2001.

[98]   Y. Park and Y. Cho, "Fair certified e-mail protocols with delivery deadline
        agreement," in *Computational Science and Its Applications - ICCSA 2004*, ser.
        Lecture Notes in Computer Science.   Springer Berlin/Heidelberg, 2004, vol.
        3043, pp. 978–987.

[99]   D. Liu, S. Qing, P. Li, and C. Yuan, "A practical certified e-mail system with
        temporal authentication based on transparent tss," in *Proceedings of the 2008
        Ninth ACIS International Conference on Software Engineering, Artificial In-
        telligence, Networking, and Parallel/Distributed Computing*, ser. SNPD'08.
        Washington, DC, USA: IEEE Computer Society, 2008, pp. 285–290.

[100]   M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party
        (extended abstract)," in *Proceedings of the 4th ACM conference on Computer
        and Communications Security*, ser. CCS'97.   New York, NY, USA: ACM, 1997,
        pp. 1–5.

[101]   M. Crispin, "Internet Message Access Protocol - Version 4rev1," RFC 3501
        (Proposed Standard), Internet Engineering Task Force, mar 2003, updated
        by RFCs 4466, 4469, 4551, 5032, 5182, 5738, 6186. [Online]. Available:
        http://www.ietf.org/rfc/rfc3501.txt

[102]   J. Myers and M. Rose, "Post Office Protocol - Version 3," RFC 1939 (Standard),
        Internet Engineering Task Force, May 1996, updated by RFCs 1957, 2449,
        6186. [Online]. Available: http://www.ietf.org/rfc/rfc1939.txt

[103]   J. Klensin, "Simple Mail Transfer Protocol," RFC 5321, Internet Engineering
        Task Force, Oct. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5321.txt

[104]   Geneva-Law, "Convention Providing a Uniform Law for Checks,"
        http://www.jurisint.org/en/ins/255.html, 1931.

[105]   W.-J. Hsin and L. Harn, "Simple certified e-check with a partial PKI solution,"
        in *Proceedings of the 43rd annual Southeast regional conference - Volume 2*,
        ser. ACM-SE 43.   New York, NY, USA: ACM, 2005, pp. 185–190.

[106]   C. Tsao, C. Chen, and C. Ku, "An Electronic Bearer Check System," *IEICE
        Transactions on Communications*, vol. 85, pp. 325–331, 2002.

[107]   W.-K. Chen, "Efficient on-line electronic checks," *Applied Mathematics and
        Computation*, vol. 162, no. 3, pp. 1259 – 1263, 2005.

[108]   V. Pasupathinathan, J. Pieprzyk, and H. Wang, "Privacy enhanced electronic
        cheque system," *IEEE International Conference on E-Commerce Technology*,
        pp. 431–434, 2005.

[109] C.-C. Chang, S.-C. Chang, and J.-S. Lee, "An on-line electronic check system with mutual authentication," *Computers and Electrical Engineering*, vol. 35, pp. 757–763, 2009.

[110] D. Chaum, "Online cash checks," in *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 288–293.

[111] D. Chaum, B. den Boer, E. van Heyst, S. Mjølsnes, and A. Steenbeek, "Efficient offline electronic checks (extended abstract)," in *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*. Springer-Verlag New York, Inc., 1990, pp. 294–301.

[112] R. H. Deng, Y. Han, A. B. Jeng, and T.-H. Ngair, "A new on-line cash check scheme," in *Proceedings of the 4th ACM conference on Computer and communications security*, ser. CCS '97. New York, NY, USA: ACM, 1997, pp. 111–116.

[113] A. d. Solages and J. Traoré, "An efficient fair off-line electronic cash system with extensions to checks and wallets with observers," in *Financial Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998, vol. 1465, pp. 275–295.

[114] S. Kim and H. Oh, "A new electronic check system with reusable refunds," *International Journal of Information Security*, vol. 1, pp. 175–188, 2002.

[115] ——, "An Offline Check System with Reusable Refunds," *IEICE Transactions on Communications*, vol. 86, no. 3, pp. 1136–1139, 2003.

[116] Y. Xu and J. Liu, "Electronic check system design based on nfc," in *Management and Service Science, 2009. MASS '09. International Conference on*, 2009, pp. 1–4.

[117] M. Hinarejos, J.-L. Ferrer-Gomila, G. Draper-Gil, and L. Huguet-Rotger, "Anonymity and transferability for an electronic bank check scheme," in *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2012, pp. 427–435.

[118] H.-T. Liaw, J.-F. Lin, and W.-C. Wu, "A new electronic traveler's check scheme based on one-way hash function," *Electronic Commerce Research and Applications*, vol. 6, pp. 499–50, December 2007.

[119] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proceedings on Advances in cryptology*, ser. CRYPTO, vol. 435. Springer-Verlag New York, Inc., 1989, pp. 239–252.

[120] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups (extended abstract)," in *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, vol. 1294. London, UK: Springer-Verlag, 1997, pp. 410–424.

[121] M. Barbeau, J. Hall, and E. Kranakis, "Detecting impersonation attacks in future wireless and mobile networks," in *Secure Mobile Ad-hoc Networks and Sensors*, ser. Lecture Notes in Computer Science, M. Burmester and A. Yasinsac, Eds. Springer Berlin / Heidelberg, 2006, vol. 4074, pp. 80–95.

[122] N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocols for fair exchange," in *Proceedings of the 4th ACM conference on Computer and communications security*, ser. CCS. New York, NY, USA: ACM, 1997, pp. 7–17.

[123] J. Zhou, R. Deng, and F. Bao, "Some remarks on a fair exchange protocol," in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, vol. 1751. Springer-Verlag, 2000, pp. 46–57.

[124] I. Ray and I. Ray, "Fair exchange in e-commerce," *ACM SIGecom Exchanges*, vol. 3, pp. 9–17, 2002.

[125] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643 – 666, 2004.

[126] Georgios Loukas and Gülay Öke, "Protection Against Denial of Service Attacks: A Survey," *The Computer Journal*, vol. 53, no. 7, pp. 1020 – 1037, 2010.