
KUNGL
TEKNISKA
HÖGSKOLAN


**HØGSKOLEN**
**I GJØVIK**

**NISlab**
Norwegian Information
Security Laboratory

# Private and Accountable Storage in Distributed and Dynamic Environments

Espen Torseth
espen@torseth.net


Institutionen för
Data- och Systemvetenskap

Examensarbete 20 poäng
i data- och systemvetenskap
inom magisterprogrammet i informations- och kommunikations säkerhet,
Kungl Tekniska Högskolan

Examensarbete
Nr 2004-x-164
2002

**Abstract**

One of the major challenges in current technologies for distributed computing is how to preserve the policy applied to data when the data moves around in the distributed environment. Most technologies today seem to assume that either there is a global storage infrastructure suitable for all users, or that the policy is trivial to transfer and express equally well on any storage system. When you add the dynamic nature of these distributed environments and the need for both privacy and accountability, it is obvious that both these assumptions are flawed.

Here an alternative, and somewhat novel, approach to distributed storage is presented. This new approach is based on observing the commonalities between different distributed computing environments, e.g. Grids, mobile agents and peer-to-peer networks, and by separating the policy enforcement from the data storage. Privacy and accountability will be shown to be achievable in this new approach by combining techniques from anonymous remailers and peer-to-peer networks with a trusted third party and partially trusted observers.

An abstract model for a generalized distributed computing node was made to disentangle the approach for storage from any specific distributed computing system. The thesis specifies a protocol for providing pseudonyms for users, which is specified and analyzed formally with Casper and FDR2. A system for anonymous messaging is given, but is not formally analyzed. Finally the storage system is described in sufficient detail for possible analysis or implementation.

**Sammendrag**

En av de store utfordringene i dagens teknologier for dstribuert prosessering er hvordan skal policy knyttet til data bevares når data flyttes omkring i distribuerte miljø. De fleste nåværende teknologier ser ut til å anta at enten så eksisterer det en global lagringsinfrastruktur som passer alle brukere, eller at policy er enkelt å flytte og utrykke likeverdig godt på hvilket som helst lagringssystem. Når du legger til den dynamiske naturen til slike distribuerte systemer og behovet for både personvern og etterrettlighet, så er det åpenbart at begge disse antagelsene er feilaktige.

Her blir en alternativ, og noe ny, tilnærimngsmåte presentert. Denne nye tilnærimngsmåten blir basert på observasjon av likheter mellom ulike miljø for distribuert prossessering, f.eks. Grid, mobile agenter og peer-to-peer nettverk, og ved å skille ut håndheving av policy fra selve lagringen av data. Personvern og etterrettlighet vil bli vist at er mulig i denne nye tilnærimngen ved å kombinere teknikker fra anonymiserende epost-systemer og peer-to-peer nettverk med tiltrodde tredjeparter og delvis tiltrodde observatører.

En abstrakt model for en generalisert distribuert prossesseringsnode ble laget for å løsrive tilnærimngsmåten fra et spesifikt system for distribuert prossessering. Oppgaven spesifiserer en protokoll for å utstede pseudonymer til brukere, som blir spesifisert og formelt analysert ved hjelp av Casper og FDR2. Et system for anonymiserte meldinger blir beskrevet, men ikke formelt analysert. Til slutt blir tilnærimngsmåten for lagring beskrevet i nødvendig detaljeringsnivå for mulig analyse og implementering.

# Preface

The origin of this thesis started with a suggestion from Prof. Snekkenes and Audestad that I should "'...look a Grids and security..."'. At first this seemed rather straightforward, but when after studying the Grid literature it became more and more obvious that there was more fundamental and general problems with large distributed and dynamic environments. When I tried to identify what I believed to be more general problems, I started to examine peer-to-peer- and agentbased-technologies. With help from Prof. Audestad I also got insight into how distributed objects like CORBA and TINA was conceived and implemented.

During this inital phase of the work I also realized that what interested me most was the seemingly incompatible requirements of *privacy* and *accountability*, and how these two can be balanced and implemented in a large scale distributed and dynamic, and potentially self-configurating, environment. Since my background also involves many hours of analyzing protocols, sometimes even to help debug applications, I have an inclination to focus on the *protocols* involved in a system.

While scratching my head on what the thesis finally would be on, I was lucky enough to have the opportunity to have interesting discussion with a colleague, Kåre Langedrag, and Øyvind Kolås at HiG. Kåre and I started to discuss the latest Internetworms and how simplistic they really were. We continued with a informal brainstorm on what the ultimate Internetworm would be like. Kåre, as a software developer, quickly realized that what we were talking about, in reality was a distributed operating environment. Later the same week I was at HiG and while waiting for Prof. Audestad, I talked to Øyvind. I brought up my indecision on a thesis topic and what Kåre and I had come up with, and Øyvind gave me some interesting feedback on both topics that finally made me realize what I had lurking in the back of my head.

I wanted to find the commonalities in different distributed computing platforms and see if there were any common and more fundamental problem areas to research. The final topic was decided as late as June 8th 2004, just 3 weeks prior to the due date. This is a clear indication that available time and the ambition levels might have been somewhat incompatible ☺.

During the work on this thesis, starting in the fall of 2003, I also be-

came the union representative for NITO[1] at ErgoIntegration[2], my current employer. This was not a smart thing to accept.... The same fall ErgoIntegration downsized the staff by approximately 10%, with all the extra work this means for union representatives. Spring 2004 brought on the biannual negotiation of wages and working conditions. As a friendly advise to others; don't combine work, studies, familiy, and union representation – something must yield. For my part it was mainly the time availble for this thesis.

I have to thank Prof. Snekkenes and Audestad for their patience and trust while I was wandering around in the literature and problem space. The PhD students at NISLab for helpful talks and discussions, Hanno Langweg, Geir Dyrkolbotn and Nils Kalstad Svendsen. Gavin Lowe for crucial and timely assistance to get the Casper tool running. My boss, Bjørn Berthelsen, also deserves credit for giving enough flexibility at work while finishing the thesis and ErgoIntegration for financing my studies. The office gang at work for their encouragement and humor: Geir Myhre, Ludvig Nedregård, Jan Arve Gran, Espen Bøe, Terje Engen, and Jens Libæk.

Finally I want to thank my children, Aase Mari (9) and Anna Emilie (3), and my wife, Anita, for keeping me connected with reality while I was writing the thesis. Putting on band-aids, making dinner, and vacuuming can do wonders when one is stuck in an abstract world of distributed storage and security.

[1]The Norwegian Society of Engineers

[2]A part of ErgoGroup (http://www.ergo.no/)

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The goal of this thesis was to explore information security in distributed computing and in particular how storage is solved. There are many systems and concepts for distributed storage, but few, if any, that try to balance individuals privacy needs with societies accountability needs.

## 1.1 Background

The origin of the thesis was a suggestion by the professors at NISlab, Einar Snekkenes and Jan Audestad, to " *. . . look at Grids and security. . . ´´*. This was a rather broad topic, so in an effort to narrow it down to something interesting and manageable, we tried several angels. The first was to look at transactions within a Grid, the next was trusted or secure distributed processing, and some other topics that were totally uninteresting. During this process two things happened; first we broadened the literature study to other technologies for distributed computing and second was that *storage* seemed to be an area that was less studied than other topics.

The author has always been intrigued by the problem of combining strong privacy protection with strong accountability, especially in dynamically changing environments. Thus the final topic became *private and accountable storage in distributed dynamic environments.*

An other important influence for the final topic and research questions is the authors experience as a systems designer. One of the issues that almost never finds a satisfactory solution is when data is transferred from one organization to an other. There are no sensible way of transferring the policy for the data's usage. The "solution´´ commonly used is to create a contractual agreement between the two organizations that in principal just says that the receiving organization will do its best to protect the data from unacceptable usage.

## 1.2 Research questions

In an effort to explore the topic of private and accountable storage in distributed dynamic environments, we chose the following research questions:

1. How do current systems for distributed computing address storage challenges?
2. How can policy be guaranteed to follow the data it controls access to?
3. How can security, flexibility, privacy, and accountability all be simultaneously be achieved for storage in distributed dynamic environments?
4. What are the minimal assumptions needed to achieve question two and three?

The first question is aimed at narrowing the scope of the thesis. Question two should give a basic understanding of the limitations on attaching policy and data to each other. The third aims at finding a generalized system model, rather than a specific solution for a specific system. Question four goes directly at what assumptions has to be made to achieve the goals in question two and three, and gives a basis to conclude on the feasibility for a practical and secure implementation.

## 1.3 Originality

Current efforts to tie policy and data together typically requires a centralized policy infrastructure and secures all but the individuals interests. Here we present a system model that gives individuals the possibility to control their own data, regardless of where and how it is stored. This is achieved without compromising the flexibility for the parties involved.

The use of an active component, i.e. a mobile agent, to enforce policy and act as a conduit to data in a generalized setting, is novel. Other models capable of this exists, but as far as we have identified, none of them has made use of this capability in an active manner.

There is a treatment of pseudonymity, rather than anonymity, to help balance the needs of an individual with the needs of a society. All operations seen by others are made under a pseudonym, and the interaction between a user and its storage agent is anonymized. The anonymity is achieved while preserving the identification of the user to the storage agent. This is novel since most or all other anonymity research also tries to create an environment where every participants has full anonymity with the respect to all other participants.

Finally there was made an effort to formally analyze a pseudonymity protocol. Our literature study found no such protocols formally analyzed elsewhere.

## 1.4 Thesis structure

To have generalized results on how privacy and accountability can be achieved in large scale distributed and dynamic environments, the thesis emphasizes the equalities of different technologies for distributed computing. The main purpose is to separate the theory of how privacy and accountability can be realized from any particular technology. This is described in chapter 2 and chapter 3.

As a means for later specification and analysis, the thesis tries to describe distributed systems as a minimal and abstract model. Chapters 4 and 5 are used for this. Abstract models are given for both the general distributed environment and the individual nodes.

The major parts, chapters 5 and 6 deals with what protocols are needed and what their requirements are. For any protocol a possible specification, or a suggestion of important issues, is given. Some basic security analysis are done on specified protocols to explore if they are feasible to analyze.

Conclusions and and further work is in chapter 7, and supplemental material are found in the appendices.

Throughout the thesis any *definitions* or *assumptions* are given in the context were they are first used.

## 1.5 Method

This thesis is a pure theoretical work. Thus the main method has been literature studies. The proposal of a system model are mainly based on the authors experience as a systems designer at ErgoIntegration and sound engineering principals. Finally the proposed model and critical parts is analyzed. A formal analysis have been striven for, but since this is quite time consuming, it has only been tried on one critical protocol in the system model.

Since, as noted in chapter 1.1, the problem we started with was both wide and open, there was in reality few other methods that could have been used. Most of the time spent on the thesis has been used on finding and reading relevant, and not so relevant, literature. The search was gradually narrowed down as the author gained better understanding of the subject matter. This method will often enable one to identify the underlying challenges, but is also quite risky since it is very time consuming. An other risk is that while searching for general or more fundamental problems, is that one might end up with finding nothing at all.

## 1.6 Ethics

Any security system that tries to give privacy protection, can typically be used by criminals to hide their data too. Similarly can any system for ac-

countability probably be used to create a totalitarian surveillance system. Both these extremes are problematic, but often system designers chose to land on one or the other based on some sort of logic or belief for that particular problem instance. Ideally these should be balanced.

The majority should be able to enjoy the benefits of their privacy, without a big brother watching over their shoulder. At the same time the majority also wants proper authorities to be able to investigate crimes and other unwanted activity.[1] A possible compromise would be systems that enabled authorities to investigate the crimes, but that thwart large scale surveillance.

[1] What is considered a *crime* or *unwanted activity*, will wary with time and culture.

# Chapter 2

# State of the art

## 2.1 Distributed dynamic systems

There are many products, designs, and research on distributed systems. Often the emphasis is on how they differ, rather than the similarities. Here the attention will be on their similarities, to demonstrate that the security challenges might better be solved on a higher abstraction level, before being implemented. In 2.1.6 it will be shown how from the view of any entity in a distributed environment, the functionality of the closest interface to a communicating party the only reality that entity knows of the other party.

### 2.1.1 Grid

Grid technology has gotten a lot of attention from both academic [18, 29] and commercial [2] environments. The academic community is seeking a cheaper and better way to attack the grand challenge problems, and the commercial interests lies in more efficient use of commodity hardware to reduce operational costs or replace expensive specialized computers. Grids are in many ways a refinement of earlier ideas [68] of meta-computing[1], but has gone beyond that. The main source of information on Grids is the Global Grid Forum [81] (GGF). Their main reference platform is the Globus Toolkit [85], but this is only one view of what a Grid is. Alternatives can be found in the Legion project[83] which is more agent-like and [79] Condor which has its roots in harnessing unused CPU-cycles.

Grid technology is already available from several commercial vendors. Sun, IBM, HP, Intel, and Oracle all have different Grid based products available. There are also several new companies which specialize in Grid technology.

Most Grid efforts today have a focus on GGF's OGSA[2] [18, 29] concept.

[1] i.e. building a computer of computers.

[2] Open Grid Services Architecture

The main goal of OGSA is to provide for interoperability and transparency for Grid services. OGSA as such does not address storage, it is only a framework for actual services.

The driving force behind Grid research seems to be a desire to harness and share wasted computing resources across organizations and nations. In the Grid context security initiatives are primarily concerned with access control, encrypting communication channels, and the integration of PKI and Kerberos. Little or no consideration is given on how policy can be transferred in an effective manner when data is transferred or utilized by different organizations.

GSI[3] [4] is the security infrastructure of the Globus toolkit. What is interesting to note, is that a strikingly similar security infrastructure was presented in [32] as early as 1989.

### 2.1.2 Agents

Agents have been extensively [57] studied. There are many projects finished and running on the subject. Here this will not be as thoroughly discussed as Grids, but we will show that there exists important similarities in the basic goals of agent-based systems and Grids.

An agent is in essence a autonomous piece of code and data that can, within limits of each agent based system, move freely between participating nodes to harness resources or provide services. Grids also allows for dynamic reconfiguration of where and how a job is done, and also provides a service oriented view of resources.

As for Grids, there is little or no research done on how to tie data and its policy together. In agent based technologies this should be easier to accomplish, since an infrastructure for code, and implicitly data, mobility already exists.

### 2.1.3 Peer-to-peer

Peer-to-peer has had a lot of attention the last few years. The origins of peer-to-peer-technology are more anarchistic than Grids and agents. Early peer-to-peer systems were primarily made to enable users to share, often illegal, files easily and out of public control. Early systems [58, 52] either had scalability problems or were not pure peer-to-peer systems. Now there are more serious research and product development efforts on peer-to-peer technologies. Suns JXTA [71] effort is one important arena today. In the open source community peer-to-peer also seems to have matured and issues on scalability have been addressed in some systems, e.g. [14].

Peer-to-peer also have a basic goal of utilizing distributed resources and providing services to the participating nodes. Today the service has primarily

[3] Grid security infrastructure

been file sharing, but as Suns JXTA shows peer-to-peer is not limited to that. Especially JXTA shows that peer-to-peer actually has many similarities with Grid and agent based systems.

### 2.1.4 Telecommunications and distributed objects

A very informative example of a distributed dynamic system is the telecommunication industry. The TINA[4] Consortium [15] can also serve as a good example of a system of distributed objects.

Telecommunications has a long history [6] in making and managing distributed systems. The GSM cellular phone system is a good example of how to create a massively distributed and interoperable system for speech and data communications. Thus telecommunications can teach us many valuable lessons in systems design. Unfortunately telecommunications does not address the problem of policies concerning data transferred over the network as such. They do address policy issues, but then as a means of supporting billing.

TINA, which is very similar to CORBA[5], does not seem to address storage and usage policies with respect to data [19]in general. They have a rather old draft [13] on information modeling concepts, but does not explicitly address these policy issues.

### 2.1.5 Comparison of the different technologies

These different technologies for distributed computing are not easy to compare. Here the goals is to show similarities, and not in any way ranking them. Table 2.1 shows that there are differences in privacy and accountability, but this seems to stem more from the goals and assumptions of early instances of the technologies, rather than inherent limitations. Metering is not that easy to generalize. Depending on the supporting infrastructure, there can be inherent limitations in different technologies for distributed computing. Services and resource sharing is trivially shown to be the basic goals for all the reviewed technologies. No of the technologies do explicitly support a mechanism to bind data and its policy together, but some does implicitly support this.

### 2.1.6 Conclusion

In all of these distributed systems any entity sees only the interface of other entities. The implementation details behind that interface is typically invisible and ignorable for other entities. In essence an entity can not by simply

[4] Telecommunications Information Networking Architecture
[5] Common Object Request Broker Architecture

**Table 2.1:** Comparison of distributed technologies

| | OGSA | Legion | Agents | P2P | Telecom. | TINA |
|---|---|---|---|---|---|---|
| Privacy | Weak | Possible | Possible | Weak | Varies | Possible |
| Accountability | Weak | Possible | Possible | No | Yes | Yes |
| Metering | Strong | Possible | Possible | No | Strong | Strong |
| Services | Yes | Yes | Yes | Yes | Yes | Yes |
| Resource sharing | Yes | Yes | Yes | Yes | Yes | Yes |
| Policy tied to data | No | Possible | Possible | No | No | Possible |

observing messages to and from other entities, know how they are implemented. In one instance a service might be run as a dedicated system. The same service might also be reachable as a virtual host on a shared computing cluster, or as a downloaded local applet. Since an entity thus can not make any assumption on how a service is implemented, every entity must be able to enforce their policies by themselves. This is illustrated in figure 2.1, were it is clear that the users in reality have no way of knowing how their data are stored at a storage provider.

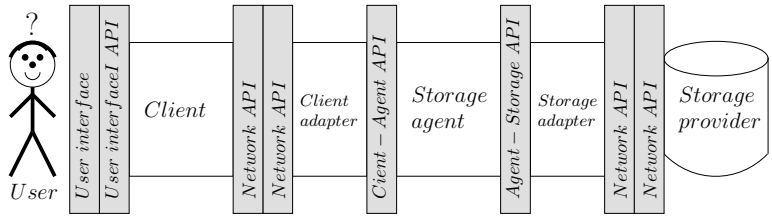


**Figure 2.1:** Entities and interfaces

One way of illustrating this point is by utilizing the same philosophical experiment that Plato used in The Republic [60]. Instead of people chained in a cave that is only shown shadows of real objects, we can think of a piece of code that only sees an API. That particular piece of code can not be assured that the API actually gives access to the *real* service it requests or just a simulated result. There is also no way a piece of code can be sure that it communicates outside its own computer or not. Thus the *knowledge* we assume an entity has in large distributed and dynamic systems, is primarily based on our own knowledge and assumptions in that system.

A very illustrative example is the Honeynet project [86]. If a honeypot or honeynet is to be useful to asses the capabilities of a capable adversary, it is obvious that the virtual environment the adversary sees must emulate the system the adversary expects. This emulation must be so effective that

both automated and manual by the adversary checks will come out the same as for the real environment. An earlier example is Bill Cheswick's analysis of the actions made by a hacker called Berferd in [12][6].

All these examples all illustrate that today the assumptions most storage systems build upon are outdated or inappropriate for the task. Thus we should explore other ways of protect our data.

## 2.2 Protocols

Security protocols exists in abundance. Thus we narrow the scope to protocols for privacy and non-repudation and analysis techniques. Even with this limitation of scope, we can only hope to scratch the surface of these topics.

### 2.2.1 Privacy and accountability

There exist many protocols for privacy and accountability. Since our main concern here is conditional privacy, we are more interested in anonymity and pseudonymity protocols rather than pure confidentiality protocols. It is also trivial to show that a global system for tamper resistant logs is unfeasible, so the emphasis for accountability protocols will be on non-repudation and multi party non-repudation.

#### Privacy

Protocols for *privacy protection* in the sense used here was not found. What we found was that for our definition of privacy protection current solutions or efforts result in *systems* rather than protocols. For the protection of the confidentiality aspect of privacy there are many protocols in use and theory. Two good examples of current production protocols are IPSec[7] [38, 41] and SSL[8]/TLS[9] [39, 16]. Both have been criticized as being to complex and some security flaws have been found [30, 78] for some, sometimes quite obscure, configurations and in the basic protocols.

**Assumption 2.1.** *Privacy protection is not reached with a protocol alone – a complete system is needed.*

#### Non-repudation

Non-repudation is an area of active research. The first protocols to achieve both fairness and efficiency [87, 69] are quite recent. The main goal of non-repudation is to create a evidence that some action by an entity occurred.

[6] This is the granddaddy of all subsequent honeypots.
[7] Internet Protocol Security
[8] Secure Socket Layer
[9] Transport Layer Security – The standardizations of SSL by IETF

Non-repudation is often the basis for e-commerce protocols [21, 72] over open networks. Actually non-repudation is just a way of creating accountability with the help of a cryptographic protocol. This aspect of non-repudation is more relevant for our purposes.

Current research on non-repudiation seems to be focused on the protocols themselves and on their applications in e-commerce. In [24] the use of agents as mediators between endpoints is discussed. Even though e-commerce is the main focus here, the usage of mediators, with varying trust levels, is novel.

Most efficient non-repudation protocols use a trusted third party to create a manageable trust system. Though non-repudation can be achieved without a trusted third party, this typically will result in an inefficient protocol overall.

**Assumption 2.2.** *An efficient and fair non-repudation protocol must use a trusted third party as its basis of trust.*

#### Multi party non-repudation

Multi party non-repudation is a generalization of non-repudation [42]. As for normal non-repudation protocols, multi party non-repudation can be based on an optimistic approach [43, 35], were a trusted third party is only utilized when disputes occur.

A new property in multi party non-repudation is *exclusion-freeness*indexexclusion-freeness. In [36] both *weak* and *strong* exclusion-freeness is defined.

**Weak** At the end of a protocol run, an excluded participant is able to prove to an external adjudicator that it was excluded.

**Strong** At the end of a protocol run there are no excluded participants.

Both weak and strong exclusion-freeness can be utilized by distributed systems to handle situations where multiple parties must agree that an action occurred.

### 2.2.2 Protocol analysis

Security analysis of cryptographic protocols has been and is an area of active research. Here we will only give a high level overview [26, 49] of this complex and intriguing topic. In [49] Meadows gives a much more complete of the state of the art than here. Here the purpose is to create at context for later discussions in this thesis.

It is interesting to reiterate the words of Needham and Schroeder in [54], since this paper has inspired, directly or indirectly, the topic of this chapter.

> Finally, protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal

> operation. The need for techniques to verify correctness of such protocols is great, and we encourage those interested in such problems to consider this area.
> *Roger M. Needham and Michael D. Schroeder*

A striking example is an attack [44] on the Needham-Schroeder protocol developed in [54] that was found by Gavin Lowe in 1996, 28 years later.

### The Dolev-Yao attacker

Before Dolev and Yao's seminal paper [17], there was no publically described abstract attacker for cryptographic protocols. Other attackermodels have been described [74], but either these have been shown to be equivalent or weaker than the original. In [10, 11] Cervesato shows that the Dolev-Yao attacker is the most powerful one.

The Dolev-Yao attacker has the power to:

- delete
- replay
- fake
- redirect

any message between parties, only limited to cryptographic constraints.

One topic that seems to be missing is on how the Dolev-Yao attacker fits in a security analysis of non-repudation protocols. An underlying assumption for the Dolev-Yao attacker is that the adversary is an other entity than the legal participants. It is implicitly assumed that all other participants are honest and well behaved. This is in stark contrast with the dominating justification for non-repudation protocols – sometimes the legal participants try to cheat. The implication of this observation is that one should be very careful with the interpretation of results from the analysis methods described here, if we apply them to non-repudation protocols.

### Logics

BAN logic [8] is the earliest attempt on formal analysis of cryptographic protocols. The logic presented by Burrows, Abadi and Needham was limited to analysis of *authentication* protocols. This fact has sometimes been overlooked by later criticism of the logic.

The GNY logic [34] is an extension of BAN logic to capture more scenarios than only authentication, but at the cost of making the logic more complex. This complexity is eased by the existence of a GUI based tool. SPEAR II [63] was made by Elton Saul as part of his master thesis at the University of Cape Town.

SVO logic [73] by Syverson and Oorschot is a newer logic. This logic unifies aspects from GNY and other logics, while at the same time it retains much of the simplicity of the original BAN logic.

#### Model-checking and state enumeration

Several authors have used different model checkers and state enumeration techniques to analyze security protocols. Catherine Meadows NRL[10] protocol analyzer [50] is one of the successful examples.

An other example of this approach is Casper [45, 46] which uses FDR2[11] [47] to check the generated CSP[12] [37] code. It is one of the few tools that are freely[13] available. The distribution comes with a easy to follow tutorial and user guide [46], and has a fairly user friendly input and output syntax. The concept of using CSP a underlying tool for analysis of security protocols is well documented and presented in [26] by Peter Ryan et al. The combination of a supporting book and sensible syntax was the main motivation for choosing Casper as the method for protocol analysis in this thesis.

#### Other approaches

Spi calculus [1] is an extension to pi calculus to support reasoning on security protocols. Here the attacker does not need to be explicitly specified. This might make spi calculus capable of analyzing scenarios were all participants are misbehaving, rather than under just one attacker model.

Strand spaces [25, 22] is a graph-theoretic interpretation of the Dolev-Yao attacker. This approach is fairly new, but has a very appealing simplicity and elegance.

## 2.3 Trusted processing

Trusted processing is a non-trivial problem to achieve locally, and even more so in an distributed environment. There exists several products, designs, and ideas that claim to achieve this. Here a few possibilities are presented along with the abstract model used for this thesis. The main designs for trusted remote computing seems to be:

- Trusted operating operating environments (i.e. operating systems, sandboxes or virtual machines)
- Trusted hardware modules

[10] Navy Research Laboratories
[11] Failure Divergent Refinement
[12] Communicating Sequential Processes
[13] Both in the sense of freedom and free of charge.

- Mathematical transformation

In thesis the abstract model used is based on the first bullet, but to give the reader a general idea of the possibilities the other two is presented as short sketches of the concepts.

### 2.3.1 Trusted operating environments

All security schemes need some sort of trusted environment to do critical processing in. In chapter 5.2 sandboxes are described in detail. Sandboxes is only one way such an environment might be made. Thus here we present alternative and complementary technologies.

### 2.3.2 Evaluation schemes

The basic idea behind evaluation schemes is that by having someone other than the maker of a system perform an evaluation of that system, we will gain more trust in the correct performance of that system. In many ways evaluation schemes are just a way of formalizing the time old engineering practice and academic principle of peer review.

Most books on information or computer security [48, 33, 59] contain a good introduction to the different evaluation schemes. Here we just state that Common Criteria (CC) is considered the state of the art. CC was made an ISO, ISO 15408, standard in 1999.

Even though evaluations can be an important element to reach a sufficient level of trust, it is by no means a clear link between a evaluation level and the resulting level of trust. Trust is a very subjective topic. What or who we trust is more often based on what and who we know, rather than an arbitrary certificate by an evaluation company. Thus an evaluation will add little or nothing to the resulting level of trust, if the users do not have an existing trust relationship with the evaluated system, evaluator, system supplier, etc.

### 2.3.3 Trusted hardware modules

Trusted hardware has been, and is, a key part of many high security systems. As a platform for trusted computation it usually takes two forms. Either it is embedded in a CPU or as an external hardware module. Both concepts have been used with success and failures, but in essence they are not that dissimilar. Both work on the basic assumption that hardware is less vulnerable to tamper than software. Sometimes this is correct, other times not.

Smart cards are very illustrative here. They are often used as a token to store private keys and perform security critical parts of a protocol, e.g. sign a hash of a document. Even though a smart card might be unhackable today, it must nonetheless be provided with the correct input to produce

reliable output. As long as the surrounding system can be subverted to perform arbitrary actions, it clearly follows that a smart card can be fooled into signing any hash it is presented with.

Smart cards and other hardware based approaches, does all or parts of sensitive computations in hardware that is unaccessible to other entities on the same platform. The main challenge is how do you get the application in to the trusted hardware and how can you interact with it without the possibility for compromise. Ross Anderson has several informative examples on how things fails in [61] and at his homepage [62].[14] Other sources for information on how things fail can be found a Peter Neumann's homepage [55] and The Risks Digets [51] wich he moderates.

### 2.3.4 Mathematical transformation

The expression *mathematical transformation* was chosen, as it has a broader application than *cryptographic methods*. The use of cryptographic methods applies to most every form of trusted computing in one way or the other. Here we look at how mathematics can transform a sensitive computation to a non-sensitive one. The two examples given illustrate the two basic approaches. We can transform the representation of a program and data into another form or problem space that disguises the original sensitivty. The other option is to split the program up in several smaller units, where the smaller units are not sensitive.

In [7] Tønnes Brekne shows that it is possible to perform computations with an encrypted program on an encrypted Turing Machine. The thesis is on encrypted automata and shows that it is possible, but does not show how to do in an cryptographically strong manner.

An example of splitting a sensitive computation into several non-sensitive parts is given in [75]. Here the author shows how computations can be split up in $N$ tasks and run them on $K$ computers (where $K > N$), and that the computation will be successful as long as the number of cheaters is below a given threshold. In some ways this is related to both secret sharing and techniques for high availability systems.[15]

## 2.4 Tying policy to data

### 2.4.1 Content based information security

In [27] the concept of content based information security (CBIS) is introduced. NATO[16] with their coalition partners face challenges in exchanging

[14] In general Anderson shows that most things fail...

[15] Both these presentations are very simplified, and interested readers are recommended to read the original texts themselves.

[16] North Atlantic Treaty Organization

information in peace restoring and peace keeping missions. This problem is not isolated to NATO's military coalition forces, but also extends to civilian agencies and humanitarian organizations. In peace operations all these parties must, to varying degrees, exchange sensitive information to achieve the ultimate goal of peace.

The experiences gained from current and past operation shows that the information infrastructure use today is inefficient for the kind of information sharing peace operation need. Thus NATO has started research on CBIS to find alternative approaches.

The architecture described for CBIS calls for the use of trusted hardware and a centralized security manager. Thus even if the basic idea is similar to the concept presented in this thesis, there is a difference in approach. Here we strive for flexibility and user empowerment.

### 2.4.2 Digital rights management

Digital rights management (DRM) is a controversial and confusing topic. The basic aim is to tie the copyright owners policy to copyrighted material, such that a consumer is forced to comply with that policy. Most of the interesting literature is either requires a membership in one of the many industry consortia or the signing of a non-disclosure agreement. Ross Anderson provides a useful FAQ [5] on the subject and links to relevant websites and news stories. Here we just note that while the aim is to bind data and policy together, digital rights management has a clear bias on protecting the copyright owners.

# Chapter 3

# Theory

In any non-trivial distributed system, there are some basic challenges that must be addressed. In 2 it will be shown that these challenges are common for all major initiatives on distributed computing, but that the different initiatives addresses them in a to narrow context and that this hinders reuse of protocols for privacy and non-repudation.

Based on personal experience and observations as a designer at a service provider, we tend to focus on the most pressing problems, without trying to identify commonalities with other similar problems. This has a negative effect on exploiting synergies and developing common solutions to common problems. Put into harsher words, we are good at finding *a* solution or fix, but not identifying *the* problem.

One of the main goals with the thesis is to show that identifying similarities in different technologies can help us identifying the more fundamental issues, rather than just concentrating on how to solve it for one particular system. Thus the primary goals in this thesis is to show:

- That different technologies for distributed computing essentially try to solve the same basic problems.
- That protocols for privacy and non-repudation can be designed for a generalized distributed environment, and easily be adapted to the different distributed computing environments.
- That such protocols can be analyzed with formal methods.

Secondary goals have been to gain a general understanding of:

- different models for distributed computing.
- how formal analysis of security protocols are done.
- the limits of formal analysis.

- that it is fruitful to analyze commonalities, rather than specialties.
- engineering security into complex distributed computing environments.

To keep the thesis within a reasonable scope, *storage* was chosen as the focus here.

## 3.1 Privacy and accountability

*Privacy* and *accountability* are often naively viewed as two opposing extremes. In some cases this view is correct, but not complete. The primary goal for privacy is to secure an individuals right to a private sphere. Accountability has the primary goal of guaranteeing a way of placing responsibility for actions. When we consider the goals, it is obvious that they are not two opposing extremes, but rather two non-trivial goals that are not easy to reach in complex environments.

### 3.1.1 Privacy

An individuals right to privacy is legally protected in most of the world. The European data protection act of 1995 [76] is a prime example. Privacy is also considered as a basic human right in article 12 of the The Universal Declaration of Human Rights [53]. However there is also no doubt that privacy exists in different degrees and forms.[1] The two basic aspects of privacy, we claim, are *anonymity* and *confidentiality*. Privacy is a combination of these to aspects, but here we will focus primarily on anonymity. The main reason is that individuals right for confidentiality of their sensitive data, usually already is secured by laws or policies. The laws and policies seldom discuss to what extent individuals should be identified or not.

#### Anonymity

Anonymity is a non-trivial, and at times controversial, subject. It rises many philosophical, social and political questions. The views given here are strongly influenced by the author's nationality and personal beliefs and values. This thesis is written at a stable time in a stable democracy with a strong protection of privacy both legally and culturally. Thus the issue on an individuals right to privacy is not a problematic issue, but there are some debates on to what extent an individual has a right for anonymity.

In general well behaved individuals do not have to identify themselves. It is when laws, policies, or other regulations are broken, that individuals are required to be identified. Sometimes identification, with varying success,

[1] Different cultures, societies, countries, etc., all have different expectations and values concerning privacy.

is used as a preventive measure. Bruce Schneier has in his Crypto-Gram newsletter [65, 66, 64] given many pros and cons on this subject. Here it will for simplicity be assumed that it is acceptable for honest players in a distributed system to be anonymous. This implies that it is acceptable for dishonest players to be identified.

**Assumption 3.1.** *It is acceptable for honest entities in a distributed environment to be anonymous.*

Privacy can be said to range from total identification to anonymity. Where ideal privacy fits on the scale can be debated depending on culture and legal context. It is however easy to argue that total identification of every individual is not ideal and that anonymity for everyone equally is not ideal. This trade off is illustrated in figure 3.1.

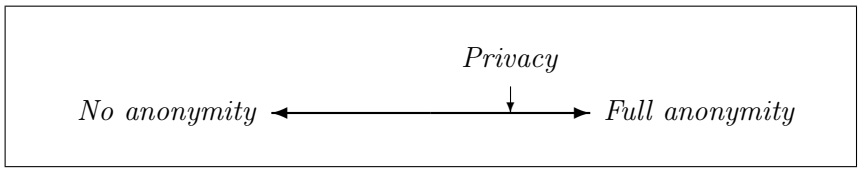


**Figure 3.1:** Privacy

Figure 3.1 also illustrates how privacy can be viewed as *conditional anonymity* rather than a seperate issue.

### Confidentiality

Confidentiality is defined here as *prevention of unauthorised disclosure of information*[33, page 5].

**Definition 3.2.** *Confidentiality is the prevention of unauthorized disclosure of information.*

For privacy purposes confidentiality is mainly used to protect sensitive information about individuals from unauthorized viewing. In most western countries there are legal obligations on anyone who possesses such information to protect it adequately. This aspect of privacy has been extensively studied before, and it will not be treated in depth here. There are plentiful of books on the subject, especially the cryptographic aspects, and the authors favorite introductory texts are [70, 31, 3, 67]. In the rest of the thesis the term *privacy* should thus be interpreted as *how to secure honest individuals from unnecessary identification*. Where confidentiality is needed it will be noted where applicable.

**Definition 3.3.** *Privacy is the prevention of unnecessary or unauthorized identification of individual persons.*

### 3.1.2 Accountability

Often *accountability* is seen as a key factor for a commercial system. Without accountability there is no liability either, and thus nobody are responsible for unacceptable actions. In commercial environments there have to exist an acceptable and reasonably safe method for placing responsibility. This can be achieved in different ways:

**Cryptography** For a thorough treatment on the cryptography of non-repudation protocols see [87].

**Logging** Logs in many forms has been used to provide accountability in many scenarios and settings. Financial instations are a prime example of this form of accountability for both their employees, customers, and regulatory instations.

**Procedures** Procedures typically support accountability by providing a log of activity, or by ensuring that critical actions are under dual control. The generating of root keys on a CA is a prime example.

**Policies** Policies support accountability by placing the responsibility for flaws on someone. European laws concerning sensitive data on individuals are prime examples.

None of these facets of accountability can by themselves provide a complete solution to guarantee accountability. They have to be carefully coordinated to give effective and efficient accountability in any form of system for distributed computing.

*Non-repudation*is basically a way of securing accountability in systems. Since the main focus for non-repudation is to cryptographic provide some sort of evidence to keep some entity accountable for an event or action. It is interesting to view non-repudation as one way of representing the degree of accountability, see figure 3.2. Non-repudation can also be claimed to provide *conditional accountability* rather than just a cryptographic form of evidence.

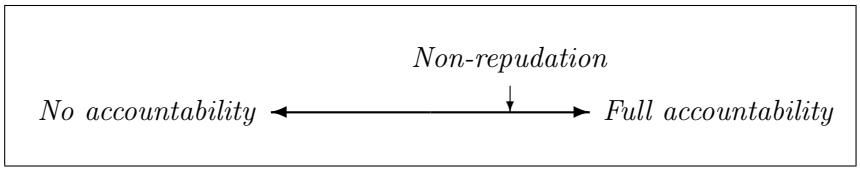


**Figure 3.2:** Non-repudation

### 3.1.3 Privacy vs. Accountability

For any general use and complex commercial system, there always will be a trade off between individual's privacy, their accountability, and the system's general functionality and efficiency. To try and identify a sensible balance between privacy and accountability, we propose to view these two factors as axis in a cartesian diagram. This gives us four quadrants $(A - D)$ that illustrate the different basic trade offs.

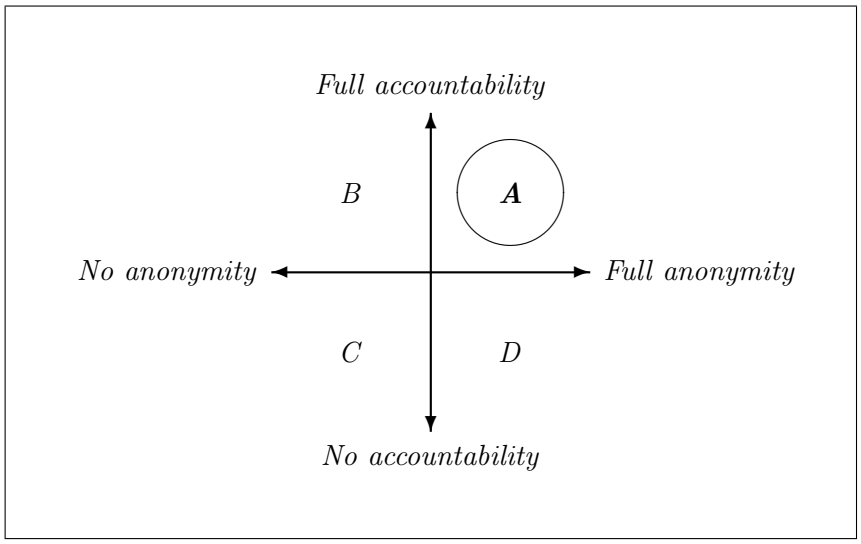


**Figure 3.3:** Privacy vs. Accountability

Figure 3.3 shows four basic trade offs in the four quadrants. The goal here is to reach a point in quadrant $A$ (indicated by a circle), where potentially there is an optimal trade off between privacy and accountability. The four basic trade off scenarios are:

**A** High accountability and high privacy can be reached, and is probably a very good mix to base thriving commercial systems on top of.

**B** Here accountability is high and privacy low. This is applicable for systems where accountability is crucial, and privacy is either unwanted or unneeded. An example can be military command and control systems.

**C** Low accountability and low privacy is probably best exemplified by the current status of the Internet.

**D** Low accountability and high privacy, i.e. anonymity, has been supported for multiple years through different means. Good examples are anonymous [84, 77] e-mail systems.

## 3.2 Confinement in dynamic distributed systems

In any distributed system a user asks, or rather *should* ask: "How can I trust that my data and applications haven't been tampered with or compromised?" The owners of resources in any distributed system should ask: "How can I be assured that a user does not abuse or deny using my resources?" Both these questions point back to Lampson's *confinement problem*defined in [9]. If a high levels of privacy or accountability is to be reached, there has to exist at least some basic support for separation between tasks. In [48, page 439] Matt Bishop restates Lampson's original paper as two requirements and three definitions:

- **Requirements**
  - The server must ensure that the resources it accesses on behalf of the client include only those resources that the client is authorized to access.
  - The server must ensure that it does not reveal the client's data to any other entity not authorized to see the client's data.
- **Definitions**
  - The *confinement problem* is the problem of preventing a server from leaking information that the user of the service considers confidential.
  - A *covert channel* is a path of communication that was not designed to be used for communication.
  - The *rule of transitive confinement* states that if a confined process invokes a second process, the second process must be confined as the caller.

These requirements and definitions places the main burden of enforcement on the server. In traditional client-server architectures this was probably sufficient. These architectures tend to implicitly regard the servers as trusted entities and the clients as potentially untrustworthy. This works quite well as long as the environment is rather static. In these static environments the trust can be achieved by having manual procedures and policies in place to replace any technical shortcomings of the security infrastructures.

## 3.3 Are distributed dynamic systems different?

In highly dynamic environments as Grid, peer-to-peer or agent based networks, the use of manual procedures will not scale. These environments can be self-configurating and in many cases there will be no single authority that can be a centralized basis for trust.

**Definition 3.4.** *A distributed dynamic environment is any computing environment that is: Distributed, dynamically configuarted, and partiall or fully self-configurating.*

The basic change is on of basic assumptions; before the implicit assumption was based on a server serving several clients and that someone was accountable for the correct operation of that server. This implicit assumption can easily be traced back the famous Ware-report [28] from 1970 which was written in a environment of terminals and mainframe computers. Today the distinction between a client and server often is rather blurred, and the accountability watered out since we have moved from dedicated local systems to global services. In the global setting it might not be possible to identify the entity who was the root cause of a failure. Thus basic assumptions have changed.

No attempt to reformulate Lampson's[9] work will be done, but as a result of the change in assumptions on can argue that in these environments *all* entities must be able to detect, document and possibly resolve breaches of policy.

The focus here will be on how two conflicting sets of requirements possibly can be combined in these complex systems. On one hand it is generally acknowledged that people have a right of *privacy*, but at the same time it is also expected that a person is *accountable* for their actions. In a commercial setting this typically translates to that someone has to pay for what happens.

*Privacy* can be achieved by creating an environment where everyone is anonymous. Such systems are technically possible [84, 77], but the consequence is a practically total loss of accountability.

*Accountability* can be achieved by several means. Any form of traces left by a person can be used to hold somebody accountable. A combination of logging and authentication has been the main solution the last decades. Another possibility is to rely on *non-repudation* by cryptographic means. Here the primary purpose has been to create an undeniable connection between a subject and its actions. This undeniable property can be problematic where privacy is a concern, while at the same time it might be crucial to secure the privacy of others. For distributed environments with no centralized management, it is probably easier to have a non-repudation mechanism scale than to rely on having to analyze logs[2] from several participants.

The real challenge then becomes to find a balance between privacy and accountability. Any automated system or design will rely on some faith and basic trust, and on external manual systems to handle the non-automated situations that might occur. Thus any concept for privacy and accountability in distributed computing environments will ultimately rely on an existing judicial system to resolve disputes. There also has to be some basic faith in

[2] As far as I know there is no form of logging that is generally accepted as *trusted* or *tamper-proof*

the correct behavior of at least some of the trusted third parties involved. If every trusted third party is unreliable and corrupt, a scalable and practical solution is probably much more difficult to find.

## 3.4 Storage in distributed dynamic environments

A good summary of the challenges facing storage model in distributed dynamic environments can be found in chapter 8 of work package 5 [23] of the DataGrid project[80]. Here four areas of open issues are listed: interactive services, policies, access control, and objects. The essence of these challenges are quite similar to the research question listed here in chapter 1.2.

One area that can, and probably will, create problems are *naming*and *name resolution*indexresolution, namefor data items. A good description of the problem is given by Ross Anderson in [61, page 125]. Here we have chosen to assume that naming and name resolution is solved by other means, but this is clearly an area of work for any attempt of implementing this model.

**Assumption 3.5.** *A practical, stable, secure, and robust system for naming and name resolution exists and is accessible for all entities in a distributed environment.*

# Chapter 4

# Environment

Any distributed environment can at a high level be described as clients which accesses services. This applies to peer-to-peer-based environments also; here each node acts as a client and a service provider simultaneously. All distributed environments must also provide some basic methods for service and client discovery and user, client, and service authorization. These methods can themselves be services, but will typically have some form of special status or configuration applied to them.

The basic assumption is that users use brokers to access services. Users, brokers, and services uses catalogs and trusted third parties to facilitate their cooperation and secure interaction.

**Assumption 4.1.** *Users in distributed environments must use one or more special services, here called brokers, to get access to the services and resources they need.*

**Assumption 4.2.** *All resources and services are presented to the users as a form of service.*

**Assumption 4.3.** *The special services needed by all entities in a distributed environment are:*

**Catalogs** *The basis for naming and name resolution.*

**Brokers** *The basis for access management, metering, and billing.*

**Trusted third parties** *The basis for creating trust between entities.*

The legend of figure 4.1 is:

**U** An user and its client.

**N** Any network that gives an user $U$ access to a broker $B$.

**B** A service broker.

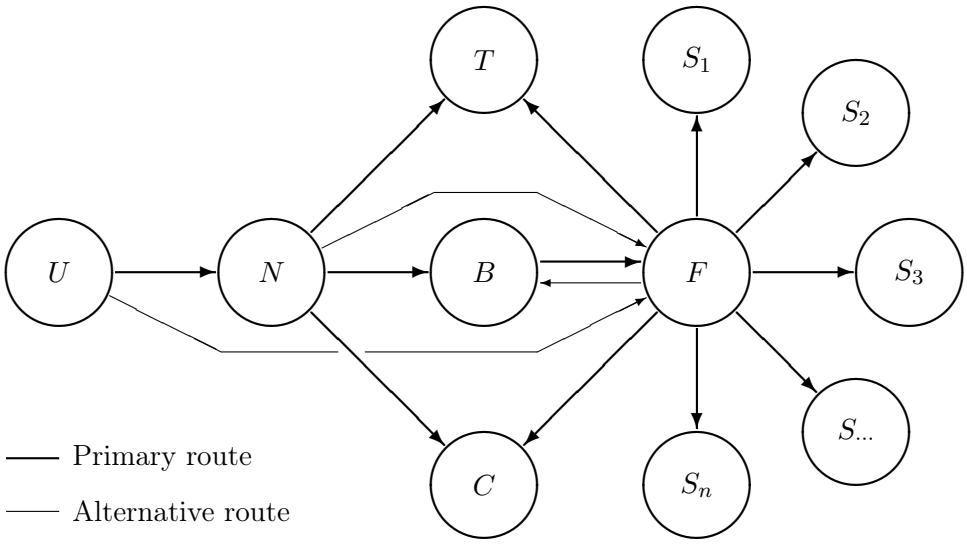


**Figure 4.1:** Entities

**F** A network fabric that supports QoS[1], and that gives brokers $B$ control over QoS parameters.

**C** A catalog service for meta data needed by other entities.

**T** A trusted third party service.

**S** An unspecified service.

Catalogs, trusted third parties, networks, fabrics, and brokers, are services. Thus figure 4.1 could have been expressed as:

- Let $U$ be a member of the set of all users $\mathcal{U} \Rightarrow U \in \mathcal{U}$
- Let $S$ be a member of the set of all services $\mathcal{S} \Rightarrow S \in \mathcal{S}$
- Let $\rightarrow$ be an operator that means *has access to*
- Then:

$$
\begin{array}{ll}
U \in \mathcal{U} \wedge S \in \mathcal{S} & \\
\Downarrow & \\
\forall U : \exists S_B \in \mathcal{S} & \text{For all } U \text{ there is at least a service } S_B \text{ that is in } \mathcal{S} \\
\Downarrow & \\
U \rightarrow S_B \rightarrow \mathcal{S} & U \text{ can access every } S \text{ in } \mathcal{S} \text{ via } S_B
\end{array}
$$

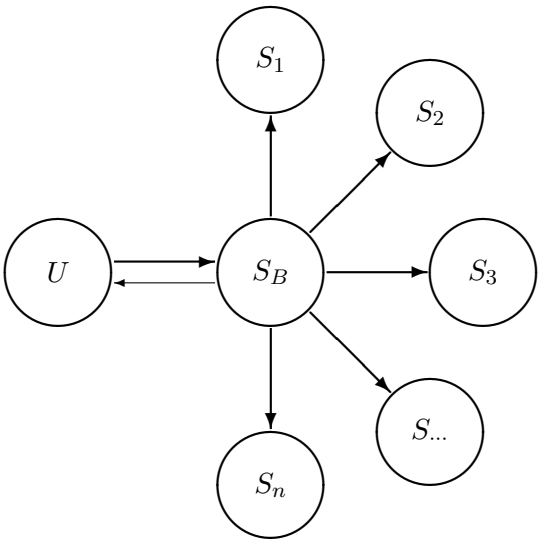


**Figure 4.2:** Simple entities

Figure 4.1 could then have been simplified to figure 4.2:

Unfortunately this variation does make our notation less intuitive, even though it is as correct as the chosen one. Any notation that potentially has to use subscripts on subscripts, e.g. a time stamping trusted third party would become $S_{T_{TS}}$, will be very difficult to read. Thus the choice of separate letters for the key services $N$, $B$, $F$, $C$, and $T$.

The other distinction that might seem odd is the separation of *network* and*fabric*. Network is used for any network, but fabric is the controlled network that interconnects brokers and resources, and that is not directly accessible to a user.[2] A *network* $N$ is any communication facility capable of transporting messages between parties, but without any guarantees at all. A *fabric* is any communication facility that can transport messages between parties with negotiated and enforced QoS parameters.

For the purpose of this thesis it is sufficient to view a storage resource as any resource that can store a bit string in a sufficiently reliable way and that can be retrieved through the fabric. Thus there is no distinction made between RAM, disk, tape, etc.

**Assumption 4.4.** *Storage is only characterized by the QoS and SLA*[3] *parameters on its performance, permanence, and protection.*

[1] Quality of Service

[2] This does not rule out the possibility that the *fabric* and the *network* sometimes might be the same physical or logical entity.

[3] Service level agreement

## 4.1 General assumptions

Any system will ultimately rest on a set of assumption made by its designers. These assumptions usually are the Achilles' heel of systems. Ross Anderson gives plenty examples in his book on security engineering [61]. If a proposed system are to be analyzed by anyone other than the original designers, these assumptions must be documented in as much detail as possible. Even though this is striven for here, there are always a possibility for that some implicit assumptions are undocumented because the designer believes that particular assumption to be obvious or insignificant[4].

### 4.1.1 Dispute settlement

Any automated system with several users and user groups will need an external system for any final ruling on how to treat suspicions and claims of misbehaving users or services. It seems to impossible to create a completely self contained system for accountability. Since the basic idea of accountability hold people responsible for their actions, all rules in a system, i.e. its policy, will ultimately be subjectively interpreted by humans. Thus it is impossible to have a fully automated system for accountability, because there is always a chance that somebody just did not know that their actions where unacceptable for others. This problem will always be present, and for large systems it will be unavoidable.

This is how our societies are organized. If two parties disagree on who is responsible for an action, they can get a ruling from the judicial system. Thus there already is an existing external system for settling conflicts, and any proposed system for accountability should support that existing system, rather than trying to replace it. This implies that any accountability system should be concerned with gathering usable evidence for use in a court of law, rather than trying to stop all misuse. The same observation is made in [87] and other texts on non-repudation or electronic commerce [21].

Another implication is that a court of law, at least in the western tradition, the evidence has to remove any "shadow of a doubt" to rule against a defending party. This can easily be viewed as a statistical test. As long as the evidence shows that there is a small, but not insignificant, possibility that someone else could have performed the contested act, the defendant is viewed as innocent. This can possibly be used to create system for accountability that instead of securing *every* action, secures *enough* actions to be within satisfactory statistical bounds for a claim against someone.

In this thesis all agreements between a service provider and a user, goes through a broker infrastructure. The brokers main objective are to facilitate fair billing and metering amongst the providers and users.

[4]Or the designer simply was not aware of that an assumption was made.

### 4.1.2 Billing

It is assumed that there exist a functioning system for billing, i.e. the proposed system should gather accountable metering data and run the financial settlement on existing payment infrastructures. The main rationale behind this is to reduce the overall complexity.

**Assumption 4.5.** *Billing and payment use existing infrastructures.*

## 4.2 General goals

The primary goals for this thesis is to give users access to their data:

- From any client
- From anywhere
- At anytime
- Only restricted by the access policy for a data item
- Provide users with sufficient privacy

Main goals for the support of the needs of service providers and authorities, is to provide sufficient user accountability to support:

- Metering and billing of storage services
- Investigation of unwanted activity

Basically we try to safeguard the privacy of honest users and simultaneously support investigation of suspected crimes.

## 4.3 Users

Users are expected to have access to one ore more clients, and that they interact with multiple services during a session.

### 4.3.1 Assumptions

The user has access to a client that enables the user to:

- Do basic cryptographic computations
- Store some longterm secrets (i.e keys) safely
- Send, receive, and respond to messages to and from other entities

There is made no distinction between the user and its terminal or client. For the other entities they are for all practical purposes inseparable for any specific session. When a broker or other actor are authenticating or authorizing users, the combination of the users identity, terminals identity, and their relative (i.e. in- or outside a perimeter) should be taken in to account before any final decision on authorization is made.

The last assumption is that the user has limited processing and bandwidth compared to service providers. Thus any protocol has to limit the users and their terminals interaction in a protocol to the bare minimum.

### 4.3.2 Goals

Users primary goals are:

- Privacy
- For data and applications:
  - Confidentiality (if its sensitive)
  - Integrity
  - Availability
- The possibility to detect tampering or dishonest entities
- Be able to dispute false or wrong billing

## 4.4 Services

Services may be of the following types:

**Fabric** $S_F$ is any resource capable of transporting data from one location to another.

**Processing** $S_P$ is any resource capable of doing computations, i.e. raw CPU.

**Application** $S_A$ is any application made available to users. This service typically requires $S_F$, $S_P$, and $S_S$ to be useful.

**Storage** $S_S$ is any form of storage service available.

**Input** $S_I$ is any other form of input service.

**Output** $S_O$ is any other form of output service.

These service types are sufficient for the treatment of security protocols and system model in this thesis.

### 4.4.1 Assumptions

The internals of a service are invisible to users and brokers. Chapter 2.1.6 provides more background on this.

Services are made available by *service providers* that may or may not want compensation for the use of the services.

Service location, capabilities, cost, etc., is published in catalogs available to all other entities.

### 4.4.2 Goals

Services as such has no goals. The service providers, i.e. service owners, has multiple goals:

- Accurate and accountable metering
- Flexibility in service delivery
- Efficient dispute settlement
- Absence of liability for users usage of services[5]

## 4.5 Brokers

Brokers are the users only way to gain access to services through the fabric. For the sake of this thesis brokers has vastly more computational power than the users terminals.

### 4.5.1 Assumptions

All brokers are assumed to be able to meter the resources used by a user when accessing a service and its resources. Metering is done with *observers* as described in chapter 5.2.

It is also assumed that both the user and service provider agrees on at least one broker that they both trust to meter fairly. This makes brokers a kind of trusted third party.

### 4.5.2 Goals

Brokers has the same goals as services, see chapter 4.4.2.

[5] E.g. a storage service do not want to held liable for a user storing pirated software at their facility.

## 4.6 Trusted third parties

Trusted third parties are key to achieving efficient non-repudation protocols[87]. There are several kinds of trusted third parties that can be useful. A list of some typical ones are:

**Certificate Authority** $T_{CA}$ is an entity that signs other entities public keys to verify that that public key belongs to that entity.

**Time stamp** $T_{TS}$ is an entity that attach a cryptographically secured time stamp to a bit string presented to it from other entities.

**Notary** $T_{NO}$ is an entity that will notaries, and optionally store, a bit string presented to it from other entities. Typically $T_{NO}$ signs the bit string with the other entities signatures attached.

**Key distribution** $T_{KD}$ is an entity that will distribute cryptographic keys to other entities as needed or requested. Typically $T_{KD}$ has access to all keys.

**Key translation** $T_{KT}$ is an entity that will translate cryptographic keys for other entities as needed or requested. Typically $T_{KT}$ has access to all keys and is situated between the other entities.

**Key storage** $T_{KS}$ is an entity that will store[6] cryptographic keys for other entities as needed or requested. $T_{KS}$ might have clear text copies of the keys, but this is not an absolute requirement.

All these can be viewed as a kind of service. Most of the assumptions and goals of services, chapter 4.4.2, will apply to trusted third parties to, but trusted third parties have extra assumptions and goals. Thus it was chosen to separate services and trusted third parties to simplify later discussion and analysis.

Even though a trusted third party is *trusted*, that trust *should not* be blind. It is critical for any security design or architecture that one has a very clear understanding of what a trusted third party is trusted to do.

### 4.6.1 Assumptions

The different types of trusted third parties are assumed to be trusted for the type of task they perform, and not for any other tasks. Thus a time stamping trusted third party is not trusted to provide e.g. key storage.

[6]This is the original trusted third party.

### 4.6.2 Goals

The goals of trusted third parties are as services the same as for services in general, chapter 4.4.2, but in addition trusted third parties also has to protect their trustworthiness. Loss of trust in a trusted third party can come from both actions by the trusted third party itself or by attacks from the outside. If the trusted third party fails internally, little or nothing, can be done to reestablish its trust[7]. If however the attack comes from the outside, either as a semantic or direct attack, a trusted third parties goal will be to safeguard its reputation as a *trusted* third party.

## 4.7 Catalogs

Catalogs are a service, but are critical for all other entities, including other catalogs, for finding the necessary meta data in different settings. Thus catalogs will have more emphasis on integrity and availability than most other services.

### 4.7.1 Assumptions

A catalog is assumed here to be non-malicious, i.e. it does not tamper with the entries within itself.

Catalogs support both naming and name resolution for other entities.

### 4.7.2 Goals

Catalogs have the same goals as services, see chapter 4.4.2.

[7] Although Verisign seems to have lost very little trust even though they issued code signing certificates to a faked Microsoft request.

# Chapter 5

# System model

We showed in chapter 2, there are other efforts to have policy follow data. The other efforts, and digital rights management in particular, has a focus on securing an institutions rights rather than an individual.

## 5.1 Architecture

Figure 5.2 shows the overall architecture. All actions done by the user are initiated via the users client.

A prerequisite is that users use a trusted third party to get a pseudonym issued. The trusted third party will disclose the real identity of a user if proper evidence is presented and proper procedures are followed. This pseudonym is then embedded in or transferred to the *storage agent*. The storage agent then uses this pseudonym as the identity presented to all other entities in the system.

Communications between the users client and storage agent is encrypted, i.e. they create a secure channel. Data produced by the user at the client is transferred in as clear text within the existing secure channel to the storage agent. The storage agent then choses, with the assistance of *brokers*, a suitable *storage provider* to store the data on.

The storage provider sends a *storage observer* to the storage agent. This observer is used to create a signed hash of the data that will be stored at the storage provider facility. This hash is taken over the unencrypted data. This enables the storage provider to respond to request by authorities etc. to check for specific data. Before the data is transferred to the storage provider it is encrypted, and the hash made by the storage observer and the pseudonym used, is cryptographically tied to data. Thus a storage provider can only answer with a pseudonym, even if the hash matches that of the data in interest. This facilitates investigative measures, but does thwart a big brother like surveillance in the architecture.

If an other entity is allowed to use the data, this can be achieved by at

least two fundamentally different methods. One is to hand over the necessary keys to the other entity, but this obviously is in contradiction with our goals. The other is to create a limited storage agent that gives the other entity a interface which the other entity can operate on data via. This is not described further here, but is clearly one of the more difficult problems not explored here.

Finally all entities use the catalog to store names and metadata. The catalog also serves the purpose of resolving names into addresses.

## 5.2 Processing

Achieving trusted processing at an external resource is not trivial. As an aid for this thesis an abstract model of a sandboxed remote environment was used to ease the mental process.

In this abstract environment the outer sandboxis there to protect the native platform from the clients actions inside the contained sandboxes. The inner sandboxesare there to protect the clients applications etc. from the native environment, the outer sandbox, and other clients inner sandboxes. This idea is not revolutionary; this concept has been used by IBM, Digital, and others to create security systems for a variety of scenarios, see [48] for examples. The observeris inspired by a similar concept used in the CAFE project [21, page 186], but here it is used as tool to assist brokers in getting their metering data. An interesting effect of having an observer that gathers the metering data, is that if metering is to costly for a e.g. grand challenge computation, a user, broker and service provider might agree to have a statistical sampling in stead of full metering without any fundamental changes in the underlying model.

Here the outer sandbox is what a service provider uses to facilitate the possibility to sell or lease unused services or resources to other entities. The inner sandbox gives the users of the services an ability to monitor their own actions and compare this with their later billing data. It also enables a user to detect attempts to tamper with its application or data. The observers give brokers a view of the transactions between the inner and outer sandboxes, so that brokers can e.g. gather metering data or detect foul play.

Other possible approaches for trusted computations are briefly discussed in 2.3.

## 5.3 Storage agents

Here we are concerned with how data is accessed, not how it is stored on any particular medium. The key issue is how we can create a firm link between the data and its associated usage policy. Our model is a suggestion on how

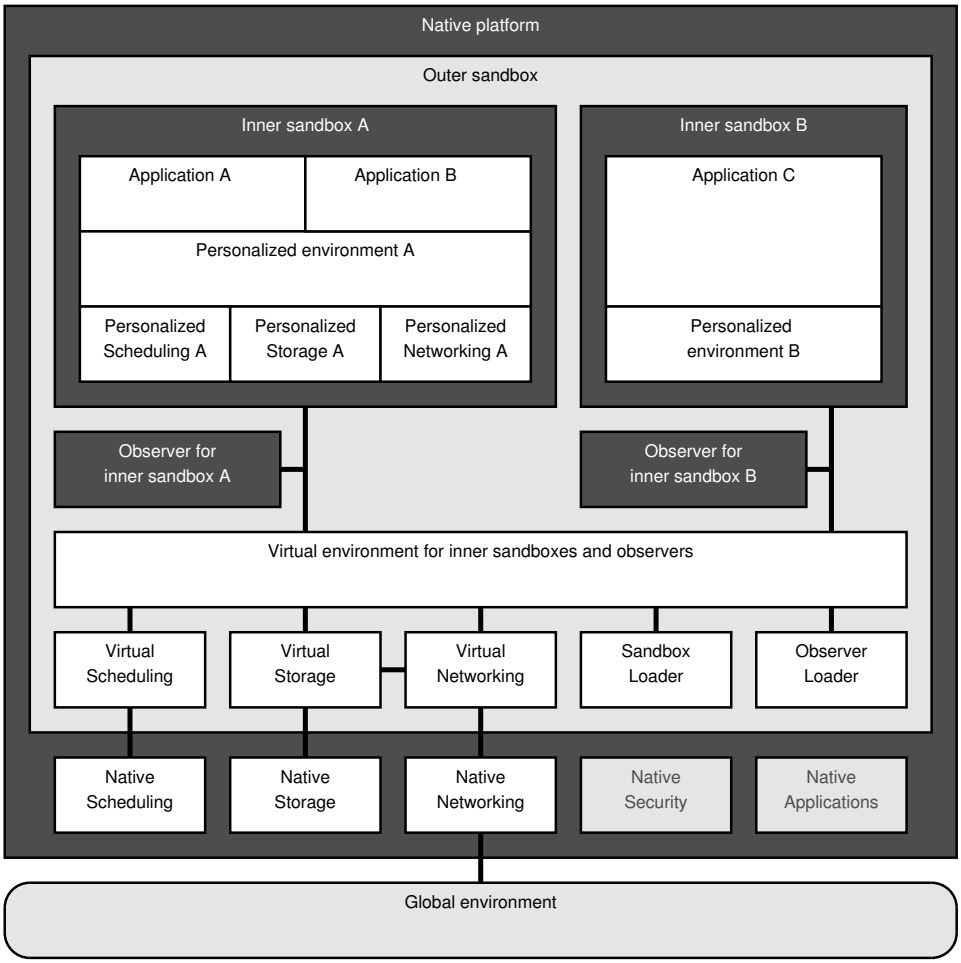


**Figure 5.1:** Sandbox

this can be achieved by using a storage agent[1]. An obvious result of having a storage agent operating on behalf of a user, is that it somehow must be able to do its processing in a trusted manner. In the previous section, 5.2, we have describe the model used as a basis for this thesis.

Here the *user*uses one or more *clients*to access one or more *storage agents*that provide access to one or more *storage providers*via suitable *storage adapters*.

The user, trusted third party, and catalog will not be further discussed here as they are sufficiently treated in chapter 4.

[1]An implementation does not have t be based on agent or mobile agent technologies, but the word *agent* best describes the concept presented here.

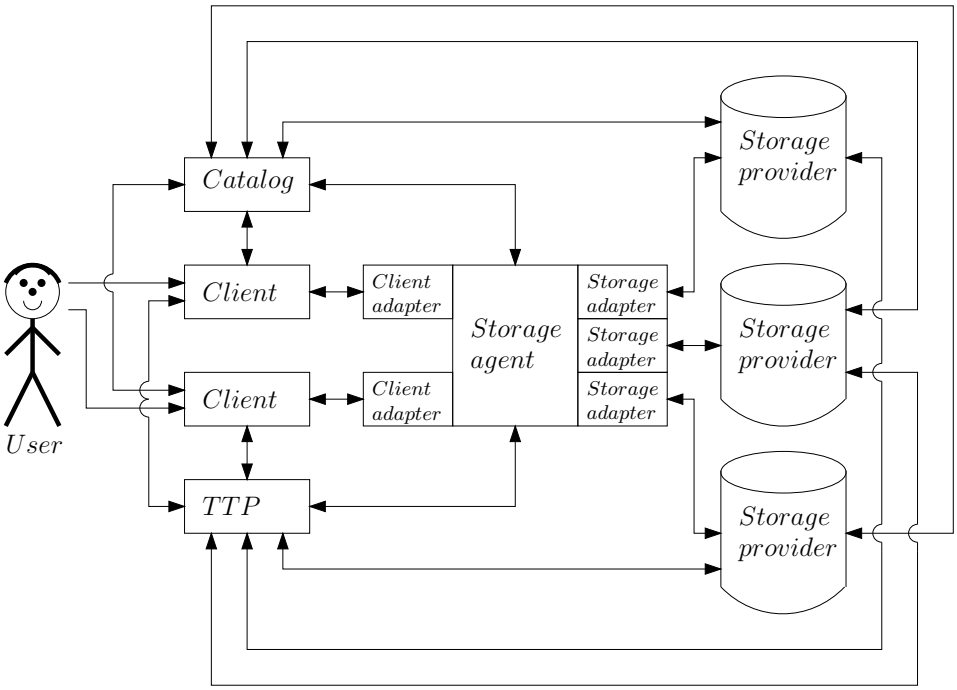


**Figure 5.2:** Storage model

### 5.3.1 Client

Clients can here be any platform or code that provides a user with a usable interface to interact with a storage agent. Thus we make no other assumptions on the clients than that they are able of some form of communication with a client adapter, catalog, and trusted third party, and that they provide the user with a user interface suitable for the task at hand.

If we want to protect the users privacy, the client and client adapter must provide a secure channel[31] over the underlying communication infrastructure. Our only assumption on this secure channel, is that it must use a mechanism, cryptographic or other, that effectively thwart attacks on the channel by a Dolev-Yao attacker.

A users clients are one of few entities that by default sees the users data in clear text. The data is encrypted in transit to the storage agent with a secure channel and when it resides at a storage provider. This protects the users privacy.

The final assumption is that the client can execute in an environment trusted by the user.

### 5.3.2 Storage agent

Storage agents are the enforcers of policy on data access and usage. They also serves the role of brokers for the access to storage services. Storage agents act on the behalf of one or more users, and have access to all keying material used to encrypt users data. The storage agents does their processing in an trusted environment.

Storage agents are allowed to spawn copies of themselves, as long as they are registered in a catalog, to either increase availability for its users, or to increase the throughput in data transfers. This opens for the possibility for users to instruct the storage agent to transfer data without the using the users client platform or bandwidth.[2]

#### Client adapter

The users only interface to a storage agent, and ultimately the storage provider, is via the clients client adapter. Thus there has to exist some authentication and authorization mechanism between the client and client adapter, or more precisely between the user and storage agent.

The rationale behind having multiple client adapters are to maintain the important aspects of flexibility for users and future proofing a system model. If there was a fixed interface between the client and storage agent, any significant change in communication technology between them, would cause a major change in interface. It would also make this model less flexible in providing the users real options in how they want to access and use their data.

The interface between the client adapter and the storage agent must be fixed. This will probably be one the main challenges to create if one wishes to implement this model.

### 5.3.3 Storage adapter

Flexibility, as for client adapters, is the main rationale for storage adapters, too.

Figure 5.3 does not show the observer from an impartial broker, but this is necessary when fair metering is needed. The main goal is to show how a storage provider can get a fingerprint, i.e. hash, of the unencrypted data. The *storage observer* is similar to a brokers observer, but is only allowed to transmit a hash and a signed hash back to the storage provider. Before the data is stored the storage adapter binds the policy to data. The policy is provided to the storage adapter by the storage agent.

All user keys, $k$, $P_U$ and $S_U$ is provided to the storage adapter by the storage agent. The keys, $P_S$ and $S_S$, for the storage observer is embedded

[2]This functionality has existed in FTP for many years, but without our security model.

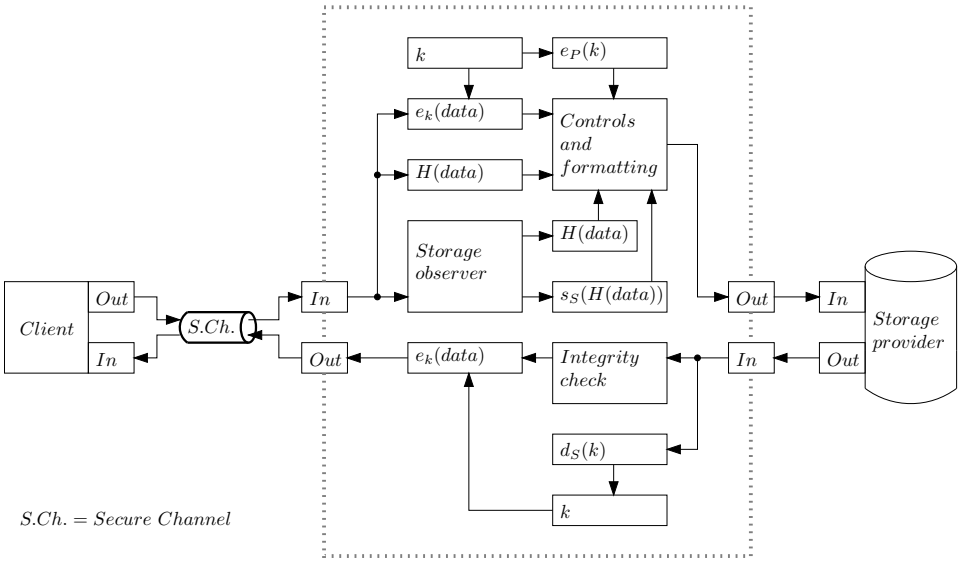


**Figure 5.3:** Storage adapter

in it when the storage provider transmits it to the storage adapter.

The resulting data file stored at the storage provider will look something like this:

$DataID \| Pseudonym \| H(data) \|$
$s_{S_U}(H(policy \| data)) \| s_{S_S}(H(data)) \|$
$e_{P_U}(k) \| e_k(policy \| data)$

**DataID** An unique ID for the data item

**Pseudonym** The pseudonym of the user owning the data item

**U** User

**S** Storage provider

**policy** Usage policy for the data item

**data** The data

An implementation would have to refine on these items.

The storage adapter gives a good starting point for finding a possible solution to the problem, see chapter 5.1, of giving other entities access to the data. Probably

### 5.3.4 Storage provider

The storage provider offloads most of the cryptography to the storage observer. This lowers the processing power needed at the storage provider, while the work load of the storage agent only increase with an extra hash and signing operation.

With our scheme the storage providers should not be liable for what data users store at their facilities. They have never seen the unencrypted data, but still can support an investigation.

## 5.4 Privacy enhancing network

If we are to guarantee privacy in spite of powerful adversaries, we have to provide some means to thwart traffic analysis. Traffic analysishas a long, and effective history, starting at WWI and playing a key role in WWII[40]. The traditional defenses have been either to hide the communication[61] or to have all channels operating at full speed at all times. In a commercial setting neither are particularly practical or cost effective. Thus we have to have an other way to introduce enough noise in the communication channels to thwart traffic analysis.

Based on ideas drawn from the Mixmaster[84] project and I2P[82], we propose here to create an peer-to-peer privacy enhancing network where the endpoint of a logical channel can identify each other while all other nodes or attackers have no effective means to identify which nodes are talking to each other.

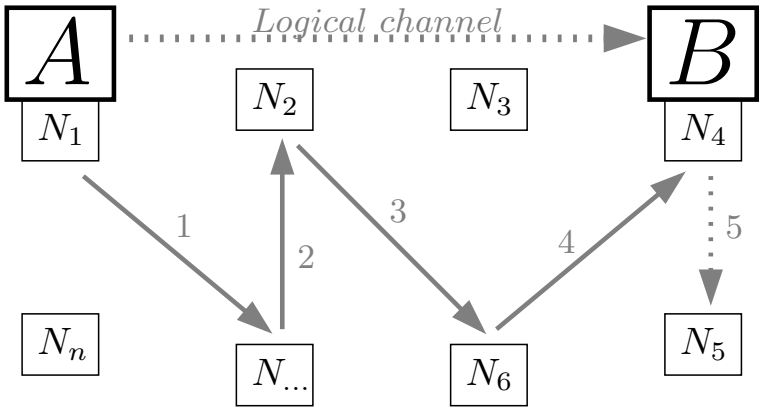


**Figure 5.4:** Privacy enhancing network

Here $A$ sends a message to $B$ via a subset of the peer-to-peer nodes. If this message is to be protected from the intermediate nodes, it is obvious that the message has to be encrypted. We also have to hide the eventual

endpoint from the intermediate nodes. Thus we have to describe message formats and protocols to achieve this. These protocols is not thoroughly analyzed here, but some suggestions and claims are made.

The basic message format is a header with routing information and a payload that can contain anything. The header must as a minimum have to contain information on:

- The address of the next hop node
- A minimum waiting period
- A maximum waiting period
- An integrity check for the payload

All these items and the payload must then be encrypted under the receiving nodes public key to stop an external observer from following a bit pattern end to end.

A countermeasure against traffic analysisis to create sufficient noise in the timing between events. Here the sender can for each hop specify a minimal and maximal waiting period before the message is transferred to the next hop. If there there are numerous enough messages this will decrease an adversaries ability to correlate incoming and outgoing messages from a node.

An other possible countermeasure is to have each node encrypt the message, which is already encrypted, again, but attach some random amount of padding. This will create further problems for adversaries in correlating incoming and outgoing traffic at a node. This will stop a possible weakness in the scheme proposed here, since a message grows smaller for each hop, but at the cost of more processing at each hop. As a compromise we suggest that by allowing for a trailing set of decoy messages, an adversary can not identify which node in a chain of messages that received the real message and not a decoy.

The header contains no information on who the sender is at all. This stops dishonest nodes from gathering information for traffic analysis, but at the same time opens for denial of service attacks. How we can stop such attacks, or at least minimize their impact, is not considered here. Other peer-to-peer networks tries to solve the problem of dishonest participants, often called leeches, in a number of ways, but here no effort has been made to apply them to this model.

As shown in figure 5.4, a message should go through a number of hops between endpoints, and possibly have a trail of decoy messages, to hide the true communicating parties. Thus a message can look something like this:

Here it is easy to see that a possible weakness in this scheme lies in that the node that receives a message might be identified by the fact that the outgoing message is significantly smaller than the incoming message. This

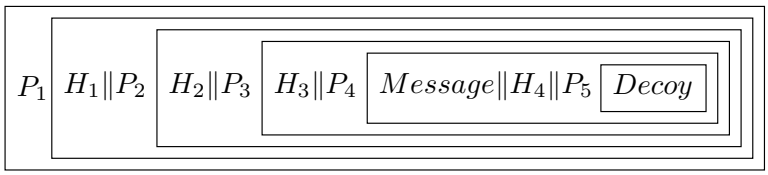


**Figure 5.5:** Message format example

can be thwarted by having multiple decoys of varying sizes, but this will decrease the overall performance.

# Chapter 6

# Protocols

There will be several sub-protocols involved in any reasonably large and complex distributed environment. When we then add a very high level of dynamics to the mix, the need for sub-protocols soon becomes apparent. In RFC 1925 [20] it is claimed that:

> . . . In protocol design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away. . .
> *Ross Callon (ed.)*

Even though this is one of the classic April Fool's RFCs, the principle of simplicity is one of the most important basis for any sensible security system. This is emphasized by both by Ross Anderson in [61] and Peter Neumann in [56].

## 6.1 Flow patterns

One method to identify potential protocols in a design, system, model or architecture is to create *flow patterns*. This is simply drawings that shows the interaction between the different entities, and serves as an informal method to facilitate further refinement of needed protocols.

In figure 6.1 the interactions discussed here are drawn black. The interactions in gray is not discussed here, but will probably be essential for a possible implementation. Interactions with dotted gray lines are considered to be out of scope. Flow patterns can be refined further by e.g. drawing the information flow in each interaction between two participants in detail to identify protocol steps.

The interaction identified so far are:

**A** User ↔ Client – The user interface

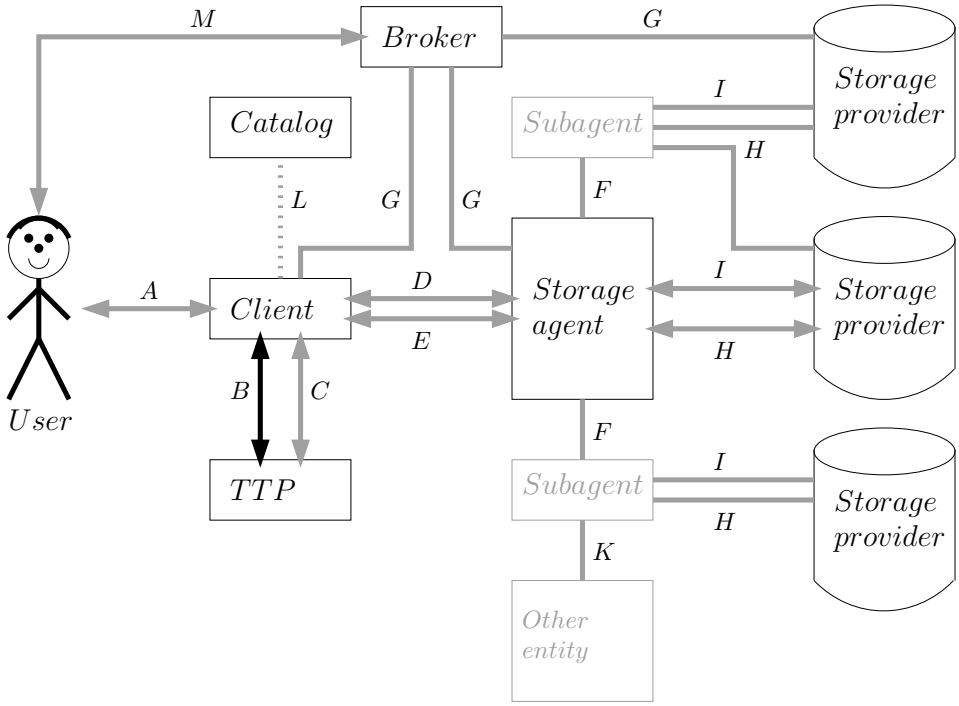


**Figure 6.1:** Flow patterns

**B** Client ↔ Trusted third party – Request for and issuance of a pseudonym and its certificate

**C** Anyone ↔ Trusted third party – Check of certificate validity

**D** Client ↔ Storage agent – Embedding of pseudonym and initial keying

**E** Client ↔ Storage agent – Data transfer, usage, and storage

**F** Storage agent ↔ Subagent – Creation and keying

**G** Anyone ↔ Broker – Service agreement, observer creation, and metering

**H** Storage agent ↔ Storage provider – Acceptance of storage observer

**I** Storage agent ↔ Storage provider – Use of data storage services

**K** Subagent ↔ Other entities – Data access interface and policy enforcement

**L** Anyone ↔ Catalog – Storing and retrieving meta data, names, etc.

**M** Broker ↔ User – Billing and payment

Even for a relatively small system and little, it is easy to find an abundance of interactions. All these interactions might need their own protocol or sub-protocol. However for an efficient implementation, we would have to use a lot of resources on protocol design. In particular one should try to minimize the protocols involved, and if possible use existing protocols.

Here the goal was to try and identify potential protocols, and if possible do a formal analysis of some of them. With our primary goal of protecting users privacy, it is easy to see that the creation of a pseudonym is critical for all subsequent actions. If we can not create a sensible pseudonymity protocol, the system as a whole fails to reach this goal. Thus we have only specified and analyzed potential candidates for pseudonym issuance protocols.

## 6.2 Pseudonymity

Full anonymity would probably gain malicious users more than honest users, the goal is to provide strong pseudonymity, i.e. guarantee that an individual can remain anonymous as long as there are no disputes or other strong reasons to identify individuals.

An efficient pseudonymity scheme seems to require that a users trusts at least one trusted third party unconditionally to perform a pseudonymizing function correctly for them. A key question then is when should a user acquire a pseudonym:

- At the start of every session?
- When needed, i.e. just before sensitive actions?
- Whenever a user wants to be *invisible*?

The first is the simplistic view that whatever a user chooses to do, is the users business. However, before making this choice, one should carefully consider if there might be to high performance impacts for constrained clients.

**Definition 6.1.** *A constrained client is a client that has severe limitations in processing, storage or bandwidth resources.*

The second option will probably be the most efficient in terms of resources used for pseudonymity, but rises at least a couple of concerns.

- How can we define what is *sensitive* in a global society?
- Can this model support *traffic analysis* so that pseudonymity is in reality lost?

These concerns will probably make this option unrealistic because of general privacy concerns. The main problem seems to that the users will not have

direct control on the choices, and that they will be at the mercy of choices made by others.

The last option seems most promising since it empowers users. Thus if a user accepts the performance penalty, they can chose to be pseudonymized at all times, or they can choose to wait until they perform actions they themselves deems sensitive. A remaining problem is if the general population of users can make informed choices on what is sensitive or not.

### 6.2.1 Pseudonymity flow pattern

The interactions for a pseudonymity service is shown in figure 6.2.

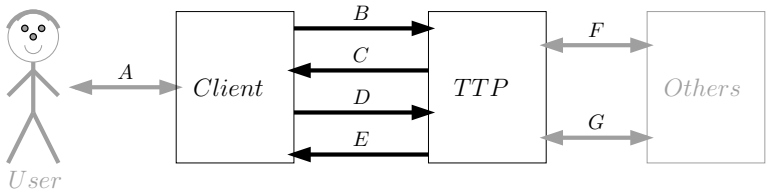


**Figure 6.2:** Pseudonymity flow patterns

**A** User ↔ Client – The user interface

**B** Client ↔ Trusted third party – Identity of client and user

**C** Trusted third party ↔ Client – Identity of trusted third party

**D** Client ↔ Trusted third party – Attributes of pseudonym

**E** Trusted third party ↔ Client – Signed certificate

**F** Anyone ↔ Trusted third party – Check of validity of pseudonymity certificate

**G** Anyone ↔ Trusted third party – Dispute settelment

Items A, F, and G are out of scope of the core pseudonymity issuance. Thus we will try to create an efficient and secure[1] protocol for items B through E. Protocol efficiency is often measured in the total number of steps involved, so we will strive for the lowest possible number of steps.

[1]Secure in the sense that it passes the tests of the chosen analysis methodology.

### 6.2.2 Pseudoprotocol

The protocol can be described at a high level as:

1. User contacts trusted third party to obtain a pseudonym
2. Trusted third party authenticates and authorizes the user
3. The user specifies the parameters the pseudonym should contain
4. The trusted third party issues the pseudonym

After the protocol run the trusted third party stores the pseudonym with the users real identity until it expires. Expiry here is the expiry time of the pseudonym plus a reasonable period after that expiry so that possible disputes can be solved.

A trivial way of creating pseudonyms is to issue short term digital certificates with the subjects name as a randomized string the trusted third party can use to resolve which real user it was issued to. Another possibility might be to use hash-chains in some form. A protocol based on hash-chains might be better suited for constrained client platforms, but hash chains are not easy to model in Casper. Thus we will base our protocols on RSA-like algorithms.

We will try two approaches. First we try to do the protocol (v.1) with just two messages. The second version (v.2) uses a total of four messages, where the first two authenticates the participants and the last two is the actual data exchange.

The complete Casper scripts are found in the appendices. The use of Casper and FDR2 are described in detail in [46] and [26], and will not be restated here.

### 6.2.3 Description of protocol fields

For all the protocols in this chapter, the following names and notations will be used:

$U$ Identity of the user

$P$ Identity of the trusted third party

$nx$ A nonce $x$ generated by participant $X$

$data$ The request for a pseudonymity certificate

$f(data)$ The certificate made by applying function $f$ on the $data$[2]

$P_X$ Participant $X$'s public key

[2]A better way of modelling certificate request in Casper was not found by the author.

In the Casper scripts and final analysis the user is called *Ulrich*, the trusted third party is called *Peter*, and the intruder is called *Mallory*. Names prefixed with $I\_$ are instances where Mallory acts as an *imposter*.

### 6.2.4 Protocol v.1

The first challenge was how to model a *certificate request* and the corresponding computation of the *certificate reply*.

The first attempt:

**Table 6.1:** Protocol v.1

| Message # | Direction | Message |
|---|---|---|
| 1 | $U \rightarrow P:$ | $U, P, (U, P, data)_{P_P}$ |
| 2 | $B \rightarrow A:$ | $U, P, (U, P, f(data))_{P_U}$ |

### 6.2.5 Analysis v.1

This protocol is vulnerable to a man-in-the-middle (MIM) attack.[3]

```
0.          ->  Ulrich   : Mallory
1.  Ulrich  -> I_Mallory : Ulrich, Mallory, {Ulrich, Mallory, Data}{PK(Mallory)}
1. I_Ulrich ->   Peter   : Ulrich, Peter, {Ulrich, Peter, Data}{PK(Peter)}
2.  Peter   -> I_Ulrich  : Peter, Ulrich, {Peter, Ulrich, f(Data)}{PK(Ulrich)}
  The intruder knows Data

Goodbye
```

The cause of the attack is the lack of liveness checks and weak mutual authentication.

### 6.2.6 Protocol v.1.1

Here we try to thwart the MIM attack, by utilizing a nonce to verify that the trusted third party is alive, and to prevent session replays.

The second attempt:

**Table 6.2:** Protocol v.1.1

| Message # | Direction | Message |
|---|---|---|
| 1 | $U \rightarrow P:$ | $U, P, (nu, U, P, data)_{P_P}$ |
| 2 | $B \rightarrow A:$ | $U, P, (nu, U, P, f(data))_{P_U}$ |

[3] This the actual output from Casper.

### 6.2.7 Analysis v.1.1

This protocol is also vulnerable to a MIM attack.

```
0.         ->  Ulrich   : Mallory
1. Ulrich  -> I_Mallory : Ulrich, Mallory, {Nu, Ulrich, Mallory, Data}{PK(Mallory)}
1. I_Peter ->   Peter   : Peter, Peter, {Nu, Peter, Peter, Data}{PK(Peter)}
2.  Peter  ->  I_Peter  : Peter, Peter, {Nu, Peter, Peter, f(Data)}{PK(Peter)}
  The intruder knows Data

Goodbye
```

The cause of the attack thus seems to be caused by weak mutual authentication.

### 6.2.8 Protocol v.2

Our final attempt sacrifices the efficiency of the two first attempts, and uses a total of four messages. The first two attempts to thwart the MIM attack by having an exchange of encrypted nonces prior to the actual certificate request and issuance.

Our third attempt:

**Table 6.3:** Protocol v.2

| Message # | Direction | Message |
|---|---|---|
| 1 | $U \rightarrow P:$ | $U, P, (U, P.nu)_{P_P}$ |
| 2 | $B \rightarrow A:$ | $U, P, (P, U, nu, np))_{P_U}$ |
| 3 | $U \rightarrow P:$ | $U, P, (U, P, nu, np, data)_{P_P}$ |
| 4 | $B \rightarrow A:$ | $U, P, (P, U, nu, np, f(data))_{P_U}$ |

### 6.2.9 Analysis v.2

This version of the protocol passes the analysis made by Casper and FDR2.

### 6.2.10 Discussion

Have we proved anything? In [49] Meadows restates the fact that protocol security problem is undecidable. Thus we can not prove that a protocol is secure in general, only that it is immune to attacks under a set of assumptions.

A more important issue is whether the author has used the Casper tool correctly or not. This is the first attempt to analyze protocols outside the Casper user manual and tutorial. There is a high probability that the author has missed some finer points on how to effectively use the tools.

The area that causes most concern, was the inability to model certificate issuance in a good manner. Here we chose to simulate it by just using a hash-function on the request. This can have an impact on the result of the analysis.

The final question is whether or not we have found a *new* protocol. This exact protocol is probably new, but it is not a pure pseudonymity protocol in the sense used here. It looks more like a protocol for general use for issuing certificates on-line to known users.

# Chapter 7

# Conclusion

## 7.1 General conclusions

We have shown that storage challenges can be solved in other ways than tradition approaches. Some new theory on the subject of privacy and accountability has put forth, and been applied to system model presented here.

A proposed protocol for issuance of pseudonyms has been made and analyzed with a tool set for formal analysis. However this protocol is probably not a pure pseudonymity protocol in our sense, but rather a protocol for issuance of certificates to known users.

## 7.2 Answers to research questions

### 7.2.1 Question 1

*How do current systems for distributed computing address storage challenges?*

In general we found that current systems do not effectively enforce the policy that applies to a particular data item after it has been transferred. There are several security infrastructures available, but few, if any, address the binding of policy to data items.

### 7.2.2 Question 2

*How can policy be guaranteed to follow the data it controls access to?*

We have not shown that we can *guarantee* that the policy will follow data in every situation. A system model, survey of the state of the art, and some theoretical contributions have been made. These can be utilized to explore this question further. We have shown that it might be *possible* to create such systems.

### 7.2.3 Question 3

*How can security, flexibility, privacy, and accountability all be simultaneously be achieved for storage in distributed dynamic environments?*

In general we have not given a general proof of this. We have demonstrated, mainly through the system model, that such system *can* exist.

### 7.2.4 Question 4

*What are the minimal assumptions needed to achieve question two and three?*

A complete minimal set of assumptions was not found. However a few assumption is found to be necessary for the overall security for any distributed dynamic system:

- A trusted and tamper proof processing environment
- At least one trustworthy trusted third party
- Security protocols that are immune against powerful adversaries

In general we can say that any security system rests on the first of these three assumptions.

## 7.3 Further work

During our work on this thesis there were many interesting topics that was not explored. Here we present a summary of the some of them:

- Create a prototype of system model, in particular the sandboxed processing with observers presented in chapter 5.2.
- Explore if a statistical or probabilistic approach to information security can create more efficient security models for distributed dynamic environments.
- Do a more thorough analysis of how agent based storage compares in security and functionality to other forms of distributed storage.
- Explore how average Jane and average Joe can be educated or empowered to make informed decisions on their security needs and risk acceptance level.
- Explore if it is possible to enumerate or calculate what level of decoy messages that are necessary to effectively hide real traffic in an anonymity network depending on the size and utilization of that network.

- Explore how Denial-of-Service attacks can be limited or thwarted in peer-to-peer networks or in general.
- Reasoning on attack models for non-repudation systems, and possibly other security areas than just for cryptographic protocols.

# Appendix A

# Notation

The notations used in this thesis is mainly taken from the notations used in [87] and [26] combined with influences from numerous other books, reports and articles.

**Table A.1:** Notation

| Notation | Explenation |
|---|---|
| $X \| Y$ | Concatenation of two messages $X$ and $Y$. |
| $[X]$ | Message $X$ is optional. |
| $H(X)$ | A one way hash function applied to message $X$. |
| $g_K(X)$ | An integrity mechanism for message $X$ with key $K$. |
| $f_x$ where $x \in \{1, 2, \ldots, n\}$ | A flag in a message. |
| $SENV_K(X)$ | A secure envelope that is generated by using symmetric cryptographic techniques and allows the holder of secret key $K$ to authenticate the integrity and origin of message $X$. |
| $e_K(X)$ | Encryption of message $X$ with key $K$. |
| $d_K(X)$ | Decryption of message $X$ with key $K$. |
| $s_{S_A}(X)$ | Principals $A$'s digital signature on message X with the private signature key $S_A$. |
| $S_A$ | Principals $A$'s private signature key. |
| $V_A$ | Principals $A$'s public verification key. |
| $P_A$ | Principals $A$'s public encryption key. |
| $P_A^-$ | Principals $A$'s private decryption key. |
| $A \rightarrow B : X$ | Principal $A$ sends message $X$ to principal $B$. |
| $A \leftarrow B : X$ | Principal $A$ recives message $X$ from principal $B$. |

# Appendix B

# Casper script - Protocol v.1

```
#Free variables
U, P : Agent
data : Nonce
f : HashFunction
PK : Agent -> PublicKey
SK : Agent -> SecretKey
InverseKeys = (PK, SK)

#Protocol description
0.    -> U : P
1. U -> P : U, P, {U, P, data}{PK(P)}
2. P -> U : P, U, {P, U, f(data)}{PK(U)}

#Processes
INITATOR(U, data) knows PK, SK(U)
RESPONDER(P)  knows PK, SK(P)

#Specification
Secret(U, data, [P])
Secret(P, data, [U])
Agreement(U, P, [data])
Agreement(P, U, [data])

#Actual variables
Ulrich, Peter, Mallory : Agent
Data : Nonce

#Functions
symbolic PK, SK

#System
INITATOR(Ulrich, Data)
RESPONDER(Peter)
```

```
#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Ulrich, Peter, Mallory, PK, SK(Mallory)}
```

# Appendix C

# Casper script - Protocol v.1.1

```
#Free variables
U, P : Agent
data, nu: Nonce
f : HashFunction
PK : Agent -> PublicKey
SK : Agent -> SecretKey
InverseKeys = (PK, SK)

#Protocol description
0.    -> U : P
1. U -> P : U, P, {nu, U, P, data}{PK(P)}
2. P -> U : P, U, {nu, P, U, f(data)}{PK(U)}

#Processes
INITATOR(U, data, nu) knows PK, SK(U)
RESPONDER(P)  knows PK, SK(P)

#Specification
Secret(U, data, [P])
Secret(P, data, [U])
Agreement(U, P, [data])
Agreement(P, U, [data])

#Actual variables
Ulrich, Peter, Mallory : Agent
Data, Nu : Nonce

#Functions
symbolic PK, SK

#System
INITATOR(Ulrich, Data, Nu)
RESPONDER(Peter)
```

```
#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Ulrich, Peter, Mallory, PK, SK(Mallory)}
```

# Appendix D

# Casper script - Protocol v.2

```
#Free variables
U, P : Agent
data, nu, np : Nonce
f : HashFunction
PK : Agent -> PublicKey
SK : Agent -> SecretKey
InverseKeys = (PK, SK)

#Protocol description
0.    -> U : P
1. U -> P : U, P, {U, P, nu}{PK(P)}
2. P -> U : P, U, {P, U, nu, np}{PK(U)}
3. U -> P : U, P, {U, P, nu, np, data}{PK(P)}
4. P -> U : P, U, {P, U, nu, np, f(data)}{PK(U)}

#Processes
INITATOR(U, data, nu) knows PK, SK(U)
RESPONDER(P, np) knows PK, SK(P)

#Specification
Secret(U, data, [P])
Secret(P, data, [U])
Agreement(U, P, [data])
Agreement(P, U, [data])

#Actual variables
Ulrich, Peter, Mallory : Agent
Data, Nu, Np : Nonce

#Functions
symbolic PK, SK

#System
```

```
INITATOR(Ulrich, Data, Nu)
RESPONDER(Peter, Np)

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Ulrich, Peter, Mallory, PK, SK(Mallory)}
```

# Bibliography


[1] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[2] Ahmar Abbas. *Grid Computing: A Practical Guide to Technology and Applications*. Charles River Media, Inc., 2004.

[3] Paul van Oorschot Alfred J. Menezes and Scott Vanstone. *Handbook of Applied Cryptography*. Discrete Mathematics and Its Applications. CRC Press LLC, 1997.

[4] The Globus Alliance. GSI Documentation, June 2004. http://www-unix.globus.org/toolkit/docs/3.2/gsi/index.html.

[5] Ross Anderson. 'Trusted Computing' Frequently Asked Questions, August 2003. http://www.cl.cam.ac.uk/ rja14/tcpa-faq.html.

[6] Jan A. Audestad. *Telecommunications: Systems, Technologies and Applications*. Telenor Corporate University / Norwegian University of Science and Technology (NTNU), 2nd. edition, 2002.

[7] Tønnes Brekne. *Encrypted Computation*. PhD thesis, Norwegian University of Science and Technology, July 2001.

[8] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication, from proceedings of the royal society, volume 426, number 1871, 1989. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*. 1996.

[9] Butler W. Lampson. A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[10] I. Cervesato. The dolev-yao intruder is the most powerful attacker, 2001. citeseer.ist.psu.edu/cervesato01dolevyao.html.

[11] Iliano Cervesato. MSR, Access control, and the Most Powerful Attacker. citeseer.ist.psu.edu/cervesato01msr.html.

[12] Bill Cheswick. An evening with berferd, in which a hacker is lured, endured, and studied. *Proc. Winter USENIX Conference*, January 1992. http://research.lumeta.com/ches/papers/berferd.ps.

[13] H. Christensen and E. Colban (ed.). Information modelling concepts. TINA Baseline, Telecommunications Information Networking Architecture Consortium, April 1995. http://www.tinac.com/specifications/documents/imc94-public.pdf.

[14] Bram Cohen. The Official BitTorrent Home Page. http://bitconjurer.org/BitTorrent/, June 2004.

[15] Telecommunications Information Networking Architecture Consortium. TINA-C Homepage. http://www.tinac.com/, June 2004.

[16] T. Dierks and C. Allen. The TLS Protocol - Version 1.0. http://www.ietf.org/rfc/rfc2246.txt, January 1999.

[17] Danny Dolev and Andrew C. Yaou. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[18] Ian Foster (ed.) and Karl Kesselman (ed.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers/Elseviere Inc., 2nd. edition, 2004.

[19] Martin Chapman (ed.) and Stefano Montesi. Overall Concepts and Principles of TINA. Tina baseline, Telecommunications Information Networking Architecture Consortium, February 1995. http://www.tinac.com/specifications/documents/overall.pdf.

[20] Ross Callon (ed.). The Twelve Networking Truths. RFC 1925, April 1996.

[21] Donald O'Mahony et al. *Electronic Payment Systems for E-Commerce*. Computer Security Series. Artech House, Inc., 2nd edition, 2001.

[22] F. Javier Thayer et al. Mixed strand spaces. In *12th IEEE Computer Security Foundations Workshop*, volume 27(2), pages 72–82. IEEE Computer Society Press, June 1999.

[23] John Gordon et al. Wp05: Mass storage management. Technical Report DataGrid-05-D5.2-0141-3-4, CERN, SARA, PPARC, February 2002.

[24] Jose A. Onieva et. al. Agent-mediated non-repudiation protocols. *Electronic Commerce Research and Applications*, 2004. Article in press.

[25] Joshua D. Guttman et al. Strand Spaces: Why is a Security Protocol Correct? 1998 IEEE Symposium on Security and Privacy, May 1998.

[26] Peter Ryan et al. *The Modelling and Analysis of Security Protocols: The CSP Approach*. Addison-Wesley, 2001.

[27] Robert C. Logsdon et al. Content Based Information Security (CBIS). White paper / NATO - Allied Command Transformation, 2003.

[28] Willis H. Ware et al. Security Controls for Computer Systems. Technical report, Defence Science Board Task Force on Computer Security, February 1970.

[29] Fran Berman et al. (eds.). *Grid Computing: Making the Global Infrastructure a Reality*. Wiley Series on Paralell and Distributed Computing. John Wiley & Sons, Ltd., 2003.

[30] Niels Ferguson and Bruce Schneier. A Cryptographic Evaluation of IPsec. http://www.schneier.com/paper-ipsec.html, 1999.

[31] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. Wiley Publishing, Inc., 2003.

[32] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The digital distributed system security architecture. In *Proc. 12th NIST-NCSC National Computer Security Conference*, pages 305–319, 1989.

[33] Dieter Gollman. *Computer Security*. John Wiley & Sons Ltd, 1999.

[34] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning About Belief in Cryptographic Protocols. In *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, 1990.

[35] Nicolás González-Deleito and Olivier Markowitch. An Optimistic Multiparty Fair Exchange Protocol with Reduced Trust Requirements. In *Proceedings of the 4th International Conference on Information Security and Cryptology*, volume 2288 of *Lecture Notes in Computer Science*, pages 258–267. Springer-Verlag, December 2001.

[36] Nicolás González-Deleito and Olivier Markowitch. Exclusion-Freeness in Multi-party Exchange Protocols. In *Proceedings of the 5th Information Security Conference*, volume 2433 of *Lecture Notes in Computer Science*, pages 200–209. Springer-Verlag, September 2002.

[37] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985. http://www.usingcsp.com/.

[38] IETF. IP Security Protocol (ipsec) Charter. http://www.ietf.org/html.charters/ipsec-charter.html, June 2004.

[39] IETF. Transport Layer Security (tls) Charter. http://www.ietf.org/html.charters/tls-charter.html, June 2004.

[40] David Kahn. *The Codebreakers - The Story of Secret Writing*. Scribner, 2nd. edition, 1996.

[41] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. http://www.ietf.org/rfc/rfc2401.txt, November 1998.

[42] Steve Kremer and Olivier Markowitch. A multi-party non-repudiation protocol. pages 271–280. IFIP World Computer Congress, Kluwer Academic Publishers, August 2000. 15th International Conference on Information Security (SEC 2000).

[43] Steve Kremer and Olivier Markowitch. Optimistic non-repudiable information exchange. *Lecture Notes in Computer Sciences*, 2015:109–122, December 2000. 3rd International Conference on Information Security and Cryptology (ICISC 2000).

[44] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.

[45] Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

[46] Gavin Lowe. Casper: A Compiler for the Analysis of Security Protocols. http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Casper/, June 2004.

[47] Formal Systems (Europe) Ltd. Formal Systems (Europe) ltd. http://www.fsel.com/, June 2004.

[48] Matt Bishop. *Computer Security : art and science*. Pearson Education, Inc., 2003.

[49] Catherine Meadows. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. http://citeseer.ist.psu.edu/586062.html.

[50] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[51] Peter G. Neumann (moderator). The risks digest. http://www.risks.org/.

[52] Napster, LLC. Napster.com. http://www.napster.com/, June 2004.

[53] United Nations. The Universal Declaration of Human Rights. http://www.unhchr.ch/udhr/lang/eng.htm, December 1948.

[54] R. M. Needham and M. Schroeder. Using encryption for authentication in large computer networks. *Communications of the ACM*, 21(12):993–999, 1978.

[55] Peter G. Neumann. Homepage. http://www.csl.sri.com/users/neumann/neumann.html.

[56] Peter G. Neumann. Practical Architectures for Survivable Systems and Networks. http://www.csl.sri.com/neumann/survivability.html, June 2000.

[57] University of Liverpool and University of Southampton. Agentlink.org - European Coordination Action for Agent-Based Computing. http://www.agentlink.org/, June 2004.

[58] OSMB, LLC. Gnutella.com, June 2004. http://www.gnutella.com/.

[59] Charles Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. Pearson Education, Inc., 3rd. edition, 2003.

[60] Plato. *The Republic*. Translated by Benjamin Jowett, http://www.constitution.org/pla/republic.htm (June 2004), Circa 360 BC.

[61] Ross Anderson. *Security Engineering*. John Wiley & Sons, Inc., 2001.

[62] Ross Anderson. Homepage. http://www.cl.cam.ac.uk/users/rja14/, June 2004.

[63] Elton Saul. SPEAR II Project Summary. http://www.cs.uct.ac.za/Research/DNA/SPEAR2/, June 2004.

[64] Bruce Schneier. Crypto-Gram Newsletter (April 15, 2004). http://www.schneier.com/crypto-gram-0404.html.

[65] Bruce Schneier. Crypto-Gram Newsletter (December 15, 2001). http://www.schneier.com/crypto-gram-0112.html.

[66] Bruce Schneier. Crypto-Gram Newsletter (February 15, 2004). http://www.schneier.com/crypto-gram-0402.html.

[67] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., 2nd edition, 1996.

[68] Larry Smarr and Charles E. Catlett. Metacomputing. *Communications of the ACM*, 36(6):44–52, 1992.

[69] Olivier Markowitch Steve Kremer and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25:1606–1621, 2002.

[70] Douglas R. Stinson. *Cryptography : theory and practice.* Chapmann Hall/CRC, 2nd edition, 2002.

[71] Inc. Sun Microsystems. jxta.org. http://www.jxta.org/, June 2004.

[72] Eun Kyo Park Sungwoo Tak, Yugyung Lee. A software framework for non-repudation services in electronic commerce based on the internet. *Microprocessors and Microsystems*, 27:265–276, 2003.

[73] P. Syverson and P. van Oorschot. On unifying some cryptographic protocol logics. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 14–28, 1994.

[74] Paul Syverson, Catherine Meadows, and Iliano Cervesato. Dolev-Yao is no better than Machiavelli.

[75] Doug Szajda, Barry Lawson, and Jason Owen. Hardening Functions for Large Scale Distributed Computations. In *IEEE Symposium on Security and Privacy*. Institute of Electrical and Electronics Engineers, Inc., May 2003.

[76] European Union. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In *Official Journal L281*, pages 31–50. November 1995.

[77] Unknown. Mixmonion: A Type III Anonymous Remailer. http://www.mixminion.net/, June 2004.

[78] David Wagner and Bruce Schneier. Analysis of the ssl 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 29–40, http://www.schneier.com/paper-ssl.html, November 1996. USENIX Press. Revised April 15, 1997.

[79] Website. Condor Project Homepage. http://www.cs.wisc.edu/condor/, June 2004.

[80] Website. The datagrid project. http://www.eu-datagrid.org/, June 2004.

[81] Website. Global Grid Forum, June 2004. http://www.ggf.org/.

[82] Website. I2P, June 2004.

[83] Website. Legion: A Worldwide Virtual Computer. http://legion.virginia.edu/, June 2004.

[84] Website. Mixmaster. http://mixmaster.sourceforge.net/, June 2004.

[85] Website. The Globus Alliance, June 2004. http://www.globus.org/.

[86] Website. The Honeynet Project. http://www.honeynet.org/, June 2004.

[87] Jianying Zhou. *Non-repudiation in Electronic Commerce*. Artech House computer security series. Artech House, Inc., 2001.

# Index