*Technical Report*

UCAM-CL-TR-489
ISSN 1476-2986

Number 489

# Designing a reliable publishing framework

Jong-Hyeon Lee

April 2000

This technical report is based on a dissertation submitted January 2000 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Wolfson College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

*http://www.cl.cam.ac.uk/techreports/*

ISSN 1476-2986

# Summary

Due to the growth of the Internet and the widespread adoption of easy-to-use web browsers, the web provides a new environment for conventional as well as new businesses. Publishing on the web is a fundamental and important means of supporting various activities on the Internet such as commercial transactions, personal home page publishing, medical information distribution, public key certification and academic scholarly publishing. Along with the dramatic growth of the web, the number of reported frauds is increasing sharply. Since the Internet was not originally designed for web publishing, it has some weaknesses that undermine its reliability.

How can we rely on web publishing? In order to resolve this question, we need to examine what makes people confident when reading conventional publications printed on paper, to investigate what attacks can erode confidence in web publishing, and to understand the nature of publishing in general.

In this dissertation, we examine security properties and policy models, and their applicability to publishing. We then investigate the nature of publishing so that we can extract its technical requirements. To help us understand the practical mechanisms which might satisfy these requirements, some applications of electronic publishing are discussed and some example mechanisms are presented.

We conclude that guaranteed integrity, verifiable authenticity and persistent availability of publications are required to make web publishing more reliable. Hence we design a framework that can support these properties. To analyse the framework, we define a security policy for web publishing that focuses on the guaranteed integrity and authenticity of web publications, and then describe some technical primitives that enable us to achieve our requirements. Finally, the Jikzi publishing system—an implementation of our framework—is presented with descriptions of its architecture and possible applications.

# Declarations

This dissertation does not exceed the limit of sixty thousand words prescribed by the Computer Laboratory Degree Committee.

Except where otherwise stated in the text, this dissertation is the result of my own work and is not the outcome of work done in collaboration.

It is not substantially the same as any I have submitted for a degree, diploma or any other qualification at any other university, and no part of it has been, or is currently being, submitted for any such qualification.

Note 1. All products and company names mentioned in this dissertation may be the trademarks of their respective owners.
2. Web links which appear in footnotes and the bibliography of this dissertation are correct at the time of going to press. (Printed on 7 January 2000)

# Acknowledgements

First, I am deeply indebted to Dr Ross J. Anderson, my supervisor, for his advice, sincere support, helpful comments and for enabling me to have financial support under the grant 'Resilient Security Mechanisms' from the Engineering and Physical Sciences Research Council for which I am very grateful. Without his support, this dissertation would have never been finished.

Some of the ideas presented here were clarified during meetings of the security research group in the Computer Laboratory. I am grateful to Prof. Bruce Christianson, Prof. Dieter Gollmann, Prof. E. Stewart Lee, Prof. Roger M. Needham and Prof. David J. Wheeler for their insightful comments and discussions in the meetings.

Many thanks to Dr Bruno Crispo, Charalampos Manifavas, Dr Václav Matyáš Jr and Dr Fabien A. P. Petitcolas for the enjoyable collaboration in 'the Global Trust Register', to Prof. Francesco Bergadano for his collaboration in 'the Guy Fawkes protocol', to Tutor Patricia Hyndman at Wolfson College, Cambridge for her kind support and to Prof. Sungpyo Hong and Prof. Youngju Choie at POSTECH for their advice and encouragement.

The helpful support of the system administrators and the secretaries of the Computer Laboratory were appreciated. I should also thank all the persons who make my stay in Cambridge much more meaningful and enjoyable.

Special thanks to Che-Hao Albert Chang, Shaw C. Chuang, Abida Khattak, Markus G. Kuhn, Ulrich Lang, Susan Pancho, Dr Michael R. Roe, Francesco M. Stajano, Kan Zhang, Shireen Anderson and Graeme Fairhurst. Especially, I thank Dr Geraint Price for his careful proofreading and helpful discussions.

Finally my deepest heartfelt thanks to my parents, grandmother and wife, to whom I dedicate this dissertation, and warmest thanks to my daughters with love. Without their sincere support I could have never come this far.

# Contents

# List of figures

# List of tables

# Glossary

This list defines some technical terms used in this dissertation.

| | |
|---|---|
| **anonymity** | the state of one's identity being or remaining unknown to most others |
| **authenticity** | the quality of being known to be genuine |
| **authentication** | the action of demonstrating a proof of genuineness |
| **availability** | the capability of being made use of |
| **confidentiality** | the condition of being kept secret |
| **digital signature** | a value that enables to identify the originator and the content integrity of a digital object |
| **hash function** | a function that produces outputs of a fixed length for inputs of arbitrary length |
| **integrity** | the condition of not having been modified |
| **object** | a passive entity used by subjects |
| **one-way function** | a function that is easy to compute but whose inverse is computationally infeasible |
| **one-way hash function** | a hash function that is one-way |
| **principal** | a subject who uses a system or service |
| **publication** | an object that has been published |
| **publicity** | the state of being known to the public |
| **publishing** | the action of making publicly known |
| **reliability** | the quality of being able to have confidence |
| **resilience** | the ability of a system to recover quickly from a fault |
| **security model** | abstract description of system behaviour |
| **security policy** | a set of rules governing how principals can access the system |
| **subject** | an active entity that can initiate requests for resources and use them |

# Chapter 1

# Introduction

## 1.1 The thesis

The thesis of this research is that web publishing can be more reliable when its integrity, authenticity and availability are improved. In order to prove our thesis, we construct a framework that improves these properties of web publishing and present an implementation of the framework.

Note that throughout the dissertation, we use the term 'reliable' publishing in the sense that people can have confidence in using publications. In this dissertation, we will examine what makes people confident in using conventional publications, investigate what attacks undermine confidence in web publishing, and understand the nature of publishing including both conventional and web publishing. As a result, we will propose a framework and implementation that help us improve the reliability of web publishing.

## 1.2 Motivation

How can we rely on web publishing? The web is now a common environment for network services and traditional services are migrating onto the web; the web is becoming a part of our life and its territory keeps growing. Simultaneously we face a number of fraud cases on the web, and we presume that the number of frauds is going to increase sharply along with the rapid expansion of the Internet. Furthermore, the web is not ready to convey the level of reliability provided by conventional publishing media, since web publishing is at quite an early stage.

The web is a valuable tool for publishing on the Internet, but carries little guarantee of the publication's reliability. Since the original purpose of the Internet was neither commercial transactions nor personal publishing, the problems of web publishing and its reliability were not considered seriously.

When the project 'World Wide Web' started at CERN[1] in 1990, some naïve HTTP[2] browsers were developed but not used worldwide. With a click-and-connect graphical user interface, Mosaic for X Windows, developed by Andreessen in 1993, became a trigger to wide use of the Internet. The adoption of the simple but strong markup language HTML[3] is another success factor of the web. In March 1993, HTTP traffic measured 0.1% of NSF[4] backbone traffic and became 1% of it in six months;[5] now it dominates Internet traffic.

The development of web technologies means that people can publish their ideas widely, easily, quickly and cheaply. This benefit attracts more people and many businesses online. The Internet is no longer a place for academics and is becoming a part of general public's life. Commercial services migrating to the web include banking, shopping and entertainment, but on the other hand, we face many frauds attacking weaknesses of web publishing.

In the 1997 Annual Report[6] of the US Securities and Exchange Commission (SEC), three Internet publishing fraud complaints are filed; these cases are about publishing investment newsletters on the web which distribute false

[1]The European Laboratory for Particle Physics

[2]HyperText Transfer Protocol, the base protocol for the web; see [42].

[3]HyperText Markup Language, the base language for the web; see [89].

[4]The National Science Foundation; the NSF backbone is a wide area network which mainly connects academic organisations.

[5]`<http://www.w3.org/History.html>`

[6]`<http://gopher.sec.gov/asec/annrep97/annrep97.htm>`

and misleading information to subscribers. The number of such frauds has been increasing. On 28 October 1998, the SEC announced the filing of 23 enforcement actions against 44 individuals and companies across the USA for frauds over the Internet and deceiving investors.[7] Most of the alleged frauds were about the distribution of false information and they have undermined confidence in Internet publishing. The SEC regards these frauds as a serious threat[8] and is now running a central database that provides certified information. However, there is no unique information source to satisfy the demand of stock investors. Providing a means to verify the authenticity of a newsletter is more important than proving a central database controlled by the authority.

In 1997, Mentor Network, a California-based firm, opened a web site to collect money with the name of a children's charity and set up a classical pyramid marketing scheme[9]— the victim invests money, then recruits others so that a stream of cash flows back to older investors from new ones. The US Federal Trade Commission reacted when it found the online charity scam had misused investment out of a million dollars. Mentor was not alone and such a fraud makes people unwilling to rely on any charity asking help over the Internet. On the Internet, we are not confident about to whom we are talking, the best we can do is just assume we are talking with the person to whom we want to talk.

In 1998, a California man, Bowin, was sentenced to ten years in prison for conducting a fake stock offering over the Internet.[10] It is perhaps the harshest jail sentence for Internet securities fraud in history. He offered to sell shares of a technology company over the Internet from late 1996 to early 1997. He advertised his company with false information and about 150 people fell for it.

We have seen some publishing fraud cases on the Internet but they are the tip of the iceberg. We need to consider what the weaknesses of web publishing are, i.e., what makes frauds easier on the web than in real world.

[7]`<http://techlawjournal.com/seclaw/81028secpr.htm>`

[8]`<http://www.sec.gov/consumer/cyberfr.htm>`

[9]`<http://www.zdnet.com/yil/content/mag/9708/pappas9708.html>`

[10]Wall Street Journal, 9 November 1998

The lack of evidential force is one worry. Although there is a huge amount of information on the web and people regard it useful, people are reluctant to accept it as legal evidence. By its very nature, web publishing is a dynamic process and nothing is guaranteed. Can we find a way to provide evidential force for web publishing? This is one of the problems tackled by this research.

On the other hand, there are problems caused by misbehaviour of users without bad intentions; for example, it is common to find links on web pages that have wrong or out-dated references, so called 'link rot', since people keep changing their web sites without notice and they also change address frequently. Nowadays web links are cited in news articles and even in scientific papers but there is no guarantee that the cited links will last long. The web links used in footnotes in this dissertation may not be exceptions.

How do we obtain reliable web publishing? We believe that reliability on the web cannot be simply imposed by some existing authority such as the government or banks. Reliability can be established by the accumulation of empirical successes from trials, and empirical success is achieved by a plausible system design. System designers should clarify what attacks the reliability of a system and which properties are required to make it reliable. We will investigate threats and weaknesses of web publishing and extract requirements for reliable publishing on the web. Then we will design a mechanism that can perform web publishing successfully under these requirements.

## 1.3 Previous work

Web publishing has not been intensively studied from the security point of view and we cannot find many related works, but some work on electronic publishing and publishing frameworks inspired us.

Apart from security concerns, electronic publishing has been considered and developed in the context of academic scholarly journal publishing by Ginsparg's preprint server [46] and Loughborough University's ELVYN [98]. Snook's DODA [102] presents a document management scheme in view of systems security.

Ginsparg's preprint server, the Los Alamos e-print archives, is a repository for the circulation of academic paper preprints mainly in physics; preprints are a form of pre-publishing before refereeing for journal publication. This speeds up discussion of preprints. Mechanically, the server accepts all preprint submissions, stores them and makes them available on the web. In Science, Taubes [104] pointed out that this service shows a possible model to cut the high cost of scholarly journal publishing. Its success inspired similar services in other fields, such as Southampton's Cognitive Science Eprint Archive.[11] In order to compete with major commercial journal publishers and meet academics' demands, these services are evolving and adding features, such as peer review to enhance the quality of preprints. They satisfy the basic needs of academics—fast communication and low cost publishing—but no security aspect is considered. Security of these services is maintained by academic trust not by mechanisms.

Loughborough University's ELVYN[12] is an electronic journal publishing and distribution mechanism for libraries. They are mainly interested in displaying and browsing publications since multimedia publishing could not be easily achieved when the project started in 1990. The project has evolved through three stages. In the report on the third stage, they expect that combining ELVYN with web publishing will boost its usage. It also deals with analysis of usage patterns and costs of publishing.

Usually studies of electronic scholarly publishing deal with issues of copyright, economic structure and success-failure models; they help us understand the nature of publishing but do not cover the interests of the nature and further security issues of web publishing.

Snook's Distributed Office Document Architecture (DODA) is a security architecture for document management in a distributed environment. It presents a broad range of features for document handling. In order to handle parts of a document, she defines a 'folio' as a functional object of the document which can be text, graphic material, multimedia objects, or a composite of them, i.e., a document can be represented by a composite of folios. Object-wise document

[11]`<http://cogprints.soton.ac.uk>`, also known as CogPrints
[12]ELVYN is an acronym for ELectronic Versions – whY Not.

control is the main idea of DODA and its security features are also controlled through folios. Web documents can be handled object-wise by the nature of hypertext and it is now easier to get object-level control of documents than in the time of DODA.

These previous works related to electronic publishing, document management and security set the stage for us to investigate web publishing.

## 1.4 Direction

We will examine factors that weaken the reliability of web publishing and try to understand the nature of web publishing in three directions: first, we will clarify the meaning of security properties and review security policy models; secondly, we will investigate the nature of publishing and make comparisons between conventional publishing and electronic publishing; thirdly, we will see major concerns of practical electronic publishing applications.

To reach the essence of the problem, we will ask fundamental questions: why is web publishing a problem?; in which respects is it different from conventional publishing on paper?; how can we rely on web publishing?; eventually, how can we build conventional publishing-level reliability in web publishing?

We will then construct a publishing framework that can improve reliability. Our framework consists of a publishing policy and some technical primitives that enable us to realise our policy. In order to verify our framework, we will present an implementation of the framework.

The structure of this dissertation is shown in Figure 1.1. We present the thesis of the dissertation in Section 1.1. In order to understand the thesis, we investigate related subjects to web publishing in three directions in next three chapters. We present our answer to the thesis in Chapter 5, and provide an implementation in Chapter 6. Conclusions of the thesis and each chapter are given in the last chapter.

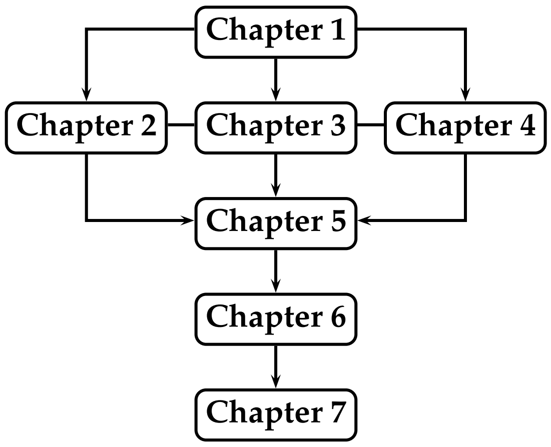


**Figure 1.1. Overview of the structure of the dissertation:** Chapter 1 describes the thesis of the dissertation, motivation and some background studies, and Chapter 2, 3 and 4 run almost parallel to reach our hypothesis, by examining fundamental security properties, the nature of publishing and publishing applications, respectively. Chapter 5 sets the thesis up and proposes our solution, and Chapter 6 presents an implementation of our proposal to show our proposal is valid and the thesis is achieved. Chapter 7 provides conclusions of the thesis and each chapter.

## 1.5 Synopsis

The rest of this dissertation is arranged as follows:

- Chapter 2 examines definitions of security properties to clarify the nature of the properties, reviews major security policy models and discusses the relationship between the properties and the security policies.
- Chapter 3 investigates the nature of publishing and the publishing process by investigating conventional and electronic publishing mechanisms. It extracts publishing requirements from the investigation and examines the relationship between security and publishing.
- Chapter 4 presents a series of web publishing applications and some example publishing mechanisms which we have proposed in conferences

and journals; the mechanisms include electronic payment, voting, digital signature and key certification services.

- Chapter 5 presents a framework for web publishing which consists of a publishing policy and some technical primitives; the policy requires integrity and authenticity of web publications; the primitives support the policy and availability of publications.

- Chapter 6 presents an implementation of the framework proposed in Chapter 5 and discusses design and implementation issues; application-level services of the mechanism are also described.

- Chapter 7 presents the conclusion of the thesis and conclusions of other chapters.

# Chapter 2

# Security properties and policies

In order to clarify the requirements of security systems, we examine security properties and policy models relevant to publishing. This helps us understand the relationship between security properties and security policies, and shows what we can achieve when we support a certain security property. For example, when a policy model supports only integrity, the definition of integrity will clarify what we can do and what cannot within the model.

First, we will deal with three major security properties: integrity, authenticity and confidentiality. From their dictionary definition to others' interpretation and practical implementation, we investigate their properties, which are fundamental in systems requiring security and a publishing system is no exception.

We then investigate three concepts which are desirable properties of publishing systems: publicity, anonymity and availability. We focus on these properties as seen from a publishing perspective and subsequently there may be discrepancies between our concepts and the conventional understanding of them. Unlike other properties listed here, availability is a system issue and we examine methods to provide it.

This investigation will help us understand what the concerns of a publishing system are and how such systems can be made more reliable.

## 2.1 Integrity

Integrity[1] is the condition of not having been modified. It assumes timeliness; when we mention integrity, it implies the uncompromised condition of an object in a certain period. When any object in a system is not modified in a certain period, we say the system provides the quality 'integrity' of the object. The integrity of an object is independent of the subject who created it. Even though the creator of an object modifies it, its integrity is lost. Whether the creator can change it or not is not an integrity issue but an authorisation issue. Integrity is a fundamental property of the object itself.

Mayfield et al. [73] attribute integrity to two terms: data and systems. Data integrity is concerned with preserving the meaning of information, with preserving the completeness and consistency of its representations within the systems, and with its correspondence to its representations external to the system. Systems integrity is defined as the successful and correct operation of computing resources; in their definition, systems integrity is related more closely to high availability, fault-tolerance and robustness rather than data and information processing. In this dissertation, we do not use the terminology 'integrity' in its systems definition.

In their seminal paper [32], Clark and Wilson define integrity in a practical way: no user of the system, even if authorised, may be permitted to modify data items in such a way that assets or accounting records of the company are lost or corrupted, thus compromising their integrity.

Schneier [101, p. 2] describes integrity intuitively in the context of message exchange as follows: it should be possible for the receiver of a message to verify that it has not been modified in transit; an intruder should not be able to substitute a false message for a legitimate one. This description applies to integrity-preserving communications but is not enough for us since integrity is independent of principals.

[1]**integrity** 1. the condition of having no part or element taken away or wanting; undivided or unbroken state; material wholeness, completeness, entirety 2. the condition of not being marred or violated; unimpaired or uncorrupt condition; original perfect state; soundness [105]

Gollmann [48, p. 5] defines integrity as the prevention of unauthorised modification of information. Unlike other definitions above, his definition assumes an authorisation in the context of integrity. Similarly, the International Telecommunication Union defined data integrity as the property that data has not been altered or destroyed in an unauthorized manner in the Recommendation X.800 [56], but dropped the requirement for authorisation in the Recommendation T.411 [57], which is similar to our understanding.

To keep the integrity of an object, we can think about two approaches: protecting an object from tampering and detecting the tampering. The former is active protection against tampering and the latter passive. The study of tamper resistance focuses on the protection of an object from tampering and the tamper evidence approach on the detection of tampering.

Initially, tamper resistance was studied for military purposes and tamper resistant devices were designed to be destroyed when tampered; at the most attackers can break the device but cannot obtain the secret inside.

Tamper evidence is focused on the detection of changes. To see changes in the object, it is clear that we need a mechanism to compare the current condition and the original condition. Some techniques have been developed to make the comparison such as cryptographic hashing, fingerprinting [71] and digital watermarking [63].

## 2.2 Authenticity

Authenticity[2] is the quality of being known to be genuine. In other words, it is a quality that can be verified to be original. That we say the authenticity of something assumes that there is a way to prove that it is genuine. The action of demonstrating this proof is called authentication.

[2]**authenticity** 1. as being authoritative or duly authorised. 3. as being what it professes in origin or authorship; as being genuine; genuineness
**genuine** 3. really proceeding from its reputed source or author; not spurious 4. having the character or origin represented; real, true, not counterfeit, unfeigned [105]

From the definition, we can think about genuineness in two types : genuineness of an object and that of the author of the object. We call the authenticity of them object authenticity and subject authenticity, respectively. Object authenticity includes the integrity of an object; authentication of an object includes verification of both its authorship and whether it has been modified or not. Message authentication is an example of verifying object authenticity.

Unlike object authenticity, subject authenticity is defined by physical and logical characteristics of a subject. Authentication of a subject is the process used to identify the subject using its characteristics, including height, fingerprints, iris patterns, names, addresses, affiliation information, cryptographic keys, email addresses and some identification numbers. Because such metrics of authentication are predominantly used as a means of facilitating access control, when authentication and authorisation are used in actual implementations, their scope often overlaps considerably. Whenever we use an identifier to represent a subject such as public key or email address, a gap between the subject and its identifier is introduced. This gap represents the precision of the authentication scheme; it is believed that logical identifiers are less tightly bound with the subject than physical identifiers.

The International Telecommunication Union defines standard recommendations about various telecommunication applications and their interoperability. Their major concern is communication and definitions are mainly derived from practical applications. Recommendation T.411 [57] defines authenticity by the property that the claimed data source can be verified to the satisfaction of the recipient; the satisfaction may depend on applications. Authentication is defined by the provision of assurance of the claimed identity of an entity in Recommendation X.811 [58]. They also define data origin authentication by the corroboration that the source of data received is as claimed in Recommendation X.800 [56].

Gollmann [48] defined subject authentication as the process of verifying a claimed identity. Schneier [101, p. 2] roughly explains authentication as follows: it should be possible for the receiver of a message to ascertain its origin, while an intruder should not be able to masquerade as someone else.

## 2.3 Confidentiality

Confidentiality[3] is the condition of being kept secret. The statement "an object is secret" means that an object is known to only a certain number of people specified. So confidentiality assumes a partition of principals: one group of people knows the object and the other group of people does not. We observe that the movement between the two groups is always one-way; the only thing that passes is knowledge, not ignorance. When there is an unauthorised communication, the confidentiality of the object is broken.

Authorisation procedures for access to an object are important to maintain its confidentiality. A method to achieve confidentiality is to restrict access by using a conventional security policy, and another method is to control keys for cryptographic algorithms. The combination of both methods is also common.

In the world of conventional publishing, information is written on paper and its confidentiality largely depends on physical access control to its repository, such as a safe or vault. For highly confidential paper documents, mechanical or hand-written ciphers had been used to make it difficult to read them. In electronic publishing, the superficial situation is not so different: access control is still important and ciphers are used for higher secrecy. However, there are some changes: the use of ciphers is much easier than before, the knowledge about cryptography becomes more public and as a result, there is a wide choice of high quality ciphers[4] publicly available on the Internet. The general public can easily access ciphers to encrypt their documents. There are also cryptographic tools widely used on the network such as SSL [43], SSH [110] and PGP [111]; the first two provide encrypted channels during communi-

[3]**confidentiality** 1. confidential quality; state of being confidential
**confidential** 2. of the nature of confidence; spoken or written in confidence; characterised by the communication of secrets or private matters – *confidential communication*: a communication made between parties who stand in a confidential relation to each other, and therefore privileged in law [105]

[4]For example, the Advanced Encryption Standard (AES) candidates; AES will be the replacement of DES [79]. The National Institute of Standards and Technology in the USA has been organising a contest to select the replacement and announced five finalists including MARS, RC6, Rijndael, Serpent and Twofish.

cation on the network and the last provides various cryptographic functions such as key generation, encryption and digital signature. Cryptography is becoming common.

Though we agree with Gollmann's argument [48, p. 203] that cryptography just translates a communication confidentiality issue to a key management issue, it is an important building block to support confidentiality of electronic data. He describes confidentiality as capturing the aspect of computer security that unauthorised users should not be learning sensitive information [*ibid.*, p. 6]. Pfleeger [88] characterised confidentiality as follows: only authorised people can see protected data. Recommendation T.411 [57] of the International Telecommunication Union defines it by the property that information is not made available or disclosed to unauthorized individuals, entities or processes. We found that the definition of confidentiality is convergent.

Although confidentiality is an important issue in computer security and many studies have been made so far, we believe that confidentiality is not an essential property of publishing, and hence we do not investigate it further. Unlike other properties, confidentiality will be only partially discussed later.

## 2.4 Publicity

Publicity[5] is the state of being known to the public. It has a complementary aspect to confidentiality in terms of knowledge transfer; publicity does not limit knowledge transfer but confidentiality does. Publishing is the action of realising publicity. Note that since publicity is not usually considered in a security context, there is no agreed terminology.

Let us consider the conventional publishing process. An author has an idea and writes a draft of the idea; if his writing is good enough and he is lucky,

[5]**publicity** the quality of being public; or the condition or fact of being open to public observation or knowledge
**public** 1. of pertaining to the people as a whole. 4. that is open to, may be used by, or may or must be shared by, all members of the community; generally accessible or available. 5. open to general observation, sight, or cognizance [105]

the draft is accepted by a publisher to print in a tangible form, say a book. Then the book is delivered to book shops through the publisher's distribution network. Here, we see another aspect of publishing which will change in the computer network era.

Firstly, publishing conventionally implies the distribution of frozen copies of an idea; at the time of publishing, the fact that the author said what was printed in the publication becomes frozen and can never be changed. Although there is a means by which the author can change his mind later in the form of errata or revisions, a clear rule is that previously printed and distributed copies are not changed; they become a part of the history of his idea and evidence of what he said. Published matter is accumulated not destroyed.

Secondly, computer networks widely adopted throughout the world change the whole structure of the publishing process. Neither selection by a publisher nor the distribution network of the publisher is necessary. If one has an idea, one can publish it through the network to the world at the speed of light. This changes the nature of publishing, but we can hardly call the published matter 'frozen' since we can change it at any time; nobody keeps a history. We have obtained a low-cost method of publishing, but coincidentally we lost the immutability of published matter.

We pointed out a couple of fundamental problems caused by the media change from paper to the computer network. In Chapter 3, the nature of publishing will be investigated and discussed, and a more extensive comparison between paper and web publishing will be given.

## 2.5 Anonymity

Anonymity[6] is the state of an object's identity – or even its existence – being or remaining unknown to most others. If the object's existence is not known

[6]**anonymity** the state of being anonymous
**anonymous** 1. nameless, having no name; of unknown name 2. bearing no author's name; of unknown or unavowed authorship [105]

to people, anonymity may look similar to confidentiality, but it is clear that the mechanism is different. Confidentiality of an object is usually maintained using shared secrets to access the object, and the sharing methods define the group of principals who know the secret and can access the object. Anonymity of an object is not defined by the same means; anonymity is usually kept by blocking knowledge transfer and the blocking methods define the group of principals who do not know the object. Consider a message transfer. If we can block all metadata transfer, except message transfer itself between principals involved, we can send a message anonymously. To avoid the leakage of metadata, such as the identity of the sender, recipients, sending time and intermediate relaying principals, we usually assume some trust relationship between principals.

Another aspect of anonymity that distinguishes it from confidentiality is its goal. Assume that there is a leaflet on a table. In terms of confidentiality, the content of the leaflet is open, but from an anonymity point of view, we might not know the author of the leaflet or the person who put it on the table. The goal of confidentiality can be principals or data used by principals; i.e., we want to keep principals themselves or data itself secret, but that of anonymity is mainly the trace of data or the relationship between a principal and his data, i.e., metadata.

Anonymity is applicable for principals and metadata. A principal includes a person or process which carries out an action, and metadata include a deed, event, timestamp and involved principals. Principal anonymity usually means source anonymity, i.e., the source of an action is not known to its destination. Anonymous mail is an example of principal anonymity.

Protecting metadata depends on the group of people who know it. Suppose a person in a city under siege wants to inform people outside of the current situation, and sends a message through an anonymiser on the Internet. Partial metadata can remain in the intermediate message relaying servers. Some servers may collude to trace the message source or another party may collect traffic data between them. This constructs a data transfer chain, and we have a way to trace back to the origin of the transferred knowledge along with the chain; anonymity is hard to control.

Although metadata anonymity plays an important role in electronic commerce, anonymity has been commonly regarded in terms of principal anonymity rather than metadata anonymity.

## 2.6 Availability

Availability[7] is the capability of being made use of. In computer terms, 'available' means a service should be provided whenever requested; depending on system requirements, it may converge to the nonstop service. Since it is always possible to face arbitrary faults, it is natural to assume faults and system stops. In this respect, availability is closely related to fault-tolerance, resilience, or high speed recovery.

International Standard ISO 7498-2 [55] gives a definition of availability as the property of being accessible and usable upon demand by an authorised entity. The Canadian Trusted Computer Product Evaluation Criteria [26] gives a similar definition: availability is the property that a product's services are accessible when needed and without undue delay. Recommendation G.972 [59] of the International Telecommunication Union defines it by the ability of the system to be in a state to perform adequately at a given instant of time within a given time interval.

For common Internet services, it is necessary to provide the service continuously without interruption since customers are worldwide and the service is used 24 hours a day. Previously, high availability, nonstop service or fault-tolerance had been asked of only a few critical systems such as military and aerospace systems. The Internet makes us consider this issue within the commercial sector.

Recently value-added services have been implemented based on the Internet, such as electronic payment, intelligent databases and network-based value-

[7]**availability** the quality being available; capability of being employed or made use of **available** 1. capable of producing a desired result; of avail, effectual, efficacious 2. of advantage; serviceable, beneficial, profitable 3. capable of being employed with advantage or tuned to account; capable of being made use of, at one's disposal, within one's reach [105]

added telecommunication services. It is almost infeasible to keep a system from failing in a networked service environment; we should assume that systems can often fail. Even though computer systems which provide a virtually nonstop service are available on the market, the cost will not be affordable for average Internet service providers. Building a system in which recovery can happen quickly, i.e., which is highly resilient, is more practical and efficient for average commercial use, especially web publishing.

We survey two types of mechanisms to obtain such resilience: process-group mechanisms and application-layer mechanisms. The former is a way to obtain resilience via a group of processes[8] executing redundant functions. The latter has each principal performing separate functions and playing a different role in the whole service.

Two prototypes of process-group resilience mechanisms are reported: Proactive Security and Rampart. The former was studied by a group of researchers [45, 53, 54] at the IBM Research Center. Their main ideas are two-fold: periodic refreshment of secret data and distribution of the secret data. The latter has been mainly studied by Reiter and his colleagues [90, 91, 92] at the AT&T Labs–Research. Redundancy to mask corrupt servers and build high-integrity services is the main concept in their prototype system, Rampart. Brief descriptions of both prototypes appear in Section 2.10.

Conventionally, application-layer mechanisms have been used in secure systems under a distributed computing environment. We find trials and implementations of application-layer resilience mechanisms in fault tolerant system design and distributed system design. For such mechanisms, each module in the system plays a different role and no heavy redundancy is assumed. These mechanisms share the idea 'separation of duty'.

An example of separation of duty in security applications is Crispo and Lomas's certification authority [36] that the function of the conventional certification authority is split into two subfunctions: revocation authority and certification authority. This sort of mechanisms can be reinforced by technical measures against local failures.

[8]A *process* is a principal that participates in group operations

## 2.7 Security policy models

The security policy in a system defines the conditions under which subject accesses are mediated by the system reference monitor [3, p. 91]; it is a statement of the security we expect the system to enforce [88, p. 271]; it provides the foundation of a security infrastructure and provides important guidance to assist with the increasing interconnection among organisations [25, p. 63]. It is usually represented as a set of principles or rules governing how to access the system and how to operate it.

A security model is an abstract description of system behaviour. A security policy describes system-specific dependencies; a security model is more abstract than a security policy. If it is intended as a general guide for many different types of computing applications and environments, then we refer to it as a model. If it is a specific description of required computing behaviour, then we refer to it as a policy.

In the description of security models, a subject is defined as an active computer system entity that can initiate requests for resources and use the resources to complete some computing task. In an operating system, subjects are typically processes or tasks. An object is defined as a passive computer system repository that is used to store information. In an operating system, objects are typically files and directories.

We present a survey of security models and their corresponding policies.

### Bell-LaPadula model

In 1973, Bell and LaPadula [15] devised a security model to prevent disclosure threats, especially from Trojan horse attacks, which is known as the Bell-LaPadula, or BLP model. It was proposed for military purposes and its main concern is confidentiality preservation in data access. This model has become the most influential security model and stimulated researchers to consider many variants such as Biba, System Z and Chinese Wall.

The BLP model is a multi-level security model which has the property that subjects can read down and write up, but never vice versa. The BLP model enforces two properties:

**The simple security property** subjects may not read objects at a higher level; it is also known as no read up (NRU) property.

**The *-property** subjects may not write objects to a lower level; it is also known as no write down (NWD) property.

### System Z: tranquillity

McLean turned the Bell-LaPadula model into System Z [74] by adding the property that a user can ask the security officer to temporarily declassify any object from high to low; thus the low level subject can read any high level object without breaking the BLP rules. System Z was the first serious critique of BLP.

It shows the virtue of introducing a property called tranquillity; the strong tranquillity property states that security labels never change during system operation, while the weak tranquillity property states that labels never change in such a way to violate a defined security policy.

### Chinese Wall

In 1989, Brewer and Nash [24] introduced a policy model for the commercial sector called Chinese Wall. The Chinese Wall model is based on commercial discretion and legally enforceable mandatory control. Although it is designed for commercial organisations, this model is based on the BLP model. It defines the simple security property and the *-property in a different way.

The Chinese Wall model is mainly concerned about conflicts among datasets under competing relations. For example, when a subject in an accountancy firm who accessed a dataset of an oil company $A$ would like to access a dataset of another oil company B, this is prohibited since B is A's competitor. If the subject wants to access a dataset of an advertisement agency C, the knowledge of oil company A's dataset does not matter. In this case, the Chinese Wall is created for that particular subject around the dataset in company A, and any dataset within the same conflict of interest class is regarded as being on the wrong side of this wall.

## Biba integrity model

Confidentiality and integrity are in some sense dual concepts: confidentiality is a constraint on who can read a message, while integrity is a constraint on who may have written or altered it.

In the sense of such duality, Biba [21] introduced an integrity model in a dual form of the BLP model in the mid 1970s, known as the Biba integrity model. The goal of the model is to protect corruption of high level objects from low level subjects. In terms of integrity, information may only flow downwards. The Biba model is the BLP model upside-down. We refer to these rules as the no write up (NWU) and no read down (NRD) rules: subjects may not write objects at a higher level and subjects may not read objects at a lower level, respectively.

## Clark-Wilson model

In 1987, Clark and Wilson [32] introduced an integrity model motivated by the way commercial organisations control the integrity of their paper records in a non-automated office setting. This is known as the Clark-Wilson model, or the CW model.

The CW model is expressed in terms of a finite set D (for data) that includes all the data items on a given computer system. Clark and Wilson partitioned D into two disjoint subsets, a set of constrained data items (CDI) and a set of unconstrained data items (UDI).

Subjects are included in the model as a set of entities that can initiate so-called transformation procedures. A transformation procedure is defined as any non-null sequence of atomic actions. An atomic action is defined as a non-interruptible execution that may result in a change to some data item. A CDI can only be changed via transformation procedures.

In the Clark-Wilson model, integrity validation procedures are introduced to validate that any given CDI has the proper degree of integrity and authentication procedures are mandatory to initiate a transformation procedure.

### British Medical Association model

In 1996, Anderson [8] investigated threats in current medical information management for the British Medical Association, and proposed a security policy for clinical information systems. This model focuses on access control, patient privacy and confidentiality management. It has a horizontal structure in access control rather than a vertical hierarchy as used in the BLP model. The model consists of nine security principles that describe use of the access control list for each clinical record, patients' right to access the list, clinicians' responsibility to inform any modification to the list of the patient, integrity condition for the record before the expiry date and the need of audit trails for all access to the records.

## 2.8 Properties and policy models

The Bell-LaPadula model is designed for confidentiality of objects and restricts read and write access between different security levels. According to the simple security property (NRU), low subjects are not allowed to read high objects; it implies that there are objects whose existence is not known to low subjects and hence BLP requires confidentiality of high objects. The *-property (NWD) prevents high subjects from writing high information on an objects which can be accessible by low subjects.

The Chinese Wall model changed the hierarchical model of BLP to a horizontal model with classes with different interests; depending on the interest of the class, access is granted. Between the classes that conflict their interests, confidentiality of an object must be kept to subjects of the other. Integrity is not considered.

The Biba model is designed for integrity as a dual concept of BLP. Integrity here means that information may only flow downwards unlike in BLP. The model prevents any compromise of high level objects from low level subjects, but it does not specify any rule to restrict attacks on integrity by subjects at the same level. Furthermore, we cannot control integrity failure carried out by the creator of an object in this model. However if the integrity of an object is defined independently of its creator, this model cannot provide proper control. In applications where even the creator of an object cannot break its integrity, such as publishing, this model is not appropriate.

The Clark-Wilson model introduces a partition of data items: $CDI$ and $UDI$. It is required that subjects must be identified and authenticated; that objects must be manipulated by authorised programs and subjects must execute allowed programs. An audit log has to be maintained. The main property to protect in this model is integrity and the model requires procedures for authentication and integrity validation. This model also protects the integrity of system invariants, e.g., total amount of money in the system and enforces separation of duty.

In the British Medical Association model, the existence of the access control list implies the need for confidentiality and authenticity of subjects on each object. By the nature of medical records, integrity of objects for a specified period is mandatory.

Any security policy which restricts access on resources requires authorisation procedures which need identification of subjects. Since subject authentication is a strong means of identification, authenticity of subject can be assumed as a fundamental property in security models.

In view of the time scale, we can see integrity in two types: temporary integrity and persistent integrity. The former means integrity which is required for a specified finite period and the latter for eternity. Persistent integrity is appropriate for history-accumulating applications such as newspapers and legislation, but does not fit for applications that need consecutive revisions and disposal after document expiry such as customer billing data. Integrity control must be done in a different manner depending on its purpose.

As demonstrated in Section 2.1 and 2.2, integrity can be regarded as a part of object authenticity, and authenticity is a primary requirement for publishing such as authorship of publications. Both are fundamental properties for publishing. Confidentiality is not a major concern in publishing since knowledge distribution is its main purpose.

## 2.9 Properties and publishing

Consider typical publishing applications: newspapers, medical directories, legislation and certificates. Every case requires not only integrity but also authenticity, or authorship. It is commonplace that we wish to identify the reporter who wrote an article,[9] the doctor who wrote a treatment protocol, the legislator who voted for an amendment proposal in a legislation process and the issuer of a certificate. The name of the person in charge usually has a strong influence on the trustworthiness of the object. As mentioned above, object authenticity interacts with integrity and subject authenticity can be used to verify authorship. Authenticity is mandatory in publishing as well as integrity.

There has always been a need for applications which collect and analyse historical information for public reference. Nineteen Eighty-Four [86] by George Orwell shows a potential threat from a totalitarian regime: a single-party dictatorship in which Big Brother controls everything, changing the past (and facts already published) to control the public. A cluster of globally distributed publishing servers providing concrete integrity may help prevent such manipulation of history.

[9]Although, in some circumstances such as whistle-blowing, maintaining anonymity is a highly desirable property.

Publishing-specific properties like publicity and anonymity must be considered in secure publishing frameworks. We have criticised the negative influence of anonymity in Chapter 1, but it is clear that anonymity exists on the Internet and some applications require it. Let us discuss its positive aspects and ways to control it. Among the major security models, anonymity is not considered at all but there are anonymous publishing applications. Electronic voting is one case: it needs anonymity of voters to achieve secret ballots, but voting schemes should support universal verifiability. So these schemes usually publish some information to check the number of votes or their integrity. Auction systems are similar.

There is additional value to anonymity when the conventional paradigm migrates to a digital paradigm. For example, when we use banknotes, we do not care about the traceability of our spending not because of technical reasons but because of economic reasons. Since each banknote has its own serial number, there is a way to trace all banknotes. However, nobody does it because it is both expensive and not usually necessary. The situation with electronic cash is different; basically each transaction party has a facility to record transaction details and the cost is not high at all. Even duplicating digital money costs nothing. Furthermore, each party's logging facility can be connected and accessible on the network. This increases the probability of transaction information leakage. Being digital makes many things simpler and easier to use, which can make them vulnerable.

If the tracing process can be done at a low price, it may interest organisations like market research firms and advertisement agencies as well as the Inland Revenue and the World Bank. It may reveal criminal activities, but it will attack privacy simultaneously. Eventually, abuse of such traces will lead to distrust in electronic transactions. This is a completely different environment for users and it is an obstacle to the uptake of electronic economy. Potential threats also reside in people's minds; they are afraid of being accused by accident. It is not easy to persuade people to move to the electronic economy. Anonymity can help reduce the threats in their minds.

## 2.10 Notes on security primitives

We present brief notes for some security primitives to supplement investigation on security properties carried out in the prior part of this chapter. This notes include descriptions about digital signatures, one-way hash functions, Proactive security and the Rampart. The first two are related to authenticity and integrity, and we survey their definitions as given by others. The last two are ways to obtain resilience and high availability, and we describe their design ideas.

### 2.10.1 Digital signatures

A digital signature[10] is a value that enables to identify the originator and verify the content integrity of a digital object. Public key cryptography is the typical way to provide such signatures.

Digital signatures are used in subject authentication; we can identify the signer using his key. It may not be a perfect problem-solver; the key may be stolen, and the mapping between the key holder and the key is vulnerable. Temporary possession of a key is sufficient to impersonate the original key holder.

Digital signatures are also used in object authentication. Signature schemes enable us to check whether the object has been compromised or not. They play a role in integrity verification as well as in subject authentication; digital signatures are comprehensive means of authentication.

Digital signatures are one of the most widely used methods of authentication as well as passwords and a series of important studies have been carried out on in this topic. We survey the definitions of digital signature offered in the literature to date.

**Diffie and Hellman** introduced the concept of digital signature in their sem-

[10]**signature** 2. the name or special mark of a person written with his or her own hand as an authentication of some document or writing
4b. a distinguish mark of any kind [105]

inal 'New Directions' paper [40]: *it must be easy for anyone to recognise the signature as authentic, but impossible for anyone other than the legitimate signer to produce it.* At the time when this paper was written, the only known way of doing this was using Lamport's one-time signature [67]. A decade later, Diffie gave another definition [39]: *a way of demonstrating to other people that (a message) had come from a particular person.*

**Goldwasser, Micali and Rivest** gave a more involved description that explicitly mentions a number of algorithms and their properties: a key generation algorithm, a signature algorithm, and a verification algorithm. The signature algorithm produces a signature using as input, the message, the key and possibly other information (such as a random input); however, in their definition [47], the algorithm produces only a single output.

This definition excludes the large class of arbitrated signatures that were already well known and in use by that time (for example, Akl's signature [2]) as well as most of the special purpose signature constructions that require interaction, such as undeniable signatures, designated confirmer signatures and oblivious signatures [101].

**Naor and Yung** refined the approach of Goldwasser, Micali and Rivest, by cutting the complexity theoretic requirement of the construction [78]; it was finally reduced by Rompel [95] to the existence of one-way functions. However, like Goldwasser, Micali and Rivest, their definitions also fail to deal with signatures that use interaction.

**Pfitzmann** provided an extensive study of disparate signature schemes in [87]. She concluded that the general definition of signature is a process with a number of access points — typically for the signer, the recipient and the court. Time is a necessary component, although logical time—in the sense of a 'global notion of numbered rounds'—is sufficient [87, p. 54]. Special access points can be added for risk bearers such as certification and revocation authorities.

**The ITU (International Telecommunication Union)** defined digital signature in the following two ways: a form of seal associated with a specified part of a document which provides proof of uniqueness of the identity of the originator, or source, who applied the seal; it supports non-repudiation of origin of the sealed, i.e., signed, part, in Recommendation T.411 [57]; a cryptographic transformation of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery, e.g. by the recipient, in Recommendation X.800 [56].

### 2.10.2 One-way hash functions

A one-way function is a function that is easy to compute but computationally infeasible to find any of the values that may have been supplied to the computation; a hash function is a function that maps values of arbitrary length to values of a fixed length. A one-way hash function is a hash function that is one-way. It is a fundamental building block in cryptography and plays an important role in subject authentication and object integrity validation.

We list a series of authentication schemes using one-way hash functions.

**Needham** introduced a subject authentication scheme [108, pp. 129–132] without transmitting a password in the Cambridge time-sharing system: the server stores the user's password $p$; when authenticating, the user sends $h(p)$ where $h$ is a one-way hash function and the server calculates $h(p)$; if they match, the user is authenticated. In 1997, Needham [81] introduced a similar scheme for banking transactions.

**Haller** developed S/Key [51] that provides subject authentication, especially authorisation. Its operation is as follows: a user chooses a random number $r$ and calculates $h^i(r)$ for $1 \leq i \leq n$ with a one-way hash function $h$; the user keeps all $h^i(r)$ and the server stores $h^{n+1}(r)$; when request authorisation, the user provides $h^n(r)$ and then the server calculates $h(h^n(r))$ and compares it with $h^{n+1}(r)$; if they match, the user is authorised. This is an extension of Needham's scheme.

**Molva et al.** designed is an authentication and key distribution system named KryptoKnight [76]. It uses a message authentication code $mac$ that can be either a block cipher DES [79] or a one-way hash function MD5 [94].

Authentication in KryptoKnight is carried out by challenges and their responses: a user $A$ chooses a nonce $N_a$ and sends $\{A, N_a\}$ to a server $S$; the server challenges with $\{mac(N_a, N_s, N_a \oplus S), N_s\}$ where $N_s$ is a nonce chosen by $S$; then the user responses with $mac(N_a, N_s)$. Hence both sides can authenticate each other.

**Anderson et al.** introduced the Guy Fawkes protocol [9] applicable to subject and object authentication. Its operation is as follows: a principal chooses a random codeword $X$, calculates $Y = h(X)$ where $h$ is a one-way hash function and constructs a message $M$ containing $Y$ and some job description; then the principal computes $Z = h(M)$ and publishes it anonymously; the principal does the job specified in $M$ and then reveal $M$. When he provides the message $M$ and the codeword $X$, people can verify whether he did the job by calculating the published hash $Z$. The Guy Fawkes protocol provides a serialised mode and this mode can be used to detect main-in-the-middle attacks on Diffie Hellman key exchange and thus to set up an confidential channel indirectly. In this channel, if a attacker cannot participate in it from the start, he cannot join in the channel later. A detailed description of this scheme will appear in Section 4.7.

### 2.10.3 Proactive security

As a result of recent advances in cryptography and the integration of cryptographic algorithms into secure communication protocols, it becomes more effective to attack the end systems and the weaknesses of the protocol implementation. With the openness of the Internet, these attacks have become more feasible.

Two plausible solutions to these threats arise: one is periodic refreshment of secret data, and the other is to distribute the secret data among multiple servers using secret sharing and threshold techniques.

Periodic replacement of data is not always feasible, specifically for long-standing secret data, such as long-term keys, signatures and certificates. Also, the distribution of data among several servers does not secure against breaks into the entire system throughout the lifetime of the system, which may be very long.

Proactive security is designed to handle such situations. The model of proactive security does not assume that all systems are always secure, i.e., they are never controlled by the attackers. Instead, it considers cases where some components of the system may be broken into. Furthermore, these protocols do not even require identification of when the system is broken into; instead, sometimes they proactively invoke recovery procedures, hoping to restore security to systems and cause the attacker to lose control. These proactive protocols combine the idea of periodic updates with techniques of secret sharing.

We can find examples of proactive security which already exist, such as periodic change of password, one-time passwords and key refresh protocols. Server synchronisation is also an application: a group of servers performs secret sharing and is connected to a broadcasting medium. The system is synchronized by a global clock. Time is divided into epochs of time (e.g., day, week, month, etc.) and each epoch starts with a refresh phase – shares are refreshed in every epoch.

### 2.10.4 Rampart-based services

Rampart is a toolkit for such services including techniques of atomic group multicast, reliable group multicast of Byzantine agreement[11]-type, group membership and output voting. Basically it uses state machine replication which provides outputs from replicated servers to a client; the servers in a process group perform output voting. The Rampart toolkit is located between net-

[11]In 1982, Lamport, Pease and Shostak [68] introduced a basic problem in distributed computing by taking a historical example of a battle in Constantinople (former Byzantium) between the Roman Empire and Ottoman battalions. The problem known as *Byzantine agreement* is like that: can a set of concurrent processes achieve coordination in spite of the faulty behaviour of some of them? The faults to be tolerated can be of various kinds. The most stringent requirement for a fault-tolerant protocol is to be resilient to so-called *Byzantine failures*: a faulty process can behave in any arbitrary way, even conspire together with other faulty processes in an attempt to make the protocol work incorrectly. The identity of faulty processes is unknown and it reflects the fact that faults can happen unpredictably.

work and application layers; it is mainly in the transport layer. On the top of the Rampart layer, one can build applications using the Rampart features.

As an application of the Rampart toolkit, the Ω service [93] provides interfaces for managing public and private keys in a distributed environment; for public keys, it supports registration, retrieval and revocation, while for private keys, key escrow, recovery and message decryption with escrowed keys. The goal of the Ω service is to provide a set of policy-dependent functions that can be tailored to fit a wide range of key management policies.

## 2.11 Summary

We examined definitions and underlying concepts of properties related to publishing including integrity, authenticity, confidentiality, publicity, anonymity and availability. As a preliminary study to construct a publishing policy, we reviewed major security models including the Bell-LaPadula model, System Z, Chinese Wall, the Biba integrity model, the Clark-Wilson model and the British Medical Association model. Both security properties and policy models were reviewed.

# Chapter 3

# The nature of publishing

In this dissertation, we use the term classical publishing to mean the conventional method of making printed books or periodicals on paper, distributing them via physical transport to shops, and selling them in shops on the street. In contrast, electronic publishing means authoring and distributing an idea through an electronic means over a networked environment. Here, we will investigate the nature and requirements of publishing along with the contrast between these two methods. The security aspect of publishing is also investigated.

## 3.1 Changing paradigm

As we pointed out in Section 2.4, classical publishing takes many steps to deliver a writer's idea to readers of a publication. It requires some physical equipment, professional printing skills and distribution networks which all keep the publishing cost high.

The high cost deters copying and alteration. High quality editorial and publishing techniques require expensive efforts to forge; for example, quality picture display, sophisticated font design, long-lasting paper and special purpose ink can all be expensive to imitate.

Any publisher has a target market and will investigate the customer's interest when selecting what to publish. As a result, readers usually have books meeting their expectation. Successful publishers can obtain authority in the market and the reader's expectation for their forthcoming publications is high. The more successful they want to be, the more carefully they have to select authors. The selectivity of publication is driven by the market. This is the conventional means of quality control in publishing.

An important aspect of classical publishing is persistence. When a book is published once, it lasts physically and its content is unchanged. It is like a word engraved on a stone. If we have enough distributed issues of a book, we can be sure that, once printed, the content of the book cannot be altered. Even though a fake copy is found, we can easily identify it and prove that it is not genuine.

Another advantage of classical publishing is that paper is easy and comfortable to carry and browse anywhere without additional apparatus. Neither electricity nor a specialised browser is necessary. Some electronic alternatives to paper books, such as Compaq's Virtual Book [28], have been developed and provide various functions, but they assume some additional resources such as a power source, network connection or remote document server. Their display is not as comfortable as paper yet. By far, paper has been the most comfortable medium for humans.

After the advent of the first moveable type printing press, leading to Jikzi in the east and the Gutenberg Bible in the west, the cost of publishing decreased. The typewriter reduced the cost of self-publishing and the photocopier cut the duplication cost. Computer-aided publishing tools make it affordable for individuals to publish high quality books at home. People could achieve commercial-level printing quality when they use these tools, but still lacked a means of distribution. As the photocopier provided a means of passive attacks on publishers, computer-aided publishing gave a means of active attacks, but the distribution problem still remained.

Electronic publishing provides the needed distribution capability; as a result, people can play a similar role to classical publishers. Electronic publishing removes all the barriers to displaying and distributing one's ideas; it changes almost all the characteristics of classical publishing. The Internet explosion is fundamentally about publishing; it is an electronic version of the Cambrian explosion and will provide huge diversity. People with access to the network can write their ideas without following any selection process and publish them worldwide at once. Electronic publishing redefines the power balance between authors and publishers and challenges all cost-relevant issues.

Because of the many benefits of electronic publishing, it will change many parts of the classical publishing world such as newspapers; in fact, there is a newspaper[1] which gave up publishing on paper and moved to electronic publishing altogether. It seems that the movement of major publishing sectors from classical media to online media will be fast.

We can also find a trial to make a link between the classical and the electronic. In his popular book 'Creating Wealth' [106], Thurow does not include footnotes in the book but provides a web link to reach it. Obviously the book was no longer self-contained and partially lost the benefit that it can be read without any apparatus, although the author can provide more detailed information to readers using hyperlinks in the electronic footnote. This hybrid trial is not completely successful, but is meaningful in the era of changing publishing paradigms.

However there is no system without drawbacks. One drawback with the web is that if an author wishes to change what he published yesterday, he can do so without any cost and accountability. This leads to a problem in publishing, namely the reliability of the publication. Under these circumstances we cannot verify whether a copy of the publication is current or not, and we cannot build the same level of reliability as paper media have nor achieve the same legal authenticity. This is important in electronic commerce; if documents lack legal authenticity, they cannot replace paper-based transactions completely.

The quality of publications is another problem. With the rapid growth of the

[1]Associated Press (AP) reported that Orem Daily Journal in Utah, USA gave up paper publication and turned to an electronic publishing only system on 5th August 1999. Professor Pryor at University of Southern California commented this is the first case where a newspaper turned to an online medium exclusively.

Internet, the quantity of electronic publishing is increasing enormously. Simultaneously, this increases the time taken to find trustworthy information. Odlyzko [84] pointed out that demonstrative electronic writing led to a flood of information, much of it of poor quality, and lowered the levels of understanding.

Even though we may have found reasonable-looking information, sometimes we suspect that the publication was not written by the claimed author or that the content is not as it was first published. So electronic publishing is no longer evidence of what the author said. Without evidential power, electronic publishing cannot replace conventional publishing.

Since copying electronically published materials does not cost anything, electronic publishers have lost a barrier against illegal copying. Copyright protection has become one of the central issues in electronic publishing. It requires fundamental changes in conventional understanding. We find a change in the status of copyrighted material: if we buy a book, then we can resell it after use and it is completely legal. All rights to handle this instance belong to the buyer of the instance. However we cannot buy a digital instance but a licence to use it. Hence reselling the licence is allowed but not reselling the instance itself. Electronic publishing requires a different type of copyright protection system from conventional publishing.

## 3.2 Electronic publishing

The advantages of electronic publishing are overall-cost savings, speed and huge potential extendibility. As discussed in Section 3.1, electronic publishing reduces expense in all steps of publishing. It minimises the capital cost and thus makes publishing affordable for a large number of people. It deskills the process; a primary school student can publish his homework to the world. The speed is such that we can encourage active remote discussion. This is especially helpful in academia; an example of this is Ginsparg's eprint archives [46, 103] that stores and distributes academic paper preprints, mainly in physics. Furthermore, electronic publishing may incorporate several multimedia techniques and software engineering techniques for more flexible and versatile features.

Scholarly journals can move earlier to electronic publishing than other classical publications, since most of users of the archives are involved in non-profit activities, and want faster communication and collaboration. Although journal publication itself is a profit-making enterprise, the profit mainly comes from libraries not from individual researchers or students. The main purpose of journal publishing is communication between scholars; for faster and more efficient research with a smaller budget, electronic publishing is a desirable substitute. Odlyzko [83] pointed out that the cost burden of the research libraries for journal subscription is serious, and made a prediction that the libraries will be a strong driving force for electronic journal publishing.

Since we do not need to have screening by the publisher, we can publish any idea without hindrance. This has privacy aspects; at the same time it has enlarged the freedom of speech and expression. People in Kosovo used the web to demonstrate their situation to the world in the Kosovo conflict. During the democratic process in Indonesia in 1998, electronic publishing was adopted by Indonesian students to present their ideas despite government censorship.

Electronic publishing will replace a significant part of classical publishing. Classical publishing will keep some territory because of its fundamental merits, such as the material benefits of paper, the selected quality of the content, persistent reference and straightforward evidence of authorship. Material factors such as browsing without additional devices and the familiarity of paper printing may not be replaced shortly, but the other advantages can be matched in the electronic world. To provide a system that does this is a goal of this dissertation.

Consider the quality issue of electronic publishing. It can be an issue of authority; if there is an authority who publishes selected articles or reports, the quality of its publications is recognised by consumers. In classical publishing, successful publishers have been a source of such authority and they have built their authority with reliable publications meeting the consumer's expectation. In the electronic world, there are numerous small publishers and the competition to achieve better recognition from consumers is harder than in the classical publishing market. Furthermore, the lack of reliability of publications is an obstacle to build such authority. However given reliability, authority will emerge in time.

Persistence is an important robustness issue within electronic publishing; if published material is only available for a short period, we cannot expect an accumulation of knowledge, which is a serious problem for scholarly publication. This issue will be discussed in Section 3.3. In addition to long-lasting publishing features, we need an infrastructure to browse old publications as well as new ones to achieve persistent reference and this issue will be discussed in Section 3.4.

Masquerading is another aspect; when an article is published by Mallory with the name of Alice, how can we know that this article is not written by Alice? There is thus an authenticity issue. On the other hand, Alice might later falsely claim that Mallory forged her article, so there is also a non-repudiation issue. The evidence of authorship needs to be guaranteed; we will discuss this issue further in Section 3.5.

The frozen copy issue, i.e., integrity issue of once published copy, will be discussed and a solution for the problem will be addressed in Section 5.3.

## 3.3 Long-lasting publishing

We have some hand-written books one millennium old in the library. Some modern books may last as long but there are a number of problems for persistence of publications, such as material weaknesses, the small number of printed issues and limited distribution area.

Consider threats to conventional publications. Paper can get wet and damaged easily because of its physical nature. Quality acid-free paper and long-lasting ink are expensive. Book maintenance needs attention to the surrounding environment such as temperature and humidity. Because of the cost of book manufacture and the size of book market, the number of printed issues is limited.

For these reasons, most books are destroyed over time. If a large number of issues of a book could be distributed in a wide area, local incidents may not affect its lifetime so much.

In electronic publishing, we still have the weakness of media; magnetic storage is not robust enough to survive its centenary. Persistent maintenance is needed even though the refresh process for the old copies is much simpler than in conventional publishing. Although electronic means help us overcome cost, geographic and mass-publishing restrictions, the lesson from conventional publishing is still valuable: electronic publications should be stored in a safe place and well-maintained and they have multiple copies in distributed areas. The point here is the need for a safe repository and wide distribution of the publication.

A cluster of geographically distributed repositories, periodic backup, and refreshment for stored publications in each repository can constitute an infrastructure for long-lasting publishing. For such a cluster of repositories, the recovery of faulty publications is an important issue. A mechanism for distributed storage and recovery will be presented in Section 5.5.

## 3.4 Persistent browsing

Although electronic publishing increases the feasibility of long-lasting publishing, it introduces a new problem, namely browsing environment preservation. Classical publications only need our eyes to read them, but electronic publishing requires additional apparatus. Sometimes this apparatus includes hardware systems and the operating system on which the browsing application can run. For example, keeping a VisiCalc[2] file for the Apple II is not helpful for later use because the program VisiCalc is obsolete and the runtime environment, the Apple II computer, has become an antique. If we do not keep the program and the computer, it is hard to browse the file. This is a

[2]VisiCalc is an early spreadsheet program running on a 32 KB Apple II and was developed by Personal Software Inc. in 1979. An enhanced version for IBM PC DOS is available at the original developer's web site <http://www.bricklin.com/visicalc.htm>.

general phenomenon in the computer market and is becoming more and more common since computer and software companies appear and disappear every day. Since VisiCalc was a dominant program in its age, we may find some proprietary emulators running on current computing environments. If we use a less popular program, we cannot even expect such a favour. Even using a dominant program in the market cannot be a solution to the problem.

Link rot is is a common inconvenience on the web; over time, hyperlinks may become invalid or point to irrelevant content. Unlike the threats mentioned above, this is mainly caused by mistakes rather than environmental hazards or malice, and also caused by the transient nature of web content. It is a serious enough factor to threaten persistent reference. Many causes of link rot can be listed: not paying attention to management, intentional hiding of published information, lack of resources, and critical changes of its working environment. These factors make web links transient; the destination document pointed to by a web link can be frequently changed. It is a part of the nature of web publishing. If a publication is both valuable and not intentionally hidden, it is better to have a repository which enables us to keep public.

As we can see in the case of VisiCalc, keeping the browsing environment is not efficient. It can be a solution to have a flexible mechanism that can support the instructions used in browser applications, add new instructions easily and export files to any environment supporting the mechanism. Markup languages can provide us with an answer to the problem. With the wide use of HTML, markup languages have been regarded as a proper infrastructure for web publishing. XML[3] is another standardised markup language supported by major web browsers and provides flexible extensibility and functionality.

As well as the infrastructure for long-lasting publishing discussed in Section 3.3, we now consider an infrastructure for persistent reference: distributed repositories on the network with highly flexible markup functionality. In each repository, we store published information which is backed up periodically. These repositories are networked and share information with one another. They provide information on request with a unique identifier throughout the network

[3]Extensible Markup Language, see [23].

like the URI.[4] The information published in these repositories is written in a flexible markup language which can be made extensible by defining and adding functions when necessary. We then can read published information in the repository without keeping all the browsing environment. One mechanism is the electronic equivalent of a conventional library.

Such a mechanism helps us maintain persistent reference and public availability; we can cite publications held in the server in academic papers and thus accumulate our knowledge. As a library, it provides public availability.

## 3.5 Evidence of authorship

In the past, authorship was regarded as an honour. The publisher's selection procedure makes it hard to publish one's ideas without serious effort, which is a way to maintain the quality of publications. In electronic publishing, the value of authorship is falling as authoring becomes easier and the quality of publishing is less controlled. In fact, it is not easy to distinguish a quality article from a personal memo full of unproven arguments before reading it carefully. Moreover, it is hard to know whether the idea presented in an article is genuine or copied from other publications.

Copying other web publications without permission or notice is commonplace. When we search a topic on the web, we can find several links containing the same paragraph in different sites. A copy-and-paste procedure produces a seemingly good quality article in few seconds. Authorship in web publishing is less clear than with conventional publishing.

The wide use of anonymity on the network blurs the definition of authorship; some believe that it is a virtue of the Internet. Even high quality publications can be found anonymously on the Internet: troubleshooting guides for some systems and survey papers on specific subjects. Presumably, such authors re-

[4]Uniform Resource Identifier; it consists of Uniform Resource Locator (URL) and Uniform Resource Name (URN); URL identifies resources via a representation of their primary access mechanism and URN is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable. For more detail, see [20].

gard that their publications are pure contribution to the community and that revealing the authorship is not necessary since they do not want material rewards. Technically, we have numerous ways and facilities to obscure the origin of publications such as anonymisers and spam mailers.

The commercialisation of the Internet is another reason to make people stay anonymous. If they reveal their email addresses or affiliation, they may receive numerous unsolicited commercial mails. A number of web crawling robots are in operation to collect identities on the Internet to sell to advertisers.

As a result, we have a complex situation with impersonation, unsolicited mails, anonymous publishing and unauthorised copying. To make authorship clear, it is helpful to have an authentication procedure to verify authorship, such as digital signatures. When the author wants to protect his authorship, the problem is simple, but we cannot expect that there are only fair players. As in the case of false stock market research newsletters shown in Chapter 1, if there is a publication that people want to believe but does not bear any proper authorship, it is not simple to attribute.

## 3.6 Requirements of publishing

Along with the understanding obtained in previous sections in this chapter, we list common requirements of both classical and electronic publishing.

**Integrity of content** The content of published material must not be altered after the publication. If it is necessary to change the content, errata or revised editions may be published but they do not overwrite the original publication. They are separate but linked to the original. In conventional publishing, high cost and large distribution helps reduce compromises of published content, but electronic publishing has no cost restriction, so another protection mechanism is necessary.

**Persistent reference** Each publication has a permanent identifier of reference. For example, the combination of book title, author's name, publisher, published year and edition can be an identifier. On the web, a URL can do a similar work, but it is not enough to cope dynamically changing web pages. A universally unique identifier like URI is necessary; we can define it by the combination of the server URL and a server-wide unique identifier.

**Public availability** The publication should be stored somewhere people can easily access such as libraries and government document offices. When a published document has a small number of instances, the repository is of utmost importance. The lack of a reliable repository reduces the availability and credibility of publications. The quantity and area of distribution are also important. When one has a large and wide distribution, we may not need a specific repository because we can find the original copy in many places; the distribution itself plays the role of the repository.

**Revision history preservation** In conventional publishing, it is straightforward to maintain a strict revision history. All revisions are published as an independent publication whatever the format is, e.g., errata, poster, leaflet, another edition of the book, etc. Electronic publishing is very fragile in this respect; no guarantee of strict revision control can be achieved without appropriate tools.

**Evidence of authorship** Publication without explicit authorship is rarely found in classical publications, although we may find some cases of anonymous publishing such as political leaflets and newsletters in a town under siege. Most publications are made by registered publishers and there is no motivation for them to hide the authorship. In electronic publishing, we have no strong means to identify the author of a published material. Anonymous writing is quite common on the Internet.

**Copyright protection** As discussed in Section 3.1, copyright of classical publications had been protected by the high cost of copying, but in electronic publishing this is not the case. The nature of digital data makes it easier to copy; we can hardly see the difference between copied and original digital media. There have been various approaches to solve this problem including digital watermarking and fingerprinting.

**Collaboration support** Before 1960, a large number of papers were written by a single author, but this has changed; for example, the proportion of single authored mathematical papers was 92.95% in the 1940s but it became 57.18% in 1994 [82]. This reflects the fact that collaboration among scholars is becoming more commonplace. Publishing should support shared document creation since it is a dominant means of communication and collaboration.

**Distribution** The essence of publishing is a way to propagate knowledge and the heart of publishing is the distribution network. For classical publishing, the network is supplied by the publisher and is maintained by physical interactions. In electronic publishing, it is sufficient to have access to the Internet in order to distribute ideas, and no further physical interaction is needed. Electronic publishing provides a wider and easier distribution means.

The requirements discussed here will be a guideline to design our publishing framework.

## 3.7 Publishing and security

In Chapter 1 and previous sections in this chapter, we have seen fraud cases and threats to authenticity, trustworthiness and immutability in electronic publishing. We found that they are related to integrity. Many frauds involve breaking the integrity of published information. If the content published on the web is far different from the original, the publication cannot be credible any more. Even for free academic services like Ginsparg's eprint archives [46], such a threat may be critical; if a significant number of preprints turn out to be fake or not genuine, people will not use it any more. Although a service is freely available and is provided with good will, it cannot be useful any more if it loses credibility. In commercial applications, such as web auction or purchase on the web, fraud can be more serious; it may become a crime rather than a credibility issue.

Authorship becomes more important in electronic publishing than in classical publishing. Under a fragile state of credibility, users need to decide which information is credible and which is not. Since the author of a publication is a major criterion of quality and value, a means to verify the author is desirable. In a legal case, if we can identify the author, we know, at least, which person to accuse.

From the system's point of view, high availability or accessibility of the published information is another security issue, although it is not the same level issue as integrity or authenticity. Persistent access to publications strengthens their immutability and makes attacks against their integrity more difficult; long-lasting publishing mentioned in Section 3.3 and persistent browsing in Section 3.4 support persistent access.

We discussed security properties that can make web publishing credible and reliable. The points made here will be used in the design of our publishing policy and framework.

## 3.8 Summary

We investigated the nature of publishing in the era of changing publishing paradigms. By comparing conventional and electronic publishing, we could understand the similarity and the difference more clearly. We also presented important points in web publishing including long-lasting publishing, persistent browsing and evidence of authorship. Finally, we presented a series of requirements for publishing and discussed the security issues.

# Chapter 4

# Applications of publishing

We present a series of applications of electronic publishing including electronic commerce, electronic voting, public key certification and a medical application. Following the description of publishing applications, we present related research published by the author and his colleagues as example applications including an electronic payment scheme [69], a voting scheme [70], an authentication scheme [9] and the Global Trust Register [4, 5].

## 4.1 Electronic commerce

Recently, the most popular publishing application on the network may be electronic commerce. The merchant displays products and the customer buys them on the network. Sometimes, the bank plays a role in clearing transactions between the customer and merchant. The action of displaying products on the network is a typical example of publishing.

In electronic commerce, we can see two types of publishing: one is descriptions of products and the other is promotions to assist sales. Product-relevant information includes price lists, specifications, billing and payment methods, company policy and customer support. It must be reliable as any violation of published information can be a breach of contract. Promotion is about advertisement of products or notices of events. Such information is usually vague and temporary by its nature. We focus on the protection of former type publishing.

In commercial transactions, trading conditions are critical publications. The trading conditions may describe product price and specification. Hence a reference to these conditions should appear in the receipt. In conventional commerce, we usually do not have detailed specification of products because the transaction is done face-to-face. However in electronic commerce, we pay for products before delivery without seeing them and so fear that delivered goods might be different from those advertised. When the fear turns out to be real, we need a receipt with detailed specification so that we can protect our interest. If we cannot get such a receipt, it may be an alternative to have evidence such as a frozen copy of the electronic offer including the detailed specification and perhaps the signature of a witness or a notary.

A number of real transaction cases that fail due to the lack of evidence can be found. According to the Internet Fraud Watch,[1] operated by the US National Consumers League, complaints about Internet transactions have increased 600 percent since 1997. The occurrence of fraud on the Internet is becoming a serious threat to Internet businesses. In this report,[2] web auction and merchandise are the two most frequent of the top ten online scams; for example, items bought are never delivered or are not the same as advertised. In many cases, proper invoices for legal claims are not provided; even though a buyer has an invoice or a receipt by printing the invoice/receipt page on the web, he cannot claim that it was generated by the merchant's server. In the case of non-delivery of items bought, using a credit card with a purchase protection insurance may be a potential answer, since some credit card companies repay the customer when problems happen, but this service is not universal. Eventually the high fraud rate of Internet transactions may increase transaction costs. The case of items which are not as advertised is more tricky. Invoices usually include only the name of the ordered product rather than detailed specification but sometimes the specification is essential, e.g., the amount of memory,

[1]`<http://www.fraud.org>`

[2]`<http://www.zdnet.com/anchordesk/story/story_2724.html>`

or the main processor in a computer. It is not easy to accuse a merchant without trustworthy evidence.

Payment is an important issue of electronic commerce and has been studied in various ways. The requirements of payment schemes usually include the evidence of transactions, the confidentiality of payment information and the collusion prevention. We present a mechanism for payment in Section 4.5.

## 4.2 Electronic voting

Electronic voting is not exactly the same as publishing but its implementations in practical schemes adopt various publishing mechanisms such as bulletin boards, remailers and public channels.

Among the requirements of electronic voting as will appear in Section 4.6.3, there are features that enable voters to check their voting process such as universal verifiability and precise tally. The universal verifiability is for voters to verify that their votes are processed without any compromises by other parties. To make voters ensure that the final tally is precise, voting schemes should provide a means of verifying the total number of votes and the detailed result. Bulletin boards are a popular means to publish the result of the vote.

Basically, the reason to use publishing mechanisms in a vote is to ensure integrity of votes. They also provide public availability and distribution. Depending on the purpose of a vote, persistent reference for the voting result may be needed.

Anonymity is another aspect of electronic voting, since the majority of electronic votes are of the secret ballot type. We do not want to reveal the relationship between voters and their votes during or after the voting process. So most voting schemes have or assume anonymising processes such as Chaum's MIX [29] or cryptographic alternatives. Shuffling the order of published votes in a network of bulletin boards is a way of blurring the trace of votes.

An electronic voting scheme with an escrow-like feature is given in Section 4.6. The scheme provides evidence of authorship, or who voted for what, when there is an authoritative order to reveal it.

## 4.3 Certification authority

A certification authority is a public-key repository that certifies the ownership of public keys. It plays a major role in the public key infrastructure. Recently, the number of security products using public key algorithms is increasing significantly and the role of certification authorities is becoming more important.

This application shares many aspects with publishing and needs most of the publishing requirements such as integrity of content, persistent reference, public availability, revision and revocation history, evidence of key ownership and distribution.

Once keys have been registered in a certification authority, they should not be contaminated or compromised. If a key can be changed without the certification authority doing it, its authority will be undermined.

All the revocation information should be recorded and browsable on demand so that users can see all the revocation history of keys. Since keys are usually generated with a lifetime and secret keys may be exposed by accidents or attacks, sometimes we need to revoke keys and we do not want to use obsolete keys. When a key is revoked, its certificate should become invalid for future signatures. Such information should be available from the authority.

As well as certificates, the information necessary to verify certificates should also be available from the certification authority such as certificate specifications, certificate issuing procedures, used algorithms and related technical primitives.

By definition, a certification authority is a means to distribute public keys. Evidence of authorship, i.e., in this case, key-ownership certification, is the main job of a certification authority.

We describe the Global Trust Register in Section 4.8. This is a certification authority printed on paper. It is an example of a conventional publication that supports electronic transactions and thus provides them with some of the advantages of conventional publishing.

## 4.4 Medical application

The volume of medical information published on the network is growing rapidly and the Internet is starting to be regarded as a proper medium to disseminate medical information because of easy and wide access, multimedia data handling, easy presentation and transparency over systems. When there is sensitive information not proper to publish, people may use their organisation's intranet with some access control. For public or private information, in any case, web publishing methods are an efficient means of sharing medical information.

The British National Formulary is an application that illustrates the merits and the problems of web publishing. It is a directory of medicines allowed in the United Kingdom and is the authoritative source of prescribing information. It is published every six months on paper by the British Medical Association and the Royal Pharmaceutical Society of Great Britain. Since 1994, it is also available online for doctors and pharmacists.

Unlike patient or prescription information, the formulary is not sensitive in view of privacy and confidentiality. Its integrity and authenticity are the concerns; its publishing requirements include integrity of content, persistent reference, public availability (with an access control), revision history preservation, evidence of authorship, copyright protection and distribution.

It is clear that a compromised formulary can be a mortal danger, as it contains drug dosage and other data. If this is wrong or obsolete, it may cause dangerous prescriptions. Availability of the formulary is also important if doctors are to rely on it in emergency cases. The formulary is updated periodically. When an incident that involves the formulary happens, we need to identify who was responsible for the incident and verify whether a genuine copy of the formulary was used.

The online formulary has a proprietary user interface running on Microsoft Windows[3] operating system and is available for limited users; in fact, the online version is written in a type of SGML and is browsed by its own client software called Q browser.[4] If it migrates to XML, the service can use freely available XML-enabled web browsers since current major web browsers support XML. Hence, without additional client software, the online formulary could be seen on usual web browsers.

Another concern is that in local hospitals, doctors use their local formulary in the hospital as well as the British National Formulary. It is necessary to merge or link information for the same medicine even though they reside on different databases. Since the local formulary is confidential because it reflects decisions on rationing, we should provide a distinction between the local formulary and the national formulary. From the user's point of view, it is necessary to see both formularies in one screen but with a distinct interface. A hierarchical definition of document types and their inheritance can make this job easy and XML supports such functions.

## 4.5 Contribution 1: Customer-centric payment

We present an electronic payment scheme focused on personal secret management and customer-centric transactions; this example application is summarised from our work [69] presented in the USENIX workshop on electronic commerce in 1998.

### 4.5.1 Motivation

As a means of authentication, shared secrets have been used. In such cases, the leakage of the secret is a significant threat. Hence sharing the same secret between the communication principals is not a good means of authentication.

[3]A trade mark of Microsoft Inc.

[4]A trade mark of Quartet Software Ltd.; see `<http://www.quartet.demon.co.uk/qbrowser/qbrowser.htm>`

We suggest a way of authenticating customers without transferring explicit customer secrets and provide a secure online transaction scheme with the authentication mechanism.

For user authentication, the traditional password scheme is still one of the most common methods, although notoriously vulnerable. To survive such vulnerability, some interesting ideas have been put forward, such as strengthening passwords [1], enhancement of existing password protocols [16, 17], protection for poorly chosen passwords [49] and one-time password schemes [51, 99].

Basically, we extend the idea of the personal identification number management scheme of Needham [81] which is an extension of his user-authorisation mechanism in the Cambridge time-sharing system [108, pp. 129–132].

In access control, we have to consider who generates secrets and who maintains them. Some banks generate customers' personal identification numbers (PINs), and send them out; some online shops ask customers to generate passwords for their services, and keep the passwords in their database. It is common that the personal secret is known to the service provider such as banks or online shops as a shared secret between the service provider and the customer. However, there is no reason for customers to trust service providers; in fact, this often turns out to be dangerous. We present a customer-centric transaction model in which the personal secret is generated and maintained by customers.

## 4.5.2 The online transaction scheme

We define three procedures for online transactions: registration, transaction and secret-revocation, and there are three principals: a customer $C$, a merchant $M$ and a bank $B$. The term 'merchant' includes online service providers and web publishers such as shopping malls, book sellers, online travel agents and news providers.

In the protocol description, the notation $A \longleftrightarrow B$ stands for protocols between $A$ and $B$, and $A \rightarrow B$ a message from $A$ to $B$. The symbol $:=$ denotes an assignment. Let $h$ be a one-way hash function. Figure 4.1 shows the interfaces between the principals.

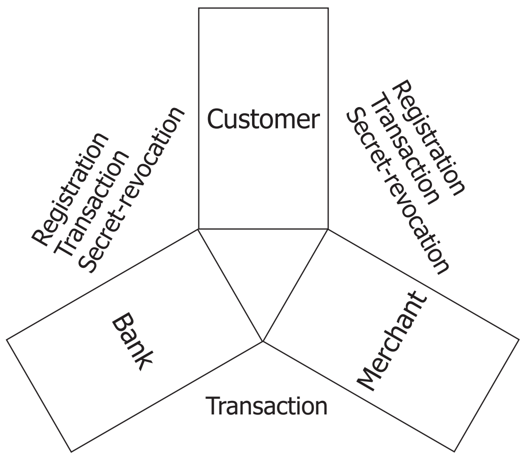


**Figure 4.1. Principals and their interfaces in the customer-centric payment scheme:** in our online transaction mechanism, we have three principals, a customer, merchant and bank, and three procedures that are applied between these principals, registration, transaction and secret-revocation. Between either the bank or the merchant and the customer, all the three procedures are applied; between the bank and the merchant, the transaction procedure is used.

### The registration procedure

The registration procedure has two steps: registration with the bank and with the merchant. First, the customer generates a random number $r_b$, chooses a secret $p$, and calculates a hash $C := h(n, b)$ of her name $n$ and date of birth $b$. Note that if the combination of $n$ and $b$ is not sufficient to uniquely identify the customer, more information can be adopted. The customer writes $r_b$, $B$, and $C$ on a diskette for private use, say $D_c$, and writes $C$ and $K_{cb} := h(r_b, p)$ on a different diskette for the registration to the bank, say $D_b$. She then sends $D_b$ to the bank $B$ with her account number $a$ in a reliable way such as by registered mail or personal delivery to a branch of the bank; this submission method may be replaced with an electronic secure channel between principals for convenience, but it is not a prerequisite.

When the diskette $D_b$ is received, the bank creates a link between the customer's account and $K_{cb}$, and sends an acknowledgement with a random number $b_c$ back to the customer. The customer stores $b_c$ on her private diskette $D_c$. This is the registration procedure with the bank; the registration with the merchant is similar. The customer generates a random number $r_m$, stores it on her private diskette $D_c$, and then sends the merchant a diskette $D_m$ containing C and $K_{cm} := h(r_m, b_c, p)$ in a similar way as with the bank. Then the merchant registers the customer's information, and sends an acknowledgement with a unique merchant secret $m_c$ back to the customer; $m_c$ is a uniquely issued value by the merchant for each customer and is used for the merchant verification by the bank. The customer stores $m_c$ on $D_c$, calculates $K_{mb} := h(m_c, K_{cb})$, and sends $(C, m_c)$ with the merchant identity M to the bank. Then the bank constructs $K_{mb}$ by $h(m_c, K_{cb})$.

**Protocol**

$\boxed{C \longleftrightarrow B}$

| | |
|---|---|
| C: | creates $r_b$ and $p$ and forms $C := h(n, b)$ and $K_{cb} := h(r_b, p)$ |
| | stores (C, B, $r_b$) on diskette $D_c$ and (C, $K_{cb}$) on $D_b$ |
| $C \rightarrow B$: | $D_b, a$ |
| B: | stores C and $K_{cb}$ with respect to $a$ |
| | creates a random number $b_c$ |
| $B \rightarrow C$: | $b_c$ |
| C: | stores $b_c$ on $D_c$ |

$\boxed{C \longleftrightarrow M}$

| | |
|---|---|
| C: | creates $r_m$ and forms $K_{cm} := h(r_m, b_c, p)$ |
| | stores (M, $r_m$) on diskette $D_c$ and (C, $K_{cm}$) on $D_m$ |
| $C \rightarrow M$: | $D_m$ |
| M: | creates account for C with $K_{cm}$ and a random number $m_c$ |
| $M \rightarrow C$: | $m_c$ |
| C: | stores $m_c$ on $D_c$ |
| | calculates $K_{mb} := h(K_{cb}, m_c)$ |
| $C \rightarrow B$: | $C, M, m_c$ |

B: constructs $K_{mb}$ with $m_c$ and $K_{cb}$
adds $K_{mb}$ and $M$ to $C$'s information

**The transaction procedure**

When a customer wants to make a transaction, she establishes a connection to the merchant by invoking client software; the software requests the secret $p$, and establishes a connection[5] to the merchant. The merchant then generates a transaction identifier $t$ and sends it back to the customer; $t$ is a nonce for preventing replay attacks and it consists of transaction time, date, and a serial number. The software calculates $h(t, K_{cm})$, where $r_m$ is from $D_c$ to generate $K_{cm}$, and sends $(C, h(t, K_{cm}))$ to the merchant.

The merchant verifies $h(t, K_{cm})$ for customer authentication. If it is valid, the merchant calculates $h(t, m_c, K_{cm})$, and sends it to the customer with an acknowledgement; the customer then compares the received value with $m_c$ for merchant authentication.

When the customer places an order, the merchant sends the payment serial number $y$ and payment amount $u$ to request payment; the customer generates a nonce $n_c$, calculates a ticket $T_{cmb} := h(t, y, u, K_{mb}, K_{cb})$, and confirms payment to the merchant by sending $(C, B, a, u, y, n_c, T_{cmb})$. The merchant requests the bank to clear the amount by sending $(M, C, a, u, t, y, n_c, T_{cmb})$ to the bank. When $T_{cmb}$ is correct, the bank transfers the amount $u$ to the merchant with an acknowledgement including $h(n_c, b_c, K_{cb})$; the value $h(n_c, b_c, K_{cb})$ is used by the merchant to generate an acknowledgement ticket for the customer. The bank must store the ticket $T_{cmb}$ for some period[6] so that it can detect replays. With the bank's acknowledgement, the merchant generates $T_{bmc} := h(t, u, K_{cm}, h(n_c, b_c, K_{cb}))$ and sends it to the customer as an acknowl-

[5]A secure connection is not necessarily required, since we have the secrets $K_{cb}$ and $K_{mb}$; if we use a secure channel between the customer and the merchant, the transaction identifier $t$ can also act a shared secret.

[6]For example, in UEPS/VISA COPAC [6], the merchant must request the transaction clearing to the bank within 21 days from the transaction date, otherwise payment will not be made. We may apply similar measures.

edgement; the customer checks it, which can be used as a proof of the purchase order and of a proof of payment in case of dispute.

We may consider a false merchant who attacks the real merchant by taking orders and not delivering; since $T_{cmb}$ contains $K_{mb}$, he cannot make monetary gain, furthermore, he must send the ticket $T_{cmb}$ back to the customer so that the customer can verify the eligibility of the merchant.

**Protocol**

| | |
|---|---|
| $C \to M$: | service invocation via client software |
| $M$: | generates a transaction id $t$ |
| $M \to C$: | $t$ |
| $C$: | calculates $h(t, K_{cm})$ |
| $C \to M$: | $C, h(t, K_{cm})$ |
| $M$: | verifies $h(t, K_{cm})$ and calculates $h(t, m_c, K_{cm})$ |
| $M \to C$: | $h(t, m_c, K_{cm})$ |
| | |
| $C \to M$: | makes an order |
| $M \to C$: | $y, u$ |
| $C$: | generates a nonce $n_c$ |
| | calculates $T_{cmb} := h(t, y, u, K_{mb}, K_{cb})$ |
| $C \to M$: | $C, B, a, u, y, n_c, T_{cmb}$ |
| $M \to B$: | $M, C, a, u, t, y, n_c, T_{cmb}$ |
| $B$: | verifies $T_{cmb}$ and calculates $h(n_c, b_c, K_{cb})$ |
| $B \to M$: | $h(n_c, b_c, K_{cb})$ |
| | clear the payment of amount $u$ |
| $M$: | calculates $T_{bmc} := h(t, u, K_{cm}, h(n_c, b_c, K_{cb}))$ |
| $M \to C$: | $T_{bmc}$ |
| $C$: | verifies $T_{bmc}$ |

### The secret-revocation procedure

When the customer wants to change her secret, she has to send old information with a new secret. We suggest a way of revocation and update of secrets: the customer sends a nonce $n_b$ with the identity $C$ to the bank, then the bank sends back a ticket $T_{bc} := h(n_b, K_{cb})$. The customer verifies the ticket $T_{bc}$, if it is valid, she requests to update her shared information by sending $(C, h(n_b, K_{cb}, b_c) \oplus K'_{cb}, h(K'_{cb}, n_b))$, where $K'_{cb}$ is the new secret; if the customer wants to change $p$ as well, she can do it by following the same procedure, but does not have to. When $h(n_b, K_{cb}, b_c)$ is correct, the bank changes the customer's information. The procedure with the merchant is similar. If the customer cannot remember her previous information, the information cannot be revoked online so revocation must be done in off-line manner.

**Protocol**

$\boxed{C \longleftrightarrow B}$

| | |
|---|---|
| $C \rightarrow B$: | $C, n_b$ |
| $B \rightarrow C$: | $T_{bc} := h(n_b, K_{cb})$ |
| C: | verifies $T_{bc}$ |
| | creates the new secret $K'_{cb}$ |
| $C \rightarrow B$: | $C, h(n_b, K_{cb}, b_c) \oplus K'_{cb}, h(K'_{cb}, n_b)$ |
| B: | verifies $h(n_b, K_{cb}, b_c)$ and $h(K'_{cb}, n_b)$ |
| | updates C's secret |
| $B \rightarrow C$: | $ack$ |

$\boxed{C \longleftrightarrow M}$

| | |
|---|---|
| $C \rightarrow M$: | $C, n_m$ |
| $M \rightarrow C$: | $T_{mc} := h(n_m, K_{cm})$ |
| C: | verifies $T_{mc}$ |
| | creates the new secret $K'_{cm}$ |
| $C \rightarrow M$: | $C, h(n_m, K_{cm}, m_c) \oplus K'_{cm}, h(K'_{cm}, n_m)$ |
| M: | verifies $h(n_m, K_{cm}, m_c)$ and $h(K'_{cm}, n_m)$ |
| | updates C's secret |
| $M \rightarrow C$: | $ack$ |

Table 4.1 shows the information stored in each principal after these procedures.

| Principal | Stored information |
|---|---|
| Customer | $C, B, M, r_b, b_c, r_m, m_c$ (p is in mind) |
| Merchant | $C, B, K_{cm}$ |
| Bank | $C, M, K_{cb}$ |

**Table 4.1. Information stored in each principal:** all principals have the communication peer identities such as $C, B$ and $M$, and the customer creates basic secrets such as random numbers and nonces. The merchant and the bank have basic linkage information for the transaction such as $K_{cm}$ and $K_{cb}$.

## 4.6 Contribution 2: Big Brother ballot

We present an electronic voting scheme which keeps the confidentiality of votes unless there is an authoritative order to publish the relationship of voters and their votes; this is a sort of escrowed voting schemes and is summarised from our work in [70]. We used a metaphor to name the scheme *Big Brother ballot* since such authoritative orders seem to be commonplace in a totalitarian regime like in the Nineteen Eighty-Four [86].

### 4.6.1 Motivation

Let us consider the following election model: there is a committee in Parliament and the members of the committee consist of representatives of several counties; they mainly stay in their county except during the Parliamentary session. Occasionally, they need to make decisions outside the session. To vote, they would usually arrange a physical meeting but this is inconvenient. Electronic voting can be an alternative; through the public network, they can participate in voting and verify the result electronically. So far, many people discussed the security requirements for electronic voting schemes.

We introduce another requirement to this model. After the voting process, if necessary, the detailed voting result can be revealed and the members can see who voted for or against, although voting was carried out in secret for fairness.

Since the members of the Parliament are representatives of their county, they may want to demonstrate their performance to the people. In this respect, the disclosure of votes later can be a requirement.

### 4.6.2 Background

Electronic voting schemes have been extensively studied in the last decade and several schemes have been suggested, although only a relatively small number of schemes have been implemented and used. We may categorise these with some criteria: the existence of anonymous channels or public channels [44, 100], single voter's vulnerability [62], method to protect privacy [19, 33], need of a trusted authority [109], practicality or efficiency [22, 34], and cryptographic features used [85].

We are interested in practical schemes with sound potential for practical implementation. The Fujioka-Okamoto-Ohta (FOO) protocol [44] is one such pragmatic scheme; it has several implementations with minor variations [35, 38, 52]. We adopt this protocol as the skeleton of our scheme and will make a variant to satisfy the model we presented. The major problem of the FOO protocol is that the protocol requires all the registered voters to participate in all stages of the protocol during voting; in practice, voters can give up at any phase during voting, and it is another way of expressing their preference which we need to reflect.

We follow Yahalom's representation [109] of the FOO protocol and its basic notation is as follows: $\{msg\}_{K_{A+}}$ represents the public key encryption of $msg$ with $A$'s public key, and $\{msg\}_{K_{B-}}$ represents the concatenation of $msg$ and a digital signature on the hash of $msg$ with $B$'s private key.

The FOO scheme uses Naor's bit-commitment [77] using a pseudorandom number generator $COM_s(msg)$ of $msg$ with a secret value $s$ and Chaum's blind signature [30] $BLI_r(msg)$ of $msg$ with a blinding factor $r$. The comma ',' denotes the concatenation between two components, the arrow $A \rightarrow B$ communication from $A$ to $B$ via any channel, and the double arrow $A \Rightarrow B$ via an anonymous channel.

### 4.6.3 Security requirements

We list the requirements that the Parliamentary Committee model has to satisfy and we describe the definition of each requirement.

**Post revealability** Votes can be revealed after voting by an authoritative order.

**Conditional privacy** All votes must be secret until officially disclosed, i.e., the secrecy of votes shall be guaranteed before the revelation.

**Universal verifiability** Every action by a voter, whether initialising a vote, actual voting or publishing a vote, is verified by the proof that the action is correctly completed; also, anyone can convince himself that the published final tally is computed fairly from the ballots that were correctly cast.

**Precise tally** All valid votes must be counted correctly; the number of invalid votes should be also notified at the end of the voting.

**Double voting protection** No voter can vote twice.

**Eligibility** No one who is not allowed to vote can take part in the ballot; in our scheme, we assume that members of the committee are registered by the administrator before voting, but any masquerading should be detected and denied.

**Fairness** Nothing must affect the voting; voting must present as accurately as possible the preference of the voters [41].

### 4.6.4 The ballot scheme

In the scheme, there are four principals: voter, administrator, counter and exposer. The voters $V_i$ are a set of principals able to cast a vote; one vote is admitted for each voter. The administrator A is a process that verifies eligibility of voters, makes a blind signature on the bit-commitment of the vote and publishes the blinded message for each voter. The counter C is a process that verifies the administrator's blind signature, verifies and counts votes, and publishes final totals. The exposer X is a process that registers key information to reveal votes, issues and publishes certificates for the registration needed in the next phase, and reveals the relationship between voters and their votes when it is requested.

We assume that all the cryptographic primitives are used in the conventional sense, that all the keys are sufficiently long, and that random values are sufficiently random and long.

Each principal in the scheme publishes lists; when a principal publishes a list, the whole list should be signed by the principal; otherwise it may be possible for attackers to manipulate it. To defend against replay attacks to later votes, we need to adopt temporary information such as nonces or timestamps; for brevity, we omit the description of this feature in each procedure and assume that nonces generated by principals are combined in the communications.

The Big Brother ballot scheme consists of six phases and they are about casting votes, preparing vote-escrow information, anonymising votes, issuing receipts, counting votes and revealing votes, sequentially. The flow of each phase is described as follows:

$\boxed{\textbf{First Phase}}$

1. $V_i$: creates her vote $\mathtt{ballot}_i$ and a random $s_i$
   creates a bit-commitment $COM_{s_i}(\mathtt{ballot}_i)$ and a random $r_i$
   creates a blinded message $M_i = BLI_{r_i}(COM_{s_i}(\mathtt{ballot}_i))$
   creates a bit-commitment $COM_{r_i}(K_{V_i+})$

2. $V_i \rightarrow A$: $V_i, \{M_i\}_{K_{V_i-}}, COM_{r_i}(K_{V_i+})$

3. $A$: verifies $V_i$ eligibility and $V_i$'s signature of $M_i$

4. $A \rightarrow V_i$: $\{M_i\}_{K_{A-}}$

5. $V_i$: unblinds with $r_i$ and verifies $\{COM_{s_i}(\mathtt{ballot}_i)\}_{K_{A-}}$
   If verification fails, claims by publishing $COM_{s_i}(\mathtt{ballot}_i)$
   and $A$'s response.

| | | |
|---|---|---|
| 6. | $A$: | publishes a first-phase-voters list of $m$ entries<br>such that each entry $i$ is $[V_i, \{M_i\}_{K_{V_i-}}, COM_{r_i}(K_{V_i+})]$ |
| 7. | $anyone$: | verifies $V_i$ eligibility and $V_i$'s signature of $M_i$ for each $V_i$ |

$\boxed{\textbf{Second Phase}}$

| | | |
|---|---|---|
| 8. | $V_i \rightarrow X$: | $\{V_i, \{r_i\}_{K_{V_i-}}\}_{K_{X+}}$ |
| 9. | $X$: | verifies $r_i$ by computing $K_{V_i+}$ from $COM_{r_i}(K_{V_i+})$<br>in the first-phase-votes list. |
| 10. | $X \rightarrow V_i$: | $cert_i = \{COM_{r_i}(V_i)\}_{K_{X-}}$ |
| 11. | $X$: | publishes a second-phase-certificates list of $n$ entries<br>such that each entry $i$ is $[i, cert_i]$ |
| 12. | $anyone$: | verifies $cert_i$ |

$\boxed{\textbf{Third Phase}}$

| | | |
|---|---|---|
| 13. | $V_i \Rightarrow C$: | $\{COM_{s_i}(ballot_i)\}_{K_{A-}}, cert_i$ |
| 14. | $C$: | verifies $A$'s signature of $COM_{s_i}(ballot_i)$ and $cert_i$<br>generates a sequence number $j$<br>publishes a third-phase-votes list of $p$ entries<br>such that each entry is $[j, \{COM_{s_j}(ballot_j)\}_{K_{A-}}, cert_j]$ |
| 15. | $V_i$: | verifies $\{COM_{s_i}(ballot_i)\}_{K_{A-}}$ in the published list |
| 16. | $anyone$: | verifies $A$'s signature of $COM_{s_j}(ballot_j)$ for each $j$ |

$\boxed{\textbf{Fourth Phase}}$

| | | |
|---|---|---|
| 17. | $V_i \Rightarrow C$: | $j, s_i$ |
| 18. | $C$: | publishes a final-receipts list of $q$ entries |

such that each entry is $[j, COM_{s_j}(\{j\}_{K_{C-}})]$

19. $V_i$: verifies $COM_{s_i}(\{i\}_{K_{C-}})$

**Fifth Phase**

20. C: obtains $ballot_i$ and verifies vote validity
publishes a final-votes list of $t$ entries and a result summary
such that each entry $j$ is $[j, \{COM_{s_j}(ballot_j)\}_{K_{A-}}, s_j, ballot_j, cert_j]$

21. $V_i$: verifies $ballot_i$ in the published list

22. $anyone$: verifies $COM_{s_j}(ballot_j)$ and $ballot_j$, for each $j$
verifies consistency of results summary and $q = t$

**Sixth Phase**

23. X: calculates $COM_{s_j}(ballot_j)$ for each $V_j$ from the first-phase-voters list
fetches $ballot_j$ in the final-votes list for $V_j$
publishes a voter-vote list such that each entry $j$ is $[V_j, r_j, ballot_j]$

24. $anyone$: verifies certificates $cert_j$ and voting details

The sixth phase is the revelation procedure which can be done with an authoritative order. If there is no need to reveal votes, we terminate the voting at the fifth phase.

### 4.6.5 Discussion

Our scheme satisfies the requirements mentioned in Section 4.6.3 and we discuss the scheme with respect to each requirement.

**Post revealability** This feature is guaranteed by the exposer; a voter $V_j$ registers a blinding factor $r_j$ with the exposer; nonetheless, the exposer cannot reveal the vote before the end of voting, since the vote is protected by the bit-commitment with a random number $s_j$; without the blinding factor registration, the voter cannot go to the third phase since the certificate from the exposer is required in the third phase; in order to issue a certificate for the submission of the blinding factor, the exposer verifies the blinding factor by calculating $COM_{r_j}(K_{V_j+})$ from the first-phase-voters list; upon demand, the exposer can reveal the relationship between voters and votes by referencing the first-phase-voters list and the final-votes list; the value $COM_{s_j}(\mathtt{ballot}_j)$ is used for validating $V_j$-entry in the final-votes list.

**Universal verifiability** For each phase, each principal, except voters, publishes information collected and anyone as well as the voters can verify the information at each phase; if a voter finds any incorrect information, she can prove where the failure happened.

**Precise tally** In the original FOO protocol, it has been assumed that all registered voters should follow all phases of the protocol, i.e., the number $m$ of published entries in the first-phase-voters list should be equal to the number $t$ of published entries in the final-votes list; we assume that any voter can give up voting at any phase, but it is then possible for the counter to make a forgery by announcing some number of votes between $p$ and $t$; in order to prevent such a forgery, we force the counter to publish $COM_{s_j}(\{j\}_{K_{C-}})$ for the received $s_j$'s; at the end of the fifth phase, anyone can verify $q = t$.

**Double voting protection** If anyone votes more than once, it is detected by the counter in the third phase and the multiple voting is denied; the administrator's blind signature is used for this purpose.

**Eligibility** The validity of voters can be checked by the nature of the digital signature; if a signature is incorrect, the false signer is not allowed to participate in voting; in the first and the second phase, the voter's signature is used for eligibility checking and in the third phase, the administrator's blind signature is used for the same purpose.

**Fairness** Since nothing is revealed before the counting of ballots, nothing can affect the voter's decision; also the counting cannot be affected by the previous phases, because it can be done after all the procedures in the previous phases are completed and the procedures do not reveal any information about ballots.

**Conditional privacy** Before the revelation, voting is kept secret, as is voters' privacy; if the sixth phase is not invoked, then the ballot remains secret.

## 4.7 Contribution 3: The Guy Fawkes protocol

We present an authentication scheme named the Guy Fawkes protocol [9] that uses hash operations, hash chains and publishing process. The publication process in the scheme plays an important role in authenticating that the signer claims.

### 4.7.1 Motivation

The Guy Fawkes protocol is a way of digital signature that requires only a small number of hash function computations. The underlying idea came to us on the 4th November 1996 while discussing how a modern day Guy Fawkes[7] could arrange publicity for his cause but without getting caught.

The naïve approach might be to telephone the newsroom of the Times and say *"I represent the free Jacobin army and we are going to carry out a liberation action tomorrow. Once we have done it, I will call using the code word 'Darnley' to state our demands"*.

This is not a particularly secure way of doing it. The message will be passed on to the police, who might remember that the state opening of Parliament is imminent and double the guard. So it would be more prudent to send a hash

[7]Fawkes conspired to blow up King James I and the Houses of Parliament in 1605; this was an attempt to end the persecution of Roman Catholics. Fawkes was caught as a result of a communication security failure (a coded letter from one of the conspirators was intercepted and deciphered). After his public execution, Parliament ordained the 5th of November as a day of thanksgiving for their narrow escape, and it is still celebrated by bonfires and fireworks.

of the message. Provided that the hash function is pseudorandom, this will not leak information. If Fawkes is now successful, he can reveal the message and find himself in possession of a very credible codeword. However, such a protocol is open to abuse by newspaper staff. So we replace the codeword with its hash.

### 4.7.2 The protocol

A basic protocol can be described as follows:

1. Select a random codeword $X$ and form its hash $Y = h(X)$
2. Construct a message $M$ = *"We are the free Jacobin army and we are going to blow up the Houses of Parliament tomorrow. The codeword by which we will authenticate ourselves afterwards will be the preimage of* $Y$*"*
3. Compute $Z = h(M)$ and publish it anonymously
4. Blow up the Houses of Parliament
5. Reveal $M$

However, the codeword $X$ can only be used once. Our innovation is to chain new codewords from old, so that Guy Fawkes can keep on identifying himself. Formally, we define this protocol by induction. Suppose that we have published $Z_i$ followed by the message $M_i$ containing $h(X_i)$, where our secret codeword is currently $X_i$. We wish to authenticate the message $M_{i+1}$. We follow the following protocol:

1. Select a random codeword $X_{i+1}$ and form its hash $h(X_{i+1})$
2. Compute $Z_{i+1} = h(M_{i+1}, h(X_{i+1}), X_i)$ and publish it
3. Reveal $M_{i+1}, h(X_{i+1})$ and $X_i$

The first codeword needs to be bootstrapped by some external mechanism; in most applications, this would be a conventional digital signature or an out-of-band authentication, perhaps using a conventional certification authority.

In the Guy Fawkes protocol, the goal is to associate a single act of authentication with a stream of future statements rather than a stream of future events. It would not be sufficient to simply use a hash chain as a set of different passwords for authenticating each political statement. Anyone who was tapping the line when the statement and its password were sent to the newsroom could alter the statement; staff in the newsroom could also substitute messages at will.

In other words, the broadcast commitment step has the critical effect of providing nonrepudiation, and gives the Guy Fawkes protocol the same effect as a digital signature. Were the Jacobins able to use asymmetric cryptography, then their first message could just as well have read *"We are going to blow up the Houses of Parliament on the 5th November. Future demands will be digitally signed and the public verification key is* W.*"* This could have been encrypted and published, with the key made known after the event.

### 4.7.3 Discussion

The authentication process of the Guy Fawkes protocol relies on published hash values. It assumes an anonymous publishing service that publishes the requested hash value transparently without exposing the publisher; this is an example of publishing requiring anonymity. If the hash value generated by the protocol initiator is published properly and is the same everywhere the initiator can prove what he claimed afterwards.

Unlike other signature schemes, the protocol is not self-contained without publishing channels, but if we can use such a publishing service, the protocol works efficiently with a small number of hash computations. In fact, we can find similar publishing services on the network, such as an advertisement section of online newspapers, newsgroups and closed user group mailing lists.

Since the protocol assumes a chronological ordering between the publication of the hash value of a statement and the event claimed in the statement, we need a means of ordering events such as a timestamping service or an ordering certifying service.

Another aspect of this protocol is verification timing. In conventional public key signature schemes, we can verify the signature of a document anytime; the time of verification is decided by the verifier. In contrast, the signer can decide the time of verification in the Guy Fawkes protocol.

The Guy Fawkes protocol may not be much of practical value compared to conventional digital signatures but there are some applications at which it excels. One of these, signing bidirectional video streams, is discussed in our paper [9]. Its value to this dissertation lies in illustrating an unexpected link between publishing and signature.

## 4.8 Contribution 4: The Global Trust Register

The Global Trust Register [4, 5] is a book containing the fingerprints of certified public keys and the names of key holders and other information associated with them. It thus implements a certification authority, but using the conventional route of printing on paper rather than electronic means. This project has been carried out with Anderson et al. between 1997 and 1998, and two editions of the register were published in 1998 and 1999.

Compiling this book was an interesting exercise in security engineering. It helped us clarify some technical and business issues associated with public key certification.

### 4.8.1 Motivation

When Diffie and Hellman published their seminal paper [40] on public key cryptography in 1976, their vision was that people would look up public keys in the phone book. Phone companies seized on this as an opportunity and developed this idea into a standard ITU-T X.500 [61] for a distributed electronic directory. The certificate format X.509 [60] developed for this directory is coming into increasingly widespread use, but the global database was never built and now will probably not be.

One reason is the diversity of telecommunications; while Britain had two phone companies in the 1970's, it now has over 200 public phone service providers, as well as many more network service providers and thousands of private corporate networks. Following this model, we find a growing number of local X.509 certification hierarchies in networks serving government, electronic document interchange and banking, as well as a number of specialist companies offering key certification services to the general public.

Yet these networks are still mostly isolated. Public certification authorities do not certify each others' keys and so there is little in the way of a general trust infrastructure available. Some certification authorities have managed to get their root keys bundled with common web browser software, but many certification authorities, especially outside the USA, have failed to persuade the software companies to include them. As a result, there is no cheap and effective way for Internet users to check the validity of public keys on which they may wish to rely. A US user, for example, cannot easily check the validity of a certificate from a server in Korea, if this is issued by a Korean certification authority whose root certificate is not included in the user's browser.

Many users have turned instead to PGP, which can be used without any preexisting infrastructure, or to proprietary products such as various EDI systems. This only adds to the complexity of key verification.

We therefore initiated a project in 1997 to solve this problem by making available, in a paper book, fingerprints of public keys together with the names and addresses of their owners. This book, which we named the Global Trust Register, contains the fingerprints of over 500 public keys and the information associated with those keys.

The process of compiling the keys in the Global Trust Register involved three steps: selecting which public keys to include, verifying that their owners were as stated in their key files and were in possession of the corresponding private key, and checking the associated information.

### 4.8.2 Conventional publishing of keys

Demonstrating the impracticality of government control of cryptography was one motive for publishing the register in the form of a book. It was not the only reason. The conventional book format publication provides other advantages.

The published information about keys is frozen. The nature of conventional publishing lets us achieve a higher integrity level than electronic publishing. The book becomes a persistence reference stored in libraries. Furthermore, it satisfies the other requirements of publishing defined in Section 3.6: public availability, strict revision history and strong evidence of authorship. Since both editions of the register are available in a dominant online book shop and the second edition is distributed by a world-class publisher, the distribution is highly tamper resistant.

Perhaps conventional publishing can help promote the use of electronic transactions with confidence; information about fragile electronic matter in a conventionally reliable container provides the conventional level of reliability to the electronic matter. The register presents a way to compensate the weaknesses of electronic publishing with the strengths of conventional publishing.

A weakness of book-format publishing of keys is timely revocation. Practically, key revocation can happen any time and the revocation has to be known to the reader of the book in time. There are several reasons to change informations published in the book such as movement, change of work and obsolete email address. For example, PGP certificates have no period of validity and so awkward structures such as 'year keys' have been invented. One potential problem is that publication dates of the Register might not always coincide with the time important year keys such as when CERT[8] keys are generated; for example, entries for the 1999 edition was closed at the end of August 1998 and it was published in March 1999. Book publishing and distribution cannot match the update speed of electronic media.

Today's real-world revocation services such as the one for credit cards have matured over many years. More and more critical certification authority prod-

[8]Computer Emergency Response Team; see `<http://www.cert.org>`

ucts provide functionality of online certificate status-check servers rather than that of the simple blacklists used by X.509 Certificate Revocation Lists [60]; yet online checks are expensive in terms of the peak communications capacity that must be provided, and a distributed blacklist has turned out to be economic with the world's largest existing revocation system, the list of lost and stolen credit cards. This is held in a distributed form with multiple levels of local stand-in processing. So we believe that well-engineered public key infrastructure services will end up combining these ideas.

However, although we were prepared to publish revocation information for the keys in the register on the web, there have been no revocations due to key compromise; there have been many address changes. This was contrary to general expectations, and may be useful to future infrastructure builders.

### 4.8.3 Confidentiality vs. integrity

As a side effect of versatile functionality of dominent public key algorithms, most users of both PGP and of X.509 products use the same key for encryption and signing although confidentiality keys often need to be managed in different ways from integrity keys. This may cause serious problems in countries which introduce key escrow or otherwise mandate government access to decryption key material; in such a case, escrowed encryption keys which are also signature keys can be used for generating fake signatures by a key escrow agency.

To avoid the problem, we found that some people have been using two PGP keys, one for confidentiality and the other integrity. It can be a warning for the key owner himself but cannot be a proper solution if the integrity key can be used for confidentiality by the nature of the underlying algorithm.

During the project, we found that some other public key based application vendors started to recognise the problem and provide different key pairs for encryption and signature.

## 4.9 Summary

In order to understand the practical aspects of publishing, we viewed a series of publishing applications including electronic commerce, voting, certification authority and medical application. As examples of publishing system building, we presented publishing-related mechanisms out of our reasearch including customer-centric payment, Big Brother ballot, the Guy Fawkes protocol and the Global Trust Register.

# Chapter 5

# Publishing policy and technical primitives

As we have seen, integrity and authenticity are the essential concerns for publishing and both properties satisfy the requirements which appeared in Section 3.6: integrity of content, evidence of authorship and revision history preservation. From an operational viewpoint, availability and persistent reference of publications are fundamental in a publishing system.

We will define a policy model that supports these requirements and present technical primitives that support operational interests as well as integrity and authenticity interests. The policy and the primitives construct a framework that makes web publishing more reliable.

## 5.1 Policy model for publishing

Based on the study of security properties and policies in Chapter 2 and the investigation of publishing in Chapter 3, we set up a policy model for reliable web publishing. The main concerns are integrity and authenticity control.

We do not include requirements for availability and persistence of publications in the policy. The mechanisms that support these properties will be presented in later sections.

As a foundation, we assume a partition for the domain $D$ of published documents: a set of controlled documents, $CD$, and a set of uncontrolled documents, $UD$; we define $UD = D - CD$ and by definition, $D = CD \cup UD$ and $CD \cap UD = \emptyset$. A member of $CD$ is controlled with respect to integrity and authenticity, and a member of $UD$ is a document that does not have any control. There is a one-way transformation from $UD$ to $CD$; if a document is once controlled, it should maintain its integrity and authenticity.

Now we define a policy for reliable web publishing. It consists of the following six principles:

## Principles

**Principle 1** Neither deletion nor replacement is allowed within $CD$.

**Principle 2** The creator of a document defines its revision access condition and only authorised principals with respect to the condition are allowed to revise it; all revisions in $CD$ must be stored and browsable.

**Principle 3** Authenticity validation procedures must be available for validating the authenticity of $CD$ members.

**Principle 4** Any action to $CD$ members must maintain the authenticity of the document.

**Principle 5** Authentication of a member of $CD$ can be performed by any user.

**Principle 6** Transformation from $UD$ to $CD$ must be one-way and the principal who transformed a document becomes the creator of the document in $CD$.

## 5.2 Analysis of the model

In brief, the policy model defines access control to delete, replace, revise and authenticate documents, in fact, deletion and replacement of documents are not allowed. A transformation map from the set of uncontrolled documents to the set of controlled ones is also defined.

The policy assumes two separate domains, CD and UD, which will provide a guideline of how to implement publishing systems with two different types of document access.

The policy protects controlled documents with the following principles:

Principle 1 provides a baseline for write access; there is no limit for the creation of new documents but any modification of existing publication is completely prohibited. When a write event occurs for controlled documents, it lasts as long as the system is alive and is never destroyed. If we need to modify a document, we make a revised copy of it; the original remains without any changes as it has been.

Principle 2 is a rule to manage revisions. The creator of a document defines its revision access condition such as 'no limit', 'only creator allowed' or 'edit allowed users list'. The authorisation decision to its revision is made with respect to this condition. All revisions are stored and accessible. The difference between two revisions can be extracted and is browsable.

Principles 3, 4 and 5 show rules for authentication; the authentication capability must be supported; actions which conflict or can break authenticity of published documents are not allowed; anyone can initiate the authentication procedure. Any publication can be authenticated by anybody and any action that can break its authenticity is not allowed.

Principle 6 assumes a transformation from UD to CD. The one-wayness of the transform means that a controlled document cannot become uncontrolled, since Principle 1 does not allow to either delete or replace CD; once a document is moved to CD, it remains there alone with all its revisions. Transforms in the reverse direction are not prohibited but they cannot affect the document already in CD. If a document is transformed to UD, then the later life of the document will not be controlled but the previous history of the document in CD is remained unchanged. Hence we may regard document movement as document copy; in fact, the document is remained unchanged in CD and a copy is newly created in UD from the original image in CD.

According to Principle 2, we have a creator of each document in CD. When we transform a document from UD to CD, we need a creator of the document. Principle 6 defines its creator by the initiator of the transformation.

An earlier version [10] of the publishing model was presented in the Security Protocols Workshop '99 and it only assumed append-only file system without an explicit policy. In the context of the policy in Section 5.1, the former model assumed that all the published documents are of CD. Authenticity-relevant principles were not explicitly specified either, in contrast, we now have detailed rules such as Principles 2, 3, 4 and 5. Since all documents are of CD in the former model, the transformation between CD and UD specified in Principle 6 was not necessary.

## 5.3 Append-only file system

Append-only file systems are rarely considered in the research literature, but are remarkably common in real life. Banks must keep records for a period of time set down by law (e.g., six years in the UK), and much the same applies to medical records, though the period for which a record must be kept depends on the condition to which it relates – cancer records, for example, are kept for the patient's lifetime [8]. Business records in general are kept for the period during which transactions can be challenged by customers or audited by tax inspectors.

Publishing is perhaps one of the oldest applications of append-only file systems. We do not expect that the archives of a newspaper or journal will ever have their content changed except in the shrinking number of totalitarian states. If an issue of a publication is found to be defamatory or in breach of copyright, the appropriate legal remedy is financial compensation rather than retrospective modification.

Similarly, in commerce, the need for finality of transactions dictates that if a bank transaction is processed in error, it is not retrospectively deleted but rather a correcting transaction is passed through the system. In the days when bank ledgers were kept on paper, they were kept in ink using bound books.

There are many mechanisms which can be used to implement an append-only file store, ranging from CD-WORM[1] drives through revision control mechanisms, which we use in our implementation, to intensive backup systems that make copies for permanent storage. In some applications, such as retail banking, the volume of transactions is large and thus the disk capacity required is fairly predictable; in others, such as medical records, the quantity of data which a doctor can type into his personal computer is negligible compared with the size of even the smallest hard disk. There might be some risk of denial-of-service attacks involving resource exhaustion, and so one might either use disk quota limits or insist on micropayments for disk writes. The whole point of frameworks like ours is that failure of the underlying server causes only service denial, not failure of integrity or authenticity.

The academic literature on append-only file systems, however, appears to be rather sparse. The earliest mention of which we are aware is the use of a public bulletin board by Benaloh [18] to ensure the serialisation of pseudonymous votes cast in a digital election. More generally, log-structured file systems [96, 97] improve the performance of some systems by ensuring faster writes at the expense of longer read times. In systems where erasure is expensive, such as flash memory, they allow an even distribution of writes in the media [64]. We might also mention the Eternity Service [7], a distributed file store in which techniques of fragmentation, redundancy, scattering and anonymity are used to provide the highest possible level of protection against denial of service attacks: in effect, an append-only file store which is highly resistant to deletion attempts. Append-only file storage satisfies Principle 1 in our policy; no document once published can be deleted or replaced.

[1]CD-WORM stands for Compact Disk – Write Once Read Multiple devices; CD-Recordable is of this kind.

Revision management is an important part of the append-only file system because it reduces the size of stored data and improves search performance. Another clear advantage is that a revision control system is able to show the difference between revisions easily. When we add revisions in a log-structured manner, we only store the updated section for each revision and some control data to restore the document. Since the introduction of the log-structured file system, there has been a series of revision management mechanisms such as SCCS,[2] RCS[3] and CVS,[4] and they can be adopted for use within an implementation of our publishing policy.

A good example application of an append-only file system is the B-money system proposed by Wei Dai [37]. It assumes that each user maintains a separate account of how much money everyone in the system owns, and in order to make a payment, one simply broadcasts a signed message saying 'I, Alice, hereby pay the sum of X to Bob'. B-money assumes a synchronous, unjammable broadcast channel, but can be implemented in our model by giving each principal a document to which they append their transactions; in fact, such a document works like a digital signature device. Secure serialisation of events is required but can be implemented in the canonical manner by using the Lamport clock.[5]

B-money has some interesting properties. For example, if someone computes the account balances of other parties incorrectly, it is himself that he puts at risk; he might accept a payment where the payer had insufficient funds, and then be embarrassed when one of his own payments was rejected by a more careful principal on the grounds that he in turn could not cover the payment.

This is a surprisingly apt model of a payment system: in the banking world,

[2]Source Code Control System by Sun Microsystems.

[3]Resource Control System by Tichy [107].

[4]Concurrent Version System by Cederqvist et al. [27].

[5]The Lamport clock [66] is a global ordering mechanism based on the assumption that the clocks are a collection of clocks synchronised by the rules: (i) two different events at the same process happen at different times on the process's clock, (ii) if Alice sends a message to Bob, then time on Alice's clock when the message is sent is less than the time on Bob's clock when the message is received. This ordering is not applicable for events not causally related although the clock may give a global ordering.

journal files are used to accumulate transactions from cash machines, cheque sorters, teller stations, etc., and these are applied overnight to the account master file in order to update it. The journals fill the same role as the transactions in B-money, and the overnight run is an optimisation which simplifies processing. To the extent that a bank's functions are mechanical rather than to do with credit assessment and risk management, it might be replaced by a B-money system plus a software agent that performs the overnight update.

## 5.4 Security markup language

There have been trials of markup languages for certain security requirements such as Wax [13] and the Eternal Resource Locator (ERL) [12]. Wax is a proprietary hypertext-based system used for medical information such as treatment protocols, drug formularies and teaching material. ERL uses embedded hash values in HTML. It is designed to support reliable electronic distribution of books on which doctors depend when making diagnostic and treatment decisions.

There is another approach to providing a digital signature-bearing publication by using SGML, a more versatile but more complicated hypertext markup language than HTML. In 1998, Kravitz [65] of IBM proposed the Signed Document Markup Language (SDML) to achieve non-repudiation of origin and integrity in the document interchange using public key signature algorithms. The specification of SDML has been submitted to the World Wide Web Consortium as a W3C note. The details of the above studies will be described in Section 6.1.

These studies show the usefulness of a security markup language. Since HTML has been widely used and web publishing has become common, the importance of markup languages becomes clear. A flexible markup language can play an important role in supporting various types of documents. For example, government documents may have paragraph-wise security labels. That is, a document is a list of paragraphs with security labels such as unclassified, classified, secret, and top-secret. To support this, we need a paragraph-wise document management mechanism. In Snook's DODA [102], she tried to accommodate this by defining a functional object unit which becomes a building block of a document. Now we have hypertext-enabled markup languages, we can achieve object-wise security label handling more flexibly and more efficiently. An implementation will be given in Section 6.4.

As a versatile and flexible markup language, XML is notable. The positioning of XML between SGML and HTML is well-defined. XML is designed to be simpler than SGML by removing syntactical ambiguity and complicated functions, and provides a greater degree of freedom in extension than HTML. So major software vendors support XML in their latest software releases and some of them are going to adopt XML as the skeleton of their document processing applications.

Thus we will use XML to construct a security markup language to define the functionality and document handling structure in our implementation of the policy.

## 5.5 Repository clustering

There is no system that never stops. If we use a highly redundant system such as Tandem computers,[6] we may experience few system failures, but the high cost would be a burden for web publishing. However, publishers want to provide their service 24 hours a day with minimum down time. We consider how to satisfy an average web publisher's demands.

In their book about fault tolerance [14, pp. 72–77], Anderson and Lee overview two core metrics within fault tolerance: Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR).

According to these metrics, our aim is to construct a system that provides longer MTBF and shorter MTTR. We have seen some successful fault tolerant features in specific applications such as modular duplication in telephone

[6]An American computer vendor that provides fault-tolerant systems and is a part of Compaq Computer Corporation; Tandem is a trademark of Compaq.

switching systems, triple modular redundancy with voting circuits like in Tandem computers, and hot-switchable modules that can be replaced without shutting the whole system down. Some of these technologies are adopted in server-range personal computer systems. As a result, the MTBF of ordinary systems has become longer and such systems are available for web publishing service providers.

From the investigation in Section 3.3, we realised that wide distribution of publications helps them last. A group of document repositories can be a solution. When we distribute data to multiple repositories, recovery procedures should be considered.

As a simplest way to achieve a wide distribution mechanism with longer MTBF and shorter MTTR, we may think about full image mirroring as PGP key servers do. It is easy to configure but expensive. If the synchronisation period between repositories is short, it may impose a large communication burden; but the longer the period is, the more failures of recovery we may face. The whole cluster may fail if the full image contains faults. So the elimination of faults is not easy.

We want something similar to RAID[7] but which implements backup rather than replication, in a transparent, efficient and scalable way. RAID is not enough to provide persistence; for example, `'rm *'` does damage of a kind that disk mirroring cannot repair. Clustering append-only file repositories can be a way coping with such errors and with damage caused by malicious software.

We propose a mechanism that provides adjustable storage efficiency and recovery performance. Suppose we are in a repository $R_0$ and there are $n$ more repositories in a cluster. First, we choose an integer $k \geq 2$ such that we can construct a subcluster of $k$ repositories. Let $m$ be the maximum integer satisfying $k \cdot m \leq n$. We construct $l$ subclusters that each subcluster has $k$ repositories, where $1 \leq l \leq m$. Each repository provides dedicated storage partitions for other repositories in the cluster. Now we describe a distribution and recovery

[7]Redundant Array of Independent Disks [72]; RAID 5 inspired to this clustering scheme; see `<http://www.raid-advisory.com>`.

scheme in a subcluster, since we apply the same scheme to all the subclusters.

There is a list of documents published in $R_0$ and we divide the list into segments $\{s_i\}_i$ of a fixed size; in the current setting, the size of a segment is 100K bytes. We need to distribute these document segments to other repositories for backup and recovery. To assign segments to $k$ repositories in a subcluster, we take $k-1$ segments $\{s_i\}_{1 \le i \le k-1}$ sequentially from the list for each assignment. We then assign $\oplus_{1 \le i \le k-1} s_i$ to $R_k$ and $s_i$ to $R_i$ for $1 \le i \le k-1$. Note that the symbol $\oplus$ means exclusive OR and the value $\oplus_{1 \le i \le k-1} s_i$, which we call parity, is used for recovery when one of $\{s_i\}_{1 \le i \le k-1}$ is faulty; if more than one of them are faulty, the parity is of no use and the recovery will be completed with another subcluster. For next $k-1$ segments, we assign $\oplus_{1 \le i \le k-1} s_i$ to $R_{k-1}$, $\{s_i\}_{1 \le i \le k-2}$ to $R_i$, and $s_{k-1}$ to $R_k$. Document distribution and storage are carried out in this procedure, repeatedly. We apply this scheme to all subclusters in the cluster. An example description of the distribution scheme is shown in Figure 5.1.

| Repositories | $R_1$ | $R_2$ | $R_3$ | $R_4$ |
|---|---|---|---|---|
| Documents | $s_1$ | $s_2$ | $s_3$ | $p_1$ |
| | $s_4$ | $s_5$ | $p_2$ | $s_6$ |
| | $s_7$ | $p_3$ | $s_8$ | $s_9$ |
| | $p_4$ | $s_{10}$ | $s_{11}$ | $s_{12}$ |
| | $s_{13}$ | $s_{14}$ | $s_{15}$ | $p_5$ |
| | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

**Figure 5.1. An example of the document distribution scheme:** this example demonstrates the way to distribute document segments from $R_0$ to members of the subcluster $R_1$, $R_2$, $R_3$ and $R_4$, where $k = 4$ and parity $p_i = \oplus_{i(k-1)-i+2 \le j \le i(k-1)} s_j$. Parity values are distributed throughout the subcluster as well as document segments. We keep recovery information, such as document length and identifiers of subclusters, corresponding segments and their parities. When a document fails, we can recover it by using its parity and corresponding segments.

If a repository in a subcluster fails, we can recover it with information inside the subcluster. If we have a failure of more than one repository, we need to recover it by using information stored in another subcluster; in this respect, the number of subclusters is important.

In the choice of $l$ and $k$, there is a trade-off between storage efficiency and recovery performance. If we increase the number $s$ of subclusters, then we can have more copies of a document distributed, but should use more resources for redundancy. If we increase the number $k$ of members in a subcluster, we can save more resources, but should accept a higher failure rate which cannot be recovered inside the subcluster.

## 5.6 Summary

In order to build a publishing framework to make web publishing more reliable, we defined a publishing policy and described supporting technical primitives including append-only file system, security markup language and repository clustering. An implementation of this framework will be presented in Chapter 6.

# Chapter 6

# The Jikzi publishing system

We now describe a prototype implementation, which we call Jikzi,[1] after the first ever book published using a moveable type printing press; this is a Buddhist text printed in Korea in 1377, some 63 years before the Gutenberg Bible.

## 6.1 Background

There are a couple of relevant works which have been carried out in the last two years: Wax and ERL. The Wax system [13] was developed to secure the online publication of medical information such as drug data. The main problems encountered were the performance penalty imposed by trying to verify the signature on a whole book when the reader simply wished to consult a sin-

[1]*Jikzi* is the name of the oldest publication by using moveable-type printing press ever found. It has a very long official name "Baek-Un-Hwa-Sang-Cho-Rok-Bul-Jo-*Jik-Zi*-Sim-Che-Yo-Jeol" and simply we call it Jikzi. This is a text on Buddhism and was written by a monk "Kyoung-Han", and published in July 1377 by his pupils. The book was printed in Korea and now there is only one copy remaining. The copy had been stored in a Korean temple until the early twentieth century, when it was moved to France by a French diplomat. Now the copy is stored in the Bibliothque Nationale de France (the French National Library) in Paris. From a historical point of view, this publication is 63 years earlier than the Gutenberg Bible. In an exhibition room of the British Library at St. Pancras, London, a presentation of Jikzi is found with the description of the Gutenberg Bible.

gle section and the difficulty of securing long-lived data such as books using relatively short-lived objects such as public keys. Wax solved these problems by associating each publisher with a tree of hashes: the leaves were hashes of the individual sections, and the nodes progressed up through chapters and books until the root, which protects a publisher's whole catalogue, is authenticated by other means, typically a one-time signature. Thus when a user opens a book at the entry corresponding to a particular drug, the entry can be verified quickly without having to check a signature on the entire book. The lesson learned from Wax was that it is a sound design principle in publishing to use hash trees and to minimise the use of signatures — preferably to a single signature on each new version of a publisher's catalogue.

The Wax design has had some success in the UK medical informatics field, but it uses a proprietary hypertext format, so the next step was to generalise it to work with HTML. The result was the ERL, or eternal resource locator [12], which can be thought of as a URL plus a hash of what you expect to find there. The author of a web page or other document can authenticate any digital object by including its ERL, and this enables the construction of webs of trust of the kind familiar from PGP but applying quite generally to objects rather than just to the names of principals. Many of the problems associated with public key infrastructures disappear; one ends up managing a few root keys and doing a lot of version control.

As discussed about security markup languages in Section 5.4, it has now become clear that the adoption of XML will make our work easier and this is the emerging framework for commercial markup languages [23]. One of the goals of the Jikzi project is thus to develop a general XML-based mechanism for embedding security properties into hypertext documents.

This brings us to other work within the same field. We mention two particular projects: Kravitz's Signed Document Markup Language [65] which defines the tags necessary to implement electronic cheques and the World Wide Web Consortium's DSig project [31] to implement digital signature technology in content marking and rating schemes. While both of these projects are useful and instructive, we need something more general; we want to deal not just with cheques but with all other kinds of bills of exchange such as bills of lading and we want to be able to deal with their inheritance properties, for example, a bank cheque is a kind of cheque, and a cheque is a kind of bill of exchange.

Governments are likely to remain wedded to the idea of having paragraph-level security labels, i.e., a document is a mixture of paragraphs at different levels such as unclassified and secret, and XML appears to be an efficient way in which the relevant document management routines can be implemented without rewriting a huge amount of software.

We also ought to be able to deal with publishing in the most general sense, which means dealing with issues of replication, persistence and control. Previous work in this area includes Snook's DODA which is aimed at computer supported cooperative work, such as software development. DODA is somewhat like a secure revision control system in which 'folios' of documents are bundled together and managed by hash and signature mechanisms according to a policy such as 'a software upgrade can only be issued when both a testing manager and a customer manager sign it'; such policies can be enforced using mechanisms such as threshold signature.

The Jikzi system was introduced as a framework for web publishing in the Security Protocols Workshop in 1999 [10] and an extension of the paper will appear in [11]. In the following sections, the details of the system will be described and some implementation issues will be discussed.

## 6.2 Goal of the system

The goal of the Jikzi publishing system is implementing the reliable publishing framework presented in Chapter 5, i.e., the Jikzi system is designed to obtain more reliable web publishing by supporting our publishing policy defined in Section 5.1 and implementing conceptual mechanisms presented in Chapter 5. It implements not only the fundamental requirements of our policy but also some useful applications to show the flexibility of the system and to extend its functionality.

In the following system description, we present the system architecture and the Jikzi markup language for secure markup. As application-level services of the system, we present some service descriptions, including a publishing service implementing our policy, a web publishing witness service, a secure access to timestamping services, an event ordering service and a global key service.

All information in the Jikzi system is basically stored and distributed in the form of documents. A document is an atomic unit of publication; it may contain metadata such as control scripts for handling or only raw data such as texts or images. A user is a person or a process accessing published documents in the system. As we defined in our policy, the Jikzi system has two types of document domains: one for controlled documents and the other for uncontrolled documents.

For controlled documents, we adopt the append-only file system described in Section 5.3. As a result, all controlled documents published in the system are basically protected from any deletion and replacement; the integrity of documents is a basic requirement. Since the underlying document storage is append-only, all the traces of writing action remain within the storage. The revision control mechanism helps maintain the history of publications and saves the storage usage. Authenticity of documents is another requirement; the author of a document is authenticated by a challenge-response procedure and its verifiable hash value is published.

On the top of the append-only file system, we build a set of security markup definitions that provides means of authentication, integrity validation, encryption and decryption for document publishing and browsing. Depending on applications, we can apply corresponding security markup. For example, electronic cheques do not need confidentiality, but double spending or duplicated use is a problem. Secret electronic mails in a firm need confidentiality. Whether the document is encrypted or not, the underlying system stores it under the publishing policy. Although confidentiality protection is not a concern in the publishing framework, users can use the security markup to support labelling, encryption or mail filtering if they do wish. Users of the system can choose appropriate markup, or build one which fits for their purpose by using primitive document type definitions. The addition of user-defined markup can be easily done and this makes the system more versatile.

In order to satisfy various demands on publishing and maintaining documents, we define various document types and primitives to be used in the type definition as well as primitive document types. Each document or service then refers to these type definitions. The set of type definitions includes primitive cryptographic means which can be used in applications that need authenticity, integrity or confidentiality. The primitive document types include actions for authentication, integrity validation, hashing, encryption and decryption.

## 6.3 Architecture

In the Jikzi system, there are two types of document domains as defined in our publishing policy: append-only and replaceable. The append-only document domain forces users not to delete or replace published documents and it is used for controlled documents. The replaceable document domain is an ordinary file system and users can store, delete and modify their documents in this domain; this is a domain for uncontrolled documents. In both domains, authentication procedures are mandatory; if we want to change or revise a document, we have to be authenticated.

We describe the architecture for the append-only document domain because the replaceable document domain is no different from a normal UNIX file system. The Jikzi system for controlled documents consists of three layers: the fundamental layer, application layer and interface layer. The system architecture is figured in Figure 6.1.

The fundamental layer provides basic features required by the publishing policy and it includes the following function blocks: Jikzi preprocessor, version manager, storage manager and storage.

The Jikzi preprocessor consists of three parts: type checker, Jikzi parser, and primitive function block. The input to the Jikzi preprocessor is written in the Jikzi markup language and standard XML, it is tested, manipulated, and then stored in an internal format called XML Primitive Format (XPF) which is an

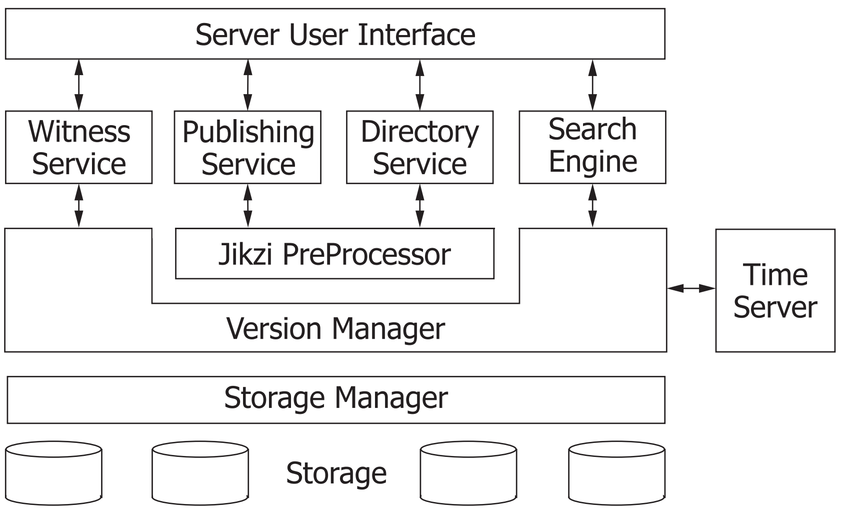


**Figure 6.1. The architecture for controlled documents in the Jikzi system:** the system consists of three layers: fundamental, application and interface layers. The fundamental layer includes the Jikzi preprocessor, version manager, storage manager and storage; the application layer includes system services such as the publishing service, witness service, directory service and search engine; the interface layer consists of the server user interface block. All user enquiries and responses to them are processed through the server user interface and transferred to services such as publishing, directory, witness services and search engine. One can use the Jikzi preprocessor to check the well-formedness of documents, or access the version manager directly. All enquiries are then controlled by the version manager and storage manager. The time server provides a consistent time; it can be either implemented in the system or an independent entity outside of the system.

extension of XML. The Jikzi system-specific actions are written in XPF. The type checker verifies whether submitted documents are well-formed. The Jikzi parser catches Jikzi specific tags and embeds the required primitive functions specified into documents. The primitive function block is a library of functions used in the system that includes algorithm for hash, digital signature, encryption and decryption. The structure of the Jikzi preprocessor is shown in Figure 6.2.

The version manager acts similarly to a revision control systems such as CVS [27]

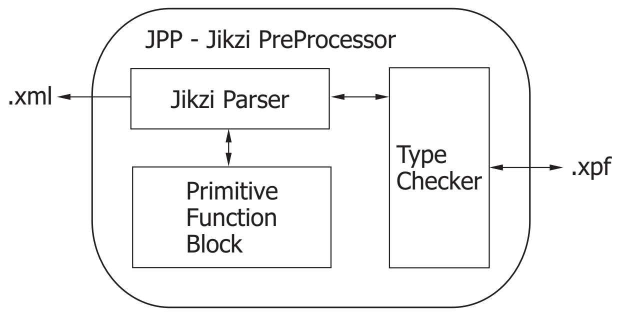


**Figure 6.2. The Jikzi preprocessor:** it processes documents written by users into well-formed documents in XML. It interacts with applications and the version manager, and consists of three subblocks: type checker, Jikzi parser and primitive function block.

and RCS [107]; it controls the version of all controlled documents and keeps traces of their update history. Documents are stored in a log-structured manner for efficiency and each revision is browsable through the server user interface.

The storage manager is the front-end to the storage and controls physical access to the storage. To implement an append-only storage, one way is to adopt writable CD-ROMs, so-called CD-Recordable, as the storage's physical media. We can write data on a writable CD-ROM only once and it lasts; cumulative writing is also possible on it. The writable CD-ROM is regarded as a usual disk partition by the operating system. The storage manager reads and writes documents on the media, and tests the integrity of the media.

The application layer provides additional services on the top of the fundamental layer. Current services include a publishing service, witness service, directory service and search engine. The publishing service allows users to publish their documents on the system. The witness service provides a signed archive of documents or web pages that have appeared on the specified web sites, and the archive can be used as an evidence for the fact that such documents were published. The directory service provides directory access to published documents for users to browse, download and verify; for example, signature verification of published documents or the result of witness can be done in this service. The search engine provides a means of searching and indexing requested publications in the system. The details of the publishing service and witness service will be given in Section 6.5

The interface layer provides an interface between users and the system services; it consists of the server user interface block. The server user interface is of web interface format so that users can access the system using their web browsers, and it provides user interfaces for the services in the application layer. Screenshots of the server user interface appear in Appendix C.

The time server enables users to affix timestamps to publications. The system can use either an external time service like the Network Time Protocol server [75] or an internal system clock. At present, we use the latter.

## 6.4 The Jikzi markup language

The Jikzi system uses its own markup language JML, or Jikzi Markup Language, an extension of XML. As we discussed in Section 5.4, XML provides flexible extensibility and is recognised by major browsers. We regard JML as a cumulative markup library; users can add their markups and share them on the server. To show how it works and to provide basic security primitives, we define some basic markups in terms of document type definitions (DTDs).

Document type definitions and application-specific document types mentioned here appear in Appendixes; document type definitions are attached in Appendix A and their examples appear in Appendix B. In order to support security capabilities, we define primitive document types: `hashList.dtd`, `signList.dtd` and `cryptList.dtd`. These document type definitions are for document hashing, signing, and encryption, respectively; and they will be used in application-specific DTDs mentioned below.

For a clean and simple description, we define a set of entities which will be frequently used in other DTDs in `stdDef.dtd`; it contains fundamental declarations and macro entities which can be included in other DTDs. All DTDs used in a document should be declared inside the document. Since we can have multiple versions of a DTD under the same name, we must specify which version of the DTD is used; otherwise, there is a possibility of an attack such as rollback of a revision of a DTD.

Using the primitive document type definitions mentioned above, we can generate DTDs for sophisticated composite documents including ERL-type documents, paragraph-wise manageable documents, electronic cheques, digital certificates and encrypted documents.

For instance, `erlDoc.dtd` supports the hash of document content and its chaining, and an example `examResult.xml` appears in Appendix B as well as other examples. A type definition `govDoc.dtd` supports paragraph-wise manageable documents usually used for tight security controls used in military applications, and basic information for official documents is defined in the DTD such as authors' names, document number, affiliation of authors, classification code for the document and paragraphs, creation date, expiry date and timestamp. An example for it is `draft.xml`. An electronic cheque is supported by `eCheque.dtd`, and the definition includes information about cheque number, account number, payee, payer(s), payment amount, currency, issue date and timestamp. Its example is `corpCheque.xml`. An X.509-type digital certificate is supported by `certificate.dtd`, and the definition includes information about certificate type, version, serial number, issuer, user, valid date and the user's key. Its example is `x509Cert.xml`. Document encryption is supported by `encDoc.dtd` and an example for it is `encMemo.xml`.

## 6.5 Application-level services

We will present a series of applications that can make effective use of the Jikzi system. These applications appear just below the server user interface in Figure 6.1. They include a publishing service, witness service, secure access to timestamping services, Lamport event ordering service and global key service.

### 6.5.1 Publishing service

The publishing service is the most fundamental service of the Jikzi publishing system that satisfies our publishing policy. It processes users' publication requests by interaction with the Jikzi preprocessor and the version manager. When incorporating a document to published, the service challenges the publication enquirer along with the authentication procedure. The enquirer chooses the domain to which the document will be belong, either controlled or uncontrolled. If the document is controlled, its well-formedness is verified when necessary and eventually, it is stored in the append-only file storage. If it is uncontrolled, it is just stored in replaceable file storage. Browsing the published documents can be done with the user interface of the directory service or the version manager.

This service is designed to satisfy the requirements of publishing presented in Section 3.6. Integrity of content is guaranteed for controlled documents by the append-only file system of the Jikzi system. Once a document is published as a controlled document, it lasts in the system.

Persistent reference is achieved combining the repository clustering mechanism, the append-only file system and the Jikzi markup language. The first two mechanisms help implement long-lasting publishing and the Jikzi markup language provides a system-independent browsing environment. Since we are publishing on the web, the conventional definition of URI can provide a unique reference identifier. Since the Jikzi system is open for the public to publish their documents, it is available publicly by its nature. The clustering mechanism makes it more available. Wide distribution of publications is given by the benefit of using the Internet and the distribution in the clustering structure helps propagation of publications.

Revision history of controlled documents is maintained by the version manager in the system. A browsing facility for the revision history is also provided. Evidence of authorship is verified when a controlled document is submitted to the system by the authentication verification procedure defined in the system. Any user can invoke an authenticity verification procedure in the directory service.

Although copyright protection is an important issue so that we take it into account as a requirement of publishing, this issue has been intensively studied and is far from our major concern. We do not provide related features in the system.

### 6.5.2 Witness service

As discussed in Section 4.1, we have many Internet frauds where items bought are never delivered or are not the same as advertised. If we have a frozen copy of the merchant's offer and the ordered item's specification, it may help us prove that we made a transaction and the transaction failed. To obtain such evidence, we need a mechanism to freeze the current state of the web publication. The witness service we present here is an answer to the problem.

If we use clustered repositories that provide the witness service, we can get multiple witnesses for a web publication on the Internet; we expect that it could have a similar effect to having multiple human witnesses in real life from the legal effect viewpoint. If we assume that there are several clusters of witness servers operated by different service providers, users can choose any of them as they wish. It is also similar to the choice of a solicitor on the high street.

When one has witnesses from different service providers, it could be strong evidence of what one had seen. If the publication is modified later without notice, one can show the previous state of the publication with confidence. The witness in this context is not an authorised server licensed by some authority like the government, but it may be enough for many purposes.

There are two types of witness services: active and passive. The active service is for CGI[2]-like program-driven procedures and the passive one for usual static HTML pages. Usually payment transactions or shopping procedures are performed using a CGI form and such procedures can be caught up by the active witness service. All other publishing written in HTML or XML containing

[2]Common Gateway Interface, a way of building an interface between the web page and an application running on the server.

texts and multimedia elements can be witnessed by the passive service.

The scenario for the service is as follows: a user requests the service by submitting the URL of the web page to be witnessed and his electronic mail address for a reply, then the server fetches the page and its relevant files, builds an archive for the page, and makes the seal for the archive. The seal contains a timestamp, the identity of the witness server, information regarding the domain name server used, IP address of the target server, the hash[3] value for each file in the archive and some request-specific information including the electronic mail address of the user and target URL. The signed archive for the requested web page can be downloaded from the system through the directory service and is stored in the append-only file storage.

The archived page can be directly displayed by web browsers since the archive contains files linked to the target URL which are one link deep, such as image files, icons and sound files. From the transaction log included in the archive, users can see the information processed by the target web server such as a language encoding type for each page and errors when extracting.

This is a means to improve the reliability of web publishing. The service can help reduce fraudulent transactions and offensive challenges by using web publications.

### 6.5.3 Distributed trust-type timestamping service

In their early paper about digital timestamping [50], Haber and Stornetta suggest two types of timestamping services. One is to constrain a centralised third party timestamping service with a hash chain of timestamps and the other is to distribute the required trust among the users of the service. The former is implemented and commercially available from Surety.com.[4] The timestamping server signs on the hash given by the request with the time and makes a hash chain of timestamping requests.

[3]In the implementation, SHA-1 [80] is used.

[4]A digital timestamping company founded by the authors of the paper [50]; it provides a service named Digital Notary; `<http://www.surety.com>`

The latter is not implemented and requires a pool of signers who can be chosen by a user; the signers in the pool should sign on the requested data with the time and return it to the user. This scheme also requires a public directory to verify timestamps by the user or challengers.

We apply this scheme to a cluster of Jikzi systems. The cluster can satisfy both requirements of the scheme: servers in the cluster can be signers and a public directory service is already built in. Furthermore, the append-only file system can make the public directory more reliable.

An implementation of this idea is as follows. Let $Y$ be the received hash value of a user's document to be timestamped and $g$ be a pseudorandom generator. Then $J := g(Y)$ is a string of numbers, where the symbol := denotes an assignment, and we divide it by $k$ segments such that $J = j_1 j_2 \dots j_k$. Let $m$ be the number of Jikzi systems $\{S_i\}_{1 \le i \le m}$ in the cluster and choose $k$ systems out of $m$ systems such that $C_i := S_{j_n \pmod m}$. The user sends her request $(Y, J)$ to chosen $\{C_i\}_{1 \le i \le k}$ and she receives in return from each server $C_i$ a signed message $\sigma_i := sign(t, Y, J)$ that includes the time $t$. Her timestamp consists of $[(Y, J), (\sigma_1, \dots, \sigma_k)]$ and is published on a Jikzi system. The $k$ signatures $\{\sigma_i\}_{1 \le i \le k}$ can be easily verified by the user or a would-be challenger.

The success of this scheme depends on the number of systems in a cluster. When the number is big enough, the collusion of all the involved systems is practically infeasible. If the number is small, it may be possible for the involved systems to collude to make a fake timestamp.

### 6.5.4 Lamport event ordering service

Consider a cluster of instances of the Jikzi system. Each system has its own clock and they can exchange a message in a predefined form. Based on the Lamport clock described in Section 5.3, we set up a service named 'Lamport event ordering service'. The service provides an order for document processing such as document submission, revision and signing. It does not provide exact global timestamps but the chronological order of events. As in the Lamport clock [66], we do not assume the existence of a precise global clock and set up an order based on communication between instances of the Jikzi system.

The fundamental idea of the event ordering service is as follows. Each event has a timestamp issued by the system where the event occurred; periodically, each system sends a signed message to all other systems in the cluster. The periodic message transmission is a way of proving causality between systems.

Consider an event $a$ which occurred in a system $A$. The system $A$ sent a signed periodic message to system $B$ after the event $a$ and then an event $b$ occurred in the system $B$. We can then agree that the event $b$ occurred after the event $a$. As with the original Lamport clock, this service also inherits the feature that we cannot say anything about the timing of events which occurred in system $B$ before receiving the periodic message. The message transmission period is the resolution of the event ordering service.

An attacker might block or delay the periodic message communication between systems so that the ordering precision may be degraded. This attack against the communication is a type of denial of service attack. We assume that the attacker cannot prevent a significant number of systems from communicating each other, and we can obtain an acceptable resolution of the ordering. Since the network is not deterministic and the attacker cannot make a perfect attack on all systems if the size of the cluster is big enough, the Lamport event ordering service can be a practical answer to ordering events.

### 6.5.5 Global key service

As an application of the Jikzi system, we design a key certificate service based on the concept of the Global Trust Register described in Section 4.8.

We cannot place the entire key certifying process of the Global Trust Register online since we need a face-to-face contact for A-rated keys under the rating rule of the register, but the basic PGP key verification process which checks the ownership of candidate keys can be done online. Even though we publish only verified keys online, we can set up a more reliable key server than the

MIT PGP public key server.[5]

Through experience gained during the Global Trust Register project, we noticed some security problems in the MIT PGP key server. In fact, there is no key certifying procedure in the MIT server so that anyone can register a key which is generated with other's name. There is no mechanism to verify the relationship between the key and the identity on the key, a string usually electronic mail address of the key holder. Another problem is that it is possible to add signatures to keys in the server without any notice to the key holder. Unless we have already noticed that it cannot be prohibited in the PGP key server, it can mislead the trust relationship between the key holder and the signer.

Since PGP has been one of the most widely used public key infrastructures, it is sensible to have a PGP public key server with proper certifying procedures. The Global Trust Register has been built on a tight key-owner relationship certifying procedure. When the register is implemented in the domain of controlled documents of the Jikzi server, we can guarantee its integrity as well as authenticity of key holders given by the key certifying rule of the register; it can be a trustworthy alternative to the MIT server.

## 6.6 Summary

We built a publishing system named Jikzi that implements the publishing framework presented in Chapter 5. We reviewed related work and described the goal of the system. The system architecture and technology such as the the Jikzi markup language are also described. Finally, we presented application-level services on the system including the fundamental publishing service, a witness service, secure access to timestamping services, an event ordering service and a global key service.

[5]`<http://pgp.ai.mit.edu>`, originally maintained by B. LaMacchia; it contains PGP public keys submitted through email or web interface without any certifying procedure; people can register fake keys with other's name; the server has multiple redundant copies on distributed servers around the world and they keep the same image by periodical mirroring.

# Chapter 7

# Conclusions

We present the conclusions of the thesis stated in Chapter 1 and the following chapters. A brief direction for future work which may follow this research is suggested.

## 7.1 A conclusion of the thesis

As stated in Section 1.1, the thesis of this research is that web publishing can be more reliable when its integrity, authenticity and availability are improved. A framework that consists of a publishing policy for improving the integrity and authenticity of web publications, and technical primitives supporting the policy and the persistent availability of web publications, was developed. An implementation of our framework was presented.

The motivation and direction of our research appear in Chapter 1, the investigation that clarifies this thesis appears in Chapter 2, 3 and 4, the proposed framework that improves integrity, authenticity and availability appears in Chapter 5, and an implementation of the framework is presented in Chapter 6. It is evident from an examination of the framework and its implementation that the thesis has been proven.

In particular, the chapters show the following:

- In Chapter 1, we examined the current situation and threats to web publishing, and briefly surveyed previous work carried out on electronic publishing to clarify the direction of our research.
- In Chapter 2, we reviewed security properties and policies to help us identify major security properties which affect the reliability of web publishing.
- In Chapter 3, we investigated the nature of publishing in both conventional and electronic media to extract fundamental requirements of web publishing with the understanding of security properties obtained in Chapter 2.
- In Chapter 4, we dealt with a series of web publishing applications to help understand practical aspects of web publishing. The applications were examined by the requirements obtained in Chapter 3.
- In Chapter 5, we presented a secure publishing policy based on the investigation in Chapter 2, 3 and 4. Technical primitives to achieve a working model implementing the policy were also presented. The policy and the primitives provided a framework for reliable web publishing.
- In Chapter 6, we presented a reliable web publishing system that implemented the framework given in Chapter 5.

## 7.2 Future work

In this dissertation, we presented a framework to make web publishing more reliable. We focused on two security properties, integrity, authenticity and availability, and provided a security policy and technical primitives to improve those properties in web publishing.

We believe that the Internet will develop in a complex way and a security policy has to be refined by interaction with applications in order to keep it useful. Applications of our framework to commercial services may help us refine it.

As a means to obtain fundamental integrity, we adopted an append-only file system. In our implementation, two mechanisms are used: revision control and CD-WORM media. CD-WORM is enough to construct our prototype but it has some difficulties in maintenance such as physical handling. Revision control is a part of the append-only file system, but it does not protect integrity alone. An interesting question is to establish an append-only file system in a simpler way without physical media support.

There is another implementation issue. In order to facilitate the Jikzi publishing server installation and to extend its functionality more flexibly, a full XML and script language-based implementation can be considered. It will also help us run the system independently of underlying operating systems because we already have multi-system support environments for XML and some script languages such as PERL.[1]

We have a fortunate opportunity to see the paradigm of publishing changing from conventional to electronic. It is interesting to observe how people build the reliability of published material in the new publishing paradigm and note which requirements are still sensible and which are not. We believe that this observation will give deeper understanding for the future design of reliable electronic services other than web publishing.

[1]Practical Extraction and Report Language, a script language portable on several variants of UNIX, Microsoft Win32 and Apple Macintosh.

# Bibliography

[1] M. Abadi, T. M. A. Lomas, and R. M. Needham, 'Strengthening passwords'. SRC Technical Note 1997-033, Compaq Systems Research Center, Palo Alto, CA, 16 December 1997. (p. 51)

[2] S. G. Akl, 'Digital signatures with blindfold arbitrators who cannot form alliances'. In *Proceedings of 1983 IEEE Symposium on Security and Privacy*, pp. 129–135, Oakland, CA, April 1983. (p. 27)

[3] E. Amoroso, *Fundamentals of Computer Security Technology*. Englewood Cliffs, New Jersey, Prentice-Hall, 1994, ISBN 0-13-305541-8. (p. 19)

[4] R. J. Anderson, B. Crispo, J.-H. Lee, C. Manifavas, V. Matyáš, and F. A. P. Petitcolas, *The Global Trust Register 1998*. Cambridge, Northgate Consultants Ltd., February 1998, ISBN 0-9532397-0-5. (pp. 45, 67)

[5] ——, *The Global Internet Trust Register 1999*. Cambridge, MA, MIT Press, April 1999, ISBN 0-262-51105-3. (pp. 45, 67)

[6] R. J. Anderson, 'UEPS – A second generation electronic wallet'. In *Proceedings of European Symposium on Research in Computer Security (ESORICS) '92*, vol. 648 of *Lecture Notes in Computer Science*, pp. 411–418, Springer-Verlag, 1992. (p. 54)

[7] ——, 'The Eternity service'. In *Pragocrypt '96*, pp. 242–252, Prague, CTU Publishing House, 1996. (p. 76)

[8] ——, 'Security in clinical information systems'. BMA Report, British Medical Association, January 1996, ISBN 0-7279-1048-5. (pp. 22, 75)

[9] R. J. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. M. Needham, 'A new family of authentication protocols'. *Operating Systems Review*, vol. 32, no. 4, pp. 9–20, October 1998. (pp. 29, 45, 64, 67)

[10] R. J. Anderson and J.-H. Lee, 'Jikzi: A new framework for secure publishing'. In *Proceedings of Security Protocols Workshop '99*, Cambridge, April 1999, to appear. (pp. 75, 85)

[11] ——, 'Jikzi – A new framework for security policy, trusted publishing and electronic commerce'. *Computer Communications*, to appear. (p. 85)

[12] R. J. Anderson, V. Matyáš, and F. A. P. Petitcolas, 'The Eternal Resource Locator: An alternative means of establishing trust on the world wide web'. In *1998 USENIX Electronic Commerce Workshop*, pp. 141–153, Boston, MA, 1998. (pp. 78, 84)

[13] R. J. Anderson, V. Matyáš, F. A. P. Petitcolas, I. E. Buchan, and R. Hanka, 'Secure books: Protecting the distribution of knowledge'. In *Proceedings of Security Protocols Workshop '97*, pp. 1–12, Paris, April 1997. (pp. 78, 83)

[14] T. Anderson and P. A. Lee, *Fault Tolerance, Principles and Practice*. London, Prentice-Hall International, 1981, ISBN 0-13-308254-7. (p. 79)

[15] D. E. Bell and L. J. LaPadula, 'Secure computer systems: Mathematical foundations'. Mitre Report ESD-TR-73-278 (Vol. I–III), Mitre Corporation, Bedford, MA, April 1974. (p. 19)

[16] S. M. Bellovin and M. Merritt, 'Encrypted key exchange: Password-based protocols secure against dictionary attacks'. In *1992 IEEE Symposium on Security and Privacy*, pp. 72–84, Oakland, CA, 1992. (p. 51)

[17] ——, 'Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise'. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 244–250, Fairfax, VA, Association for Computing Machinery, November 1993. (p. 51)

[18] J. C. Benaloh, *Verifiable Secret-Ballot Elections*. Ph.D. dissertation, Yale University, New Haven, CT, 1987, YALEU/DCS/TR-561. (p. 76)

[19] J. C. Benaloh and D. Tuinstra, 'Receipt-free secret-ballot elections'. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pp. 544–553, Montreal, Quebec, Association for Computing Machinery, May 1994. (p. 58)

[20] T. Berners-Lee, R. Fielding, and L. Masinter, 'Uniform Resource Identifiers (URI): Generic syntax'. IETF RFC 2396, Internet Engineering Task Force, August 1998. (p. 40)

[21] K. Biba, 'Integrity considerations for secure computing systems'. Mitre Report MTR-3153, Mitre Corporation, Bedford, MA, 1975. (p. 21)

[22] J. Borrell and J. Rifà, 'An implementable secure voting scheme'. *Computers & Security*, vol. 15, no. 4, pp. 327–338, 1996. (p. 58)

[23] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, 'Extensible Markup Language (XML)'. W3C Working Draft WD-xml-970807, World Wide Web Consortium, 7 August 1997, <`http://www.w3.org/TR/WD-xml-lang.html`>. (pp. 39, 84)

[24] D. F. C. Brewer and M. J. Nash, 'The Chinese Wall security policy'. In *1989 IEEE Symposium on Security and Privacy*, pp. 206–214, Oakland, CA, 1989. (p. 20)

[25] G. Bruce and R. Dempsey, *Security in Distributed Computing*. Upper Saddle River, NJ, Prentice Hall PTR, 1997, ISBN 0-13-182908-4. (p. 19)

[26] Canadian System Security Centre, *The Canadian Trusted Computer Product Evaluation Criteria*, 1993, Version 3.0e. (p. 17)

[27] P. Cederqvist, *Version management with CVS for CVS 1.9*. Signum Support AB, Linkoping, 1993, Web publication, <`http://www.gnu.org/manual/cvs-1.9/`>. (pp. 77, 88)

[28] D. Chaiken, M. Hayter, J. Kistler, and D. Redell, 'The virtual book'. Research Report 157, Compaq Systems Research Center, Palo Alto, CA, 11 November 1998. (p. 33)

[29] D. L. Chaum, 'Untraceable electronic mail, return addresses, and digital pseudonyms'. *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, 1981. (p. 47)

[30] ——, 'Security without identification: Transaction systems to make Big Brother obsolete'. *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985. (p. 58)

[31] Y.-h. Chu, P. DesAutels, B. LaMancchia, and P. Lipp, 'PICS Signed Labels (DSig) 1.0 specification'. W3C Recommendation REC-DSig-label-19980527, World Wide Web Consortium, 27 May 1998, `<http://www.w3.org/TR/REC-DSig-label>`. (p. 84)

[32] D. D. Clark and D. R. Wilson, 'A comparison of commercial and military computer security policies'. In *1987 IEEE Symposium on Security and Privacy*, pp. 184–194, Oakland, CA, 1987. (pp. 10, 21)

[33] R. J. F. Cramer, M. Franklin, B. Schoenmakers, and M. Yung, 'Multi-authority secret-ballot elections with linear work'. In *Proceedings of Advances in Cryptology – Eurocrypt '96*, pp. 72–83, Springer-Verlag, 1996. (p. 58)

[34] R. J. F. Cramer, R. Gennaro, and B. Schoenmakers, 'A secure and optimally efficient multi-authority election scheme'. In *Proceedings of Advances in Cryptology – Eurocrypt '97*, pp. 103–118, Springer-Verlag, 1997. (p. 58)

[35] L. F. Cranor and R. K. Cytron, 'Design and implementation of a practical security-conscious electronic polling system'. Technical Report WUCS-96-02, Washington University, St. Louis, MO, 23 January 1996. (p. 58)

[36] B. Crispo and T. M. A. Lomas, 'A certification scheme for electronic commerce'. In *Proceedings of Security Protocols Workshop '96*, vol. 1189 of *Lec-*

*ture Notes in Computer Science*, pp. 19–32, Cambridge, Springer-Verlag, April 1996. (p. 18)

[37] W. Dai, *B-Money*, 1998, Web publication, <`http://www.eskimo.com/~weidai/bmoney.txt`>. (p. 77)

[38] B. Davenport, A. Newberger, and J. Woodard, 'Creating a secure digital voting protocol for campus elections'. Web publication, Princeton University, Princeton, NJ, 1996, <`http://www.princeton.edu/~usgvote/technical/paper/`>. (p. 58)

[39] W. Diffie, 'The first ten years of public-key cryptography'. *Proceedings of the IEEE*, vol. 76, no. 5, pp. 560–577, May 1988. (p. 27)

[40] W. Diffie and M. E. Hellman, 'New directions in cryptography'. *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976. (pp. 27, 67)

[41] M. Dummett, *Voting Procedures*. Oxford, Clarendon Press, 1984, ISBN 0-198-76188-0. (p. 59)

[42] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, 'Hypertext Transfer Protocol – HTTP/1.1'. IETF RFC 2616, Internet Engineering Task Force, June 1999. (p. 2)

[43] A. O. Freier, P. Karlton, and P. C. Kocher, 'The SSL protocol version 3.0'. IETF Internet-DRAFT, Internet Engineering Task Force, 18 November 1996, <`http://home.netscape.com/eng/ssl3/draft302.txt`>. (p. 13)

[44] A. Fujioka, T. Okamoto, and K. Ohta, 'Practical secret voting scheme for large scale elections'. In *the Proceedings of Advances in Cryptology – Auscrypt '92*, pp. 244–251, Springer-Verlag, 1992. (p. 58)

[45] R. Gennaro, *Theory and practice of verifiable secret sharing*. Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1996. (p. 18)

[46] P. Ginsparg, 'First steps towards electronic research communication'. *Computers in Physics*, vol. 8, no. 4, pp. 390–396, January 1994. (pp. 4, 35, 43)

[47] S. Goldwasser, S. Micali, and R. L. Rivest, 'A digital signature scheme secure against adaptive chosen message attacks'. *SIAM Journal of Computing*, vol. 17, no. 2, pp. 281–308, April 1988. (p. 27)

[48] D. Gollman, *Computer Security*. West Sussex, John Wiley & Sons, 1999, ISBN 0-471-97844-2. (pp. 11, 12, 14)

[49] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer, 'Protecting poorly chosen secrets from guessing attacks'. *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 648–656, June 1993. (p. 51)

[50] S. Haber and W. S. Stornetta, 'How to time-stamp a digital document'. *Journal of Cryptology*, vol. 3, no. 2, pp. 99–112, 1991. (p. 94)

[51] N. M. Haller, 'The S/Key one-time password system'. In *ISOC Symposium on Network and Distributed System Security*, pp. 151–157, San Diego, CA, February 1994, see also IETF RFC 1704, 1760 and 1938. (pp. 28, 51)

[52] M. Herschberg, *Secure electronic voting over the World Wide Web*. Master's dissertation, Massachusetts Institute of Technology – Laboratory of Computer Science, Cambridge, MA, May 1997. (p. 58)

[53] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, 'Proactive public key and signature systems'. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pp. 100–110, Zurich, Association for Computing Machinery, April 1997. (p. 18)

[54] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, 'Proactive secret sharing, or: How to cope with perpetual leakage'. In *Proceedings of Advances in Cryptology – CRYPTO '95*, vol. 963 of *Lecture Notes in Computer Science*, pp. 339–352, Springer-Verlag, 1995. (p. 18)

[55] International Organisation for Standardisation, *ISO 7498-2: Basic Reference Model for Open Systems Interconnection (OSI) Part 2: Security Architecture*. Geneva, 1988. (p. 17)

[56] International Telecommunication Union Telecommunication Standardization Sector, *Security architecture for Open Systems Interconnection for CCITT applications*, March 1991, ITU-T Recommendation X.800. (pp. 11, 12, 28)

[57] ——, *Information technology – Open Document Architecture and interchange format: Introduction and general principles*, March 1993, ITU-T Recommendation T.411. (pp. 11, 12, 14, 28)

[58] ——, *Information technology – Open Systems Interconnection - Security frameworks for open systems: Authentication framework*, April 1995, ITU-T Recommendation X.811, also published as ISO/IEC 10181-2 in 1996. (p. 12)

[59] ——, *Definition of terms relevant to optical fibre submarine cable systems*, April 1997, ITU-T Recommendation G.972. (p. 17)

[60] ——, *Information technology – Open Systems Interconnection – The directory: Authentication framework*, August 1997, ITU-T Recommendation X.509. (pp. 68, 70)

[61] ——, *Information technology – Open Systems Interconnection – The directory: Overview of concepts, models and services*, August 1997, ITU-T Recommendation X.500. (p. 67)

[62] W.-S. Juang and C.-L. Lei, 'A collision-free secret ballot protocol for computerized general election'. *Computers & Security*, vol. 15, no. 4, pp. 339–348, 1996. (p. 58)

[63] S. Katzenbeisser and F. A. P. Petitcolas, eds, *Information Hiding Techniques for Steganography and Digital Watermarking*. London, Artech House Publishing, 2000, ISBN 1-58053-035-4. (p. 11)

[64] A. Kawaguchi, S. Nishioka, and H. Motoda, 'A flash-memory based file system'. In *1995 USENIX Annual Technical Conference*, pp. 16–20, New Orleans, LA, January 1995. (p. 76)

[65] J. Kravitz, 'SDML – Signed Document Markup Language'. W3C Note NOTE-SDML-19980619, World Wide Web Consortium, 19 June 1998, `<http://www.w3.org/TR/NOTE-SDML/>`. (pp. 78, 84)

[66] L. Lamport, 'Time, clocks, and the ordering of events in a distributed system'. *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978. (pp. 77, 95)

[67] ——, 'Constructing digital signatures from one-way function'. Technical Report SRI-CSL-98, SRI International, Menlo Park, CA, October 1979. (p. 27)

[68] L. Lamport, R. Shostak, and M. Pease, 'The Byzantine generals problem'. *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982. (p. 30)

[69] J.-H. Lee, 'A resilient access control scheme for secure electronic transactions'. In *1998 USENIX Electronic Commerce Workshop*, pp. 75–82, Boston, MA, 1998. (pp. 45, 50)

[70] ——, 'The Big Brother ballot'. *Operating Systems Review*, vol. 33, no. 3, pp. 19–25, August 1999. (pp. 45, 57)

[71] ——, 'Fingerprinting'. In S. Katzenbeisser and F. A. P. Petitcolas, eds, *Information Hiding Techniques for Steganography and Digital Watermarking*, Chapter 8, London, Artech House Publishing, 2000, ISBN 1-58053-035-4. (p. 11)

[72] P. Massiglia, ed., *The RAIDbook: A Handbook of Storage System Technology*. RAID Advisory Board, Peer to Peer Communications, 6th edition, 1997, ISBN 1-57398-028-5. (p. 80)

[73] T. Mayfield, J. E. Roskos, S. R. Welke, and J. M. Boone, 'Integrity in automated information systems'. National Computer Security Center C Technical Report 79-91, Institute for Defense Analyses, September 1991, also known as IDA Paper P-2316. (p. 10)

[74] J. McLean, 'The specification and modeling of computer security'. *IEEE Computer*, vol. 23, no. 1, pp. 9–16, 1990. (p. 20)

[75] D. L. Mills, 'Internet time synchronization: the Network Time Protocol'. *IEEE Transactions on Communications*, vol. COM-39, no. 10, pp. 1482–1493, 1991. (p. 90)

[76] R. Molva, G. Tsudik, E. van Herreweghen, and S. Zatti, 'KryptoKnight authentication and key distribution system'. In *Proceedings of European Symposium on Research in Computer Security (ESORICS) '92*, vol. 648 of *Lecture Notes in Computer Science*, pp. 155–174, Springer-Verlag, 1992. (p. 28)

[77] M. Naor, 'Bit commitment using pseudo-randomness'. In *Proceedings of Advances in Cryptology – CRYPTO '89*, vol. 435 of *Lecture Notes in Computer Science*, pp. 128–136, Springer-Verlag, 1990. (p. 58)

[78] M. Naor and M. Yung, 'Universal one-way hash functions and their cryptographic application'. In *Proceedings of the 21th Annual ACM Symposium on Theory of Computing*, pp. 33–43, Seattle, WA, Association for Computing Machinery, May 1989. (p. 27)

[79] National Institute of Standard and Technology, *Data Encryption Standard (DES)*, 30 December 1993, Federal Information Processing Standards Publication FIPS PUB 46-2, reaffirmed until 1998. (pp. 13, 28)

[80] ——, *Secure Hash Standard (SHS)*, 17 April 1995, Federal Information Processing Standards Publication FIPS PUB 180-1. (p. 94)

[81] R. M. Needham, 'The changing environment for security protocols'. *IEEE Network*, pp. 12–15, May/June 1997. (pp. 28, 51)

[82] A. M. Odlyzko, 'The slow evolution of electronic publishing'. In A. J. Meadows and F. Rowland, eds, *Electronic Publishing '97: New Models and Opportunities*, pp. 4–18, ICCC Press, 1997. (p. 43)

[83] ——, 'Competition and cooperation: Libraries and publishers in the transition to electronic scholarly journals'. *Journal of Scholarly Publishing*, vol. 30, no. 4, pp. 163–185, July 1999. (p. 36)

[84] ——, 'The evolution of electronic scholarly communication'. In C. J. Manson, ed., *Science Editing and Information Management*, pp. 3–4, January 1999, `<http://www.research.att.com/~amo/doc/evolution.communications.txt>`. (p. 35)

[85] T. Okamoto, 'Receipt-free electronic voting schemes for large scale elections'. In *Proceedings of Security Protocols Workshop '97*, pp. 25–35, Paris, 1997. (p. 58)

[86] G. Orwell, *Nineteen Eighty-Four*. London, England, Penguin Books, 1989, First published by Martin Secker & Warburg Ltd. in 1949, ISBN 0-14-027877-X. (pp. 24, 57)

[87] B. Pfitzmann, *Digital signature schemes – General framework and fail-stop signatures*, vol. 1100 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996. (p. 27)

[88] C. P. Pfleeger, *Security in Computing*. Upper Saddle River, NJ, Prentice Hall PTR, 2nd edition, 1997, ISBN 0-13-185794-0. (pp. 14, 19)

[89] D. Ragget, A. Le Hors, and I. Jacobs, 'HyperText Markup Language (HTML)'. W3C Recommendation REC-html40, World Wide Web Consortium, 24 April 1998, `<http://www.w3.org/TR/REC-html40>`. (p. 2)

[90] M. K. Reiter, 'Secure agreement protocols: Reliable and atomic group multicast in Rampart'. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 68–80, Fairfax, VA, Association for Computing Machinery, November 1994. (p. 18)

[91] ——, 'The Rampart toolkit for building high-integrity services'. In *Theory and Practice in Distributed Systems*, vol. 938 of *Lecture Notes in Computer Science*, pp. 99–110, Springer-Verlag, 1995. (p. 18)

[92] ——, 'Distributing trust with the Rampart toolkit'. *Communications of the ACM*, vol. 39, no. 4, pp. 71–74, 1996. (p. 18)

[93] M. K. Reiter, M. K. Franklin, J. B. Lacy, and R. N. Wright, 'The Ω key management service'. *Journal of Computer Security*, vol. 4, no. 4, pp. 267–287, 1996. (p. 31)

[94] R. L. Rivest, 'The MD5 message digest algorithm'. IETF RFC 1321, Internet Engineering Task Force, April 1992. (p. 28)

[95] J. Rompel, 'One-way functions are necessary and sufficient for digital signatures'. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing*, pp. 387–394, Baltimore, MD, Association for Computing Machinery, May 1990. (p. 27)

[96] M. Rosenblum and J. K. Ousterhout, 'The design and implementation of a log-structured file system'. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pp. 1–15, Pacific Grove, CA, October 1991. (p. 76)

[97] ——, 'The design and implementation of a log-structured file system'. *ACM Computing Surveys*, vol. 10, no. 1, pp. 26–52, 1992. (p. 76)

[98] F. Rowland, C. McKnight, J. Meadows, and P. Such, 'ELVYN: The delivery of an electronic version of a journal from the publisher to libraries'. *Journal of the American Society for Information Science*, vol. 47, no. 9, pp. 690–700, September 1996. (p. 4)

[99] A. D. Rubin, 'Independent one-time passwords'. *USENIX Computing Systems*, vol. 9, no. 1, pp. 15–27, 1996. (p. 51)

[100] K. Sako and J. Killian, 'Secure voting using partially compatible homomorphism'. In *Proceedings of Advances in Cryptology – CRYPTO '94*, pp. 411–424, Springer-Verlag, 1994. (p. 58)

[101] B. Schneier, *Applied Cryptography*. New York, John Wiley & Sons, 2nd edition, 1996, ISBN 0-471-12845-7. (pp. 10, 12, 27)

[102] J. F. Snook, *Towards secure, optimistic, distributed open systems*. Ph.D. dissertation, University of Hertfordshire, Hatfield, September 1992, Technical Report No. 151. (pp. 4, 79)

[103] G. Taubes, ‘Publication by electronic mail takes physics by storm’. *Science*, vol. 259, no. 5099, pp. 1246–1248, 26 February 1993. (p. 35)

[104] ——, ‘Electronic preprints point the way to author empowerment’. *Science*, vol. 271, no. 5249, pp. 767–768, 9 February 1996. (p. 5)

[105] The Philological Society of London, ed., *The Oxford English dictionary*. Oxford, Clarendon Press, 1961. (pp. 10, 11, 13, 14, 15, 17, 26)

[106] L. C. Thurow, *Creating Wealth – The New Rules for Individuals, Companies, and Countries in a Knowledge-based Economy*. London, Nicholas Brealey Publishing, 1999, ISBN 1-85788-242-3, also published as ‘Building Wealth’. (p. 34)

[107] W. F. Tichy, ‘Design, implementation, and evaluation of a revision control system’. In *the 6th International Conference on Software Engineering*, pp. 58–67, ACM/IEEE, September 1982. (pp. 77, 89)

[108] M. V. Wilkes, *Time-sharing computer systems*, vol. 5 of *Computer Monographs*. London, Macdonald/American Elsevier, 2nd edition, 1972, 1st edition was published in 1968, ISBN 0-356-03985-4. (pp. 28, 51)

[109] R. Yahalom, ‘Optimistic trust with realistic eNvestigators’. In *Proceedings of Security Protocols Workshop ’98*, vol. 1550 of *Lecture Notes in Computer Science*, pp. 193–202, Cambridge, Springer-Verlag, April 1998. (p. 58)

[110] T. Ylönen, ‘SSH – Secure login connection over the Internet’. In *the 6th USENIX Security Symposium*, pp. 37–42, San Jose, CA, June 1996. (p. 13)

[111] P. R. Zimmermann, *The Official PGP User’s Guide*. Cambridge, MA, MIT Press, 1995, ISBN 0-262-74017-6. (p. 13)

# Appendixes

## Appendix A: Document Type Definitions

### stdDef.dtd

```
<!-- stdDef.dtd: DTD for entities -->
<!ENTITY % nameList
    "foreName CDATA #REQUIRED
    surName CDATA #REQUIRED
    initial CDATA #IMPLIED">
<!ENTITY % dateList
    "year CDATA #REQUIRED
    month CDATA #REQUIRED
    day CDATA #REQUIRED
    hour CDATA #IMPLIED
    minute CDATA #IMPLIED
    second CDATA #IMPLIED">
<!ENTITY % algoList
    "algoName CDATA #REQUIRED
    version CDATA #IMPLIED">
<!ENTITY % orgList
    "orgName CDATA #REQUIRED
    dept CDATA #IMPLIED
    subsection* CDATA #IMPLIED">

<!ELEMENT dtd (dtdInfo)+>
<!ELEMENT dtdInfo EMPTY>
<!ATTLIST dtdInfo
    name CDATA #REQUIRED
    version CDATA #REQUIRED
    URL CDATA #IMPLIED
    author CDATA #IMPLIED>

<!-- end of stdDef.dtd -->
```

## hashList.dtd

```
<!-- hashList.dtd: DTD for hash lists -->
<!ENTITY % stdDef SYSTEM "stdDef.dtd">

%stdDef;

<!ELEMENT hashList (hash)+>

<!ELEMENT hash (hashInfo, hashValue)>
<!ELEMENT hashInfo (hashAlgo, url?, parent?)>
<!ELEMENT hashAlgo EMPTY>
<!ATTLIST hashAlgo %algoList;>
<!ELEMENT url (#PCDATA)>
<!ELEMENT parent (#PCDATA)>
<!ELEMENT hashValue (#PCDATA)>

<!-- end of hashList.dtd -->
```

## signList.dtd

```
<!-- signList.dtd: DTD for signing docs -->
<!ENTITY % stdDef SYSTEM "stdDef.dtd">

%stdDef;

<!ELEMENT signList (sign)+>

<!ELEMENT sign (signInfo, pKeyInfo, signature)>
<!ELEMENT signInfo (signer, signAlgo, url?, parent?)>
<!ELEMENT signer EMPTY>
```

```
<!ATTLIST signer %nameList;
    orgName CDATA #IMPLIED
    signerId CDATA #IMPLIED>
<!ELEMENT signAlgo EMPTY>
<!ATTLIST signAlgo %algoList;>
<!ELEMENT url (#PCDATA)>
<!ELEMENT parent (#PCDATA)>
<!ELEMENT pKeyInfo (#PCDATA)>
<!ELEMENT signature (#PCDATA)>

<!-- end of signList.dtd -->
```

## cryptList.dtd

```
<!-- cryptList.dtd: DTD for encrypting docs -->
<!ENTITY % stdDef SYSTEM "stdDef.dtd">

%stdDef;

<!ELEMENT cryptList (cryptInfo)+>

<!ELEMENT cryptInfo (encrypter, cryptAlgo, acl)>
<!ELEMENT encrypter EMPTY>
<!ATTLIST encrypter %nameList;>
<!ELEMENT cryptAlgo EMPTY>
<!ATTLIST cryptAlgo %algoList;>
<!ELEMENT acl (authEntry+)>
<!ELEMENT authEntry (authName, pKeyInfo)>
<!ELEMENT authName (#PCDATA)>
<!ATTLIST authName %nameList;>
<!ELEMENT pKeyInfo (#PCDATA)>

<!-- end of cryptList.dtd -->
```

## erlDoc.dtd

```
<!-- erlDoc.dtd: DTD for ERL-type docs -->
```

```
<!ENTITY % hashList SYSTEM "hashList.dtd">

%hashList;

<!ELEMENT erlDoc (dtd, docBody, hashList)>
<!ELEMENT docBody (p+, hashInfo+)>
<!ELEMENT p (#PCDATA)>
```

## govDoc.dtd

```
<!-- govDoc.dtd: DTD for government docs -->
<!ENTITY % signList SYSTEM "signList.dtd">

%signList;

<!ELEMENT govDoc (dtd, docBody, signList)>
<!ELEMENT docBody (author+, docId, classification,
    creationDate, expiryDate?, timestamp, signInfo+, paragraph+)>

<!ELEMENT author EMPTY>
<!ATTLIST author %nameList;
    authorId CDATA #IMPLIED
    %orgList;>
<!ELEMENT docId (#PCDATA)>
<!ELEMENT classification EMPTY>
<!ATTLIST classification classCode
    ( topsecret | secret | classified | unclassified ) "classified" >
<!ELEMENT creationDate EMPTY>
<!ATTLIST creationDate %dateList;>
<!ELEMENT expiryDate EMPTY>
<!ATTLIST expiryDate %dateList;>
<!ELEMENT timestamp (#PCDATA)>
<!ELEMENT paragraph (textBody, classification)>
<!ELEMENT textBody (#PCDATA)>
<!-- end of govDoc.dtd -->
```

## eCheque.dtd

```
<!-- eCheque.dtd: DTD for electronic cheques -->
<!ENTITY % signList SYSTEM "signList.dtd">

%signList;

<!ELEMENT eCheque (dtd, chequeBody, signList)>
<!ELEMENT chequeBody (chequeId, account, payee, payment,
    issueDate, notLater?, timestamp, signInfo+)>

<!ELEMENT chequeId (#PCDATA)>
<!ELEMENT account (#PCDATA)>
<!ELEMENT payee EMPTY>
<!ATTLIST payee %nameList;
    payeeId CDATA #IMPLIED>
<!ELEMENT payment EMPTY>
<!ATTLIST payment
    amount CDATA #REQUIRED
    currency CDATA #IMPLIED>
<!ELEMENT issueDate EMPTY>
<!ATTLIST issueDate %dateList;>
<!ELEMENT notLater EMPTY>
<!ATTLIST notLater %dateList;>
<!ELEMENT timestamp (#PCDATA)>

<!-- end of eCheque.dtd -->
```

## certificate.dtd

```
<!-- certificate.dtd: DTD for certificates -->
<!ENTITY % signList SYSTEM "signList.dtd">

%signList;

<!ELEMENT certificate (dtd, certBody, signList)>
<!ELEMENT certBody (certType, serialNo, issuer, user,
    validity, signInfo, uKeyInfo, extension?)>
```

```
<!ELEMENT certType (#PCDATA)>
<!ATTLIST certType version CDATA #REQUIRED>
<!ELEMENT serialNo (#PCDATA)>
<!ELEMENT issuer EMPTY>
<!ATTLIST issuer
    issuerId CDATA #REQUIRED
    country CDATA #REQUIRED
    organization CDATA #REQUIRED
    caName CDATA #REQUIRED>
<!ELEMENT user EMPTY>
<!ATTLIST user %nameList;
    userId CDATA #IMPLIED
    country CDATA #REQUIRED
    organization CDATA #REQUIRED
    dept CDATA #IMPLIED>
<!ELEMENT validity (startDate, expiryDate)>
<!ELEMENT startDate EMPTY>
<!ATTLIST startDate %dateList;>
<!ELEMENT expiryDate EMPTY>
<!ATTLIST expiryDate %dateList;>
<!ELEMENT uKeyInfo (#PCDATA)>
<!ELEMENT extension (#PCDATA)>

<!-- end of certificate.dtd -->
```

## encDoc.dtd

```
<!-- encDoc.dtd: DTD for encrypted docs -->
<!ENTITY % cryptList SYSTEM "cryptList.dtd">

%cryptList;

<!ELEMENT encDoc (dtd, encBody, cryptList)>
<!ELEMENT encBody (#PCDATA)>

<!-- end of encDoc.dtd -->
```

# Appendix B: XML samples for DTDs

## examResult.xml

```
<?xml version="1.0"?>
<!DOCTYPE erlDoc SYSTEM "erlDoc.dtd">

<erlDoc>

<dtd>
<dtdInfo name="stdDef.dtd" version="1.0"/>
<dtdInfo name="hashList.dtd" version="1.0"/>
<dtdInfo name="erlDoc.dtd" version="1.0"/>
</dtd>

<docBody>
<p>
...
The examination results for the second MB degree
examination are as follows:
...
</p>

<hashInfo><hashAlgo algoName="SHA-1"/>
<url>http://www.med.abc.ac.uk/results</url>
<parent>http://www.cert.bma.org.uk</parent>
</hashInfo>
<hashInfo><hashAlgo algoName="TIGER"/>
<url>http://www.med.abc.ac.uk/results</url>
<parent>http://www.cert.med.ac.uk</parent>
</hashInfo>
</docBody>

<hashList>
<hash>
<hashInfo>
    <hashAlgo algoName="SHA-1"/>
    <url>http://www.med.abc.ac.uk/results</url>
    <parent>http://www.cert.bma.org.uk</parent>
</hashInfo>
```

```
<hashValue>M/67+HL...02LyhP3lfGnTf</hashValue>
</hash>

<hash>
<hashInfo>
    <hashAlgo algoName="TIGER"/>
    <url>http://www.med.abc.ac.uk/results</url>
    <parent>http://www.cert.med.ac.uk</parent>
</hashInfo>
<hashValue>zSEMzoA...EAMDEbYugP/+5</hashValue>
</hash>
</hashList>

</erlDoc>
```

## draft.xml

```
<?xml version='1.0'?>
<!DOCTYPE govDoc SYSTEM "govDoc.dtd">

<govDoc>

<dtd>
<dtdInfo name="stdDef.dtd" version="1.0"/>
<dtdInfo name="signList.dtd" version="1.0"/>
<dtdInfo name="govDoc.dtd" version="1.0"/>
</dtd>

<docBody>
    <author foreName="John" surName="Smith" initial="M"
        authorId="95M1295" orgName="DTI" dept="telecom"/>
    <docId>DTI-TEL-9901-0123</docId>
    <classification classCode="secret"/>
    <creationDate year="1999" month="01" day="23"/>
    <expiryDate year="1999" month="12" day="31"/>
    <timestamp>182390214292</timestamp>
<signInfo>
    <signer foreName="John" surName="Smith" initial="M"
        signerId="95M1295"/>
    <signAlgo algoName="PGP-RSA" version="5.5"/>
    <url>http://www.dti.gov/draft</url>
</signInfo>
    <paragraph>
        <textBody>
        ...
        This is a draft for the regulation for
        ...
        </textBody>
        <classification classCode="classified"/>
    </paragraph>
    <paragraph>
        <textBody>
        ...
        Products including cryptographic means
        ...
        </textBody>
        <classification classCode="secret"/>
    </paragraph>

</docBody>

<signList>
<sign>
<signInfo>
    <signer foreName="John" surName="Smith" initial="M"
        signerId="95M1295"/>
    <signAlgo algoName="PGP-RSA" version="5.5"/>
    <url>http://www.dti.gov/draft</url>
</signInfo>
<pKeyInfo>
lQMFEDWhboWuyrPDhRvRXQEBkp4D/ivwpsci5MJQXUA
bcPOUQquOgzMpp7W5KXP1Cit9EyqaPtet+1nkaoRXYv
FQIB/eBjkcvNaA02w/mvHQRQYiAzz6kdPSn/rt9THkX
LAOsOekv
=1zy8
</pKeyInfo>
<signature>
CZ/SDEjG6wt7V3uXWbZGV0pVg5LJg8j7bONjtdDuAHy
IyeYFI87qHE=
=OTeM
```

```
</signature>
</sign>
</signList>

</govDoc>

<!-- end of draft.xml -->
```

## corpCheque.xml

```
<?xml version="1.0"?>
<!DOCTYPE eCheque SYSTEM "eCheque.dtd">

<eCheque>

<dtd>
<dtdInfo name="stdDef.dtd" version="1.0" />
<dtdInfo name="signList.dtd" version="1.0" />
<dtdInfo name="eCheque.dtd" version="1.0" />
</dtd>

<chequeBody>
    <chequeId>00883627</chequeId>
    <account>12-34-56 1234567</account>
    <payee foreName="William" surName="Hopkinson" initial="F"/>
    <payment amount="19.95" currency="GBP" />
    <issueDate year="1999" month="01" day="15" />
    <notLater year="1999" month="06" day="30" />
    <timestamp>872043082393</timestamp>
    <signInfo>
    <signer foreName="John" surName="Smith" initial="M" />
    <signAlgo algoName="PGP-RSA" version="5.5" />
    </signInfo>
    <signInfo>
    <signer foreName="Edward" surName="Thompson" initial="J" />
    <signAlgo algoName="PGP-DSS" version="5.5" />
    </signInfo>
</chequeBody>

<signList>
```

```
<sign>
<signInfo>
    <signer foreName="John" surName="Smith" initial="M" />
    <signAlgo algoName="PGP-RSA" version="5.5" />
</signInfo>
<pKeyInfo>
lQMFEDWhboWuyrPDhRvRXQEBkp4D/ivwpsci5MJQXUA
bcPOUQquOgzMpp7W5KXP1Cit9EyqaPtet+1nkaoRXYv
FQIB/eBjkcvNaA02w/mvHQRQYiAzz6kdPSn/rt9THkX
LAOsOekv
=1zy8
</pKeyInfo>
<signature>
CZ/SDEjG6wt7V3uXWbZGV0pVg5LJg8j7bONjtdDuAHy
IyeYFI87qHE=
=OTeM
</signature>
</sign>
<sign>
<signInfo>
    <signer foreName="Edward" surName="Thompson" initial="J" />
    <signAlgo algoName="PGP-DSS" version="5.5" />
</signInfo>
<pKeyInfo>
SHllb24gTGVlICgxMDI0KSA8Sm9uZy1IeWVvbi5MZWV
bcPOUQquOgzMpp7W5KXP1Cit9EyqaPtet+1nkaoRXYv
FQIB/eBjkcvNaA02w/mvHQRQYiAzz6kdPSn/rt9THkX
L12s7ejk
IEz1y
</pKeyInfo>
<signature>
E0a57bT2+xWWds0Jh3wpIqV25B6+ExJA6xnAB3Az5hd
xAEALBQYiHd
n/rt9
</signature>
</sign>
</signList>

</eCheque>

<!-- end of corpCheque.xml -->
```

## x509Cert.xml

```
<?xml version="1.0"?>
<!DOCTYPE certificate SYSTEM "certificate.dtd">

<certificate>

<dtd>
<dtdInfo name="stdDef.dtd" version="1.0"/>
<dtdInfo name="signList.dtd" version="1.0"/>
<dtdInfo name="certificate.dtd" version="1.0"/>
</dtd>

<certBody>
    <certType version="3.0">x509v3</certType>
    <serialNo>1</serialNo>
    <issuer issuerId="1000345" country="UK"
        organization="UCAM" caName="CamCA"/>
    <user foreName="William" surName="Hopkinson" initial="F"
        userId="20023813" country="UK" organization="UCAM"
        dept="Computer Laboratory"/>
    <validity>
        <startDate year="1999" month="01" day="01" hour="00"
            minute="00" second="00"/>
        <expiryDate year="1999" month="12" day="31" hour="23"
            minute="59" second="59"/>
    </validity>
<signInfo>
    <signer foreName="$$NA" surName="$$NA" orgName="CamCA"
        signerId="1000345"/>
    <signAlgo algoName="PGP-RSA" version="5.5"/>
</signInfo>
    <uKeyInfo>
    djTHQquOgMp7W5KXPshYtwIs1Cit9EqaPt+1nkaoYv
    RhsB/eBjk/dsjYTskcvNaA02w/mHQYz6kdPSn/tHkX
    FI87qHE=
    dkQ0s
    </uKeyInfo>
    <extension>NONE</extension>
</certBody>
```

```
<signList>
<sign>
<signInfo>
    <signer foreName="$$NA" surName="$$NA" orgName="CamCA"
        signerId="1000345"/>
    <signAlgo algoName="PGP-RSA" version="5.5"/>
</signInfo>
<pKeyInfo>
sLrTlQMFEDWhboWuyrPDhRvRXQEBkp4D/ivwpsci5MJQXUA
QyTdbcPOUQquOgzMpp7W5KXP1Cit9EyqaPtet+1nkaoRXYv
GhsAFQIB/eBjkcvNaA02w/mvHQRQYiAzz6kdPSn/rt9THkX
FI87qHE=
dkQ0s
</pKeyInfo>
<signature>
E0a57bT2+xWWds0Jh3wpIqV25B6+ExJA6xnAB3Az5hdkQ0s
Tb0a2IdqBxN=
=IyeY
</signature>
</sign>
</signList>

</certificate>

<!-- end of x509Cert.xml -->
```

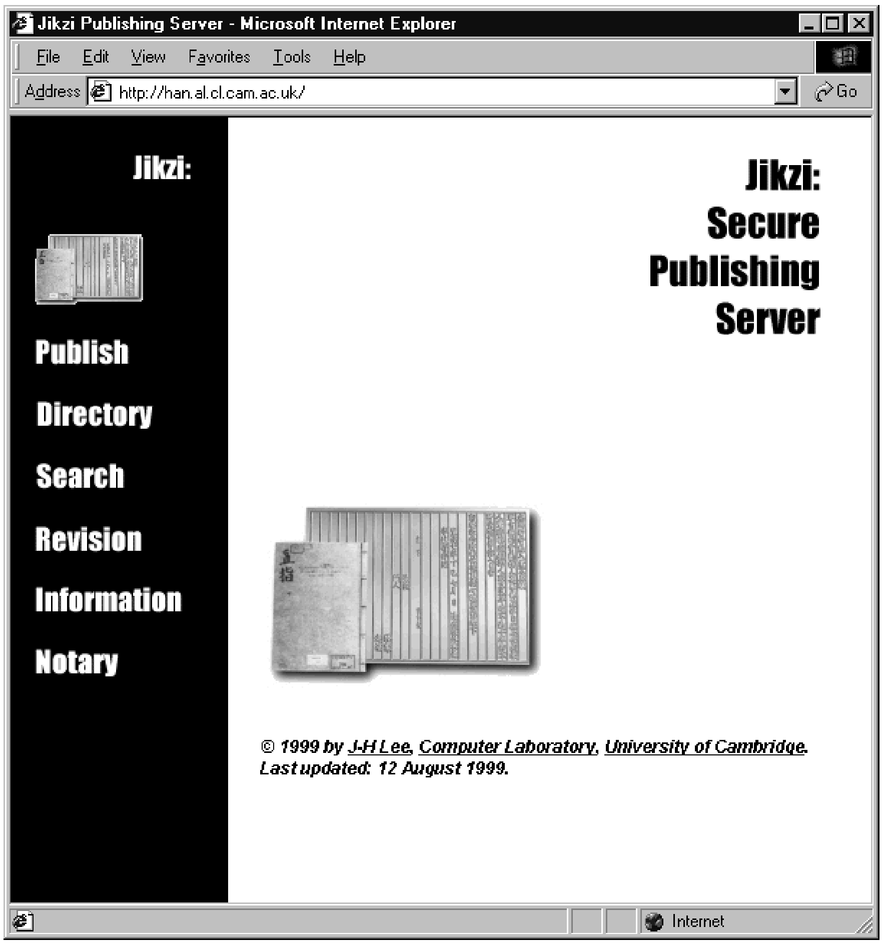
## encMemo.xml

```
<?xml version="1.0"?>
<!DOCTYPE encDoc SYSTEM "encDoc.dtd">

<encDoc>

<dtd>
<dtdInfo name="stdDef.dtd" version="1.0"/>
<dtdInfo name="signList.dtd" version="1.0"/>
<dtdInfo name="encDoc.dtd" version="1.0"/>
</dtd>

<encBody>
```

```
bcPOUQquOgzMpp7W5KXP1Cit9EyqaPtet+1nkaoRXYv
FQIB/eBjkcvNaA02w/mvHQRQYiAzz6kdPSn/rt9THkX
...

lQMFEDWhboWuyrPDhRvRXQEBkp4D/ivwpsci5MJQXUA
</encBody>

<cryptList>
<cryptInfo>
    <encrypter foreName="William" surName="Hopkinson"/>
    <cryptAlgo algoName="PGP-RSA" version="5.5"/>
    <acl>
        <authEntry>
            <authName foreName="William" surName="Hopkinson"/>
            <pKeyInfo>DhRvrRXE/TBkp4D/ivwpsci5MJ
            sjeUdcPOUQqugzp5X1it9EqPe+1naR
            WeRtS/eBjkcvNa0wmHQQiz6dn/rt9T5
            sYUi+=FI87qH
            E=dkQ0s
        </pKeyInfo>
        </authEntry>
        <authEntry>
            <authName foreName="John" surName="Smith"/>
            <pKeyInfo>
            dTrYFDWbWyPhvRQBpD/vw+pc5JuQXUA
            K/+ePUqOgMp7WKX1i9EqPt+nkaoRXYv
            Ed/aQBejkva0wmvQQiAz6kPSn/tTukX
            dMusKFIH
            RsIYk
            </pKeyInfo>
        </authEntry>
    </acl>
</cryptInfo>
</cryptList>

</encDoc>

<!-- end of encMemo.xml -->
```

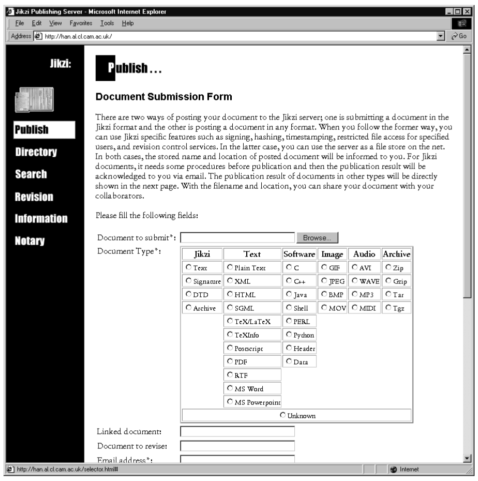
# Appendix C: The user interface

This chapter includes screen shots for Jikzi publishing server which show the user interface of the system.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
Jikzi: Secure Publishing Server
Internet

© 1999 by J-H Lee, Computer Laboratory, University of Cambridge.
Last updated: 12 August 1999.

**Figure C.1. The entrance screen for the Jikzi service:** the screen has a menu bar in the left-hand side with six menu buttons including Publish, Directory, Search, Revision, Information and Notary. The service interface for each menu is shown in the following pages.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
Browse...
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Publish . . .**

**Document Submission Form**

There are two ways of posting your document to the Jikzi server; one is submitting a document in the Jikzi format and the other is posting a document in any format. When you follow the former way, you can use Jikzi specific features such as signing, hashing, timestamping, restricted file access for specified users, and revision control services. In the latter case, you can use the server as a file store on the net. In both cases, the stored name and location of posted document will be informed to you. For Jikzi documents, it needs some procedures before publication and then the publication result will be acknowledged to you via email. The publication result of documents in other types will be directly shown in the next page. With the filename and location, you can share your document with your collaborators.

Please fill the following fields:

Document to submit*:

Document Type*:

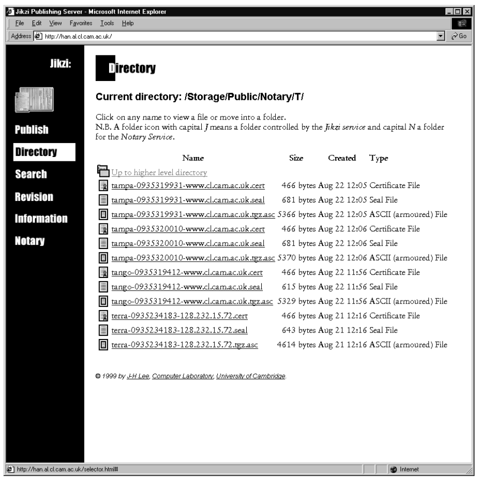
| Jikzi | Text | Software | Image | Audio | Archive |
|---|---|---|---|---|---|
| ○ Text | ○ Plain Text | ○ C | ○ GIF | ○ AVI | ○ Zip |
| ○ Signature | ○ XML | ○ C++ | ○ JPEG | ○ WAVE | ○ Gzip |
| ○ DTD | ○ HTML | ○ Java | ○ BMP | ○ MP3 | ○ Tar |
| ○ Archive | ○ SGML | ○ Shell | ○ MOV | ○ MIDI | ○ Tgz |
| | ○ TeX/LaTeX | ○ PERL | | | |
| | ○ TeXInfo | ○ Python | | | |
| | ○ Postscript | ○ Header | | | |
| | ○ PDF | ○ Data | | | |
| | ○ RTF | | | | |
| | ○ MS Word | | | | |
| | ○ MS Powerpoint | | | | |
| ○ Unknown | | | | | |

Linked document:

Document to revise:

Email address*:

**Figure C.2. Publish menu:** the screen shows the document submission form with user options to publish documents in the server.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Directory**

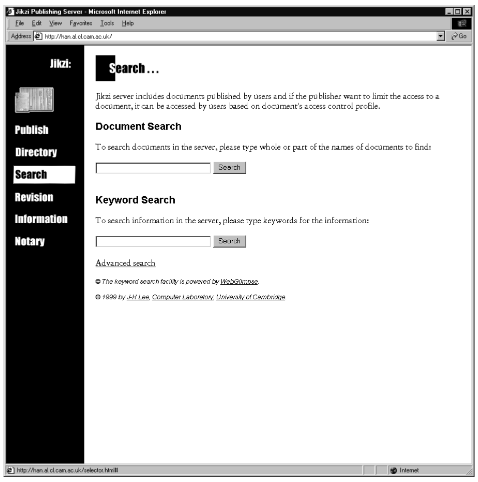
**Current directory: /Storage/Public/Notary/T/**

Click on any name to view a file or move into a folder.
N.B. A folder icon with capital *J* means a folder controlled by the *Jikzi service* and capital *N* a folder for the *Notary Service*.

| Name | Size | Created | Type |
|---|---|---|---|
| Up to higher level directory | | | |
| tampa-0935319931-www.cl.cam.ac.uk.cert | 466 bytes | Aug 22 12:05 | Certificate File |
| tampa-0935319931-www.cl.cam.ac.uk.seal | 681 bytes | Aug 22 12:05 | Seal File |
| tampa-0935319931-www.cl.cam.ac.uk.tgz.asc | 5366 bytes | Aug 22 12:05 | ASCII (armoured) File |
| tampa-0935320010-www.cl.cam.ac.uk.cert | 466 bytes | Aug 22 12:06 | Certificate File |
| tampa-0935320010-www.cl.cam.ac.uk.seal | 681 bytes | Aug 22 12:06 | Seal File |
| tampa-0935320010-www.cl.cam.ac.uk.tgz.asc | 5370 bytes | Aug 22 12:06 | ASCII (armoured) File |
| tango-0935319412-www.cl.cam.ac.uk.cert | 466 bytes | Aug 22 11:56 | Certificate File |
| tango-0935319412-www.cl.cam.ac.uk.seal | 615 bytes | Aug 22 11:56 | Seal File |
| tango-0935319412-www.cl.cam.ac.uk.tgz.asc | 5329 bytes | Aug 22 11:56 | ASCII (armoured) File |
| terra-0935234183-128.232.15.72.cert | 466 bytes | Aug 21 12:16 | Certificate File |
| terra-0935234183-128.232.15.72.seal | 643 bytes | Aug 21 12:16 | Seal File |
| terra-0935234183-128.232.15.72.tgz.asc | 4614 bytes | Aug 21 12:16 | ASCII (armoured) File |

*© 1999 by J-H Lee, Computer Laboratory, University of Cambridge.*

**Figure C.3. Directory menu:** the screen shows a list of stored files in the server repository; the screen shown above displays a list of files generated by the witness service.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
Search
Search
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Search . . .**

Jikzi server includes documents published by users and if the publisher want to limit the access to a document, it can be accessed by users based on document's access control profile.

**Document Search**

To search documents in the server, please type whole or part of the names of documents to find:

**Keyword Search**

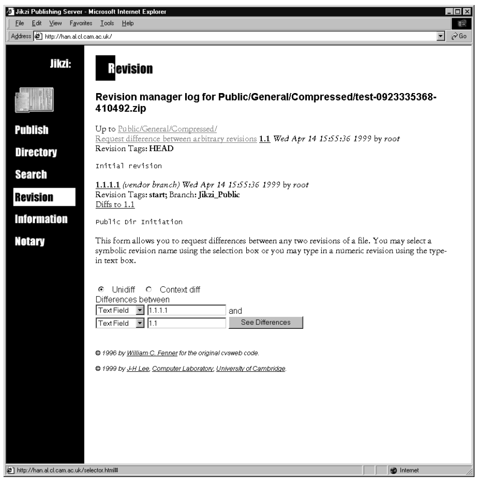
To search information in the server, please type keywords for the information:

Advanced search

*© The keyword search facility is powered by WebGlimpse.*

*© 1999 by J-H Lee, Computer Laboratory, University of Cambridge.*

**Figure C.4. Search menu:** the screen shows the form used to search for published documents. Documents can be sought by name or keyword; an advanced search facility is also provided.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
See Differences
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Revision**

**Revision manager log for Public/General/Compressed/test-0923335368-410492.zip**

Up to Public/General/Compressed/
Request difference between arbitrary revisions **1.1** *Wed Apr 14 15:55:36 1999* by *root*
Revision Tags: **HEAD**

```
Initial revision
```

**1.1.1.1** *(vendor branch) Wed Apr 14 15:55:36 1999* by *root*
Revision Tags: **start;** Branch: **Jikzi_Public**
Diffs to 1.1

```
Public Dir Initiation
```

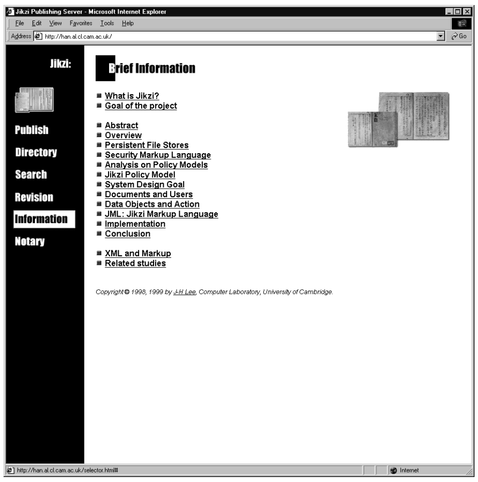
This form allows you to request differences between any two revisions of a file. You may select a symbolic revision name using the selection box or you may type in a numeric revision using the type-in text box.

[✓ Unidiff] Context diff
Differences between
Text Field 1.1.1.1 and
Text Field 1.1

*© 1996 by William C. Fenner for the original cvsweb code.*

*© 1999 by J-H Lee, Computer Laboratory, University of Cambridge.*

**Figure C.5. Revision menu:** the screen shows the revision history of a file, and users can see the differences between two versions.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Brief Information**

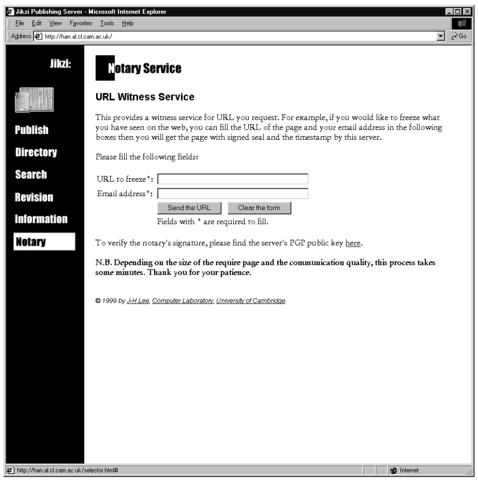
- What is Jikzi?
- Goal of the project

- Abstract
- Overview
- Persistent File Stores
- Security Markup Language
- Analysis on Policy Models
- Jikzi Policy Model
- System Design Goal
- Documents and Users
- Data Objects and Action
- JML: Jikzi Markup Language
- Implementation
- Conclusion

- XML and Markup
- Related studies

*Copyright © 1998, 1999 by J-H Lee, Computer Laboratory, University of Cambridge.*

**Figure C.6. Information pages:** the screen shows an information list about the Jikzi server; it includes background information, architecture and detailed mechanisms.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
Send the URL
Clear the form
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Notary Service**

**URL Witness Service**

This provides a witness service for URL you request. For example, if you would like to freeze what you have seen on the web, you can fill the URL of the page and your email address in the following boxes then you will get the page with signed seal and the timestamp by this server.

Please fill the following fields:

URL to freeze*:
Email address*:

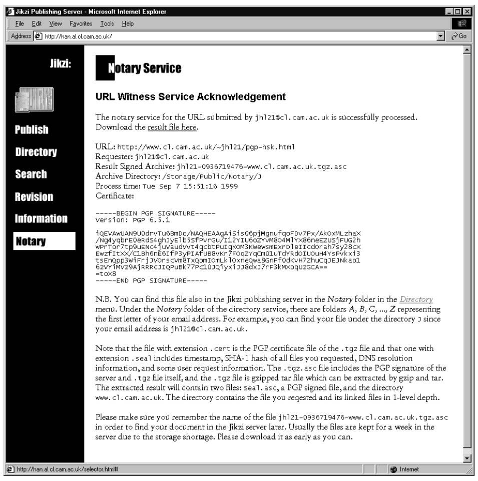
Fields with * are required to fill.

To verify the notary's signature, please find the server's PGP public key here.

**N.B. Depending on the size of the require page and the communication quality, this process takes some minutes. Thank you for your patience.**

*© 1999 by J-H Lee, Computer Laboratory, University of Cambridge.*

**Figure C.7. Notary menu:** the screen shows the user interface for the witness service; there are fields for the target URL and the user's email address for a reply.


Jikzi Publishing Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://han.al.cl.cam.ac.uk/
Go
Jikzi:
Publish
Directory
Search
Revision
Information
Notary
http://han.al.cl.cam.ac.uk/selector.html#
Internet

**Notary Service**

**URL Witness Service Acknowledgement**

The notary service for the URL submitted by `jhl21@cl.cam.ac.uk` is successfully processed. Download the result file here.

URL: `http://www.cl.cam.ac.uk/~jhl21/pgp-hsk.html`
Requester: `jhl21@cl.cam.ac.uk`
Result Signed Archive: `jhl21-0936719476-www.cl.cam.ac.uk.tgz.asc`
Archive Directory: `/Storage/Public/Notary/J`
Process time: `Tue Sep 7 15:51:16 1999`
Certificate:

```
-----BEGIN PGP SIGNATURE-----
Version: PGP 6.5.1

iQEVAwUAN9U0drvTu6BmDo/NAQHEAAgAiSis06pjMgnufqoFDv7Px/AkOxMLzhaX
/Ng4yqbrE0eRdS4ghJyElb5SfPvrGu/I12YIU6oZYvM8O4MlYX86neEZUSjFUG2h
wPrTor7tp9uENc4juVaudVvt4qcbtPuIgKOM3KwewsmExrDleIIcdOrah7sy28cX
EwzfItXX/C1Bh6nE6IfP3yPIAfUB8vKr7FOqZYqCm01uTdYRdOIUOuH4YsPvkxi3
tsEnQpp3WiFrjJVOrscVm8TxQomIOmLklOxneQwa8GnFf0dKvH7ZhuCqJEJNkao1
6zVYiMVz9AjRRRcJIQPuBk77Pc10JQiyxiJJ8dxJ7rF3kMXoqUzGCA==
=toX8
-----END PGP SIGNATURE-----
```

N.B. You can find this file also in the Jikzi publishing server in the *Notary* folder in the *Directory* menu. Under the *Notary* folder of the directory service, there are folders *A, B, C, ..., Z* representing the first letter of your email address. For example, you can find your file under the directory `J` since your email address is `jhl21@cl.cam.ac.uk`.

Note that the file with extension `.cert` is the PGP certificate file of the `.tgz` file and that one with extension `.seal` includes timestamp, SHA-1 hash of all files you requested, DNS resolution information, and some user request information. The `.tgz.asc` file includes the PGP signature of the server and `.tgz` file itself, and the `.tgz` file is gzipped tar file which can be extracted by gzip and tar. The extracted result will contain two files: `seal.asc`, a PGP signed file, and the directory `www.cl.cam.ac.uk`. The directory contains the file you reqested and its linked files in 1-level depth.

Please make sure you remember the name of the file `jhl21-0936719476-www.cl.cam.ac.uk.tgz.asc` in order to find your document in the Jikzi server later. Usually the files are kept for a week in the server due to the storage shortage. Please download it as early as you can.

**Figure C.8. Witness service result:** the screen displays a result of the passive witness service for a requested web page.