On the Application of Identity-Based Cryptography in Grid Security

Hoon Wei Lim

Technical Report
RHUL–MA–2006–2
2 May 2006

Royal Holloway
University of London

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

On the Application of Identity-Based Cryptography in Grid Security

Hoon Wei Lim

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London

2006

Declaration

These doctoral studies were conducted under the supervision of Prof. Kenneth G. Paterson and Dr. Matthew J.B. Robshaw.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Hoon Wei Lim
January, 2006

Acknowledgements

The past three years have been thus far the most exciting, challenging, interesting, and rewarding part of my life. I am very thankful that I have crossed paths with many wonderful people who have helped me in many ways in my pursuit of a PhD in Royal Holloway.

First and foremost, I would like to express my deepest gratitude to Kenny Paterson and Matt Robshaw for their invaluable supervision, unfailing enthusiasm, and respectful professionalism in guiding me through the countless difficult times that I encountered in my study. In particular, I thank them for their endless constructive feedback on my work and for nurturing the right research attitudes in me.

I am very grateful to Jason Crampton, Steven Galbraith, Chris Mitchell, Fred Piper, Geraint Price, Scarlet Schwiderski-Grosche, Allan Tomlinson and Peter Wild for their guidance and support. I am also indebted to David Mireles, Jacob Schuldt and Qiang Tang for their very helpful discussions, and many thanks to Ivona Brandic, Peter Gardfjäll, Maura Paterson, Johan Tordsson and Po-Wah Yau for proof-reading parts of this thesis.

To my mum, I cannot thank her enough for all her love and support. I wish that she will always be happy and healthy. To my brother and sister, I hope that they will excel in their future undertakings.

Many thanks to all my colleagues (especially those in Room 355) and friends for their good company and encouragement. Their presence has certainly lightened up my life and I will never forget their sharing and caring.

Finally, this thesis is dedicated to my beloved grandmother, Soo Khoon Chai, who passed away in October 2004.

Abstract

This thesis examines the application of identity-based cryptography (IBC) in designing security infrastructures for grid applications.

In this thesis, we propose a fully identity-based key infrastructure for grid (IKIG). Our proposal exploits some interesting properties of hierarchical identity-based cryptography (HIBC) to replicate security services provided by the grid security infrastructure (GSI) in the Globus Toolkit. The GSI is based on public key infrastructure (PKI) that supports standard X.509 certificates and proxy certificates. Since our proposal is certificate-free and has small key sizes, it offers a more lightweight approach to key management than the GSI. We also develop a one-pass delegation protocol that makes use of HIBC properties. This combination of lightweight key management and efficient delegation protocol has better scalability than the existing PKI-based approach to grid security.

Despite the advantages that IKIG offers, key escrow remains an issue which may not be desirable for certain grid applications. Therefore, we present an alternative identity-based approach called dynamic key infrastructure for grid (DKIG). Our DKIG proposal combines both identity-based techniques and the conventional PKI approach. In this hybrid setting, each user publishes a fixed parameter set through a standard X.509 certificate. Although X.509 certificates are involved in DKIG, it is still more lightweight than the GSI as it enables the derivation of both long-term and proxy credentials on-the-fly based only on a fixed certificate.

We also revisit the notion of secret public keys which was originally used as a cryptographic technique for designing secure password-based authenticated key establishment protocols. We introduce new password-based protocols using identity-based secret public keys. Our identity-based techniques can be integrated naturally with the standard TLS handshake protocol. We then discuss how this TLS-like identity-based secret public key protocol can be applied to securing interactions between users and credential storage systems, such as MyProxy, within grid environments.

Contents

List of Figures

List of Tables

Abbreviations

API:	Application Program Interface
BDH:	Bilinear Diffie-Hellman
CA:	Certification Authority
CDH:	Computational Diffie-Hellman
CRL:	Certificate Revocation List
DF:	Delegation Factory
DDH:	Decisional Diffie-Hellman
DH:	Diffie-Hellman
DKIG:	Dynamic Key Infrastructure for Grid
DN:	Distinguished Name
EPR:	EndPoint Reference
GDH:	Gap Diffie-Hellman
GGF:	Global Grid Forum
GRAM:	Grid Resource Allocation and Management
GSI:	Grid Security Infrastructure
GSS-API:	Generic Security Service Application Program Interface
GT:	Globus Toolkit
HIBC:	Hierarchical Identity-Based Cryptography
HIBE:	Hierarchical Identity-Based Encryption
HIBS:	Hierarchical Identity-Based Signature
HTTP:	HyperText Transport Protocol
IBC:	Identity-Based Cryptography
IBE:	Identity-Based Encryption
IBS:	Identity-Based Signature
ID:	Identifier
ID-SPK:	Identity-Based Secret Public Key
IKIG:	Identity-Based Key Infrastructure for Grid
MAC:	Message Authentication Code
MJF:	Managed Job Factory
OCSP:	Online Certificate Status Protocol
OGSA:	Open Grid Services Architecture
OGSI:	Open Grid Services Infrastructure
PKG:	Private Key Generator
PKI:	Public Key Infrastructure
PMA:	Policy Management Authority
SPK:	Secret Public Key
TA:	Trusted Authority
TCP:	Transmission Control Protocol
TLS:	Transport Layer Security
VO:	Virtual Organisation
WS:	Web Services
WSDL:	Web Services Description Language
WSRF:	Web Services Resource Framework
WWW:	World Wide Web
XML:	eXtensible Markup Language

Chapter 1

Introduction

Contents

This chapter gives an overview of the thesis. We provide the motivation for our research and describe the contributions of this thesis. In this chapter, we also present the overall structure of the thesis.

1.1 Motivation

Continual improvements to computing power, storage capacity and network bandwidth are allowing computing technologies of previously unheard of levels of sophistication. Nevertheless, there remain increasing demands for access to more computational power and resources, much of this being driven by the demands of large and complex new problems. Grid computing [67, 70] has been proposed as a mechanism to meet such demands. In essence, grid computing aims to provide an infrastructure allowing access to a wealth of sharable resources such as processing power, storage, databases, applications and any other devices (hardware) or components (software). All of these can be reached by remote users wishing to solve urgent and/or resource-intensive problems in computational science and engineering, experimental science, industrial engineering, corporate communications and so forth.

Provision of appropriate security within a grid environment seems to be more challenging than most conventional distributed systems. Public key infrastructure (PKI) technology is presently deployed in most grid implementations as it is perceived as a stable and mature technology which is widely supported and can be easily integrated with different applications on various platforms. In the Globus Toolkit (GT) [62, 81], the leading toolkit used in developing many grid systems, proxy certificates [176] have been designed and deployed in addition to standard X.509 public key certificates [101]. This compensates for some of the shortcomings in the conventional PKI setting and provides additional properties that align with the requirements for secure communications among grid entities within a dynamically changing environment. The motivations for the use of proxy certificates are twofold: (i) to limit exposure of long-term credentials, and (ii) to enable single sign-on (or unattended authentication) and delegation services. It is not clear, however, if the extensive use of certificates in the hierarchical PKI setting forms the basis of the best possible security architecture for grid environments.

Independent of grid computing, a variant of traditional public key technologies called identity-based cryptography (IBC) [25, 159] has recently received considerable attention. Through IBC, an identifier which represents a user can be transformed into his public key and used on-the-fly without any authenticity check. The potential of IBC to provide more immediate flexibility to entities within a security infrastructure and its certificate-free approach may well match the dynamic qualities of grid environments. In other words, it seems that the development of IBC may offer more lightweight and flexible key usage and management approaches within grid security infrastructures than does a traditional PKI. The aim of this thesis is to conduct a detailed investigation into the use of IBC in place of the PKI-based Grid Security Infrastructure (GSI) that is adopted in the GT. Our main focus is to find out if the identity-based approach can offer a feasible alternative and is a better solution than the PKI-based GSI.

1.2 Contributions

The application of IBC in grid security is an emerging and interesting area but the potential of IBC has only been partially investigated to date. The contributions of this thesis can be summarised as follows.

- We propose a fully identity-based key infrastructure for grid systems. It inherits attractive properties from IBC such as being certificate-free and having small key sizes. We present a customised identity-based authenticated key agreement protocol for grid environments. We also design a lightweight one-pass delegation protocol that supports short-term identity-based keys. Our protocols support single sign-on, authenticated key agreement and credential delegation in a natural way. In our proposals, we identify and discuss the performance trade-offs in terms of computational and communication costs between our proposals and current approaches.

- We propose a dynamic key infrastructure for grid applications. This is a hybrid certificate/identity-based approach which aims to resolve the key escrow issue encountered with the fully identity-based approach. Using this approach, most of the benefits that identity-based techniques offer can be preserved, while eliminating key escrow from the infrastructure. We examine the performance trade-offs in this hybrid approach as compared to the fully identity-based approach and the GSI.

- We introduce and study the concept of identity-based secret public keys. Our identity-based approach allows secret public keys to be constructed in a very natural way using arbitrary random strings, removing the structure found in, for example RSA or Diffie-Hellman keys. Using this and other new properties of identity-based secret public keys, we propose password-based authentication protocols and provide informal security analyses on these protocols. We also extend our work to integrating our identity-based techniques with the standard TLS handshake protocol. Such a protocol, in turn, seems to fit nicely into MyProxy, an on-line credential storage system for grid applications.

Parts of the research findings in this thesis have been (or will be) published by Springer-Verlag in Lecture Notes in Computer Science (LNCS) series [43, 116, 117, 118] and by IEEE Computer Society Press [115].

1.3 Organisation of Thesis

The remainder of this thesis is organised as follows.

Literature review: We present background material on grid security and identity-based cryptography in Chapters 2 and 3, respectively. These are prerequisites for understanding our new proposals for security infrastructures suited to grid applications. In Chapter 2, we first present the concept of grid computing, covering topics such as the evolution of grid computing, grid properties, grid architecture, the role of web services in grid computing, and grid applications. Subsequently, we describe grid security requirements and the security technologies used to meet these requirements in current grid implementations. We also provide technical details of the Globus Toolkit and the MyProxy system which are widely used in grid applications.

In Chapter 3, we give an overview of identity-based cryptography. We cover topics such as the comparison between certificate-based and identity-based public key infrastructures, pairing concepts, some identity-based cryptographic schemes that we will adopt in our proposals, and general performance and implementation aspects of identity-based cryptosystems. In addition, we also review existing or proposed applications of identity-based cryptography, in particular in email systems.

Security infrastructures for grid applications: Chapters 4 and 5 present our proposals of security infrastructures/architectures for grid environments. In Chapter 4, we begin by explaining current issues in PKI-based grid security architectures. We then give an overview of our proposal for an identity-based key infrastructure for grid (IKIG) and explain how is it different from current related work in the literature. That is followed by the detailed design of our IKIG proposal, which includes defining how various security services (supported by the GSI) can be provided by IKIG. We examine the key management aspects of IKIG and give informal security analyses of the protocols that underpin our IKIG proposal. Subsequently, we present an analysis of the performance of the protocols and compare the results with the performance of the GSI. We also discuss the impact of our IKIG proposal on web services security, some implementation issues, and limitations of our identity-based approach.

In Chapter 5, we introduce the concept of a dynamic key infrastructure for grid (DKIG). The aim of this proposal is to resolve the key escrow problem that IKIG inherits from its use of IBC. Chapter 5 begins by giving the motivation for and an overview of our DKIG proposal. We then review some related work in the literature. This is followed by the details of our DKIG proposal, including description of the authenticated key agreement and delegation protocols that DKIG supports. We provide details of key management aspects of DKIG. Subsequently, we present security and performance analyses of our DKIG proposal. Finally in Chapter 5, we highlight some practical issues and limitations of DKIG.

Secret public key protocols revisited: In Chapter 6, we revisit the concept of secret public keys and introduce the notion of identity-based secret public keys. Note that this chapter is rather independent of Chapters 4 and 5, and the reader should be able to understand most of the material presented in this chapter without reading Chapters 4 and 5. The aim of Chapter 6 is to examine new properties of identity-based secret public keys and how these properties can be used in constructing a password-based analogue of the TLS handshake protocol. We begin by reviewing the history of secret public keys and some related work. We then study the early proposals for secret public key protocols and the attacks on them. This gives us motivation for investigating identity-based secret public keys. In what follows, we introduce new properties of identity-based secret public keys. We also present two-party and three-party identity-based secret public key protocols. Subsequently, we show how our identity-based techniques can be expanded to support the use of passwords in an analogue of the standard TLS handshake protocol. Finally in this chapter, we discuss the application of our TLS-like identity-based secret public key protocol in the MyProxy system within a grid environment.

Conclusions: In the final chapter of this thesis, Chapter 7, we give concluding remarks about our proposals in Chapters 4, 5 and 6. These include the problems that we have studied, the importance of these problems, and a summary of our research findings. We also provide some suggestions for future work related to our proposals.

Chapter 2

Grid Computing and Its Security

Contents

This chapter provides an overview of grid computing and the associated security concerns. We discuss some grid security issues relevant to our research. To help the reader understand better the security requirements of grid systems, we also develop an example scenario of a grid application and discuss the security aspects of this scenario.

2.1 Computational Grids

Grid computing [63, 70] has emerged as a fast-evolving and important field which has gained substantial attention from multidisciplinary researchers worldwide due to its broad applicability. It is seen as the next generation computing technology, offering virtually "unlimited" resource sharing for computationally intensive advanced science and engineering problems. Figure 2.1 shows a classic computational grid environment which provides pervasive access to different types of computational resources. Examples of these resources are: processing power, storage capacity, network bandwidth, scientific instruments, databases, applications, and any other hardware devices or software components which may be needed to complete resource demanding and computationally complex tasks. After close to a decade of research and development, Foster [58, 59] has put forward the view that the grid vision of using computational power like other utilities, such as water and electricity is feasible based on a combination of technology trends and research advances.

Computing technologies have improved dramatically and become considerably cheaper in their costs over the last two decades. A personal computer in 2001 is as fast as a supercomputer of 1990 [58]. Even though computing power, storage capacity and network bandwidth are improving steadily, they cannot cope with the increasing

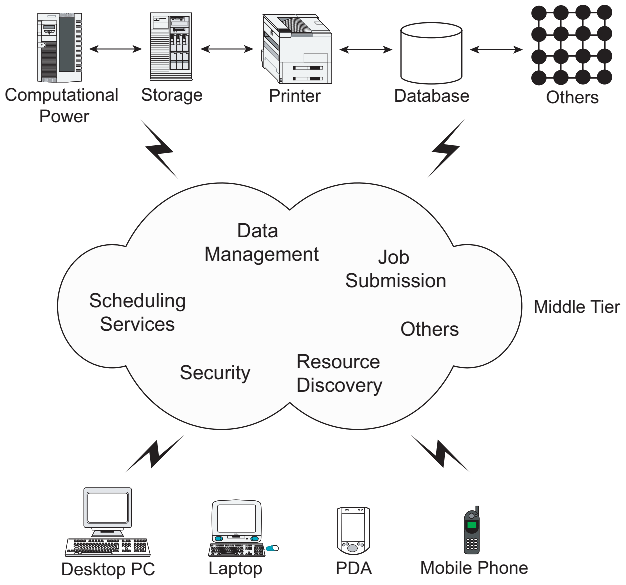

Figure 2.1: A typical computational grid environment.

demand for access to more computational resource for solving large and complex new problems. The emergence of grid computing provides a timely boost to the advancement of computing technology in developing better ways of enhancing human endeavour.

2.1.1 Definition of Computational Grid

The term grid[1] was first used in the mid-1990s to denote a distributed computing infrastructure for advanced science and engineering applications [65]. It implies a common and uniform means of providing computer programs with the required amount of computational resources, analogous to providing electrical power to household appliances. That said, the concept of sharing distributed resources is not new. In 1965, Vyssotsky *et al.* [183], the designers of the Multics operating system, envisaged a computer facility operating "like a power company or water company". Licklider and Taylor [114] also anticipated grid-like scenarios in 1968. Since the late 1960s, much work has been devoted to developing distributed systems. From the launch of Arpanet in 1969, to the Internet era through the 1980s, and to the World Wide Web (WWW) era during the 1990s, networking and computing technologies have improved by leaps and bounds. With the technological advancement that we have seen, the concept of distributed resource sharing may no longer be merely a myth but an achievable reality.

In [63], Foster and Kesselman, the pioneers of modern grid development, wrote:

> *A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities.*

In a subsequent publication [70], co-authored with Steve Tuecke, they refined the definition to address social and policy issues, stating that grid computing is a technology concerned with *coordinated resource sharing and problem solving in dynamic,*

[1] We refer to *grid computing* as a type of distributed computing technology, while a *computational grid* indicates the physical infrastructure that supports grid computing. Note that we also use a more general term *grid* to represent both grid technology and infrastructure.

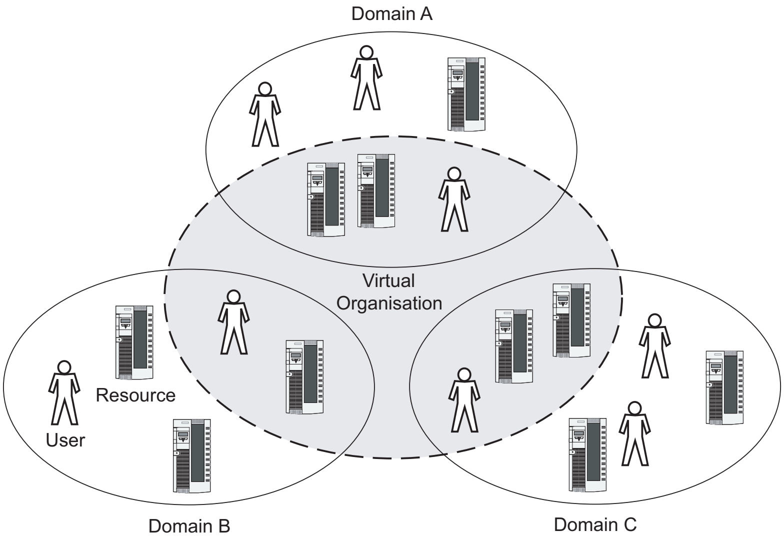

Figure 2.2: A VO consists of resources and users from different domains.

multi-institutional collaborations. The term *virtual organisation* (VO) [64, 70] is often used for these collaborations because of their distributed and usually ephemeral nature.

To be more specific, a virtual organisation (VO) is a set of users and resources that span multiple domains and are governed by a set of defined sharing rules, as shown in Figure 2.2. This sharing is, necessarily, highly controlled, with resource providers and users defining clearly and carefully what is shared, who is allowed to share, and the conditions under which sharing occurs [65, 70]. VOs can vary greatly in terms of scope, size, duration, structure, distribution and capabilities being shared.

2.1.2 Evolution of Grid

Grid computing has evolved quickly over the past decade. The early grid efforts started in the mid-1990s were known as *metacomputing*. The metacomputing projects were partly driven by the need to link a few US national supercomputing centres and aimed to provide computational resources to a range of high-performance applications. They were the forerunners of grids as we recognise them today. Two representative projects were FARNER and I-WAY [60]. In contrast to today's grids, they were mainly proprietary solutions.

The experience gained in the metacomputing projects convinced grid pioneers of the viability of constructing a global-scale distributed computing infrastructure capable of supporting diverse applications requiring massive amounts of computation and data. This vision was documented in [64] which set out the expectation of second generation grid systems. The focus of grid development at this stage was the building of middleware components that could support heterogeneous, scalable and adaptable distributed computing environments. The grid solutions became more generic and open than before as a result of the involvement of a larger research community working to address numerous implementation challenges. Some of the important new technologies (or projects) that emerged during this period are: Globus Toolkit (GT) versions 1 and 2 [62] which evolved from I-WAY; Legion [91]; Condor [177]; UNIform Interface to COmputer REsources (UNICORE) [7]; Storage Resource Broker (SRB) [149]; Common Object Request Broker Architecture (CORBA) [138]; and Peer-to-Peer computing (P2P) [46]. It is worth noting that some of these technologies such as CORBA and P2P are not intended to be used in grid computing, but are nevertheless technologies related to grid.

This second generation of grid computing technologies provided the inter-operability that was essential to achieve large-scale computation. As grid solutions were further explored, other aspects of the engineering of the grid became apparent. One of them was the desire to have re-usable components containing information about resources and a flexible way of assembling these components. Therefore, the third generation of grid systems involved increasing adoption of a *service-oriented model* (which we will discuss more in Section 2.1.4) and attention to *metadata*[2]. The technology enabler for this is web services which have received strong industry support. An important milestone is the development of the Open Grid Services Architecture (OGSA) which adapts the design of grid systems to web services. This is still a continuing process as web services technologies are still evolving, as with grid computing. GT versions 3 and 4 are two examples of OGSA-compliant implementations. Further background on the evolution of grid computing can be found in [153].

[2] Metadata defines the structure of data. It provides information about the content, quality, condition and other characteristics of data.

2.1.3 Grid Properties

Having briefly discussed how grid computing has evolved into the technology that we see today, one can easily understand that grid computing represents both an extension of and enhancement to current distributed computing technology. For example, CORBA, an enterprise distributed computing technology, only supports resource sharing within a single organisation but not across multiple domains [153]. Although P2P computing is effective for large-scale, geographically dispersed participants, its resource sharing architecture concentrates mainly on files and compute cycles [61]. On the other hand, grid computing aims to overcome these limitations by using a new architecture that we will describe in Section 2.1.4.

We list the properties inherent in grid systems as follows.

- *High-performance/throughput orientation.* A grid system is viewed as a single virtual machine (or distributed supercomputers) with massive computational power to cater for resource intensive and complex tasks.
- *Parallelism.* Parallel computations have always been essential for scientific and advanced engineering applications. Through the use of parallel computing technologies such as Message Passing Interface (MPI) [130, 131] and Parallel Virtual Machine (PVM) [78] for writing portable parallel programs, complicated tasks that comprise many independent and repetitive subtasks can be performed in parallel in a grid system.
- *Heterogeneity.* A grid system allows users to access and utilise resources located at different domains with dissimilar communication mechanisms. This inter-operability enables execution of tasks at different locations and on different platforms to be transparent to the users.
- *Scalability.* It is envisaged that a grid system allows the establishment of highly flexible sharing relationships among grid users and resource providers through various collaborative structures such as client-server and P2P, along with more complex models like sharing via intermediaries or brokers [65]. A grid system should be capable of supporting a reasonably large number of users with minimal performance degradation.

- *Dynamicity & adaptability.* A grid system should provide a highly dynamic and adaptive collaborative environment. Grid users and resource contributors may join and leave a VO over a short period of time. Ideally, a grid system should be adaptable enough so that the unavailability of a resource due to full utilisation or a technical failure would trigger a search for an alternative resource which meets the required specification.

Given these features, it seems that the grid research community must push the computing technology boundaries if the grid vision is to be truly realised. Current distributed computing technologies, such as web services, P2P, CORBA and PVM all have their own limitations when deployed individually. It is still not clear how difficult the challenges are for computational grids to function like electric power grids, if grid computing is developed using or evolved from these technologies.

2.1.4 Grid Architecture

After about a decade of focussed research and development, and experimentation, considerable consensus has emerged among the grid community on the requirements and architecture for grid computing. Because of the large-scale and heterogeneous nature of grid, inter-operability has always been the central architectural issue. Standardised protocols, defining the content and sequence of message exchanges used to request remote operations, have emerged as an essential means of achieving inter-operability.

In what follows, we describe the grid architecture proposed by Foster *et al.* in [63], which has now become the foundation for all grid systems.

The Layered Grid Architecture. The original layered grid architecture proposed by Foster *et al.* is shown in Figure 2.3. A brief description of each component is as follows:

- The *fabric* layer provides the lowest level of access to actual resources and implements the mechanisms that allow those resources to be shared and utilised.

- The *connectivity* layer defines the communication and security protocols required for network data transmissions between different fabric layer resources.
- The *resource* layer builds on the connectivity layer, implementing protocols that enable secure negotiation, initiation, monitoring, accounting and payment for sharing operations on individual resources.
- The *collective* layer deals with the coordination of multiple resources by providing functions such as resource discovery and scheduling.
- The *application* layer is where grid applications are implemented by utilising services defined in any of the previous layers.

Figure 2.3 also shows the mapping between the grid and Internet protocol architectures. They are similar in that components within each layer of the protocol architecture share common characteristics with each other, where each layer can build on the capabilities and behaviour provided by the layer below. The major difference lies in the interaction between layers of the architecture. In the Internet protocol, each layer generally interacts only with the layer above or below. For instance, a message created at the application layer can only be passed down to the transport layer. In the grid architecture, applications are able to call services defined at any lower layers. At each layer, there are Application Program Interfaces (APIs) implemented by Software Development Kits (SDKs) that allow exchanges of protocol messages with the appropriate services at different layers. Useful services such as resource discovery, data access, resource scheduling and so forth are normally not directly mapped to a protocol at a specific layer, but may combine protocol

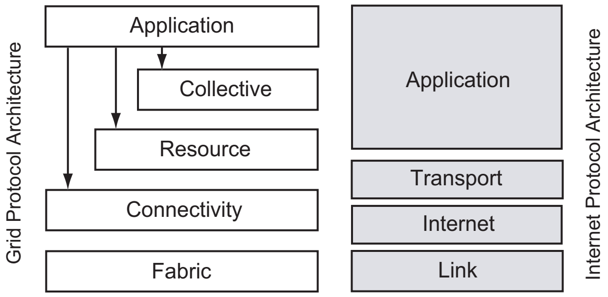

Figure 2.3: The layered grid architecture and its mapping to the Internet protocol architecture [63].

operations from more than one layer [63]. As illustrated in Figure 2.3, APIs at the application layer have the ability to directly call other APIs at the lower layers.

The Open Grid Services Architecture. By 2001, the rapidly increasing uptake of grid computing technologies among the gradually enlarging grid community had resulted in the emergence of the Global Grid Forum (GGF) as a standards body. One of the early activities of the GGF was developing the Open Grid Services Architecture (OGSA) [68], which aims to define a common, standard and open architecture for grid systems. A set of standard interfaces for services applicable to grid applications such as job management services, resource management services, and security services are specified in OGSA. These services are termed *grid services.* These are web services with extended features that allow them to support grid applications. However, OGSA is only a high-level architectural view of what grid services are and this has spawned another standard called the Open Grid Services Infrastructure (OGSI), also developed by GGF. OGSI gives a formal and detailed technical specification of what a grid service is.

From an organisational perspective, the OGSA services can be viewed within the context of the layered architecture of Figure 2.3. For instance, at the connectivity layer, the OGSA services are defined by a small number of service definitions that address critical issues such as authentication, credential mapping and policy verification. Meanwhile, at the resource level, there is also a set of service definitions for data access, job submission, bandwidth allocation and so on [66].

2.1.5 The Role of Web Services in Grid

Generally speaking, web services can be defined as *self-contained, self-describing modular applications that can be published, located and invoked* across the web [152]. Web services perform functions which can be anything from simple requests to complicated business processes. Publishing services on the web is different from publishing information on a website. Information displayed on a website is intended for humans. Information which is available through web services will always be accessed by software and not directly by a human.

We now look at a few basic building blocks of web services: eXtensible Markup Language (XML) [34], SOAP[3] [92], and Web Services Description Language (WSDL) [44].

- XML was created as a structured, self-describing way to represent data that is totally independent of application, protocol, operating system, or even programming language. It was initially developed to overcome the limitations of HTML, which is good at describing how data should be displayed but is poor at describing what the data that is being displayed is. XML, being text-based, is now very important to web services because it is human readable. No tools are needed to parse and render the data, and simple text tools and editors are sufficient for its manipulation.
- SOAP was created as a way to transport XML from one computer to another via a number of standard transport protocols. HTTP is the most common transport protocol and is prevalently used by the web. The SOAP model allows a clean separation between infrastructure processing and application processing of messages. It provides an envelope into which an XML message is placed. This envelope is a uniform container that can then be carried by a variety of transports.
- WSDL is an XML-based interface definition language that defines the set of operations that web services provide and the structure of their related SOAP messages. A WSDL file has a *what* section, a *how* section and a *where* section. The *what* section specifies the input and output structure of messages. The *how* section defines how the messages should be packaged in the SOAP envelope and how the envelope should be transferred. It also defines what information should be included in the SOAP header. The *where* section describes a specific web service implementation and ways to find its endpoint (or address).

Web services, being platform-independent, are the technology of choice for Internet-based applications with loosely-coupled[4] clients and servers. This, in turn, makes them the natural choice for building the next generation of grid-enabled applications.

[3] SOAP used to stand for Simple Object Access Protocol, but according to the W3C SOAP 1.2 specification [92], SOAP is now just a name and is no longer an acronym.

[4] Loosely-coupled services in the context of web services means that even if they are used within incompatible system technologies, they can still be joined together on demand to create composite services, or disassembled just as easily into their functional components.

In addition, web services provide support for dynamic discovery and composition of services in heterogeneous environments through WSDL. These are also important functionalities for grid applications [71].

However, web services do have certain limitations. The most notable one is that web services are *stateless*[5], which is in contrast with one of the most important requirements of OGSA, i.e. the underlying middleware of a grid system must be *stateful*. Thus, as mentioned earlier in Section 2.1.4, grid services are web services with improved characteristics. Apart from being stateful, other improvements included in OGSA (and OGSI) are notifications, portType extensions and life cycle management, to name just a few. An example of a complete implementation of OGSI is the third version of the Globus Toolkit (GT) [71].

Although OGSI grid services appear to solve some of the limitations of web services when adopted within a grid environment, they do have other drawbacks which act as a barrier to the convergence of web services and grid services. As OGSI is object oriented, there are compatibility issues with current web services tools (web services are not supposed to be object oriented) [165]. To improve grid services' chances of finally converging with web services, a new standard called Web Services Resource Framework (WSRF) [84] was proposed in early 2004 as a substitute for OGSI. The aim of WSRF is to integrate itself into the family of web services standards, instead of being a "patch over web services" like OGSI was. In other words, OGSA will be based directly on web services instead of being based on OGSI grid services. WSRF defines conventions for managing state so that applications can discover, inspect and interact with stateful resources in standard and inter-operable ways. This, in turn, provides the stateful services that OGSA needs. This improvement has been implemented in the latest version of the Globus Toolkit, GT4 [81].

In order to provide a web service with the ability to keep state information while still keeping it stateless, a separate entity called a *resource* (different from an actual computational resource) is introduced. A resource will store state information of a web service. The pairing of a web service with a resource is termed a WS-Resource and it will be given an address called an *endpoint reference* (EPR). Hence, WSRF

[5] Here 'stateless' means that a web service cannot remember or keep information, from one invocation to another. Although web services can in theory be either stateless or stateful, they are usually stateless and there is no standard way of making them stateful.

is a collection of specifications that relate to the management of WS-Resources.

Despite the seemingly promising role of web services in grid computing, it is widely recognised that transmitting XML data can be significantly less efficient than binary code [45]. Even so, this shortcoming is usually acceptable in most situations where bandwidth is not a significant constraint. We will discuss in more detail how the performance issues in web services affect the design and development of grid applications in Section 2.3.

2.1.6 Grid Applications

Grid concepts and technologies were first developed to facilitate resource sharing in far-flung scientific collaborations among universities and research institutions. Computational grids are normally used for resource intensive and computationally demanding scientific simulation or analysis, and complex problems which require dynamically constructed collaborative environments such as climate change simulation, modelling of high energy and nuclear physics, analysis of large statistical samples in astronomical research and so forth. Table 2.1 presents a classification of grid applications into the five types identified by Foster and Kesselman in [63].

In order to help develop these applications, there are numerous government-funded grid projects with the backing of industry players. These projects are both devel-

Table 2.1: Five classes of grid applications [63].

Category	Characteristics	Examples
Distributed supercomputing	Very large problems needing lots of CPU, memory, etc	Distributed interactive simulation, stellar dynamics
High throughput	Harness many otherwise idle resources to increase aggregate throughput	Chip design, parameter studies, cryptographic problems
On demand	Remote resources integrated with local computation, often for bounded amount of time	Medical instrumentation, network-enabled solvers, cloud detection
Data intensive	Synthesis of new information from many or large data sources	Sky survey, physics data, data assimilation
Collaborative	Support communication or collaborative work between multiple participants	Collaborative design, data exploration, education

oping core technologies and deploying production grids. Some of the recent major projects initiated in the US and Europe are as follows:

- *Globus* [81] is aimed at bringing together a large community of organisations and individuals to conduct research in and development of the fundamental technologies for grid computing. The Globus Toolkit is an open source software toolkit being developed by the Globus Alliance (the Globus Toolkit will be discussed in more detail in Section 2.3). Globus is supported by US government agencies such as the Defence Advanced Research Projects Agency (DARPA), the National Science Foundation (NSF), the US Department of Energy, and the National Aeronautics and Space Administration (NASA). It is also being supported by leading corporate organisations such as IBM, Microsoft and Cisco Systems.
- *TeraGrid* [172], completed in September 2004, was a multi-year effort to build and deploy one of the world's largest and fastest distributed infrastructures for open scientific research. The project, which was sponsored by NSF, currently connects nine supercomputing centres in the US. This brings together over 40 teraflops[6] of computing power, nearly 2 petabytes[7] of rotating storage, and specialized data analysis and visualization resources into production. These resources are interconnected at 10 to 30 gigabits/second via a dedicated national network.
- *EU DataGrid* [56] is a project funded by European Union (EU) to build the next generation computing infrastructure. The project, completed in March 2004, partly serves as a testbed for the European Organisation for Nuclear Research (CERN)'s Large Hadron Collider (LHC) Computing Grid Project [111], the world's largest and most powerful particle accelerator. Many of the products and technologies of the DataGrid project are included in newer EU grid projects such as Enabling Grids for E-SciencE (EGEE) [55].
- *UK e-Science* [133], also known as National e-Science, is a UK government-industry programme funded by the Department of Trade and Industry (DTI) and the Research Councils. The term e-Science refers to large-scale science that will increasingly be carried out through distributed global collaborations

[6] A teraflops is equal to one trillion floating-point operations per second.

[7] 1 petabyte = 10^3 terabytes = 10^6 gigabytes.

enabled by the Internet. It is envisaged that grid computing can realise the e-Science vision by setting up nation-wide computational grids and data resources.

Apart from the above grid projects, there are close to a hundred other on-going grid projects and initiatives around the globe [89].

Foster *et al.* [68] expect that computational grids, which are initially important for scientific and technical computing applications, will become essential for commercial distributed computing applications as well, including enterprise application integration and business-to-business (B2B) partner collaboration over the Internet. In the same way that the WWW began as a technology for scientific collaboration and was later adopted for e-business, it is believed that grid technologies will follow the same path. This is evident from the support provided by industry heavyweights such as Microsoft, IBM, HP and Sun Microsystems with their respective products, Microsoft .NET, IBM Websphere, mmGrid and Sun ONE. GRIDtoday [90] named the five industries where grid may have the biggest impacts as financial services, health care, energy, manufacturing and entertainment. However, it still remains to be seen if grid can change the world to the same extent as the Internet. The development of a new technology will always have the ‘chicken and egg’ problem: applications are needed to drive the research and development of the new technology, but certain applications are difficult to develop in the absence of stable and mature technology.

2.2 Grid Security

Security has become a vital and omnipresent issue for any distributed system. Grid systems, operating in networked, open environments, are no different. Since grid offers uniform access to resources that may be widely distributed geographically and organisationally, securing virtual organisations and access to the remote resources seems to be much more complicated than before [162].

In the remainder of this section, we will begin by providing the security requirements for grid applications. Most of them are also requirements in typical distributed systems, but some are unique to grid environments. This will be followed by brief descriptions of some existing security technologies deployed in grid implementations. Utilising current technologies is seen to be a rational choice in order to build a grid system in a less expensive manner. In addition, with existing technologies, a grid system can be implemented within a shorter time-frame as compared to investing in new technologies. We will then discuss some emerging security technologies that the grid community has adopted for grid implementations.

2.2.1 Grid Security Requirements

For better exposition of the security requirements for grid environments, we consider an example grid application. Our example concerns a scientific experiment called Compact Muon Solenoid [88, 162]. In the experiment, conducted at the Large Hadron Collider in the CERN Laboratory, Switzerland, the collected data is to be analysed by more than 2,000 physicists at more than 150 universities and laboratories located in 34 countries. Clearly, the security challenges mainly come from dissemination, processing and sharing of the data.

Before the data is transmitted from one domain to another, the authenticity of the requestor must be verifiable so that only an authorized requestor is able to access the available resources. When dealing with dissemination and access of data across many different resource centres in different countries, integration of security mechanisms and policies becomes a requirement. Also in many cases, data confidentiality and integrity can be vital to safeguard the physicists' research findings. When it comes to processing data, high-end computational resources generally require high investment and thus it is desirable that their usage can be tightly controlled, possibly through access control mechanisms. This is essential in balancing the resource usage between users from the physical organisations and the physicists from the virtual organisations. While, in certain circumstances, some research laboratories may have long-term trust relationships with each other, in most cases, remote resource access is anticipated to be short-term, typically on the order of hours or days. These trust relationships can be defined in the policies targeted at virtual organisations as well as hosting resources. From the data sharing aspect, trust establishment between entities of various universities and laboratories plays a crucial part in grid security.

Policy enforcement can be rather complicated, as expressing and exchanging policies within a virtual organisation involves remote entities with different security mechanisms and access privileges. All these security challenges must be considered when designing a sound grid security architecture.

With a rough idea of what security requirements might be expected in a grid application, we now compile a more formal list of grid security requirements deemed to be essential for supporting scalable, dynamic and distributed virtual organisations [69, 103, 104, 162, 173] .

- *Entity authentication.* In a grid environment, there are several types of entities that need to be authenticated. The most common are individual users who utilise grid resources and hosts which provide resources and services. Sometimes, a system administrator or an organisation can be thought of as an entity in his own right.
- *Single sign-on.* Participants in a grid environment often need to co-ordinate and communicate with multiple resources to accomplish a single job. It may be overly burdensome to the participants if they have to manually authenticate with each resource, e.g. by typing in a passphrase. Therefore, they should only need to perform an authentication process once to the first resource when a job is initiated, and not have to perform any further authentication with other resources.
- *Delegation.* The constantly changing size and membership of a virtual organisation encourages delegation of credentials and access rights from a job requestor to an intermediary such as a grid gateway or a resource broker, or directly to a target resource. This helps to achieve unattended authentication, i.e. authentication without any physical intervention from the requestor. This is particularly useful when a job takes a relatively long period of time to complete. Also, by delegating authority from the requestor to another entity, remote execution of the job becomes transparent to the requestor. This augurs well for the realisation of the grid vision in which computational grids are seen as uniform resources by the users regardless of the users' physical locations.
- *Credential life-span and renewal.* To limit the risk of compromising a user's credential when the user is performing or using single sign-on and delegation

services, the credential must be limited to a reasonable lifetime. In many cases, it may be difficult to predict accurately the credential lifetime required for a specific job. In those cases where the job takes a longer execution time than the lifetime of the issued credential, the user needs to be notified before the credential expires so that it can be renewed by the user.

- *Data confidentiality and integrity.* As with other standard distributed systems, protection of sensitive information from exposure to unintended parties can be critical. For example, this may be the case for health care applications, intellectual property oriented experiments, digital rights protected content delivery and so forth. Many modern security mechanisms which provide data privacy also offer data integrity protection, preventing unauthorized and undetectable modification of data by malicious parties.

- *Authorization and access control.* Grid applications require access to resources which may be located in different organisational domains with different owners. The access rights of a user should be based on authorization policies defined by his virtual organisation, the resource owners and local system administrators. This process is usually complementary to entity authentication so that policy enforcement can be targeted at a specific user.

- *Integration and inter-operability.* Each organisational domain has its own security infrastructure which supports a set of security mechanisms. These may be different from other domains or hosting environments. These mechanisms are typically based on existing security technologies which have been well-established, some of which will be discussed in Section 2.2.2. Hence, a practical grid security architecture must be able to support and integrate with current security mechanisms. Similarly, adoption of new standards or technologies should place high emphasis on solving potential integration issues that may arise. In addition, inter-operability between communicating parties should be achievable at various levels such as transport-level, message-level and service-level [71].

- *Trust relationships.* Establishing trust relationships between entities within a grid environment is arguably one of the most important security requirements. Trust can be expressed in the form of policies. For example, users or system administrators can decide whether or not to trust their Certificate Authority

(CA) based on the Certificate Policy and Certification Practices Statement (CP/CPS) issued by the CA. Trust relationships between CAs can be established through a Policy Management Authority (PMA)[8]. At the user level, a job requestor who has delegated his credential to a remote resource has to trust the resource to not misuse the delegated authority that it possesses.

- *Miscellaneous.* There are additional security requirements which need to be considered when designing a grid security solution, such as firewall traversal, security assurance, accounting, auditing, user privacy, security management, policy management and so forth. Among these, mechanisms capable of allowing data to cleanly traverse firewalls without compromising local control of firewall policy is crucial for implementing cross-domain grid systems. This has partly driven the adoption of SOAP message security (which we will discuss in Section 2.2.3). For more details about other security requirements and a more comprehensive discussion of grid security requirements in general, see [162, 173].

In this thesis, our focus will be on the first five items from the above list, namely: (i) entity authentication; (ii) single sign-on; (iii) delegation; (iv) credential life-span and renewal; and (v) data confidentiality and integrity. Furthermore, we will look at the integration of our proposals with current grid security architectures.

2.2.2 Existing Security Technologies for Grid

In this section, we intend to cover only security technologies or mechanisms relevant to the scope of this thesis. These include public key infrastructure (PKI), proxy certificates and their role in grid environments, the Transport Layer Security (TLS) protocol and how it is used to support grid security infrastructure, and some basic concepts about RSA encryption and signature schemes.

[8] A Policy Management Authority (PMA) is established among regional CAs to help facilitate the development of trust between the CAs. Examples of PMAs are European Grid PMA, Asia Pacific Grid PMA and The Americas Grid PMA. The International Grid Trust Federation which comprises regional PMAs, in turn, fosters a global trust relationship between PMAs.

Public Key Infrastructure (PKI). In 1976, Diffie and Hellman [51] introduced public key cryptography, a concept that offers an alternative to conventional secret key cryptography. Later on, the concept of PKI was introduced as a means of supporting security services using public key encryption and digital signature techniques. The practicality of a public key cryptosystem relies on the assurance of the authenticity of the public keys of the users. In order to achieve this, public key certificates are used to bind public keys with their respective owners' identities using digital signatures generated by a trusted third party.

In order to (partly) facilitate the growth and development of PKIs, many standards such as Lightweight Directory Access Protocol (LDAP) [185], Secure/Multipurpose Internet Mail Extension (S/MIME) [150] and X.509 [101] have been proposed which help to define an appropriate certificate framework within a PKI. By using X.509, an ITU Telecommunication Standardization Sector (ITU-T) standard for PKI, a public key is bound to a user's respective Distinguished Names (DNs) through an X.509 public key certificate issued by a CA. Figure 2.4 shows an example of an X.509 certificate which includes information such as the issuer, subject, validity period, public key and signature of a CA. A certificate, which is approximately 2 kilobytes in size, is usually transmitted across the network in PEM[9] encoding format.

A PKI (in this thesis we assume X.509-based PKI) usually exists in a hierarchical structure whereby there might be a few intermediate CAs between a user and a root CA which all users are expected to trust. The CA who issues certificates to its users normally operates at an organisation's domain level. Above this CA is an intermediate CA that, in turn, certifies the domain level CA. A root CA, normally a worldwide reputable and trusted party, sits at the top of the PKI hierarchy. This means that to check the validity of a user's certificate, a verifier needs to trace and verify all the certificates from the end of a chain back to the beginning, at the root CA whom the verifier can trust. This can be done through certification path validation procedures specified in [101].

Currently, PKI is the most widely used security infrastructure for grid implementations [69, 173]. One early development of a PKI-based security infrastructure for

[9] PEM stands for Privacy Enhanced Mail and it was published as a set of standards in the early-1990s to secure emails. It has now been superseded by S/MIME but it is still a widely used 7-bit ASCII format to encode certificates.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 19 (0x13)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=UK, O=eScience, OU=Authority,
                CN=CA/emailAddress=ca-operator@grid-support.ac.uk
        Validity
            Not Before: Jun 23 11:34:34 2004 GMT
            Not After : Jun 23 11:34:34 2005 GMT
        Subject: C=UK, O=eScience, OU=RoyalHollowayLondon, L=ISG,
                 CN=HoonWeiLim
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:be:5d:c1:4b:49:37:f0:58:7b:08:c6:f4:3d:25:
                    58:39:a6:8c:90:2c:ce:e4:56:ca:92:11:29:6e:28:
                    d7:81:8f:0a:a1:f5:38:6b:8f:21:9d:36:67:3a:66:
                    d8:bb:a6:86:a8:ee:c1:a9:91:a4:3e:6e:5f:5d:2e:
                    75:bd:31:aa:96:ee:16:a4:9a:ec:a7:03:11:89:ed:
                    17:ba:d1:52:43:98:ce:0e:d1:59:23:0c:0a:8b:fd:
                    0a:f4:b9:99:60:9d:94:16:62:cc:94:8d:f1:4f:ac:
                    e2:a4:45:94:87:33:74:90:0a:53:a6:0b:c8:9f:18:
                    15:8e:46:de:e5:a5:31:10:ad
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape Cert Type:
                SSL Client, SSL Server, S/MIME, Object Signing
    Signature Algorithm: md5WithRSAEncryption
        21:ad:81:31:82:d9:f4:25:31:99:9a:96:88:b8:8a:6f:cb:26:
        e3:aa:f4:4c:c9:41:35:40:f7:5f:0c:2c:d2:3a:5f:35:f9:dd:
        31:fb:92:3a:63:a2:67:0d:ee:3c:f0:f7:da:1a:0d:9a:c4:06:
        a6:b0:a8:8f:b2:10:21:a7:3b:05:86:e7:f9:b4:09:2a:15:a2:
        5f:80:ca:c7:e4:44:54:47:03:5e:3a:91:00:d4:4f:6c:59:cd:
        94:32:86:61:56:6f:01:a1:00:3a:b9:6b:c1:fe:e5:82:35:20:
        bf:25:fb:cd:76:29:02:5d:14:96:9f:f9:df:28:e1:65:66:e3:
        5c:05
-----BEGIN CERTIFICATE-----
MIICJzCCAZCgAwIBAgIBEzANBgkqhkiG9w0BAQQFADBYMREwDwYDVQQKEwhHcmlk
d2lzZTEWMBQGA1UECxMNS3Jha293IGJyYW5jaDEYMBYGA1UECxMPb2F4YWNhLmlu
dHJhbmV0MREwDwYDVQQDEwhWZXJpZmllcjAeFw0wNDA2MjMxMTM0MzRaFw0wNTA2
MjMxMTM0MzRaMEQxETAPBgNVBAoTCEdyaWR3aXNlMRYwFAYDVQQLEw1LcmFrb3cg
YnJhbmNoMRcwFQYDVQQDEw5Lcnp5c3p0b2YgV2lsazCBnzANBgkqhkiG9w0BAQEF
AAOBjQAwgYkCgYEAvl3BS0k38Fh7CMb0PSVYOaaMkCzO5FbKkhEpbijXgY8KofU4
a48hnTZnOmbYu6aGqO7BqZGkPm5fXS51vTGqlu4WpJrspwMRie0XutFSQ5jODtFZ
IwwKi/0K9LmZYJ2UFmLMlI3xT6zipEWUhzN0kApTpgvInxgVjkbe5aUxEK0CAwEA
AaMVMBMwEQYJYIZIAYb4QgEBBAQDAgTwMA0GCSqGSIb3DQEBBAUAA4GBACGtgTGC
2fQlMZmaloi4im/LJuOq9EzJQTVA918MLNI6XzX53TH7kjpjomcN7jzw99oaDZrE
BqawqI+yECGnOwWG5/m0CSoVol+AysfkRFRHA146kQDUT2xZzZQyhmFWbwGhADq5
a8H+5YI1IL8l+812KQJdFJaf+d8o4WVm41wF
-----END CERTIFICATE-----
```

Figure 2.4: A sample X.509 certificate and its PEM (base64 encoded) format.

grid applications was the Globus Toolkit's Grid Security Infrastructure (GSI) (we will discuss GSI at length in Section 2.3). Apart from PKI, there are a small number of grid projects which use Kerberos [136] as the backbone of their security infrastructures. Kerberos was an early development of authentication and authorization services using symmetric key techniques. It is based on the Needham-Schroeder key establishment protocol [134]. It is generally believed that Kerberos, being based on symmetric key cryptography, is more efficient compared to PKI. However, for a highly dynamic environment such as a computational grid, Kerberos is considered fairly rigid and inflexible because it requires the explicit involvement of site administrators to establish inter-domain trust relationships or to create new entities [69, 187]. Therefore, PKI is preferred for grid applications, while Kerberos seems to be best suited for intra-domain security. In order to achieve inter-operability with PKI-based systems, the Kerberos-based grid projects make use of a Kerberised client-side program, called KX.509, to acquire X.509 certificates using a client's existing Kerberos ticket [109, 128].

Proxy Certificates. There also exists a variant of X.509 certificates called *proxy certificates* [176]. The use of proxy credentials is a common means of allowing an entity A to grant to another entity B the right for B to be authorized to others as if it were A, in which case B is said to be acting as a proxy on behalf of A [135]. Tuecke *et al.* developed a X.509 proxy certificate profile [176] which works nicely with the current X.509 public key certificate profile [101] in PKI-based systems. They used the terminology *proxy certificate* to denote a short-term certificate that is derived[10] from a normal X.509 public key certificate (or another proxy certificate) for the purpose of providing restricted delegation of rights within a PKI-based system. A proxy certificate contains a new extension called `ProxyCertInfo` which differentiates it from a standard public key certificate. Within the `ProxyCertInfo` extension, there exists a `ProxyPolicy` field that specifies some policies on the use of the certificate for the purposes of authorization [176]. Proxy certificates can be very useful for enabling single sign-on and rights delegation in situations where the user wants to access certain remote servers in an unattended fashion. Reasons for wishing to do this include a potentially long processing time. Another important motivation for using proxy certificates is to limit the exposure of long-term credentials (which normally need to be updated yearly) by using proxy credentials with much shorter lifetimes (typically on the order of hours or days).

The process of a user A creating a proxy certificate for herself, as described in [176], can be summarised as follows:

1. A generates a new public/private key pair.
2. The key pair is used to create a request[11] for a proxy certificate.

[10] When we say certificate B is derived from certificate A, we mean certificate B is digitally signed using the private key related to certificate A.

[11] A certification request consists of three parts: (i) certification request information; (ii) a signature algorithm identifier; and (iii) a digital signature on the certification request information. The certification request information consists of the entity's DN, the entity's public key, and a set of attributes providing other information about the entity [107].

3. A proxy certificate, signed by the long-term private key associated with A's long-term public key (or a short-term private key associated with another proxy certificate), is created in response to the request. During this process, the proxy certificate request is verified to ensure that the requested proxy certificate is valid.

When a proxy certificate is created as part of a delegation from A to B, this process is modified by having B perform steps 1 and 2, then B passes the request to A over an authenticated and integrity protected channel. Subsequently, A performs step 3 and passes the proxy certificate back to B. For proxy certificates, path validation is very similar to standard path validation of public key certificates, as specified in [176]. Note that a valid path begins with a standard public key certificate that has already been validated by public key certificate validation procedures specified in [101].

Transport Layer Security (TLS). The TLS protocol [50] was developed from the Secure Sockets Layer (SSL) protocol version 3.0 [72], which was in turn developed by Netscape. It is currently one of the most widely used secure communications protocols on the Internet. The primary goal of the TLS protocol is to provide data confidentiality and integrity for communications between two parties, a client and a server. The protocol comprises two layers: the TLS record protocol and the TLS handshake protocol.

The TLS handshake protocol allows the server and the client to authenticate each other and to negotiate a key exchange method, cryptographic algorithms, and keying material from which further keys can be derived as necessary. Figure 2.5 shows a flow chart representing the TLS handshake protocol. The protocol involves the following steps [50]:

1. Exchange hello messages to agree on cryptographic algorithms, exchange random values and check for session resumption[12].

[12] Both the client and the server can create a new session or resume an old session if they both still store the old session identifier and state. If they are willing to establish a new connection using the resumed session state, both parties must proceed directly to the `Finished` messages.

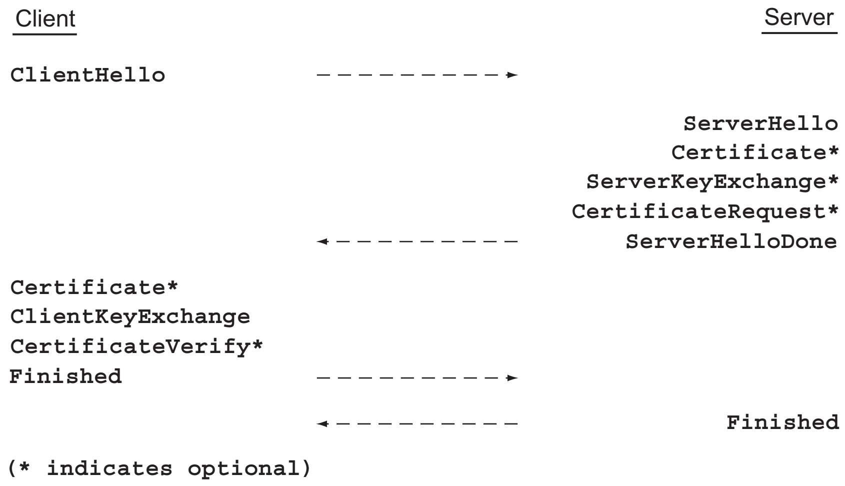

Figure 2.5: Message flows for a full TLS handshake [50].

2. Exchange the necessary cryptographic parameters to allow the client and the server to agree on a pre-master secret.

3. Exchange certificates and cryptographic information to allow the client and the server to authenticate themselves.

4. Generate a master secret from the pre-master secret and exchanged random values.

5. Provide security parameters to the record layer, such as symmetric encryption and MAC algorithms, a compression method, a master secret, and the exchanged random values.

6. Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

On the other hand, the TLS record protocol provides the actual security for communications between the client and the server. It preserves confidentiality and integrity of data encapsulated in the record layer.

We will discuss more details about the contents of the TLS handshake protocol messages when we present our proposals in Chapters 4 and 5.

The RSA Cryptosystem. The first algorithm known to be suitable for public key encryption, as well as signing, is the RSA cryptosystem [151].

In setting up an RSA cryptosystem, each user performs the following steps to generate his public/private key pair.

1. Generate two large distinct random primes p and q, where $|p| \approx |q|$.
2. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
3. Select a random integer e, where $1 < e < \phi(N)$ and $\gcd(e, \phi(N)) = 1$, where gcd denotes the greatest common divisor.
4. Compute the unique integer d such that $ed \equiv 1 \bmod \phi(N)$ and $1 < d < \phi(N)$.
5. The public key is (N, e) and the private key is d.

Suppose Alice wants to encrypt a message m for Bob. Alice creates the ciphertext c by computing $c = m^e \bmod N$, where (N, e) is Bob's public key. To decrypt the ciphertext, Bob calculates $m = c^d \bmod N$. The relationship between e and d ensures that Bob can correctly recover m by performing this calculation.

If Alice wants to sign a message with her private key, she can create a digital signature $s = m^d \bmod N$. Here d is Alice's private key. To verify the signature, Bob checks if the message m can be recovered by calculating $s^e \bmod N$. Here e can be obtained from Alice's public key (N, e). Note that the given example is a "naive" RSA cryptosystem. Practical implementations must employ secure padding schemes and hash functions to prevent various types of attacks, see for example [18].

In practical applications, it is common to choose a small public exponent for the public key. This makes encryption faster than decryption and verification faster than signing. With the typical modular exponentiation algorithms used to implement the RSA algorithm, public key operations take $\mathcal{O}(k^2)$ steps, private key operations take $\mathcal{O}(k^3)$ steps, and key generation takes $\mathcal{O}(k^4)$ steps, where k is the number of bits in the modulus [154].

2.2.3 Emerging Security Technologies for Grid

The Open Grid Services Architecture (OGSA) introduces both new opportunities and new challenges for grid security. It makes sense that OGSA security should leverage as much as possible the existing and emerging web services security standards, and to augment these only when needed. Emerging web services security specifications address standard methods for authentication and establishment of security contexts and trust relationships in WS-SecureConversation [93] and WS-Trust [94]; standard formats for security token exchange in WS-Security [132] and SAML [37]; and expression of web service security policy in WS-Policy [156] and XACML [129]. These specifications have been exploited by grid developers to create uniform and inter-operable methods to be used in grid security. For more details and examples of OGSA security and web services security, see [162].

We remark that the adoption of web services security technologies is orthogonal to the application of identity-based cryptography in grid security. The use of web services seems to be mainly driven by the critical need for inter-operable and scalable grid systems. Therefore, integration between web services security and identity-based cryptography will not be covered at length in this thesis. We only intend to discuss the possible impacts of identity-based cryptography on the underlying fundamental XML security technologies from which web services security are built. These include XML Signature and XML Encryption. In the remainder of this section, we will discuss some basic features of XML Signature and XML Encryption, and the message-level security that is provided by WS-Security.

XML Signature. XML Signature [53] provides an XML syntax defined for digital signatures. An XML Signature can be applied to some or all the content of one or more XML documents or SOAP messages (refer back to page 25 for the definition of SOAP). Also, a single XML document can contain multiple XML Signatures. They are represented by a `Signature` element which has the structure shown in Figure 2.6. (Notation: ? denotes zero or one occurrence; + indicates one or more occurrences; and $*$ represents zero or more occurrences.)

Generating an XML Signature involves two phases: reference generation and sig-

```
<Signature Id?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI?>
      (<Transforms>)?
      <DigestMethod>
      <DigestValue>
    </Reference>)+
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

Figure 2.6: The shorthand XML schema for `<Signature>`.

nature generation. Given a data object, the reference generation operation applies a `Transform` algorithm to the data object, calculates a digest value over the resulting data object, and creates a `Reference` element. Subsequently, the signature generation operation creates a `SignedInfo` element. It then performs canonicalization[13] on the `SignedInfo` element before signing it with the algorithms specified in `SignedInfo`. It then constructs a `Signature` element.

Verifying an XML Signature follows the following process. Given the `SignedInfo` element, the reference validation operation performs canonicalization and obtains the data object pointed to in the `Reference` element. It applies a `Transform` algorithm and computes a digest value with the algorithm specified in the `Reference` element. The calculated digest value is compared with the `DigestValue` in the `Reference` element. The verification fails if there is any mismatch. Subsequently, the signature verification operation obtains the verification key from the `KeyInfo` element. Using the output of the `SignedInfo` canonicalization, it then runs the algorithm specified in the `SignatureMethod`, taking as input the `SignatureValue`. The signature verification passes if and only if the resulting value matches the value in the `SignedInfo` element.

[13] Canonicalization is a process of converting data that has more than one possible representation into a standard representation. This is to ensure that two similar data objects with different XML data representations will give the same digest value. This is essential for preserving data integrity, a requirement for the signature scheme to work properly.

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievalMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>
```

Figure 2.7: The shorthand XML schema for `<EncryptedData>`.

XML Encryption. Meanwhile, XML Encryption [54] is designed to keep all or part of a SOAP message secret. It also allows a sender to encrypt different sections of an XML document with different keys, thus making different sections of the document available to different recipients. Figure 2.7 shows the shorthand schema for an XML Encryption.

In many ways, the `EncryptedData` element is simpler than the `Signature` element because it is more self-contained. The `EncryptedData` element represents a single resource being encrypted and it does not contain pointers to multiple items. On the other hand, the `Signature` element contains a collection of pointers that represent what is being signed.

To encrypt XML data, the XML Encryption scheme first chooses an encryption algorithm for the `EncryptionMethod` element. It then specifies an encryption key and serialises the message data to obtain octets that are input to the encryption algorithm. Then the actual encryption of the data is performed and the result is placed in the `CipherData` element before the associated `EncryptedData` element is built.

For decryption, the XML Encryption scheme obtains the necessary information for decryption such as the encryption algorithm and the associated key information from the `EncryptedData` element. The corresponding encryption key is extracted from the `KeyInfo` element before the actual decryption of the `CipherData` takes place.

SOAP Message Security. The confidentiality and integrity of SOAP messages can be protected through WS-Security. The aim of WS-Security is to use existing technologies to provide a common format for security of SOAP messages. Message-level security protects messages in a different way from transport-level security. The former allows each message to be encrypted or signed independently and thus self-protected. Thus, message protection persists wherever the message travels. However, the latter only provides security to messages transmitted from one point to another in a secure channel, e.g. established using the TLS protocol. The downside of the flexibility that message-level security offers is that it results in a rather complex message that employs many security standards such as XML Signature, XML Encryption, and a common security token such as an X.509 certificate. This may raise considerable performance issues when transporting secure SOAP messages from one entity to another. The performance may be worse when the message has multiple signatures or data encrypted with different keys targeted at different recipients.

Security-related information targeted at a specific recipient is provided through a SOAP security header. There are three types of elements that may make up the header:

(i). Security tokens: These are data used for authentication or authorization. Examples of security tokens are username/password, X.509 certificate and Kerberos ticket.

(ii). Signatures: These are usually obtained as the result of signing part or all of the SOAP body using XML Signature.

(iii). `ReferenceList` or `EncryptedKey`: This is a reference list that contains references to all the possible different `EncryptedData` elements, or key information used by a WS-Security processor to decrypt the encrypted data.

Many grid scenarios require end-to-end message protection over a route that traverses one or more intermediate components such as firewalls or routers. To meet the needs of these scenarios, message-level security is preferred over transport-level [162]. A message may need to traverse a few intermediate components before reaching its intended recipient. As we have described earlier, message-level security has the flexibility of protecting only parts of the message. That means the information that an intermediate component requires to process the message can be transmitted in clear while sensitive data can still be encrypted.

2.3 Globus Toolkit

The Globus Toolkit (GT) [62, 81] is currently the *de facto* standard open source software toolkit for building grid systems. In the late 1990s, GT2, which uses protocols that leverage existing Internet standards for transport, resource discovery, and security, pioneered the creation of inter-operable grid systems and enabled significant progress on grid programming tools. In 2003, GT3 which implemented the emerging OGSA and extended GT2 technologies was released. OGSA aligns grid development with broad industry initiatives in service-oriented architectures and web services. Grid services employed in GT3 are modified web services to suit the statefulness and other requirements needed for grid applications. This has resulted in some drawbacks such as incompatibility of grid services with certain web services tools because of the object-oriented nature of grid services. Therefore, WSRF, another new grid standards has been specified to define a generic and open framework for modelling and accessing stateful resources using web services. The first WSRF implementation in GT4 was made available in April 2005.

GT4 comprises various components/services to support grid applications. These can be summarised as follows:

- *Common runtime* is a component which provides a set of fundamental libraries and tools which are needed to build both services (non-web services) and web services.
- *Security* consists of security services such as authentication, authorization and delegation that are supported by the Grid Security Infrastructure (GSI). These security services ensure secure job submissions and data transmissions between

grid entities.

- *Data management* facilitates management of large sets of data within virtual organisations by using tools such as GridFTP (Grid File Transfer Protocol) and RFT (Reliable File Transfer).
- *Information services* are commonly referred to as the Monitoring and Discovery Services (MDS). These contain a set of components used to discover and monitor resources within a grid environment.
- *Execution management* deals with initiation, scheduling, monitoring, coordination of jobs through the Grid Resource Allocation and Management (GRAM) service and other components.

In what follows, we give a brief overview of two fundamental components in the GT that are relevant to this thesis: GRAM and GSI.

2.3.1 Key Concepts of GRAM

The Grid Resource Allocation and Management (GRAM) service provides a single interface for requesting and using remote system resources for execution of jobs. The most common use of GRAM is remote job submission and control. Figure 2.8 shows an overview of the basic components of the GRAM service. With GT4 GRAM, there will be one or more compute elements that reside on each resource. To invoke a job using GRAM, a user creates a job request, describes the job to be run, and sends it to a GRAM adapter running on the resource using a Java container[14]. The GRAM adapter maps the request onto an appropriate request to a local scheduler. The local scheduler has an interface called Managed Job Factory (MJF) which has access to a MJF WS-Resource[15]. The MJF can create one or more managed job instances to execute the job(s) submitted by the user. Each new managed job in turn has a WS-Resource which will be given a handle, called an endpoint reference (EPR). The user can use this handle to query his job's status, terminate the job,

[14] A visual GUI window which can be used to hold Java components.

[15] A WS-Resource is used to represent state associated with a web service, e.g. a Managed Job instance. This allows other components to use WSRF and other specifications such as WS-Notification and WS-Addressing to access state information of the web service, manage lifetime, perform notification and so forth.

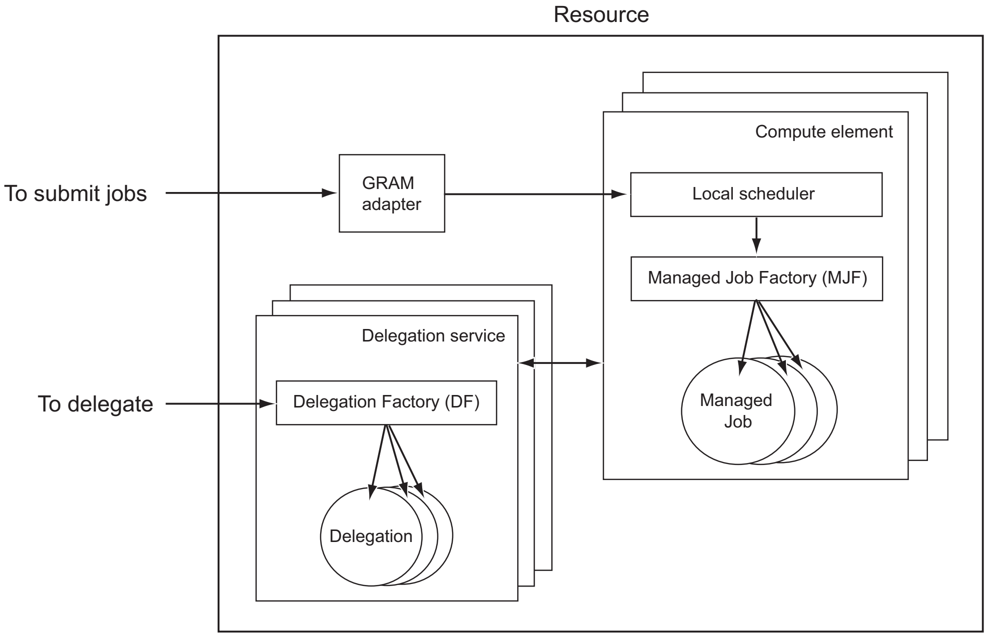

Figure 2.8: Basic components of the GT4 GRAM service.

obtain notifications of output produced by the job and so on [83]. This handle can be communicated to other entities who can perform the same operation if authorized to do so.

In GT4, a credential delegation mechanism can be used by the user to delegate his credential to the GRAM service for additional remote operations. This credential delegation service is managed by a Delegation Factory (DF) which has access to a DF WS-Resource that stores information about the certificate chain associated with the DF. The DF can in turn create one or more delegation instances. Each Delegate WS-Resource contains a credential. Note that the user can either forward his credential together with the job request, or upload it to a delegation service associated with the GRAM service. The latter allows re-use of the user credential for multiple short-lived jobs.

2.3.2 Key Concepts of GSI

The security services provided by the GT rely upon a security architecture called the Grid Security Infrastructure (GSI)[16]. This is based on PKI technology. The focus of the GSI is primarily on authentication, message protection, and single sign-on and credential delegation through proxy credentials [69, 146, 186, 187].

In grid applications which employ the GSI, each entity is assigned a unique identity or DN and issued a public key certificate signed by a Grid CA. As of 30 August 2005, there were approximately 35 CAs[17] operating in the European DataGrid (EDG) and Enabling Grids for E-SciencE (EGEE) projects, roughly at the scale of one CA for each nation. These CAs service grid users which comprise mostly researchers and scientists throughout the whole continent of Europe. Note that no root CA has been setup and trust relationships between CAs are established through the European Grid Policy Management Authority (PMA). Depending on the number of the users, there may be at least a few Registration Authorities (RAs), typically key persons from research institutions who lead the development and maintenance of the projects in each nation, who assist the CA(s) in user validation and registration. For example, there are currently about 50 RAs[18] in the UK who work closely with the UK e-Science CA.

Public key certificates are used to support authentication and key agreement protocols, such as the TLS protocol. In addition, proxy certificates are also used for single sign-on and delegation. Before a user submits a job request, he must create a proxy certificate which includes generating a new public/private key pair and signing the proxy certificate with his long-term private key. This newly created proxy certificate can then be used for repeated authentication with other grid entities, without the need to access the user's long-term private key. For rights delegation from a user A to a target service provider X, three steps are required: (i) X creates a new public/private key pair and sends a request (that is signed with the new private key) to A; (ii) A verifies the request using the new public key, creates a new proxy certificate

[16] GSI was originally an acronym for Globus Security Infrastructure in [69]. It later came to be more commonly known as Grid Security Infrastructure when the GT was gradually becoming the *de facto* toolkit in the grid community.

[17] Source: `http://marianne.in2p3.fr/datagrid/ca/ca-table-ca.html`, last updated on 30 August 2005.

[18] Source: `https://ca.grid-support.ac.uk/`, last updated on 5 September 2005.

and signs it with her current proxy credential (short-lived private key); and (iii) A forwards the new proxy certificate to X [186]. This three-step delegation process is based on the proxying process described on page 36. Note that A can impose some constraints in the `ProxyPolicy` field of the proxy certificate, which specifies what X can and cannot do. A has to trust that an entity to which X presents this proxy certificate will impose the constraints.

In the GSI setting, each user has a long-term RSA public/private key pair with $N = 1024$ bits. The short-term keys for the user's proxy credential have only 512-bit moduli. This substantial reduction of key sizes is driven by the fact that an RSA key generation is a computationally expensive operation. It is shown in [186] that generating a key pair with 512-bit moduli can reduce the processing time by approximately 77% of the time required for a 1024-bit key pair. Given the latest progress in algorithms for factoring integers, a 512-bit modulus provides only marginal security from a concerted attack. For longer-term security, 1024-bit or larger moduli should be used [127]. However, since the proxy credential has a relatively short lifetime, it is believed that the reduction in security implied by using only 512-bit moduli poses an acceptably low risk in grid systems. In terms of key management, it is the responsibility of users to keep control of their respective private keys. Depending on the Grid CA's policy, a user's long-term public/private key pair and its associated certificate are usually updated and renewed yearly. Should a user realise or suspect that his private key has been compromised, then the holder himself is responsible for the notification of the exposure to the Grid CA, in order to have his certificate revoked. To improve accessibility of the user's private key from anywhere at anytime and to increase protection of the key, online credential repositories, such as MyProxy [15, 137], have been developed. The MyProxy server stores users' long-term credentials and it is anticipated that the server receives round-the-clock monitoring from security specialists. This will be discussed more in Section 2.4.

The TLS protocol is one of the most important security enablers for PKI-based grid architectures such as the GSI. The GSI has been built on the Generic Security Service Application Program Interface (GSS-API) [119] and incorporates GSI-enabled OpenSSL [140] to support proxy certificates. Examples of the RSA-based cipher suites[19] are `TLS_RSA_WITH_RC4_128_MD5`, `TLS_RSA_WITH_DES_CBC_SHA` and so on.

[19] A cipher suite defines a cipher specification supported in the TLS protocol.

The following steps, adopted from [187], describe how the GSI facilitates a secure job invocation from a user to a remote resource through a GRAM service.

1. The user creates a job instantiation request and signs it with his proxy credential. The signed request and the user's credentials (both long-term and short-term) are sent to a GRAM adapter on the target resource.

2. The GRAM adapter verifies the signature on the request and the identity of the user who sent it based on the credentials that it received. It then determines the local account (handled by a local scheduler) in which the job should be run based on the requestor's identity using the *grid-map file*[20].

3. The local scheduler passes the job request to the MJF. Once the MJF starts up, it uses the Grid Resource Identity Mapper (GRIM)[21] to generate a set of proxy credentials. This set has embedded in it the user's grid identity, local account name, and local policy to help the user verify that the MJF is appropriate for his needs.

4. The MJF also verifies the signature on the request to ensure that it has not been tampered with and validates if the requestor is authorized to run in the local user account. When all checks pass, the MJF instantiates a managed job service for the job request and returns an EPR to the user.

5. The user connects to the managed job service to initiate the job. The user client and the managed job service perform mutual authentication using the TLS protocol, with the user using his proxy credential and the managed job service using the proxy credential that the MJF acquired from GRIM. The user verifies the managed job service as having a credential issued by an appropriate host and containing a grid identity matching its own, thus ensuring that the managed job service is running on the right host and the correct account. If all checks succeed, a secure TLS channel is established between the user and the managed job service for further data exchange. At this point, the user may, at his discretion, delegate his credential to the managed job service for later use when necessary. A delegation service is created by the DF to store

[20] A local configuration file containing mapping from global identities to local identities.

[21] A setuid program that accesses the local host credentials and uses them to generate a set of proxy credentials for the MJF.

the user's credential. The WS-Resource associated to this service also has a unique EPR which will be returned to the requestor.

Should the user wish to submit multiple short-lived jobs, he can upload his proxy credential to the Delegation service and re-use it before its expiry.

In this thesis, we will re-use the above secure job invocation process when presenting our alternative proposals for grid security infrastructures in Chapters 4 and 5. Our main focus will be on tackling some of the issues encountered in grid-supported PKI with various alternative key management techniques. We examine security services such as authentication, single sign-on, delegation and key agreement that are normally supported by PKI and certificates. We will also use one of the simplest access control methods through a grid-map file in our discussion for the sake of completeness, even though the GT also offers a more complicated authorization component called Community Authorization Service (CAS) [146, 147].

GT4 supports both transport-level and message-level security. The former, which is the default option, entails SOAP messages being sent over a network connection through a secure TLS channel, while the latter uses the WS-Trust and WS-SecureConversation specifications to protect data confidentiality and integrity of SOAP messages. The recommended use of transport-level security instead of fully web-services based security is driven by the relatively poor performance of message-level security implementations [188]. It is not clear when this issue will be solved. This in turn motivates our focus on transport-level security throughout the thesis.

2.4 MyProxy

There are two common approaches to protecting a private key in a PKI. First it can be protected by storing it in a file with restricted access, e.g. storing it in an encrypted file with a decryption passphrase. Alternatively, the private key can be stored on a hardware token (e.g. a smart card) that is usually protected by a PIN. While it seems that this method provides the best security, it may be an expensive option due to the need to deploy hardware support for using the token.

The second approach is to limit the exposure of the private key by regularly issuing a new short-lived public/private key, usually on the order of hours or days. It is a convention in cryptography and computer security that if a cryptographic key is short-lived, then the level of protection which is afforded to it can be eased. For example, the temporary private key can be stored unencrypted on a local file system, protected only by file system permissions. Currently, the majority of grid applications use proxy credentials in the form of X.509 proxy certificates for single sign-on and delegation services. Even so, the limited security of typical desktop systems may put users' long-term private keys at risks through hacking, trojan horses, viruses and other types of malicious software.

MyProxy [15, 137], a web-based grid portal, has been designed as an online credential repository which stores users' long-term credentials. This seems to better prevent attackers from stealing users' private keys, as it is expected that a MyProxy system will receive round-the-clock monitoring from security specialists. Whenever a user wants to create a proxy credential, he must authenticate himself to the MyProxy server using a passphrase which he shares with the MyProxy server, and request a new proxy credential to be issued to him. Furthermore, being web-based, the MyProxy server can be accessed from any computer with Internet access.

The core protocol for the MyProxy system between a user and the MyProxy Server is as follows [15].

1. The user establishes a TCP connection to the server and initiates a server-authenticated TLS handshake protocol. A full TLS handshake is not mandatory as in most cases, a user does not have an existing X.509 credential.

2. Once the TLS handshake is complete and a secure channel is established, the user sends a request message to the server. The request contains the protocol version, the command (e.g. retrieve, store, or remove a proxy credential), a username, a passphrase (an ASCII password used to protect the stored proxy credential) and a lifetime.

3. If all checks succeed, the server will return '0' to indicate success or '1' with an error text that suggests otherwise.

Subsequently, should the user want to retrieve a new proxy certificate, he can generate a new public/private key pair and forward the public key to the server through the established secure channel. The server will then create a new proxy signed with the user's stored private key and return it to the user. Note that the MyProxy server encrypts the user's long-term private key in the MyProxy repository with the user-chosen passphrase included in the request message using an authenticated symmetric encryption scheme. The passphrase is not stored in the MyProxy repository. Thus, the client must send the correct passphrase to the server so that it can decrypt the private key correctly.

Recently, there has been an increasing interest in and support for one-time passwords in the MyProxy system. These mitigate the risk of passwords being stolen by keystroke loggers, trojan horses, or other means.

2.5 Example Scenario

Here, we give a simple grid scenario, as shown in Figure 2.9, which captures some of the common security requirements within a grid environment.

Alice, who is a member of CryptoGrid, wants to run a differential cryptanalysis simulation which may take days to complete. She first accesses a trusted MyProxy server where she has stored her long-term credential, to request a fresh proxy cer-

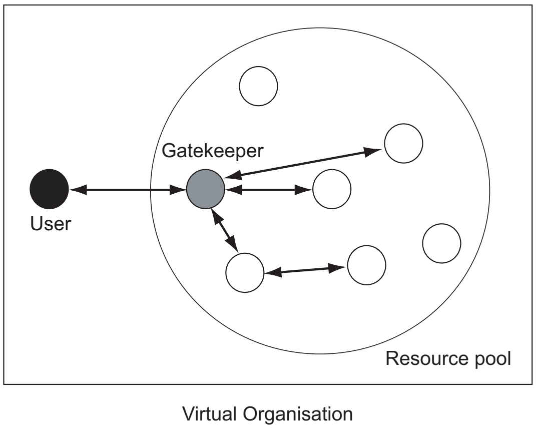

Figure 2.9: Job invocation and interaction within a grid environment.

tificate. She then submits her job to the CryptoGrid Gatekeeper through a Globus GRAM client. The Gatekeeper verifies Alice's signed request and checks if Alice is authorized to access the resources that CryptoGrid provides by consulting its local grid-map file. Once the check passes, Alice and the Gatekeeper perform mutual authentication using their respective long-term and proxy certificates. Subsequently, Alice delegates her credential to the Gatekeeper through the secure channel that they have established by signing a new proxy certificate which contains the Gatekeeper's short-term credential. Every time Alice's job needs to gain access to a new resource, the Gatekeeper will present this certificate to prove that it is acting on Alice's behalf.

Based on the job description, the Gatekeeper queries a local replica catalogue to determine suitable resources to run the simulation. Once that has been located, the Gatekeeper passes on the job description to a hosting server. Before the Gatekeeper submits the job to the hosting server, it must perform mutual authentication with the server using the delegated credential from Alice. In many cases, the Gatekeeper may need to further delegate Alice's credential to the hosting server which requires access to other resources to complete Alice's job. For instance, the job may require additional data from a database server. Also, Alice may monitor the progress of the job and possibly change her mind about where or how it is executing. Upon completion of the job, Alice will be notified by the Gatekeeper and the results will be sent back to her. Finally the Globus GRAM client will clean-up information in the job submission scripts and remove temporary settings that coordinated the job.

2.6 Summary

We are moving into a future in which the physical location of computational resources does not really matter. It is believed that VOs have the potential to dramatically change the way we use computers to solve complex and resource intensive problems. In this chapter, we have given an introduction to the fundamental concepts of grid computing. We also discussed the role of web services in grid computing and outlined some applications. Security issues in grid computing have been identified as major challenges in making computational grids widely used infrastructures. We have examined these issues, along with the security requirements that they lead to. We have also outlined the security technologies in use in current grid systems. To give a better view of actual grid deployment and usage, we also included some illustrations of the Globus Toolkit, its security components and an example scenario.

Chapter 3

Identity-Based Cryptography

Contents

This chapter provides a background study of identity-based cryptography. We review some basic concepts of pairings and some cryptographic primitives used in identity-based cryptosystems. We also discuss the performance and implementation issues for identity-based cryptographic schemes, which may have an impact on the practicality of the schemes. Some existing applications of identity-based cryptography are presented to help in the understanding of the prospects for and benefits of identity-based cryptosystems.

3.1 A Short History

Identity-based cryptography (IBC) was first introduced by Shamir [159] in 1984. Instead of generating and using a random public/private key pair in a public key cryptosystem such as RSA, Shamir conceived the idea of using a user's name or his network address as a public key, with the corresponding private component being generated by a trusted key generation centre. In fact, any type of identifier, e.g. email address, social security number, telephone number and so forth, can be used, so long it can uniquely identify the user and is readily available to the party that uses it. The main motivation for this approach is to eliminate the need for certificates and the problems that they bring. Since a user's public key is based on some publicly available information that uniquely represents the user, an identity-based cryptosystem can do away with public key directory maintenance and certificate management. Despite the novel and ambitious conception, Shamir was only able to develop an identity-based signature (IBS) scheme based on the RSA primitive. The construction of an identity-based encryption (IBE) scheme was left as an open problem. Since then, there were numerous attempts to realise Shamir's vision of identity-based encryption, such as those in [49, 124, 139, 171, 175, 178]. However, none of these proposals were fully satisfactory. Either they did not provide adequate security or they were not feasible to implement in practical environments. Meanwhile, there were further proposals for IBS schemes in [57, 95], also based on the RSA primitive.

Only in the early 2000's did the emergence of cryptographic schemes based on pairings on elliptic curves result in the construction of a feasible and secure IBE scheme. This area began with the novel work of Sakai *et al.* [155] on pairing-based key agreement protocols and signature schemes, and subsequent work on the three-party key agreement protocol by Joux [106]. Boneh and Franklin [25] then presented the first practical and secure IBE scheme based on the Weil pairings. These three key contributions have stimulated the development of a wide range of pairing-based cryptographic schemes and protocols. Following the publication of [25] (an extended version appears in [26]), a number of IBS schemes [39, 99, 142] and hierarchical identity-based encryption (HIBE) and signature (HIBS) schemes [80, 100] were proposed. Also, proposals for identity-based authenticated key agreement (IAKA) protocols (e.g. [42, 163]), identity-based signcryption (IBSC) schemes (e.g. [32, 112, 122]) and many other identity-based cryptographic schemes soon appeared in the literature. It is worth noting that apart from Boneh and Franklin's seminal work, there is also another feasible and secure solution for IBE due to Cocks [47]. The security of Cocks' scheme is based on the Quadratic Residuosity problem. However, although the scheme of [47] is computationally viable to implement, it is generally considered very costly in terms of communication overheads due to significant message expansion. More background on cryptography from pairings can be found in [143] and a good source of research papers on this subject is available at [11].

3.2 Certificated-Based PKI and Identity-Based PKI

In Chapter 2, we gave a review of the conventional certificate-based PKI that supports the widely used RSA encryption and signature schemes. Here we are going to describe, at a relatively high-level, how an identity-based PKI works and what its key differences from the traditional PKI are. The subsequent sections will then delve into more details, uncovering some of the underlying concepts concerning pairings and describing the identity-based cryptographic primitives that we will employ throughout this thesis.

We begin by looking at a simplified version of the Boneh-Franklin IBE scheme. This is defined by four algorithms, as follows:

Setup: Given a security parameter, the algorithm generates a set of system parameters (which will be made public) and a master secret (which it keeps private).

Extract: This algorithm is run to extract the private key corresponding to a given public key. It takes the system parameters, the master secret and an arbitrary identifier ID (public key string) as input, and returns a private key.

Encrypt: This algorithm uses the system parameters and an ID to encrypt a message, and generates a ciphertext.

Decrypt: Using the system parameters, a private key and a ciphertext as input, this algorithm returns a plaintext (or possibly an indication that the decryption process has failed).

The Setup and Extract algorithms are normally executed by a Private Key Generator (PKG), while the Encrypt and Decrypt algorithms are carried out by users. The PKG in turn will be managed and controlled by a Trusted Authority (TA), a trusted third party roughly equivalent to a CA in a traditional PKI. Suppose that Alice wants to send a message secretly to Bob using an IBE scheme. She does not need to first verify the authenticity of Bob's public key by retrieving Bob's public key certificate (which must take place in a conventional PKI). Instead Alice simply encrypts the message with Bob's ID, e.g. '`bob@example.com`'. Clearly, Alice needs to know the system parameters of Bob's TA. If Bob does not already possess the corresponding private key, he has to obtain it from his TA. If the TA is satisfied that Bob is a legitimate receiver, it takes its system parameters, master secret and Bob's ID to extract a private key, which will then be used by Bob to decrypt the ciphertext.

The major technical difference between a certificate-based PKI and an identity-based PKI is the binding between the public/private keys and the individual. This can be achieved by using a certificate in the traditional PKI. In the identity-based setting, the public key is bound to the transmitted data while the binding between the private key and the individual is managed by the TA. A public key based on an identifier can be constructed on-the-fly at any time, even before its matching private component is computed. In terms of key generation, the conventional PKI allows either a user or his CA to create public/private key pairs. However, it is only the TA that can compute private keys in the identity-based setting. This inevitably implies that an identity-based PKI has an escrow facility, which may or may not be desirable. Boneh and Franklin suggested in [25] that key escrow can be circumvented by using multiple TAs and threshold cryptography. On the other hand, because of this built-in feature, the user always needs to set up an independent secure channel with his TA for retrieving private key material. For key revocation, Boneh and Franklin proposed the use of date concatenated with the user's identifier to achieve automated key expiry. This may obviate the need for a revocation mechanism. However, it has the disadvantage of increasing the TA's workload, since the TA is required to regularly generate private keys and deliver them to its users. We will examine whether this key revocation approach is sufficient for grid applications in Chapter 4. Table 3.1 summarises the above comparison between traditional PKIs and identity-based PKIs. A more detailed discussion comparing the two architectures can be found in [144]. For discussion of inter-operation between the certificate-based and the identity-based settings and some of the issues that may

Table 3.1: Key differences between a certificated-based PKI and an identity-based approach.

Feature	Certificate-based PKI	Identity-based PKI
Public key generation	Using random information	Using an explicit identifier
Private key generation	By a user or the CA	By the PKG
Key certification	Yes	No
Key distribution	Requiring an integrity protected channel for distributing a new public key from a user to his CA	Requiring an integrity and privacy protected channel for distributing a new private key from the TA to its user
Public key retrieval	From a public directory or from the key owner	On-the-fly based on the key owner's identifier
Escrow facility	No (except when key generation is run by the CA)	Yes

then arise, see [148].

There are applications which do not tolerate key escrow as in the case of an identity-based cryptosystem. This and other traditional PKI issues such as key revocation have inspired other new models of infrastructures for supporting public key cryptography. These include the certificate-based encryption scheme of Gentry [79] and Al-Riyami and Paterson's certificateless public key cryptography [5]. In Gentry's model, a user's private key comprises two components: (i) a component which is selected by the user and is kept private; and (ii) a component which is time-dependent and issued by a CA on a regular basis. The second component can be sent across a public channel. The first matching public component can be made publicly available in the same manner as a standard public key, whereas the second public component can be computed by other users using only public parameters of the scheme. This approach does not suffer from the key revocation and escrow problems that both traditional PKIs and identity-based PKIs encounter, assuming that the CA issues a partial private key to its users in every time period, e.g. hour or day. Al-Riyami and Paterson combined elements from the traditional PKI and the identity-based approach, with a technical approach that is rather closely related to that of [79]. A user's private key in their certificateless setting also consists of two components: (i) an identity-dependent partial private key (generated in the same way which the normal identity-based approach does); and (ii) a full private key which can be produced using the partial private key and some secret known only to the user. Succinctly, both proposals in [79] and [5] make use of a user's input in addition to the CA/TA's for generating a private key, and thus the possibility of key escrow is eliminated. In a way, key revocation in these proposals has become simpler than in the traditional approach, provided the CA/TA can afford to withstand the burden of producing partial private keys regularly for its users.

3.3 Mathematical Preliminaries

In this section, we give a brief introductory mathematical background on elliptic curves and pairings. We also establish some computational problems which will be used throughout the thesis.

Let p be a large prime, m an integer with $m \geq 1$, and let $\mathbb{F}_{p^m}$ be the finite field with p^m elements. So p indicates the *characteristic* of the field and m denotes its *extension degree.* The multiplicative group of $\mathbb{F}_{p^m}$ is denoted as $\mathbb{F}^*_{p^m}$.

Let $E/\mathbb{F}_{p^m}$ be an *elliptic curve* over the field $\mathbb{F}_{p^m}$. This is commonly defined by an equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_i \in \mathbb{F}_{p^m}$ for $i = 1, 2, 3, 4, 6$. A point $P = (x, y) \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m}$ is said to be on the curve if (x, y) satisfies the above equation. Associated with the curve $E/\mathbb{F}_{p^m}$ is an additive Abelian group $(E(\mathbb{F}_{p^m}), +)$ whose elements are the points on the curve together with a null element (or a point at infinity), denoted ∞. The size of $E(\mathbb{F}_{p^m})$ is called the *order* of the curve over the field $\mathbb{F}_{p^m}$, denoted $\#E(\mathbb{F}_{p^m})$.

Suppose that $E(\mathbb{F}_{p^m})$ has a cyclic subgroup $\mathbb{G}_1$ of order q, for some prime q. Define the *embedding degree* (or security multiplier) to be the least integer $k > 0$ such that $q \mid p^{km} - 1$ and $q \nmid p^l - 1$ for all $0 < l < k$. Let $\mathbb{G}_2$ denotes a cyclic subgroup of $\mathbb{F}^*_{p^{km}}$ of order q. An admissible pairing in the context of IBC is a function $\hat{e}$ which maps a pair of elliptic curve points of $\mathbb{G}_1$ to an element of $\mathbb{G}_2$, denoted $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, and which has the following properties:

- *Bilinear*: Given $P, Q, R \in \mathbb{G}_1$, we have

$$\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R) \text{ and } \hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R).$$

  Hence, for any $a, b \in \mathbb{Z}_q^*$,

$$\hat{e}(aP, bQ) = \hat{e}(abP, Q) = \hat{e}(P, abQ) = \hat{e}(aP, Q)^b = \hat{e}(P, Q)^{ab}.$$

- *Non-degenerate*: There exists $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1$.
- *Computable*: If $P, Q \in \mathbb{G}_1$, then $\hat{e}(P, Q)$ can be efficiently computed.

A pairing can be derived from either the modified Weil pairing [25] or the Tate pairing [73]. Even though the Boneh and Franklin IBE scheme was proposed based on the Weil pairing, the Tate pairing is generally preferred since it can be implemented more efficiently than the Weil Pairing [75]. Supersingular curves are suitable for pairing-based cryptosystems since it is possible to have embedding degree $k = 2$, 3, 4 and 6 [75]. In general, smaller values for p^m are preferred because that means arithmetic on the elliptic curve $E(\mathbb{F}_{p^m})$ is faster and transmission of points on the curve requires less bandwidth. Hence, we tend to want to keep our field as small as possible and to gain our security from larger values for k [76]. For example, working on a supersingular curve with characteristic 2, we can arrange to have embedding degree $k = 4$. This may offer better performance for arithmetic operations. To reduce bandwidth requirements, supersingular curves of characteristic 3 which have $k = 6$ may be preferred. See [13] for further discussion on choosing suitable curves for relatively small values of k. We will also return to the issue of selecting curves in Section 3.5.

Using the point compression technique, the size of an element of $\mathbb{G}_1$ is equivalent to the size of its x-coordinate plus one bit (for the two possible corresponding y-coordinates). The operation that involves $P + Q$ where $P, Q \in \mathbb{G}_1$ is called an elliptic curve point addition, while aP where $a \in \mathbb{Z}_q^*$ is called an elliptic curve point (or scalar) multiplication. Note that the point multiplication aP can be computed very efficiently. However, the problem of finding a when given aP is believed to be intractable, when the curve is appropriately chosen. This problem is known as the elliptic curve discrete logarithm (ECDL) problem. The ECDL problem can be harder to solve than the well-known discrete logarithm problem over finite fields of the same size. This is the case because the best current known methods for solving the discrete logarithm problem over finite fields do not generally translate to elliptic curves. It is for this reason that many elliptic curve cryptosystems can operate with smaller key sizes and yet achieve seemingly equivalent levels of security as compared to other public key cryptosystems based on the discrete logarithm problem. Some of the relevant computational problems related to the ECDL problem can be stated informally as follows.

- *Computational Diffie-Hellman (CDH) problem*: Given $\langle P, aP, bP\rangle \in \mathbb{G}_1^3$, where $a, b \in \mathbb{Z}_q^*$, compute abP. We say that the CDH assumption is satisfied if there exists no probabilistic algorithm which can solve the CDH problem with non-negligible advantage within polynomial time, when a and b are selected at random from $\mathbb{Z}_q^*$.
- *Decisional Diffie-Hellman (DDH) problem*: Given $\langle P, aP, bP, cP\rangle \in \mathbb{G}_1^4$ for $a, b, c \in \mathbb{Z}_q^*$, decide whether $c \equiv ab \mod q$. This can be solved efficiently by verifying $\hat{e}(aP, bP) = \hat{e}(P, cP)$, provided we have an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
- *Bilinear Diffie-Hellman (BDH) problem*: Given $\langle P, aP, bP, cP\rangle \in \mathbb{G}_1^4$ for $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc} \in \mathbb{G}_2$. We say that the BDH assumption is satisfied if there exists no probabilistic algorithm which can solve the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ with non-negligible advantage within polynomial time, when a, b and c are selected at random from $\mathbb{Z}_q^*$.
- *Gap Diffie-Hellman (GDH) problem*: Solve the CDH problem in $\mathbb{G}_1$, possibly with the help of an oracle that solves the DDH problem in $\mathbb{G}_1$. We say that the GDH assumption is satisfied if there exists no probabilistic algorithm which can solve the GDH problem with non-negligible advantage within polynomial time.

It is known that the BDH problem is no harder than the CDH problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$. The converse, however, is still open [25]. The reader can consult [23] and [76] for more mathematical background on elliptic curves and pairings, respectively.

3.4 Identity-Based Cryptographic Primitives

We are now ready to give more details of a few selected IBE and IBS schemes. These schemes will be adopted in our grid security proposals later in this thesis. We will not discuss security proofs for these schemes, as these are beyond the scope of the thesis, but we assume the schemes have all been at least peer-reviewed by the IBC research community and deemed sufficiently secure to be used.

3.4.1 The Boneh-Franklin IBE Scheme

The first practical, efficient and provably secure IBE scheme is due to Boneh and Franklin [25]. This scheme quickly led to a new wave of cryptographic research in pairing-based cryptography and IBC in particular. The following four algorithms define Boneh and Franklin's basic IBE scheme:

Setup: Given a security parameter $k \in \mathbb{Z}^+$, the PKG:

1. specifies two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$;
2. chooses an arbitrary generator $P \in \mathbb{G}_1$;
3. defines two cryptographic hash functions, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_1^* \rightarrow \{0,1\}^n$ for some n; and
4. picks a master secret $s \in \mathbb{Z}_q^*$ at random and computes the matching public parameter as sP.

The system or public parameters are $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, sP, H_1, H_2 \rangle$. Here, n will be the bit length of plaintexts.

Extract: This algorithm is run by the PKG to extract a private key $sQ_{\mathrm{ID}} = sH_1(\mathrm{ID})$ when given an arbitrary identifier string $\mathrm{ID} \in \{0,1\}^*$.

Encrypt: To encrypt a message $m \in \{0,1\}^n$ under an identifier ID, the public key used is $Q_{\mathrm{ID}} = H_1(\mathrm{ID})$. The resulting ciphertext is $c = (U, V)$, where $U = rP$, $V = m \oplus H_2(\hat{e}\,(Q_{\mathrm{ID}}, sP)^r)$, and $r \in \mathbb{Z}_q^*$ is selected at random.

Decrypt: To decrypt a ciphertext $c = (U, V)$ encrypted using the identifier ID, the private key used is $sQ_{\mathrm{ID}} \in \mathbb{G}_1^*$. The plaintext m is recovered by calculating $V \oplus H_2(\hat{e}\,(sQ_{\mathrm{ID}}, U))$.

We remark that the above basic IBE scheme is not secure against adaptive chosen ciphertext attacks in the identity-based setting (IND-ID-CCA secure). It is only secure against chosen plaintext attacks (IND-ID-CPA secure), provided the BDH assumption holds [25].

The full version of the Boneh-Franklin IBE scheme, which has security in the sense of IND-ID-CCA, is as follows:

Setup: As in the basic IBE scheme. In addition, the PKG picks two more hash functions, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

Extract: As in the basic IBE scheme.

Encrypt: To encrypt a message $m \in \{0,1\}^n$ under an identifier ID, the public key used is $Q_{\mathrm{ID}} = H_1(\mathrm{ID})$. The algorithm selects a random $z \in \{0,1\}^n$ and sets $r = H_3(z, m)$. The resulting cipertext is then set to be:

$$c = \langle U, V, W \rangle = \langle rP, z \oplus H_2(g^r), m \oplus H_4(z) \rangle,$$

where $g = \hat{e}(Q_{\mathrm{ID}}, sP) \in \mathbb{G}_2$, a value which can be pre-computed.

Decrypt: To decrypt a ciphertext $c = \langle U, V, W \rangle$ encrypted using the identifier ID, the private key used is $sQ_{\mathrm{ID}} \in \mathbb{G}_1^*$. If $U \notin \mathbb{G}_1^*$, reject the ciphertext. The plaintext m is then recovered by performing the following steps:

1. compute $V \oplus H_2(\hat{e}(sQ_{\mathrm{ID}}, U)) = z$;
2. compute $W \oplus H_4(z) = m$;
3. set $r = H_3(z, m)$ and if $U \neq rP$, reject the ciphertext, otherwise accept m as the decryption of c.

The full IBE scheme of Boneh and Franklin makes use of a technique due to Fujisaki and Okamoto [74] to strengthen the basic version. This ensures that the full scheme is IND-ID-CCA secure.

3.4.2 The Cha-Cheon IBS Scheme

Shortly after the appearance of the Boneh-Franklin IBE scheme, a few IBS schemes that make use of the same keying infrastructure as Boneh and Franklin's scheme were published. These include proposals from Paterson [142], Cha and Cheon [39], and Hess [99] with different performance trade-offs.

Here, we illustrate the Cha-Cheon IBS scheme, whose security is related to the CDH problem.

Setup: As in the basic IBE scheme, except that we now replace H_2 with H_6 which is defined as $\{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.

Extract: As in the basic IBE scheme.

Sign: Given a private key sQ_{ID} and a message $m \in \{0,1\}^*$, the signer picks a random number $r \in \mathbb{Z}_q^*$, and computes $U = rQ_{\text{ID}}$, $h = H_6(m, U)$, and $V = (r+h)sQ_{\text{ID}}$. The algorithm outputs $\sigma = (U, V)$ as the signature.

Verify: Given a signature $\sigma = (U, V)$ of a message m signed by a user with an identifier ID, the verifier checks if $\hat{e}(P, V) = \hat{e}(sP, U + hQ_{\text{ID}})$. The signature is accepted as valid if and only if this equality holds.

As mentioned earlier, identity-based cryptographic schemes have built in private key escrow. It is generally, therefore, difficult to achieve true non-repudiation through the use of identity-based signatures. Additional infrastructure or fixes are needed to overcome this issue. For example, it is possible to use multiple TAs and threshold cryptographic techniques [25].

3.4.3 The Gentry-Silverberg HIBE and HIBS Schemes

Horwitz and Lynn [100] first introduced the concept of HIBE to ease the private key distribution problem and improve the scalability of the original IBE scheme proposed in [25]. They proposed a 2-level HIBE scheme with total collusion-resistance at the first level but only partial collusion-resistance at the second level (i.e. users can collude to obtain the master secret of their domain PKG). Shortly after the proposal of [100], Gentry and Silverberg [80] proposed a secure, practical and fully scalable HIBE scheme with total collusion resistance, regardless of the number of levels in the hierarchy. In the HIBC (e.g. HIBE and HIBS) setting, a root PKG is only required to produce private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level.

We describe Gentry and Silverberg's hierarchical HIBE and HIBS schemes as follows. We assume the root PKG is located at level 0.

Root Setup: The root PKG chooses a generator $P_0 \in \mathbb{G}_1$, picks a random $s_0 \in \mathbb{Z}_q^*$, and sets $Q_0 = s_0 P_0$. It also selects cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n, and $H_3 : \{0,1\}^* \rightarrow \mathbb{G}_1$. The root PKG's master secret is s_0 and the system parameters are $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3 \rangle$.

Lower-level Setup: A lower-level entity (lower-level PKG or user) at level t picks a random $s_t \in \mathbb{Z}_q^*$ which will be kept secret.

Extract: For an entity at level t with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, where $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_i \rangle$ is the ID-tuple of the entity's ancestor at level i $(1 \leq i < t)$, the entity's parent computes $P_t = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_t) \in \mathbb{G}_1$, sets the secret point S_t to be $\sum_{i=1}^{t} s_{i-1} P_i = S_{t-1} + s_{t-1} P_t$ (note that S_{t-1} is the parent's secret point given by the parent's ancestor and s_{t-1} is a secret value only known to the parent), and defines Q-values by setting $Q_i = s_i P_0$ for $1 \leq i \leq t-1$. The entity at level t is given both S_t and the Q-values by its parent.

Encrypt: Given a message $m \in \{0,1\}^n$ with the ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, the message can be encrypted by first computing $P_i = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i) \in \mathbb{G}_1$ for $1 \leq i \leq t$; then choosing a random $r \in \mathbb{Z}_q^*$; and the ciphertext is set to:

$$c = \langle rP_0, rP_2, \ldots, rP_t, m \oplus H_2(g^r) \rangle,$$

where $g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2$, a value which can be pre-computed.

Decrypt: Given a ciphertext $c = \langle U_0, U_2, \ldots, U_t, V \rangle$ encrypted using the ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, the ciphertext can be decrypted by computing:

$$m = V \oplus H_2 \left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^{t} \hat{e}(Q_{i-1}, U_i)} \right).$$

Sign: Given a private key S_t and a message $m \in \{0,1\}^*$, the signer with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$ computes $h = H_3(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, m) \in \mathbb{G}_1$ and $\sigma = S_t + s_t h$. The algorithm outputs $\langle \sigma, Q_1, \ldots, Q_t \rangle$ as the signature.

Verify: Given a signature $\langle \sigma, Q_1, \ldots, Q_t \rangle$ of a message m signed by an entity with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, the verifier checks if:

$$\hat{e}(P_0, \sigma) = \hat{e}(Q_0, P_1)\hat{e}(Q_t, h) \prod_{i=2}^{t} \hat{e}(Q_{i-1}, P_i),$$

where $h = H_3(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, m)$. To improve the verification performance, in particular in situations where many signatures from the same signer need to be verified, the values $\hat{e}(Q_0, P_1)$ and $\prod_{i=2}^{t} \hat{e}(Q_{i-1}, P_i)$ can be pre-computed. This means that there will be only two pairing computations when the verification is performed.

We remark that we have only presented a basic HIBE scheme. The full version of the Gentry-Silverberg HIBE scheme, which has security in the sense of IND-ID-CCA, is as follows:

Setup: As in the basic HIBE scheme. In addition, the PKG picks two more hash functions, $H_4 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_5 : \{0,1\}^n \to \{0,1\}^n$.

Extract: As in the basic IBE scheme.

Encrypt: Given a message m with the ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, the message can be encrypted by first computing $P_i = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i) \in \mathbb{G}_1$ for $1 \leq i \leq t$; then choosing a random $z \in \{0,1\}^n$; and setting $r = H_4(z, m)$. Subsequently, the ciphertext is set to:

$$c = \langle rP_0, rP_2, \ldots, rP_t, z \oplus H_2(g^r), m \oplus H_5(z) \rangle,$$

where $g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2$, as before.

Decrypt: Given a ciphertext $c = \langle U_0, U_2, \ldots, U_t, V, W \rangle$ encrypted using the ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, the ciphertext can be decrypted by performing the following steps:

1. compute
$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^{t}\hat{e}(Q_{i-1}, U_i)}\right) = z;$$
2. compute $W \oplus H_5(z) = m$;
3. set $r = H_4(z, m)$ and test if $\langle U_0, U_2, \ldots, U_t, V\rangle$ is a basic HIBE encryption of z using r and $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t\rangle$. If not, reject the ciphertext, otherwise accept m as the decryption of c.

It is worth noting that other HIBE and HIBS schemes are available. We have selected the schemes from [80] because they are efficient and their security is based on reasonable computational assumptions. More importantly, we observe that the HIBS scheme of [80] can be extended to an identity-based (partial) aggregate signature scheme using a simple twist. The aggregate signature scheme, converted from a 1-level HIBS scheme, can be described as follows:

Aggregate: To aggregate n signatures on n distinct messages produced by n different users (who are all at level 1 of the hierarchy used for the HIBS scheme), each user with identity ID_i (for $1 \le i \le n$) computes $U_i = s_0 H_1(\mathrm{ID}_i) + s_{1,i} H_3(\mathrm{ID}_i, m_i)$ and $Q_{1,i} = s_{1,i} P_0$. Note that $s_0 H_1(\mathrm{ID}_i)$ is the private key of the user with identity ID_i, while $s_{1,i}$ is an integer that user i selects at random from $\mathbb{Z}_q^*$. The aggregate signature is $\langle \sum_{i=1}^{n} U_i, Q_0, Q_{1,1}, \ldots, Q_{1,n}\rangle$.

Aggregate Verify: Given an aggregate signature $\langle U, Q_0, Q_{1,1}, \ldots, Q_{1,n}\rangle$ produced by n users from n messages, the verifier computes $H_3(\mathrm{ID}_i, m_i)$ for $1 \le i \le n$ and checks if:
$$\hat{e}(P_0, U) = \prod_{i=1}^{n} \hat{e}(Q_0, H_1(\mathrm{ID}_i))\hat{e}(Q_{1,i}, H_3(\mathrm{ID}_i, m_i)).$$

Note that this method can compress n signatures each with 2 group elements to $n+1$ group elements. In [189], Yoon *et al.* also presented an aggregation technique with roughly the same degree of compression for a modification of the Cha-Cheon IBS scheme. We will see a non-identity-based aggregate signature scheme with full compression (into a single signature) in Section 3.4.5.

3.4.4 The ZSM IBS Scheme with Message Recovery

In some scenarios, particularly where reducing the bandwidth of messages is important, an IBS scheme with message recovery may be useful. Zhang, Susilo and Mu [190] have recently proposed such a scheme based on the work in [4]. Their scheme is as follows.

Setup: As in the basic IBE scheme. In addition, the PKG selects k_1 and k_2 such that $|q| = k_1 + k_2$. It also defines additional hash functions $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $F_1 : \{0,1\}^{k_2} \rightarrow \{0,1\}^{k_1}$, and $F_2 : \{0,1\}^{k_1} \rightarrow \{0,1\}^{k_2}$. The system parameters are now $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, sP, H_0, H_1, F_1, F_2, k_1, k_2 \rangle$.

Extract: As in the basic IBE scheme.

Sign: Given a private key sQ_{ID} and a message $m \in \{0,1\}^{k_2}$, the signer computes:

1. $v = \hat{e}(P,P)^k$, where k is selected at random from $\mathbb{Z}_q^*$;
2. $f = F_1(m) \| (F_2(F_1(m)) \oplus m)$;
3. $r = (H_1(v) + f) \bmod q$;
4. $U = kP - r(sQ_{\text{ID}})$.

The signature σ is (r, U).

Verify: Given a signature $\sigma = (r, U)$ signed by a user with a public key $Q_{\text{ID}} = H_1(\text{ID})$, the verifier computes

$$f = r - H_1(\hat{e}(U,P)\hat{e}(Q_{\text{ID}}, sP)^r) \text{ and } m = [f]_{k_2} \oplus F_2([f]^{k_1}).$$

The verifier also checks if $[f]^{k_1} = F_1(m)$. The signature is accepted as valid if and only if this equation holds. Here $[f]^{k_1}$ denotes the left-most k_1 bits of the string f, while $[f]_{k_2}$ denotes the right-most k_2 bits of the string f.

The length of the signature is $|r| + |U| = |q| + |\text{element of } \mathbb{G}_1|$. For example, if we select q to be a 170-bit prime and the size of a group element of $\mathbb{G}_1$ is 171 bits, the length of the signature is then 341 bits. The security of this scheme is based on the hardness of the CDH problem. To obtain approximately similar security as a standard 1024-bit RSA signature and a 2^{-80} probability of a successful forgery by an adversary, $|k_1| \leq 80$ is needed if a group element of $\mathbb{G}_1$ is represented by 171 bits [190]. The size of the message is limited to k_2 bits, where $k_2 = |q| - k_1$.

3.4.5 Signature Schemes from Pairings

Apart from the aforementioned IBS schemes, we would like to draw to the reader's attention two ordinary (i.e. not identity-based) pairing-based signature schemes with attractive properties, namely, the BLS short signature scheme and the BGLS aggregate signature scheme. The BLS short signature scheme will be adopted in our hybrid certificate/identity-based proposal in Chapter 5.

The BLS short signature scheme was proposed by Boneh, Lynn and Shacham [28] in 2001. It can produce short signatures which provide equivalent security to standard signatures for environments with bandwidth constraints. The security of the scheme is based on the CDH assumption. Using the same parameters $\langle \hat{e}, \mathbb{G}_1, \mathbb{G}_2, H_1, P \rangle$ as in the previous sections, the BLS short signature scheme can be described as follows:

KeyGen: Pick an integer at random from $\mathbb{Z}_q^*$ and compute $sP \in \mathbb{G}_1$. The public key is sP and the matching private key is s.

Sign: Given a private key $s \in \mathbb{Z}_q^*$ and a message $m \in \{0,1\}^*$, compute $H_1(m) \in \mathbb{G}_1$. The signature is $\sigma = sH_1(m) \in \mathbb{G}_1$.

Verify: Given a public key sP, a message m and a signature σ, compute $H_1(m)$ and accept the signature as valid if and only if $\hat{e}(P, \sigma) = \hat{e}(sP, H_1(m))$ holds.

Note that the length of the signature is only the size of a group element of $\mathbb{G}_1$, half the size of a signature produced by the Cha-Cheon IBS scheme.

We remark that the above version of the short signature scheme may well not provide the expected level of security if we choose an elliptic curve in characteristic 2 or 3 [29, 143]. This is because of the existence of special purpose algorithms to solve the discrete logarithm problem in these cases (more details about curves selection are discussed in Section 3.5). However, this problem can be addressed using a modified version of the above scheme and non-supersingular curves, as was proposed in [29][1]. In particular, a class of curves called MNT curves can be used and levels of security around 85 bits can be achieved with a signature size of about 170 bits.

The above BLS short signature scheme can be extended to an aggregate signature scheme, as proposed by Boneh, Gentry, Lynn and Shacham in [27]. An aggregate signature is an aggregation of n signatures on n distinct messages from n distinct users, which produces a single short signature. This can be very useful for verifying certificate chains that contain signatures on distinct certificates issued by different parties. The BGLS aggregate signature scheme works as follows:

Aggregate: For the aggregating group U of n users, assign to each user an index i, where $1 \leq i \leq n$. Each user $u_i \in U$ with his private key s_i provides a signature $\sigma_i \in \mathbb{G}_1$ on a message $m_i \in \{0,1\}^*$, where $\sigma_i = s_i H_1(m_i) \in \mathbb{G}_1$. The aggregate signature is $\sigma = \sum_{i=1}^{n} \sigma_i$.

Aggregate Verify: Given an aggregate signature σ for an aggregating group U of n users, with their respective public key $s_i P \in \mathbb{G}_1$ and the original messages $m_i \in \{0,1\}^*$, ensure that the messages m_i are all distinct and accept the aggregate signature as valid if $\hat{e}(P, \sigma) = \prod_{i=1}^{n} \hat{e}(s_i P, H_1(m_i))$ holds.

An aggregate signature produced from the above scheme compresses n signatures into just a single group element of $\mathbb{G}_1$. An aggregate verification requires $n+1$ pairing computations.

3.5 Performance and Implementation Considerations

As with any other cryptosystem, the practicality of an identity-based cryptosystem is highly dependent on its ease of implementation and its level of performance. Even though Cocks [47] proposed a computationally acceptable identity-based encryption scheme about the same time as Boneh and Franklin, Cocks' scheme is too expensive

[1] We note that [29] is a full and revised version of [28].

in terms of communication overheads. For instance, transmitting a short ciphertext of 128 bits (e.g. session key) using a 1024-bit modulus may require up to 32 kilobits of keying material [47], which is roughly 13 times the size of a standard X.509 certificate. This may not be acceptable in many applications. On the other hand, identity-based cryptographic schemes based on pairings use small key sizes. Also, they are, as we shall see below, computationally almost comparable with RSA schemes [12, 157].

In what follows, we discuss parameter selection for pairing-based cryptographic schemes, pairing computation times for different parameters sets, and some of the development tools available for constructing identity-based cryptosystems.

Parameter Selection. Most pairing-based cryptographic schemes need to be working in a subgroup of $E(\mathbb{F}_{p^m})$ of sufficiently large prime order q so that $\mathbb{F}_{p^{km}}$ is a sufficiently large finite field. For current minimum levels of security, we require $q > 2^{160}$ and $p^{km} > 2^{1024}$ to resist algorithms for solving the discrete logarithm problem in $\mathbb{G}_1$ and $\mathbb{G}_2$ (as defined in Section 3.3), respectively [75, 76]. Currently, 1024-bit RSA is considered to provide roughly 80 bits of security [9]. This is believed to be roughly as secure as a cryptosystem based on the discrete logarithm problem over a field of size 2^{1024}. For example, let us consider the case of supersingular curves over fields of characteristic 2 with $k = 4$. In order to achieve a level of security roughly similar to 1024-bit RSA, we require our curve to be defined over $\mathbb{F}_{2^m}$ with $m \approx 256$ (since $k = 4$ and $(2^{256})^4 = 2^{1024}$). Similarly, if we work on an elliptic curve in characteristic 3 ($k = 6$), the curve should be defined over $\mathbb{F}_{3^m}$ with $m \approx 108$ in order to achieve a level of security roughly equivalent to 1024-bit RSA. It is, however, not possible to find curves and suitable prime order subgroups of these exact sizes due to the rarity of suitable supersingular elliptic curves. It is also worth noting that there are special purpose algorithms which can be applied to solve the discrete logarithm problem in $\mathbb{F}_{p^{km}}$ when p is 2 or 3. This may mean that larger parameters than at first appears must be chosen to obtain the required security levels [143]. Table 3.2 shows a few curves offering roughly similar levels of security to 1024-bit RSA.

It is clear from Table 3.2 that the curve defined over a finite field with size 2^{271}

Table 3.2: Different sizes of curves with their matching levels of security for RSA cryptographic schemes.

Order of Curve ($\approx$)	**Characteristic/k**	**Equivalent RSA Modulus (in bits)**
2^{241}	2/4	964
2^{271}	2/4	1084
2^{283}	2/4	1132
3^{97}	3/6	922
3^{149}	3/6	1417
3^{163}	3/6	1550

has roughly similar levels of security to 1024-bit RSA. Operations in characteristic 2 have the advantage of faster and easier implementation in hardware. However, working in characteristic 3 can achieve shorter message length than in characteristic 2. One problem with working in characteristics 2 and 3 is that, as we mentioned earlier, there are only a small number of curves and fields to choose from. Hence, there is an element of luck in the search for a suitable large prime factor of the curve order and the choice of parameters is consequently not very flexible. However, no such problems arise when working over $\mathbb{F}_p$ [76].

Pairing Computation. The dominant cost for an identity-based cryptographic scheme is the evaluation of a pairing which involves an application of the Miller algorithm [73] or a modification of it (see for example [12, 14, 77]). Table 3.3 shows some experimental results of various cryptographic algorithms obtained by Barreto *et al.* [12].

It is obvious from Table 3.3 that an RSA encryption can be performed very fast as it involves only a very small public exponent, e.g. $2^{16} + 1$. We see that a Tate pairing computation with underlying base field $\mathbb{F}_{2^{271}}$ is approximately four times slower than an RSA decryption. On the other hand, a pairing computation with underlying base field $\mathbb{F}_p$ and $|p| = 512$ is slightly faster than on a curve defined over $\mathbb{F}_{2^{271}}$ but the penalty is the need to use a bigger key size. The pairing computation time can be further reduced by more than half with pre-computation, which involves calculations of certain fixed parameters in the Miller algorithm [12]. An IBE decryption (based on the Boneh-Franklin IBE scheme in Section 3.4.1) involves one hash function (H_2) evaluation, one XOR operation, and one pairing computation. The timings from

Table 3.3: Computation times for different operations on a Pentium III 1 GHz machine [12].

Operation	**Time (in ms)**
RSA encryption, $\|N\| = 1024$ bits, $\|e\| = 16$ bits	0.4
RSA decryption, $\|N\| = 1024$ bits, $\|d\| = 1007$ bits	7.9
Tate pairing, $\mathbb{F}_{2^{271}}$, $\|q\| = 272$ bits	23.0
Tate pairing, $\mathbb{F}_{3^{97}}$, $\|q\| = 154$ bits	26.2
Tate pairing, $\mathbb{F}_p$, $\|p\| = 512$ bits	20.0
Tate pairing, $\mathbb{F}_p$, $\|p\| = 512$ bits, pre-computation	8.6
IBE encryption, $\mathbb{F}_p$, $\|p\| = 512$ bits	48.0
IBE encryption, $\mathbb{F}_p$, $\|p\| = 512$ bits, pre-computation	36.0
IBE decryption, $\mathbb{F}_p$, $\|p\| = 512$ bits	30.0
IBE decryption, $\mathbb{F}_p$, $\|p\| = 512$ bits, pre-computation	19.0

Table 3.3 shows that a pairing computation can take up to about two-thirds of the time required to perform an IBE decryption (20 ms out of 30 ms).

In general, although computing the Tate pairing is slower than performing an RSA modular exponentiation, it is perceived to be acceptably fast to implement in practice. Implementations of the Boneh-Franklin IBE scheme on smart cards by Gemplus[2], a world-leading smart card based solutions provider, is a good example [125]. There has been further work in improving the performance of pairing computations, such as [157, 158]. In particular, it has been shown in [10] that many of the characteristics 2 and 3 timings in Table 3.3 have been improved significantly since the publication of [12]. This gives hope that faster identity-based cryptosystems can be built and that the gap in performance between RSA and identity-based cryptosystems can be further reduced.

Development Libraries and Toolkits. There are currently only a few software libraries and toolkits that support implementations of identity-based cryptographic schemes. Here, we give a short review of some available libraries and toolkits.

- *MIRACL* [160] is a C/C++ library used for implementing cryptographic schemes such as RSA, Diffie-Hellman key exchange, DSA and so on. The latest ver-

[2] Gemplus International announced the world's first identity-based encryption for smart cards on 2 November 2004.

sion 4.85 of MIRACL offers full support for elliptic curve cryptography over $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$. It provides inline assembler and allows a variety of techniques to be used to improve performance. The results in Table 3.3 are based on implementations using MIRACL.

- *The Stanford IBE Library* [167] contains source code which implements the Boneh-Franklin IBE scheme. The source code is written by members of the Applied Crypto Group of the Computer Science Department at Stanford University, and is made available through the group's website. The library relies on the GNU MP library[3] and the OpenSSL library.
- *The Voltage IBE Toolkit* [180] is a set of tools that enable software developers to quickly incorporate the Boneh-Franklin IBE scheme into other applications. The toolkit, written in C, provides high-level interfaces for rapid application development as well as lower-level cryptographic APIs for advanced security operations.

Having reviewed various identity-based cryptographic schemes in Section 3.4, and discussed their seemingly promising performance in this section, we are now ready to move on to the next section to reviewing some real world applications of IBC.

3.6 Applications

Currently, two leading companies in developing and marketing identity-based security systems are Voltage Security[4] and NoreTech[5]. Voltage SecureMail was the first commercial IBE product available in the market, targeted at the financial, governmental and healthcare industries. On the other hand, NoreTech has developed prototype implementations for Symbian OS and Microsoft Smartphone OS mobile phones [125].

The Voltage SecureMail product [181] offers a secure email communication solution for enterprises. It does not require pre-enrolment of users before they can receive

[3] GNU MP is a free library for arbitrary precision arithmetic, operating on signed integers, rational numbers and floating point numbers.

[4] `http://www.voltage.com/`.

[5] `http://www.noretech.com/`.

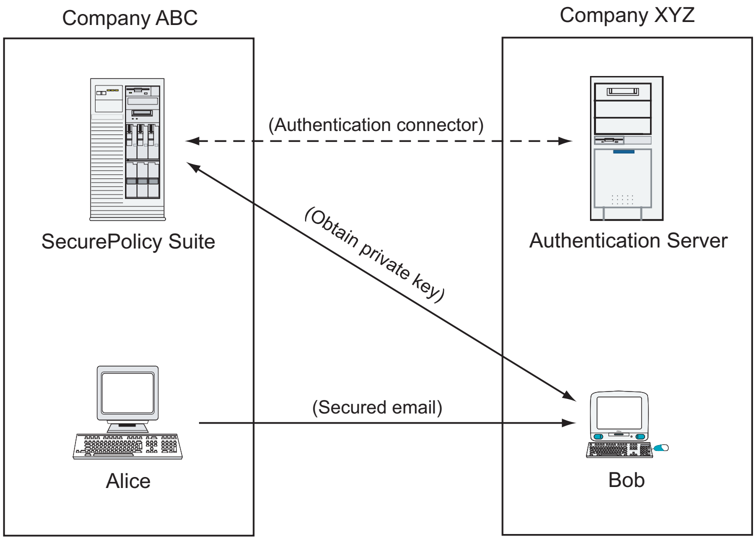

Figure 3.1: An overview of the Voltage SecureMail architecture [181].

secure emails. Suppose Alice from company ABC wants to send her customer Bob at company XYZ a confidential email. After she has composed the message, she simply types in Bob's email address '`bob@xyz.com`' and hits the "Send Secure" button. The email will be encrypted with Bob's public key and delivered to Bob, as illustrated in Figure 3.1.

If this is the first time Bob receives a secure message from Alice, he clicks on a link in the message header and downloads the Voltage SecureMail client. He then proceeds to enrol and authenticate to company ABC's SecurePolicy Suite, an administration and key generation server that supports Voltage SecureMail. Upon completion of the authentication, the SecurePolicy Suite presents Bob with his private key which can be used to decrypt and read the secured email.

Here are more details about the architecture of Voltage SecureMail [181, 182].

- *Public parameters*: The system parameters for the IBE scheme are created when the Voltage SecurePolicy Suite is initialised. The authenticated parameter set can be downloaded by users from the Voltage SecurePolicy Suite through the Voltage Domain Trust Handshake. This builds on the standard TLS handshake.
- *Authentication and policy enforcement*: The Voltage SecurePolicy Suite in-

cludes an enrolment server, which in turn integrates with existing identity management systems such as LDAP and Windows Active Directory, to authenticate users before releasing private keys. When the enrolment server receives a request for an IBE decryption key from Bob, the identity associated with the request is checked to determine to which identity group it belongs. Subsequently, a suitable authentication connector such as Windows Domain Authentication or POP3 Authentication is called to contact Bob's authentication server. The appropriate information is passed to the authentication server to validate Bob's identity. A key server in the Voltage SecurePolicy Suite will only grant Bob the private key if the checks by the enrolment server pass and if Bob complies with whatever policy is bound to the associated identifier.

- *Key management*: Private keys are derived by the key server and delivered to the users through TLS secure channels. For key update and to achieve automated key expiry, Voltage SecureEmail combines an identity and a date when constructing a public key. For instance,

  'name=bob,validity=01/10/05-02/10/05'.

  Thus, the exposure of a compromised key is limited to one day. Furthermore, if Bob resigns from his company, the Voltage SecurePolicy Suite simply stops issuing private keys to him.

Smetters and Durfee [164] also proposed another secure email application based on IBC, which can be built on existing global trust infrastructures. Instead of having one global PKG, each domain, with its own Domain Name System (DNS) server, is responsible for creating a set of IBE system parameters and distributing private keys to users within its domain. Their design make uses of DNS Security Extensions (DNSSEC), a technique used for securing the DNS, to distribute an authenticated parameter set to its domain users. The parameters include a Time-to-Live (TTL) field to force a user client to check for updates to the parameter values at reasonable time intervals. This limits the possible damage caused by exposure of the IBE master secret. To communicate with a user from a different domain, a cross-certification between the DNS servers of these domains using a PKI approach is required. User clients within a domain communicate with their PKG (handled by a DNS server) using the standard TLS protocol. To support key revocation based on an identifier, Smetters and Durfee suggested the use of a salt as part of the identifier. The salt, also published in the domain system parameters, should be a random string long enough to be unlikely to be chosen at random again. When the TTL value has expired, all user clients need to update the salt value. Work on securing IP-based network traffic using similar ideas was also presented in [164].

From the above examples, it seems that identity-based techniques are suitable for email applications. One reason for this may be that the users' email addresses can be used directly as their identifiers. In addition, sending a secure message through an email system may well be one of the easiest methods of delivering confidential information from one party to another over the Internet. Apart from email applications, there are other example applications in which IBC may be more beneficial than conventional public key cryptography. In [48], Dalton studied ways of securely managing health care records in the UK's National Health Service. The potential improvements with IBC, such as policy enforcement using identifiers, workflow management, access control based on roles or groups and so on, were considered within the health care environment. Meanwhile, Khalili *et al.* [108] proposed a key distribution technique based on the combination of IBC and threshold cryptography for ad hoc network environments. By using identifiers as the public keys, users need only propagate their identity information rather than standard full certified public keys. Stading [166] studied the use of IBC for improving key management techniques in Scribe which was originally designed for Chord [170], a P2P lookup service. His proposal made use of threshold cryptography to generate a shared master secret among some selected servers from a group of overlay network[6] nodes without the need for a trusted third party. The private key of a specific node can be computed by sending requests to each server holding part of the master secret. For more example applications of IBC, see [11].

3.7 Summary

In this chapter, we have given a literature review of IBC. We covered some background facts about the evolution of IBC. We compared a traditional certificate-based

[6] An overlay network is a type of computer network which is built on top of another network. Nodes in the overlay can be thought of as being connected by virtual or logical links.

PKI with an identity-based PKI. We also provided some mathematical background on pairings sufficient for understanding the identity-based cryptographic primitives that we will apply to grid applications in the subsequent chapters. Based on the presented performance figures for pairing computations, we have seen that identity-based cryptographic schemes are fast enough to be deployed in real life applications, with other advantages such as being certificate-free and having small key sizes. We have shown that currently secure messaging systems seem to be the ideal testbeds for identity-based cryptographic schemes. However, many other applications with different and more complex security requirements are still to be explored. This provides our motivation for studying the application of IBC in grid security.

Chapter 4

Identity-Based Key Infrastructure for Grid

Contents

This chapter presents a comprehensive investigation of the use of identity-based techniques to provide an alternative grid security architecture. We propose a customised identity-based key agreement protocol which fits nicely with the GSI and provides a more lightweight secure job submission environment for grid users. Single sign-on and delegation services are also supported in a very natural way in our identity-based architecture.

4.1 Current Problems

In Chapter 2, we reviewed current grid technologies and described the grid vision. One can imagine a VO as a giant virtual supercomputer that encompasses thousands of processing chips and memory cards, hundreds of hard disks and database files, and other resources. However, unlike an actual personal computer which has its processing parts and components integrated within a single physical machine, resources within a VO are geographically dispersed and managed under different administrative domains. Such a heterogeneous and scalable environment raises many questions on how various security mechanisms can be efficiently put in place without jeopardising the practicality of a VO.

Despite the ambitious vision, the key management that is essential for supporting these security mechanisms can be much more complex than in a conventional distributed system. To illustrate this, we refer back to the grid example scenario described in Section 2.5. Consider a single secure job submission. Before Alice submits her job request, she must make sure that her proxy credential has been created. When Alice submits a job through the Gatekeeper (GK), another proxy credential that deals with the job has to be created at the GK. If Alice delegates her proxy credential to the GK to offload some tasks such as resource allocation and data transfer, the GK would need to create a separate proxy credential signed with Alice's short-term private key. Should the GK want to further delegate Alice's proxy credential to resource X, then creation of another proxy certificate is inevitable. The extensive use of standard X.509 certificates and proxy certificates even for a single job submission is evident, as shown in Table 4.1.

Table 4.1: Various entities and the required certificates in the PKI-based GSI. An issuer who is a proxy entity is denoted by *.

Entity	No.	Subject	Certificate Type	Issuer
Alice	(i)	A	Standard	CA
	(ii)	A	Proxy	A
Gatekeeper	(iii)	GK	Standard	CA
	(iv)	GK	Proxy	GK
	(v)	A	Proxy	A^*
Resource X	(vi)	X	Standard	CA
	(vii)	X	Proxy	X
	(viii)	A	Proxy	GK*

Generation, certification and verification of public keys, distribution of certificates, and other aspects of key management cause non-trivial overheads. For instance, when the GK and resource X perform mutual authentication through the TLS handshake protocol before the job is executed, the GK must transmit the complete certificate chain consisting of: certificates (i), (ii), (iii) and (v) in Table 4.1, to X. For X to verify the short-term public key that the GK uses on behalf of Alice, X must check the signatures on all the certificates (v), (iii), (ii) and (i) to ensure that A has indeed delegated her credential to the GK. It is obvious that when the delegation chain becomes longer, the verification of the short-term public key used at the end of the chain becomes more tedious. Note that this process is repeated every time the GK communicates with a resource provider on Alice's behalf. Current key and certificate management techniques are likely to cause bottlenecks at the GK and resource X when they handle many job requests simultaneously. Hence, the scalability of the PKI approach seems to be questionable.

From the above description, we see that although current public key management techniques are workable, they seem to be rather heavyweight. On the contrary, it is an essential requirement that key management and the security architecture within a grid environment are lightweight so that a VO that serves as a single virtual machine can be constructed with minimal performance impact. This brings us to the central question investigated in this thesis: *can an identity-based key infrastructure offer a more lightweight, and thus a better, solution than the PKI-based GSI?*

4.2 Overview of Identity-Based Key Infrastructure for Grid

In order to tackle the aforementioned problems, it makes sense to investigate the benefits of a certificate-free approach to key management. With this in mind, IBC has the following attractive properties:

- *Identity-based*: The use of identity-based public keys in IBC allows any entity's public key to be generated and used on-the-fly without the need for a directory look-up or other public key distribution mechanism;
- *Certificate-free*: IBC does not make use of certificates since public keys can be computed based on some public identifiers; and
- *Small key sizes*: Since identity-based cryptographic schemes use pairings which are, in turn, based on elliptic curves, they can have smaller key sizes than more conventional public key cryptosystems such as RSA.

These interesting properties of IBC indicate the possibility of developing an alternative security infrastructure that provides greater flexibility to entities within a grid environment and that offers a more lightweight public key management approach than traditional PKI. The aim of this chapter is to propose and investigate an identity-based key infrastructure for grid (IKIG). Our proposal incorporates features that align with the security services provided by the PKI-based GSI, which is an essential requirement of any alternative proposal. The challenge of achieving this lies in the difficulty of creation and management of identity-based proxy credentials, in addition to long-term credentials that are common to most identity-based cryptosystems. By exploiting some properties from HIBC, IKIG facilitates the creation and usage of identity-based proxy credentials in a very natural way. These identity-based proxy credentials, in turn, are needed to support features that match those provided by the GSI.

Our IKIG proposal has the following features:

1. Single sign-on: As with the GSI, our IKIG proposal supports single sign-on through the use of identity-based proxy credentials. Since the users' short-term

public keys are based on some predictable identifiers and the matching short-term private keys are stored locally at the user side, user authentication can be performed without any physical intervention from the users and without the need for certificates.

2. Mutual authentication and key agreement: IKIG also supports a certificate-free authenticated key agreement protocol based on the TLS handshake. Our protocol allows mutual authentication and session key establishment between two entities in a more lightweight manner than the traditional TLS as it has small key sizes and requires no certificates.

3. Delegation: We propose a non-interactive delegation protocol which works in the same way as in the GSI, in the sense that the delegator signs a new public key of the delegation target. In addition, IKIG allows partial aggregation of signatures. This can be useful when the length of the delegation chain increases. This is a new feature not available in the GSI.

Users in the IKIG setting do not need to obtain short-term private keys from their respective PKGs. This is because the users themselves act as PKGs for their local proxy clients. Thus short-term private key distribution is not an issue in IKIG. This contrasts favourably with the applications of IBC that we discussed in Section 3.6, where private key distribution is a complicating factor.

4.3 Related Work

A number of researchers have recently started exploring the use of IBC in grid security. Our first publication on this subject [117] was an exploratory paper that described some potential benefits of IBC in a grid security architecture. It was noted in [117] that a VO within a grid environment may have a large number of members that join and leave over the time and that certificates are used extensively for every job submission. This would inevitably complicate key management and increase the bandwidth requirement of a grid system. It was also noted in [117] that these problems could be simplified by using certificate-free IBC. Moreover in the IBC setting, a user's public key can be created and used immediately without the need for a public key certificate to be forwarded to the intended recipient (normally via a TLS handshake). However, the supposedly dynamic use of identity-based keys was hindered by some traditional limitations of IBC such as key escrow and the need to distribute private keys through secure channels. More importantly, some of the essential security requirements desired in the GT, such as using proxy credentials for single sign-on and delegation, were not addressed in [117].

At about the same time, Mao [123] revisited the GSI proposed for GT2 [69] and presented an application of Sakai *et al.*'s non-interactive identity-based key distribution technique [155] within the GSI authentication framework. It was assumed in [123] that a user, who is from an average or low-end platform, may have to execute many mutual authentication sessions with different resource providers. This would potentially cause a performance bottleneck at the user client. The use of non-interactive session key establishment technique seems to significantly reduce the communication overheads between two key sharing parties. Nevertheless, this may not be the case in practice, particularly in the newer versions of the GT, i.e. GT3 and GT4. A grid user can always delegate his credential to a resource (or resource broker) which can act on the user's behalf. Thus, the performance issues discussed in [123] are only valid for a special scenario whereby a user client is required to contact and perform mutual authentication directly with many resources. Also, even though session key establishment between two parties can be achieved non-interactively in the approach of [123], these entities must obtain their private keys from the same PKG and on a regular basis, e.g. daily. Furthermore, ways of securing job submissions using proxy credentials in the identity-based approach were not addressed in [123].

Subsequently, Huang *et al.* [102] combined and extended the work of [117] and [123], and presented an identity-based security infrastructure which seems to work rather differently to the GSI. Although the authors of [102] showed how to perform credential delegation between two parties, each run of their delegation protocol involves additional secure communication with a PKG. This appears to be more costly and tedious than the current delegation techniques using PKI. In terms of access control within a grid environment, Huang *et al.* proposed a framework which requires the participation of the PKG whose task is to issue authorization assertions to its users. The need for involvement of the PKG in delegation and access control, and the fact that the grid entities must share the same system parameters in order to achieve non-interactive mutual authentication (as in [123]) seems to cause bottlenecks at the PKG and confine the scalability of the system. Huang *et al.* also investigated a secure group communication model for a grid application using Joux's one round tripartite key agreement protocol [106]. However, Joux's protocol was adopted naively without considering the need for validating the authenticity of Diffie-Hellman keying materials from group members. As pointed out in [6], without authentication, Joux's protocol is vulnerable to a simple man-in-the-middle attack.

In summary, the potential of IBC has only been partially investigated to date and none of the proposals so far satisfactorily address the requirements imposed by grid applications (and that are currently met by the GSI). In this chapter, we present a fully developed identity-based infrastructure for grid applications that does meet these requirements. Our work represents an improvement on [117] and appears in an abbreviated form in [115].

4.4 Design of IKIG

In our IKIG setting, we propose to replace the CA in the current PKI-based GSI with a Trusted Authority (TA). The TA's roles including acting as the PKG and supporting other user-related administration and management. In this section, we assume users and resource providers have registered with the same TA. Issues of inter-operation between TAs in the IKIG setting will be discussed in Section 4.8.3.

We now apply both the Gentry-Silverberg full HIBE and HIBS schemes, as introduced in Section 3.4.3, in our proposal. Figure 4.1 shows the hierarchical setting of HIBC that matches the hierarchical relationships of various entities within a grid environment, with the TA at level 0, user at level 1 and user proxy at level 2.

We note that proxy credentials must be used for secure job submissions in order to match the requirements of the GSI. These short-term credentials are used in security services such as mutual authentication and delegation. Let the TA have a master secret s_0 and let its system parameters be $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$. The TA distributes long-term private keys to its users (and resource providers) at level 1, who in turn generate short-term private keys for their own proxies at level 2, as illustrated in Figure 4.1. For example, a user A has identity-based key sets as

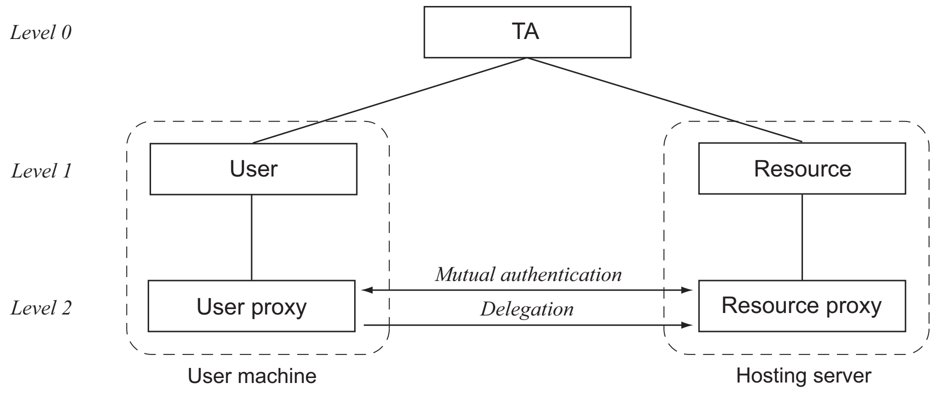

Figure 4.1: A hierarchical structure of entities in the IKIG setting.

shown in Table 4.2 in our IKIG setting.

In Table 4.2, A's long-term public key P_A is equivalent to a public key computed using a level-1 identifier in the Gentry-Silverberg HIBE and HIBS schemes. The corresponding long-term private key S_A is generated by the TA. On the other hand, A can compute a short-term private key for her proxy which is at the second level of the hierarchy. Note that the proxy's identifier is defined purely by a lifetime LT_A in some fixed format, and thus A's short-term public key $P_{\bar{A}}$ is computed based on the ID-tuple $\langle \mathrm{ID}_A, \mathrm{LT}_A \rangle$. Here, $s_A, s_{\bar{A}} \in \mathbb{Z}_q^*$ are secret values chosen by A and A's proxy, respectively. The long-term and short-term credentials of a resource, X, can be established in the same manner, for example (P_X, S_X) and $(P_{\bar{X}}, S_{\bar{X}})$, respectively. Details of key generation and various aspects of key distribution and management will be discussed in Section 4.5. For now, we simply assume that these keys are in place.

Now that we have described the keys needed by the various entities in our IKIG approach, we can illustrate how to secure the job invocation process that was discussed in Section 2.3.2. Consider a simple scenario where A, with identity-based key sets shown in Table 4.2, wants to submit a job request `J_req` to a target re-

Table 4.2: A's long-term and proxy credentials.

Credential Type	Public Key	Private Key	Secret Value
Long-term	$P_A = H_1(\mathrm{ID}_A)$	$S_A = s_0 P_A$	s_A
Short-term	$P_{\bar{A}} = H_1(\mathrm{ID}_A \Vert \mathrm{LT}_A)$	$S_{\bar{A}} = S_A + s_A P_{\bar{A}}$	$s_{\bar{A}}$

source X. Here, we assume J_req contains information such as the job description and instructions on how the job is to be executed. For simplicity in describing our approach, we assume the user can communicate with the resource provider directly, rather than through a gatekeeper or a resource broker.

4.4.1 Single Sign-On

The first step in a job submission is to create a user proxy credential. With the proxy credential, A does not need to sign on (i.e. access her encrypted long-term private key S_A with her passphrase) again until the expiry of her short-term public/private key pair. A can store her short-term private key $S_{\bar{A}}$ in a local file system accessible by her GT client so that the client can use the key as it wishes. Single sign-on can be seen as the preliminary but important step before mutual authentication or delegation are performed between A and another entity. At any point in time during the job submission session, A's client can prove possession of $S_{\bar{A}}$ on A's behalf. A's client will need to do this when challenged by other entities during the execution of key agreement and delegation protocols.

4.4.2 Authorization

A generates an identity-based signature using the Gentry-Silverberg IBS scheme on J_req with her short-term private key $S_{\bar{A}}$. The Gentry-Silverberg IBS scheme was defined in 3.4.3. She then submits the signed request to a Managed Job Factory (MJF) that resides on X through a GRAM service, by sending the following message:

$$A \rightarrow X : \mathrm{ID}_A, \mathrm{LT}_A, \texttt{J\_req}, Sig_{\bar{A}}(\texttt{J\_req})$$

Here we use $Sig_{\bar{A}}(.)$ to denote a signing operation using A's short-term private key $S_{\bar{A}}$. The signed request is of the form $\langle \sigma_{\bar{A}}, Q_A, Q_{\bar{A}} \rangle$, where $\sigma_{\bar{A}} = S_{\bar{A}} + s_{\bar{A}}h$, $h = H_3(\mathrm{ID}_A \| \mathrm{LT}_A \| \texttt{J\_req})$, $Q_A = s_A P_0$ and $Q_{\bar{A}} = s_{\bar{A}} P_0$. Note that the MJF can verify the signed request by first computing $P_{\bar{A}}$ on-the-fly (assuming it has knowledge of the system parameters) and then checking if:

$$\hat{e}(P_0, \sigma_{\bar{A}}) = \hat{e}(Q_0, P_A)\hat{e}(Q_{\bar{A}}, h)\hat{e}(Q_A, P_{\bar{A}}).$$

Subsequently, the MJF maps A's identifier to its grid-map file before granting access to A. If the check succeeds, the MJF instantiates a managed job service for the job request and returns an endpoint reference to A. Clearly, this technique does not require standard or proxy certificates to certify the respective public keys. Note that $\hat{e}(Q_0, P_A)$ and $\hat{e}(Q_A, P_{\bar{A}})$ can be pre-computed by X and may be used many times for other purposes such as key agreement and delegation. The values Q_0 and Q_A are relatively static since they are calculated using the TA's and A's long-term parameters (we will show how the Q-values can be computed in Section 4.5.2). Thus, the signature verification requires only two real-time pairing computations.

4.4.3 Mutual Authentication and Key Agreement

Before the job can be started, A must perform mutual authentication with the newly created managed job service. As we have discussed earlier in Section 2.3.2, a user must verify that he is indeed submitting his job to the right host and correct account, while the hosting server must check if the user is who he claims he is. In the GSI, this is achieved using the standard TLS protocol. We present an identity-based authenticated key agreement protocol based on the TLS handshake protocol, Protocol 1. Our protocol uses the Gentry-Silverberg full HIBE and HIBS schemes for encryption and signing operations, respectively. It assumes that the TA's system parameters are already known to the protocol participants. We make use of the terminology of [50] with small changes as appropriate.

Protocol 1 Identity-based authenticated key agreement based on the RSA TLS handshake

```
(1)  $A \rightarrow X$ :  ClientHello = $n_A$, session_id, cipher_suite
(2)  $X \rightarrow A$ :  ServerHello = $n_X$, session_id, cipher_suite
                ServerIdentifier = $\mathrm{ID}_X$, $\mathrm{LT}_X$
                ServerHelloDone
(3)  $A \rightarrow X$ :  ClientIdentifier = $\mathrm{ID}_A$, $\mathrm{LT}_A$
                ClientKeyExchange = $Enc_{\bar{X}}$(pre_master_secret)
                IdentityVerify = $Sig_{\bar{A}}$(handshake_messages)
                ClientFinished
(4)  $X \rightarrow A$ :  ServerFinished
```

Protocol 1 is analogous to a TLS protocol which uses the RSA algorithm to transport keying material from client to server, as specified in [50]. As with the current TLS specification, Protocol 1 begins with A (playing the role of client) sending X (playing the role of server) a `ClientHello` message. The message contains a fresh random number and a session identifier as shown in step (1). Here, `cipher_suite` contains a cipher specification extending those provided in TLS version 1.0 to handle the HIBE and HIBS schemes. For example, this specification could take the form `TLS_HIBE_HIBS_WITH_DES_CBC_SHA` which would define the use of HIBE and HIBS for key transport and authentication, and DES-CBC and SHA as the symmetric encryption algorithm and hash function, respectively.

X responds with a `ServerHello` message which contains a new random number and a session identifier which may or may not have the same value as in `ClientHello` message depending on the state of the session (e.g. new or resumed session). X also forwards `ServerIdentifier` which contains information, such as X's identifier and a lifetime that X chooses, that allows A to compute X's short-term public key[1]. The `ServerHelloDone` message is sent to indicate the end of step (2).

In step (3), A first forwards `ClientIdentifier` to X. She then chooses a pre-master secret and encrypts it with X's short-term public key. Note that $Enc_{\bar{X}}(.)$ denotes an encryption using the HIBE scheme with X's short-term public key $P_{\bar{X}} = H_1(\mathrm{ID}_X \| \mathrm{LT}_X)$. The encrypted pre-master secret is transmitted to X as `ClientKeyExchange`. Next, A generates a signature on `handshake_messages` for `IdentityVerify`. This message component provides entity authentication of A to X. Here `handshake_messages` refers to all handshake messages sent or received starting at `ClientHello` up to but not including this message. A completes step (3) by sending `ClientFinished` that contains a verification value. This allows X to confirm that it has indeed received the previous handshake messages with the correct contents. The verification value is computed based on a master secret K_{AX} and a hash of `handshake_messages`. The master secret K_{AX} is calculated by A

[1] Note that in both certificate-based and identity-based settings, an entity's certificate or identifier (e.g. email address) can be obtained in advance, through out-of-band mechanisms. For example, these credentials can be downloaded or found on the entity's website. In the TLS protocol, however, these credentials can be exchanged as part of the protocol messages. The key difference between the two is that a certificate needs to be verified before it can be used, whereas an identifier can be used on-the-fly. In actual implementation, `ServerIdentifier` and `ClientIdentifier` of Protocol 1 may not be needed if entities' identifiers and the associated lifetimes can be determined at the application layer when the protocol is initiated.

from the value pre_master_secret as:

$$K_{AX} = \mathrm{PRF}(\texttt{pre\_master\_secret}, \text{“}\texttt{master secret}\text{”}, n_A, n_X),$$

where PRF is a pseudo-random function specified for the TLS protocol in RFC 2246 [50]. Finally X computes its part of the verification value in ServerFinished in the last step. X authenticates itself successfully to A if and only if A receives ServerFinished and validates that the message contains the correct verification value. This implies that X has indeed retrieved the correct pre-master secret chosen by A. The master secret will subsequently be forwarded to the TLS record protocol so that further keys used for protecting application data can be derived as necessary.

Protocol 1 can be executed by A when she connects to the managed job service to initiate her job without any use of certificates, in contrast to the current PKI-based TLS protocol.

Alternative. We remark that Protocol 1 can be easily transformed into a protocol that supports Diffie-Hellman key exchange. Protocol 2 shows the identity-based version of Diffie-Hellman TLS handshake, where cipher_suite now can be specified as TLS_DH_HIBS_WITH_DES_CBC_SHA.

Protocol 2 Identity-based authenticated key agreement based on the Diffie-Hellman TLS handshake

(1)	$A \rightarrow X$:	ClientHello = n_A, session_id, cipher_suite
(2)	$X \rightarrow A$:	ServerHello = n_X, session_id, cipher_suite
		ServerIdentifier = ID_X, LT_X
		ServerKeyExchange = xP, $Sig_{\bar{X}}(xP, n_A, n_X)$
		ServerHelloDone
(3)	$A \rightarrow X$:	ClientIdentifier = ID_A, LT_A
		ClientKeyExchange = aP
		IdentityVerify = $Sig_{\bar{A}}$(handshake_messages)
		ClientFinished
(4)	$X \rightarrow A$:	ServerFinished

In Protocol 2, A and X exchange ephemeral Diffie-Hellman key values aP and xP, respectively, where $a, x \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$. Thus the key exchange takes place in the group $\mathbb{G}_1$. Note that n_A and n_X are included along with xP in ServerKeyExchange before the string is signed. This aids in preventing a replay attack using the same Diffie Hellman key value in subsequent protocol runs executed within the timeframe of LT_X. The pre-master secret is now axP.

Protocol 1 is more computationally lightweight than Protocol 2 at the user's client because in Protocol 1, A does not need to perform any pairing computations in executing the protocol. This is because the Gentry-Silverberg HIBE encryption and HIBS signing algorithms are pairing-free. On the other hand, in Protocol 2, A has to perform a signature verification on the `ServerKeyExchange` from X, which requires at least two pairing computations.

4.4.4 Delegation

A may, at her discretion, delegate her credential to X for later use when necessary. Currently, the GSI employs a two-pass delegation protocol [176, 186] between a delegator and a delegation target. We propose a one-pass delegation protocol which works in the same way as GSI in the sense that the delegator signs a new public key of the delegation target. As we depicted earlier in Protocol 1, the client can easily compute the server's short-term public key based on the server's identifier and a lifetime, and vice versa. By the same reasoning, the delegator can straightforwardly sign the delegation target's new short-term public key which will be used for delegation purposes. This can be done without having the delegation target transmit its chosen short-term public key to the delegator through an authenticated and integrity protected channel. In order to compensate for the removal of certain types of policy enforcement which could have been done through a proxy certificate, we suggest the use of a delegation token. The delegation token is a 5-tuple containing identifiers of the delegator and the delegation target, the job request, any policy which the delegator wants to enforce on the delegation target, and the validity period of the token. Let $\texttt{DelegationToken}_X = (\mathrm{ID}_A, \mathrm{ID}_X, \texttt{J\_req}, \texttt{Policy}, \mathrm{LT}_{AX})$, where LT_{AX} is a lifetime which A imposes on X. Protocol 3 then shows our one-pass delegation protocol.

Protocol 3 Identity-based credential delegation based on the HIBS scheme

$A \rightarrow X$: $\texttt{DelegationToken}_X$, $Sig_{\bar{A}}(\texttt{DelegationToken}_X)$

Note that A can naturally bind X's public key information (ID_X and LT_{AX}) to the delegation token without acquiring X's new short-term public key from X (assuming A knows X's identifier). The signed delegation token is of the form $\langle \sigma_{\bar{A}}, Q_A, Q_{\bar{A}} \rangle$, where $\sigma_{\bar{A}} = S_{\bar{A}} + s_{\bar{A}}h$, $h = H_3(\mathrm{ID}_A \| \mathrm{LT}_{AX} \| \texttt{DelegationToken}_X)$, $Q_A = s_A P_0$ and $Q_{\bar{A}} = s_{\bar{A}} P_0$, as defined by the Gentry-Silverberg HIBS scheme.

X's status as the delegation target can be confirmed by a third party by: (i) verifying the signed delegation token using $P_{\bar{A}}$, and (ii) challenging X for a proof of possession of the private component associated to $P_{\bar{X}}$. This can be achieved when X and the verifying party perform mutual authentication using Protocol 1. In fact, X can forward the signed delegation token to the verifying party as part of the handshake messages exchanged between X and the verifying party. So long as A and X share the same TA in the hierarchical IKIG setting, X can even aggregate the signature on the delegation token with other signatures in the handshake protocol in order to save bandwidth. For example, X can aggregate $Sig_{\bar{A}}(\texttt{DelegationToken}_X)$ with $Sig_{\bar{X}}(\texttt{handshake\_messages})$ that X produces in the `IdentityVerify` message of Protocol 1, using the technique which we described in Section 3.4.3. Let

$$\sigma_{\bar{X}} = S_{\bar{X}} + s_{\bar{X}} H_3(\mathrm{ID}_X \| \mathrm{LT}_{AX} \| \texttt{handshake\_messages}),$$

and as before,

$$\sigma_{\bar{A}} = S_{\bar{A}} + s_{\bar{A}} H_3(\mathrm{ID}_A \| \mathrm{LT}_{AX} \| \texttt{DelegationToken}_X).$$

Then X can create a partial aggregate signature of the form

$$\langle \sigma_{\bar{A}} + \sigma_{\bar{X}}, Q_A, Q_{\bar{A}}, Q_X, Q_{\bar{X}} \rangle.$$

Assuming $m_A = \texttt{DelegationToken}_X$ and $m_X = \texttt{handshake\_messages}_X$, the verifier, when given the aggregate signature $\langle \sigma, Q_A, Q_{\bar{A}}, Q_X, Q_{\bar{X}} \rangle$, can replace two separate signature verifications with the check:

$$\begin{aligned}\hat{e}(P_0, \sigma) = \hat{e}(Q_0, H_1(\mathrm{ID}_A)) \cdot \hat{e}(Q_{\bar{A}}, H_3(\mathrm{ID}_A \| \mathrm{LT}_{AX} \| m_A))\\ \cdot\, \hat{e}(Q_A, H_3(\mathrm{ID}_A \| \mathrm{LT}_{AX}) \cdot \hat{e}(Q_0, H_1(\mathrm{ID}_X))\\ \cdot\, \hat{e}(Q_{\bar{X}}, H_3(\mathrm{ID}_X \| \mathrm{LT}_{AX} \| m_X)) \cdot \hat{e}(Q_X, H_3(\mathrm{ID}_X \| \mathrm{LT}_{AX}).\end{aligned}$$

Clearly, this aggregation technique can save more as the length of the delegation chain increases. Should X want to further delegate A's credential to another resource provider Y, X can in principle, repeat A's actions with X becoming the delegator and Y the delegation target. X needs to sign a delegation token for Y in which case $\texttt{DelegationToken}_Y = (\mathrm{ID}_X, \mathrm{ID}_Y, \texttt{J\_req}, \texttt{Policy}, \mathrm{LT}_{XY})$. As before, a verifier will have no problem constructing the necessary short-term public keys $(P_{\bar{A}}, P_{\bar{X}})$ for verifying $\texttt{DelegationToken}_X$ and $\texttt{DelegationToken}_Y$, respectively. In order to reduce the bandwidth requirement for transmitting signed tokens from Y to the verifier as part of the TLS handshake, Y can aggregate both signed tokens together with the signed handshake messages needed for Protocol 1.

We remark that it is currently feasible to produce RSA aggregate signatures with a special instantiation of the RSA signature scheme [121]. However, not all RSA public keys can be used to achieve signature compression. For example, no two users can share the same modulus N and the RSA scheme must have the property of certified trapdoor permutations[2]. These and other mathematical constraints discussed in [121] place more burden on and trust in the CA in ensuring that the public keys have the correct structure before certification. Hence, it is not clear if such a scheme is suitable for a grid environment where a huge number of proxy certificates are issued by the grid entities themselves.

Alternative. There is another straightforward and natural delegation technique using properties from HIBC. Since A's proxy is at level 2 of the key hierarchy in the IKIG setting, she can, in principle, act as a PKG and issue a short-term private key to X as if X were an entity at level 3 below A in the hierarchy. The issuance of the private key can indicate an act of delegation if the associated public key contains information that binds X's identifier and some delegation policy enforced by A. For

[2] A trapdoor permutation is certified if one can verify from the public key that it is actually a permutation [20].

instance, A can extract $S_{\bar{X}} = S_{\bar{A}} + s_{\bar{A}} P_{\bar{X}}$ with her secret value $s_{\bar{A}}$, yielding a short-term private key matching $P_{\bar{X}} = H_1(\mathrm{ID}_A \| \mathrm{ID}_X \| \texttt{J\_req} \| \texttt{Policy} \| \mathrm{LT}_{AX})$, and send $\langle S_{\bar{X}}, s_A P_0, s_{\bar{A}} P_0 \rangle$ to X through a confidential and integrity protected channel. For a third party to verify if the delegation has taken place, the verifier: (i) authenticates X[3], and (ii) checks if X knows the private key corresponding to $P_{\bar{X}}$, through a challenge-response protocol such as the TLS handshake. This method is related to the delegation approach proposed in [41]. We opted for Protocol 3 in our proposal because it works similarly to the delegation protocol deployed in the GSI. Also, transmission of a signed delegation token in Protocol 3 does not require a secure channel that preserves data confidentiality and integrity[4]. Furthermore, confirming condition (i) in the alternative approach requires an extra step from the delegator in producing and presenting a signature that proves its authenticity.

4.5 Key Management in IKIG

Here, we look at various aspects of key management including key generation, key update, key revocation and key storage, for our IKIG proposal.

4.5.1 Parameter Generation and TA Initialization

During the initial system setup phase, the TA runs a Bilinear Diffie-Hellman (BDH) parameter generator on input a security parameter k to generate groups $\mathbb{G}_1$, $\mathbb{G}_2$ of large prime order q and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. It then performs the Root Setup of the Gentry-Silverberg HIBE and HIBS schemes to produce a master secret s_0 and its system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$.

Currently the GT uses 1024-bit RSA public keys in public key certificates and 512 bit keys in proxy certificates. Smaller key sizes are used in proxy certificates mainly because of their frequent use in job submissions and also because of the computa-

[3] This is needed to convince the verifier that he is indeed interacting with X.

[4] It is common that delegation from one party to another can be performed after they have established a secure TLS channel. However, in the latest development of the GT4 (see Figure 2.8), credential delegation can be an independent service to a job request. Hence, a delegation technique without the need of a private and integrity protected channel can be seen as an advantage.

tionally intensive nature of RSA key pair generation [186]. Since IBC primitives use much smaller key sizes and have efficient key generation algorithms, our proposal has the luxury of using parameters of roughly similar security level to 1024-bit RSA keys for both users' long-term and short-term credentials. This can be achieved by working with a supersingular elliptic curve of embedding degree 4 over $\mathbb{F}_{2^{271}}$ [75, 77]. This choice results a corresponding group of prime order q approximately equal to 2^{252}. Elements of this group can be represented using 272 bits. Since all arithmetic is carried out in fields of characteristic 2, group operations and pairing computations can be implemented efficiently [12]. In addition to the curve and group selections, we require hash functions for the Gentry-Silverberg HIBE and HIBS schemes. The outputs of H_1 and H_3 are elements of $\mathbb{G}_1$, while H_4 gives an output with approximately $\log_2(q)$ bits. Note that the size of outputs of H_2 and H_5 are dependent on the bit length of plaintexts, n, and in our case, we assume that $n = 256$, since this is sufficient for our proposed protocol messages (e.g. pre-master secrets and hashed handshake messages). We remark that all these aforementioned parameters of the TA are assumed to be bootstrapped into the grid system.

4.5.2 User Registration

When a new grid user A goes to a nearby RA, the RA performs the following steps:

1. The RA verifies A's identity by checking her passport (or an acceptable ID-card). Once the check succeeds, the RA compares A's identity with its global identity list and subsequently assigns her a distinguished string ID_A. We propose that the identifier has the form

   "`/C=UK/O=eScience/OU=RHUL/CN=Alice/Y=2006`",

   which is based on the syntax from [101].

2. The RA submits A's application to the TA through its website using the HTTPS protocol (assuming the RA has an authentic and pre-distributed copy of the TA's system parameters). One way for the TA to authenticate the RA is by using a shared secret, for example, a password. If A's application is approved, the TA generates A's long-term private key as $S_A = s_0 P_A$, where

$P_A = H_1(\mathrm{ID}_A)$ is the matching long-term public key, and sends it back to the RA through the established secure TLS channel. The long-term credential for A and the TA's system parameters are distributed to A through a temporary storage medium such as a pen drive.

3. A performs the Lower-level Setup algorithm of the Gentry-Silverberg HIBE (or HIBS) scheme, picking a random $s_A \in \mathbb{Z}_q^*$ which she will keep secret. She then defines her Q-values as $\langle Q_0 = s_0 P_0, Q_A = s_A P_0 \rangle$.

The user's client must create a proxy credential every time the user "signs on" to the grid system. In IKIG, this is done as follows:

1. A runs the Extract algorithm of the Gentry-Silverberg HIBE (or HIBS) scheme to generate a short-term private key $S_{\bar{A}} = S_A + s_A P_{\bar{A}}$, where $P_{\bar{A}} = H_1(\mathrm{ID}_A \| \mathrm{LT}_A)$ is the corresponding short-term public key.
2. She also performs the Lower-level Setup algorithm of the Gentry-Silverberg HIBE (or HIBS) scheme, picking a random $s_{\bar{A}} \in \mathbb{Z}_q^*$ which will be kept secret. The Q-values for A's proxy are $\langle Q_0 = s_0 P_0, Q_A = s_A P_0, Q_{\bar{A}} = s_{\bar{A}} P_0 \rangle$. Note that A's TA does not know $S_{\bar{A}}$ and $s_{\bar{A}}$, and thus key escrow is limited, unless the TA behaves maliciously in mounting an active attack to impersonate the user to other entities.

As described in Section 4.4.1, A's short-term private key is stored temporarily in a local file system in unencrypted form. Only then is A ready and allowed to submit job requests.

It is worth noting that long-term keys used by hosting resources can be obtained using the above manner by the resource owners.

We have now explained how the long-term and short-term keys discussed in Table 4.2 are derived and used in our IKIG setting.

4.5.3 Key Update

A's long-term public key is fixed as $P_A = H_1(\mathrm{ID}_A)$, where the associated long-term private key is $s_0 P_A$, and s_0 is the master secret of the TA. In a grid environment, it is normal practice to renew the user's long-term keys on a yearly basis. In our proposal, this can be done through the TA issuing a new private key $s_0 P_{A'}$ directly to the user through a secure channel which can be established via Protocol 1. For example, A's identifier can be updated by updating the year field as follows:

$$\mathrm{ID}_{A'} = \text{“}\texttt{/C=UK/O=eScience/OU=RHUL/CN=Alice/Y=2007}\text{”}.$$

This is a more proactive approach as compared to current practice in PKI because the TA can easily compute the user's new long-term public key without requesting a new public key from the user. However, we have to enforce a policy whereby, in the event of compromise of the user's current private key right before the issuance of a new private key, the user must, upon her discovery of the incident, contact her RA in person to obtain a new private key.

The user creates a new short-term public/private key pair every time she signs on to the system. As with the current GSI setting, we assume the default lifetime for these keys is 12 hours. These short-term keys are used for various security services such as mutual authentication, single sign-on and delegation. Upon expiry of the proxy credential, these keys will be deleted from the local file system where they are temporarily stored. Should the validity of the proxy credential need to be extended, the user needs to be notified before the credential expires so that it can be renewed by the user. This may occur, for example, when a job execution takes longer than 12 hours and this event is not foreseen by the user concerned.

4.5.4 Key Revocation

For key revocation in the GSI, the user is expected to periodically check either a certificate revocation list (CRL) stored in a trusted directory or the CA's web site, depending on the policy enforced by their local administrator. It is recommended that the user updates their local copies of CRLs at least once a day [8]. However, many users do not bother doing this in reality. This may not cause serious concern as the CA can always instruct the user's entry in a grid Gatekeeper's (or a Resource Broker's) grid-map file to be removed when the CA is notified about key compromise of the user. At the time of writing, the GSI still does not support online certificate status protocol (OCSP) [82].

In the IBC setting, a number of revocation mechanisms are possible. We could use a more fine-grained identifier [25]. For example, we could extend the user's identifier to include another field which specifies a month, such as:

"`/C=UK/O=eScience/OU=RHUL/CN=Alice/Y=2006/M=January`".

This allows automated expiry of public keys after one month (hence the window of exposure is also limited to a month). The granularity of the user identifier must not be so complex that it loses its predictability. In grid environments, only short-term keys are used to encrypt or sign data (recall in Section 2.2.2, we explained that one of the motivations for using proxy credentials in grid environments is to limit the exposure of long-term private keys). Hence, the above suggested granularity may be adequate for most applications. However, should this approach prove insufficient, e.g. in some grid applications requiring high security, then existing PKI revocation mechanisms such as CRLs and OCSP can be adapted to the IBC setting. See [144] for a longer discussion of key revocation in identity-based PKI and traditional certificated-based PKI.

Revocation of short-term keys is a minor concern here as these keys will be destroyed upon expiry of their validity periods. Again though, this might not be sufficiently quickly in high security applications. In any case, the current GSI revocation approach would also need to be improved to handle this.

4.5.5 Integrating with MyProxy

MyProxy [15, 137], which we have discussed in Section 2.4, is gaining in popularity and is widely used in real grid applications. Being web-based, it serves as an online credential storage system that enables users' long-term private keys to be accessed from anywhere through the Internet with users being authenticated through a password-based mechanism. We envisage that this kind of service could be equally useful in our IKIG approach. We show how IKIG functionality can be exploited to support MyProxy.

As in the current MyProxy architecture, we assume that users' long-term private keys are stored at the MyProxy server in an encrypted form, the encryption keys being derived from passwords shared between users and the MyProxy server. The user must first establish a secure channel with her MyProxy server through a server-authenticated TLS protocol. Then she would submit a proxy request which includes her identity, her password, and the desired lifetime of the proxy credential to the MyProxy over the secure channel. Unlike the current GSI and MyProxy approach (which requires new RSA short-term public/private key pair generation on the user side), in the IKIG setting, the MyProxy server can determine the user's new short-term public key $P_{\bar{A}} = H_1(\mathrm{ID}_A \| \mathrm{LT}_A)$ directly from the user's identifier and the indicated lifetime. If the MyProxy server is able to successfully decrypt the user's encrypted long-term private key s_A using the password, then it extracts the short-term private key $S_{\bar{A}} = s_0 P_A + s_A P_{\bar{A}}$ associated with $P_{\bar{A}}$. The MyProxy server then forwards the proxy credential $(P_{\bar{A}}, S_{\bar{A}})$ back to the user through the secure channel. It is worth noting that the MyProxy server in such a setting is, in fact, providing a distribution service for private keys to its users roughly equivalent to private key distribution from a TA to its users in a typical identity-based cryptosystem.

We remark that the design of MyProxy allows escrow of the user's private key in the original GSI setting. A malicious administrator of the MyProxy server can always intercept the user's valid password and later impersonate the user to other grid entities. Therefore, users must trust the MyProxy server to behave honestly. This places GSI operating with the MyProxy plug-in on roughly an equal footing with IKIG in terms of key escrow. We also note that, independent of the MyProxy protocol, the combined use of user passwords and the TLS protocol may have other security implications. We return to this issue in Chapter 6.

4.6 Security Analysis

One of the main objectives of designing a secure infrastructure for grid applications is to overcome security threats caused by malicious behavior of adversaries (who could be legitimate users or outsiders). By exploiting weaknesses that may be inherited or overlooked in the design of the infrastructure, the adversary aims to gain unauthorized access to grid resources, modify sensitive records or files, steal valuable or classified data and so on. When billing comes into place for commercial grid applications, the adversary will have even more incentive to attempt to get free use of grid resources. The adversary may realise his goals in causing security breaches, for example, by trying to:

- impersonate a legitimate user to a MyProxy server using guessed passwords in order to retrieve a valid proxy credential from the server;
- forge a signature and masquerade as a genuine job requestor in order to get access to a resource;
- re-use previously negotiated session keys for continued access to a resource; or
- break into a resource hosting server and modify the grid-map file residing in the server.

In this section, we informally analyse the security of the authenticated key agreement and delegation protocols (Protocols 1 and 3, respectively). The scope of the analysis is limited to our protocols since these form the core components of IKIG. In our analysis, we assume that the adversary is able to eavesdrop and manipulate, insert, re-route, and/or delete messages. The TA, however, is assumed to be curious but honest. This means the TA can intercept messages sent by its users but does not actively impersonate the users to other parties.

4.6.1 Mutual Authentication and Key Agreement

The security of the TLS protocol and its predecessor, SSL, have been well-studied. Results, for example in [110, 145, 184], show that the TLS (and SSL) protocol is adequately secure, providing it is implemented carefully. Otherwise, it could be vulnerable to various side channel attacks [38, 179]. In Protocol 1, we replaced certificates with identifiers. Here, we will discuss if the changes impact on the security of the TLS protocol.

In step (2) of Protocol 1, we have removed the `Certificate` message that the server must send to the client in the original TLS specification. It has been replaced with `ServerIdentifier` which contains ID_X and LT_X. Also, the `Certificate` message in step (3) which is supposed to contain the client's certificates has become `ClientIdentifier`, containing ID_X and LT_X. Entities A and X do not have to verify the validity of the respective identifier that they each received. Since we assume that the TA has carried out its responsibility appropriately, A and X can each trust that the exchanged identifier came from the genuine party if this party can prove possession of the associated private component. Thus, if X can successfully verify the signed handshake messages from A using a public key constructed from A's identifier, then A is authenticated to X. On the other hand, for A to authenticate X, she must receive the correct verification value in `ServerFinished` from X, which was calculated based on the pre-master secret that she has chosen. This confirms that the server has recovered the correct pre-master secret that she sent encrypted under the public key that should belong to the server. Should the adversary attempt to impersonate either the user or the server, he must obtain their respective private keys or break the Gentry-Silverberg HIBE/HIBS schemes.

We conclude that the replacement of certificates with identifiers has not weakened the security protection that the original TLS protocol offers.

4.6.2 Delegation

We saw in Section 2.2.2 that delegation in the PKI approach requires the delegation target to transport a fresh public key to the delegator. We remark that this must be carried out using an authenticated and integrity protected channel. This can be established through, for example, the TLS protocol. The failure to do so allows a simple man-in-the-middle-attack. Let A be the delegator, X be the delegation target and E be the adversary. The attack can be illustrated as follows:

$$\begin{array}{ll}
(1).\ X \rightarrow E: & PK_{\bar{X}},\ Sig_{\bar{X}}(\texttt{proxy\_request}) \\
(2).\ E \rightarrow A: & PK_{\bar{X}'},\ Sig_{\bar{X}'}(\texttt{proxy\_request}) \\
(3).\ A \rightarrow E: & \texttt{proxy\_certificate}_X
\end{array}$$

When X sends out a request for proxy certificate to A, E can simply intercept X's signed request and replace it with his. These are shown by messages (1) and (2) above. Note that X signed his request using a short-term private key $SK_{\bar{X}}$ which corresponds to a fresh short-term public key $PK_{\bar{X}}$ as an act to prove possession of $SK_{\bar{X}}$. If X's request was not transmitted to A through an authenticated channel, E can make his own request and impersonate X to A. The delegator A will, in turn, generate a proxy certificate which is intended for X but contains E's chosen public key $PK_{\bar{X}'}$, as shown in message (3). Now, E possesses a valid proxy certificate that allows him to act on A's behalf and using X's identity.

In our IKIG setting where Protocol 3 is being used, the transmission of a signed delegation token and the corresponding message does not require the same level of protection as in the case of the GSI. The adversary cannot impersonate the delegator to the delegation target unless it has knowledge of the delegator's private key. This is so because verification of the signed delegation token using a public key constructed from the delegator's identifier explicitly authenticates the identity of the sender. The only way for the adversary to succeed in impersonating the delegator or the delegation target is to break the Gentry-Silverberg HIBS scheme. It is also obvious that the Gentry-Silverberg HIBS scheme does provide message integrity. This can protect the integrity of the delegation token which contains the identities of the delegator and the delegation target.

With that, we conclude that the security of Protocol 3 appears to be as strong as the security that the Gentry-Silverberg HIBS scheme provides.

4.7 Performance Analysis

We have seen how conventional public key usage and management can be replaced by identity-based techniques in Section 4.5. There are two noticeable differences: (i) an entity's long-term or short-term public key can be used immediately without requiring any form of key authentication; and (ii) the use of certificates is completely eliminated.

Here we present a performance comparison between the standard GSI approach and our IKIG proposal, by examining two types of overhead: computational cost and communication cost. Computational cost refers to the amount of computation required to perform cryptographic operations. Communication cost indicates the total network bandwidth required for transmission of data between two parties.

Computational Costs. We first review and compare the main cryptographic operations involved in delivering security services in the standard GSI and in our IKIG proposal.

- Generation of proxy credentials: The RSA signature scheme is currently used in the GSI to provide proxy certificate signing and verification. Repeated RSA key pair generation, which includes generating two large primes[5], is easily the most computationally costly cryptographic operation in the GSI. In our IKIG proposal, generation of a short-term public key involves application of a cryptographic hash function while computation of a short-term private key requires only one elliptic curve point addition and one point multiplication.
- Authenticated key agreement: Although GT4 supports both the TLS protocol and the WS-SecureConversation/WS-Trust specifications, we will only compare our IKIG approach with the former as it has been shown that the transport-level protocol is faster than message-level protocols [161]. Considering that a mutual authentication within the grid environment requires the full TLS handshake using both the long-term and the proxy credentials of the two communicating parties, the main cryptographic tasks that the job requestor has to perform are as follows:

    1. two RSA signature verifications on the resource's long-term and short-term certificates;
    2. one RSA encryption of a pre-master secret key using the resource's short-term public key; and
    3. one RSA signature generation on some handshake messages using his own short-term private key.

[5] In practice, the way to generate large random primes is to first generate large random numbers, and then test them with a primality testing algorithm. Even though the algorithm is generally fast, it may claim that a large random number is prime when it is not. That is why the algorithm must be run enough times so that the error probability can be reduced below a desired threshold [169].

The resource is required to perform the same number of basic cryptographic operations as the requestor does. This includes decryption of the pre-master secret and verification of the requestor's signature on the handshake messages.

For Protocol 1, the user performs one encryption and one signing operation, while the resource has one decryption and one signature verification to perform. Note that even though the Gentry-Silverberg HIBS scheme requires four pairing computations for a signature verification, it is possible to pre-compute two of them and store the respective pairing values in a local cache. This can spread the costs incurred over many protocol runs. There are other computational costs involved, notably hashing, but these will turn out to be small in comparison to the costs of point multiplications and pairing computations.

- Delegation: The current GSI delegation protocol (as described in Section 2.3.2) involves an expensive RSA key generation and a signing operation at the delegation target's side. The delegator has to perform one signing operation and one signature verification operation. The verifier must perform three signature verifications[6], assuming the length of the delegation chain is one.

  On the other hand, our delegation method in Protocol 3 requires only signing of a delegation token by the delegator. Private key extraction by the delegation target is very efficient and the verifier has to perform only one signature verification.

Table 4.3 gives a more detailed view of the computational costs in terms of computation times. These timings were obtained through implementations of the RSA and HIBE/HIBS schemes based on the MIRACL library [160]. Our implementation uses the curves and parameters defined in Section 4.5.1, and were written in C/C++ and compiled with Microsoft Visual C++ 6.0. The small public exponent, $e = 2^{16} + 1$, is used for RSA encryptions (or signature verifications), and we assume that pre-computation of pairing values is possible. We use the Chinese Remainder Theorem (CRT) method for faster RSA decryptions (or signature generation). We use the eta pairing [10] for Tate pairing calculations. The time taken for a basic pairing computation using a Pentium IV 2.4 GHz processor is 3.88 milliseconds. Also, for

[6] Note that the delegation target must also prove possession of the short-term private key associated with the proxy certificate that it presented to the verifier. Since this can be achieved when the delegation target and the verifier perform mutual authentication using the TLS handshake, we omit this additional step here for the sake of simplicity.

Table 4.3: Performance trade-offs in computation times (in milliseconds) between the GSI and the IKIG settings on a Pentium IV 2.4 GHz machine.

	GSI		IKIG	
Operation	**RSA**	**Time**	**HIBE/HIBS**	**Time**
Key generation				
(a.) Long-term	1 GEN	149.90	1 EXT	1.69
(b.) Short-term	1 GEN	34.85	1 EXT	1.74
Authenticated key agreement				
(a.) Requestor	1 1024-bit VER 1 512-bit ENC 1 512-bit SIG 1 512-bit VER	2.67	1 ENC, 1 SIG	8.79
(b.) Resource	1 1024-bit VER 1 512-bit DEC 2 512-bit VERs	2.67	1 DEC, 1 VER	20.16
Delegation				
(a.) Delegator	1 512-bit SIG 1 512-bit VER	1.86	1 SIG	3.35
(b.) Delegation target	1 GEN 1 512-bit SIG	35.63	1 EXT	1.74
(c.) Verifier	3 512-bit VERs	0.84	1 VER	8.42

GEN = RSA parameter generation	EXT = HIBE/HIBS private key extraction
ENC = Encryption	DEC = Decryption
SIG = Signing	VER = Verification

simplicity, we limit the length of the delegation chain to one.

From Table 4.3, it is obvious that RSA encryption/verification in the GSI is very fast. However, we can see that key generation and delegation in the GSI are considerably slower than in IKIG. This is because of the cost of generating fresh RSA key parameters. Significant computational savings can be achieved by the resource provider in the IKIG setting when multiple distinct proxy credentials (used by different managed job services) are created simultaneously[7]. Nevertheless, it seems that Protocol 1 is slower than the actual TLS protocol. This is mainly because of the pairing computations (two pairings in both HIBE decryption and HIBS verification). Fortunately in our IKIG proposal, the computationally intensive pairing computations are performed at the high-performance resource side. Operations or

[7] Note that computing a public key in the IKIG setting is very efficient, i.e. less than a fraction of a millisecond. This cost is included in ENC/DEC and SIG/VER when we consider the HIBE/HIBS schemes.

computations at the user side are more lightweight than the server. Our proposal also offers the flexibility of spreading the cost of pairing calculations between the user and the server by adopting Protocol 2 in place of Protocol 1.

It is worth noting that the RSA and Gentry-Silverberg HIBE/HIBS schemes are very different from each other in terms of their mathematical properties. We also note that IKIG uses short-term keys which offer a greater security level than the 512-bit RSA keys do. Therefore, we believe that it is fair to conclude that the overall computational costs for the IKIG setting are comparable to those of the GSI. Recent results (see for example [10, 157, 158]) have shown improvements in computing pairings with the use of various optimisation techniques and this should give hope to faster HIBE and HIBS schemes in the near future. We remark that it is unlikely that significant algorithmic improvements for RSA computations will be forthcoming, since this field has been intensively researched for many years.

Communication Costs. We recall that in the IKIG setting, the TA's system parameters are $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$. An element of $\mathbb{G}_1$ can be represented by 272 bits since the group is defined as a subgroup of a supersingular curve of embedding degree 4 over $\mathbb{F}_{2^{271}}$. Also, H_2 and H_5 each produces an output with size 256 bits. The TA's system parameters are assumed to be pre-distributed to its users, as with pre-distribution of a CA certificate to users in the GSI setting.

Table 4.4 compares the communication costs of the standard GSI approach and our IKIG proposal. Note that we only consider the dominant communication costs between the job requestor and the resource, i.e. signed or encrypted messages and certificates, which have the biggest contributions to the network bandwidth.

In Figure 2.4, we saw that the size of a certificate for a 1024-bit RSA public key is

Table 4.4: Performance trade-offs in communication costs (in kilobits) between the GSI and IKIG.

Operation	**GSI**	**IKIG**
Authenticated key agreement	37.8	1.9
Delegation	7.4	0.8

about 2 kilobytes. Here, we ignore small fields in a certificate such as issuer, subject and validity period. Thus, for the purpose of comparison between GSI and IKIG, we estimate that the size of a standard certificate, which comprises an RSA public key and the issuer's signature, is roughly 1.5 kilobytes or 12 kilobits (after omitting small fields). We assume a proxy certificate has size 0.8 kilobytes or 6.4 kilobits. Ciphertexts and signatures produced using a short-term RSA key are 512 bits.

On the other hand, the size of a ciphertext produced by the full HIBE scheme of [80] in our IKIG setting is: $2 \times |\text{element of } \mathbb{G}_1| + 2 \times 256\text{-bit hash value} = 1056$ bits. Also, the size of a signature on a message using the HIBS scheme of [80] is: $3 \times |\text{element of } \mathbb{G}_1| = 816$ bits.

From Table 4.4, the communication cost for the TLS protocol in the GSI is estimated to be $2(12)+2(6.4)+2(0.5) = 37.8$ kilobits, since there are two public key certificates, two proxy certificates, one encrypted pre-master secret (for `ClientKeyExchange`) and one signed message (for `CertificateVerify`) being transmitted over the network[8]. In the case of IKIG, the figure of 1.9 kilobits refers to the encrypted pre-master secret and the signed message in Protocol 1[9]. This clearly shows that an execution of the TLS protocol in the GSI is significantly more costly than our Protocol 1 in IKIG in terms of bandwidth requirements.

The communication overhead for delegation between the delegator and the delegation target can be estimated straightforwardly from our discussion on delegation protocols in Section 4.4.4. The delegation protocol for the GSI requires transmission of a signed request, a short-term public key and a proxy certificate; whereas Protocol 3 has only a signed token.

Figure 4.2 shows that in the GSI, the communication cost between two entities rises quickly when delegation chains are extended, reflecting a truly flexible and scalable grid environment. (We assume that two grid entities would authenticate each other and establish a session key before delegation takes place.) In contrast, the communication cost incurred by our IKIG proposal increases at a much slower

[8] We remark that for simplicity, small components in the TLS protocol are not included in our calculation. For example, the sizes of a nonce and `cipher_suite` in the `ClientHello` message are merely 28 and 2 bytes, respectively [50].

[9] Similarly, we assume `ServerIdentifier` and `ClientIdentifier` are small fields which can be ignored in our calculation.

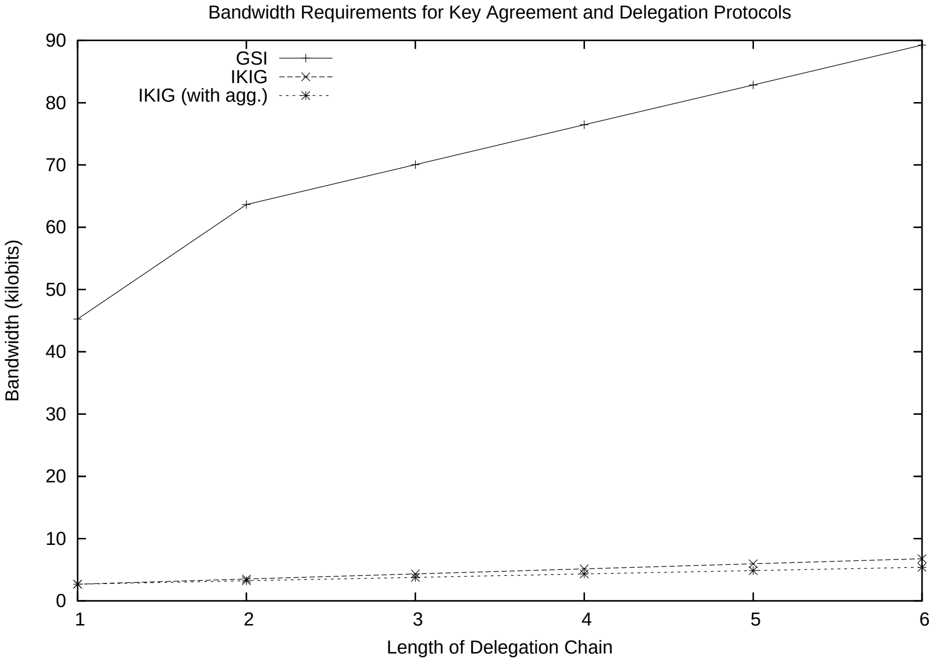

Figure 4.2: A performance graph which measures the network bandwidth required for authenticated key agreement and delegation in the GSI and IKIG.

rate. This indicates that IKIG may well be much more scalable than GSI. We also show, in Figure 4.2, that aggregation of signed delegation tokens and messages (as explained in Section 4.4.4) can play a part in saving bandwidth in the IKIG setting. In conclusion, the communication cost resulting from a job submission in IKIG is significantly lower than in GSI. The main reason for this is that in the IBC setting, there are no long-term and short-term public keys or certificates being transmitted between the user and the resource.

4.8 Discussion

In this section, we look into some practical issues surrounding the use of our IKIG approach. We discuss the possible impact of our identity-based techniques on web services security, some implementation issues, inter-operation between TAs and the limitations of our IKIG proposal.

4.8.1 Impact on Web Services Security

In GT4, the GSI supports both transport-level and message-level security. The recommended use of transport-level security as the default option instead of fully web services based security is driven by the relatively poor performance of message-level security implementations. This is clearly demonstrated in [161]. XML representations of data tend to be significantly larger than their equivalent binary formats. An XML message can expand roughly 4 to 10 times over its equivalent binary representation [45, 87]. This can substantially increase the communication costs, the latency of sending/receiving SOAP messages, and the time needed for parsing the XML data.

By using IBC in place of conventional public key techniques, we envisage that the sizes of SOAP headers may well be reduced substantially. For instance, if A wants to submit a signed job request in XML format to resource X, she would need to attach[10] her public key and certificate in the `KeyInfo` element if the RSA signature algorithm was used. This is shown in Figure 4.3. However, if we adopt the identity-based techniques, she could, in principle, simply include her identifier in the `KeyInfo` element of the XML signature. When X receives the signed XML message, it can construct A's public key merely based on A's identifier:

```
<ds:KeyInfo>
  <ds:KeyName>
    /C=UK/O=eScience/OU=RHUL/CN=Alice/Y=2006
  </ds:KeyName>
</ds:KeyInfo>
```

which is roughly 85 bytes, as compared to 1.2 kilobytes in Figure 4.3. This shows more than 90% saving in the content size of the `KeyInfo` element through the IBC approach. Note that we have not considered the case of proxy certificates and the size of signatures in this example. Therefore, the potential saving can be even

[10] Certificates are normally sent directly from the sender to the intended recipient in a dynamic environment such as grid. In other applications, it may also be appropriate to reference the sender's public key/certificate so that the receiver can obtain the sender's credential later from a public directory.

```
<ds:KeyInfo>
  <ds:KeyValue>
    <ds:RSAKeyValue>
      <ds:Modulus>
        3FFtWUsvEajQt2SeSF+RvAxWdPPh5GSlQnp8SDvvqvCwE6PXcRWrIGmV7twNf2T
        UXCxYuztUUClMIy14B0Q+k1ej2nekmYL7+Ic3DDGVFVaYPoxaRY0Y2lV8tOreyn
        WegpFbITXc8V6Y02QfR5O7Pn1/10ElslaF/TF8MQGqYE8=
      </ds:Modulus>
      <ds:Exponent>AQAB</ds:Exponent>
    </ds:RSAKeyValue>
  </ds:KeyValue>
  <ds:X509Data>
    <ds:X509Certificate>
      MIICCTCCAXagAwIBAgIQe0Sk4xr1VolGFFNMkCx07TAJBgUrDgMCHQUAMBIxEDA
      OBgNVBAMTB1Rlc3QgQ0EwHhcNMDMwODE1MDcwMDAwWhcNMDUwODE1MDY1OTU5Wj
      AkMSIwIAYDVQQDExlCb2IgQmFrZXIgTz1Cb2IgQ29ycCBDPVVTMIGfMA0GCSqGS
      Ib3DQEBAQUAA4GNADCBiQKBgQDcUW1ZSy8RqNC3ZJ5IX5G8DFZ08+HkZKVCenxI
      O++q8LATo9dxFasgaZXu3A1/ZNRcLFi7O1RQKUwjLXgHRD6TV6Pad6SZgvv4hzc
      MMZUVVpg+jFpFjRjaVXy06t7KdZ6CkVshNdzxXpjTZB9Hk7s+fX/XQSWyVoX9MX
      wxAapgTwIDAQABo1YwVDANBgNVHQoEBjAEAwIGQDBDBgNVHQEEPDA6gBABpU6Rp
      UssqgWYs3fukLy6oRQwEjEQMA4GA1UEAxMHVGVzdCBDQYIQLgyd1ReM8bVNnFUq
      D4e60DAJBgUrDgMCHQUAA4GBAF4jP1gGDbaq3rg/Vo3JY7EDNTp0HmwLiPMLmdn
      B3WTIGFcjS/jZFzRCbvKPeiPTZ6kRkGgydFOuCo5HMAxIks/LtnKFd/0qYT+AOD
      q/rCrwSx+F+Ro2rf9tPpja9o7gANqxs6Pm7f1QSPZO57bT/6afiVm7NdaCfjgMp
      hb+XNyn
    </ds:X509Certificate>
  </ds:X509Data>
</ds:KeyInfo>
```

Figure 4.3: A sample of the KeyInfo element based on RSA which is approximately 1.2 kilobytes.

more significant if the SOAP header contains multiple signatures and/or certificates. Moreover, an identity-based public key is predictable, self-describing and human-readable. Thus no special tools are needed to parse and render the key information[11]. This is indeed a very desirable property and it matches a fundamental objective of XML.

Another possible impact on web services security is a more lightweight TLS-like key agreement protocol based on the WS-SecureConversation [93] and WS-Trust [94] specifications. With similar reasoning to that behind the design of Protocol 1, simple identity-based security tokens can be used in place of signed security tokens in the web services setting. We also envisage that the use of IBC may well be able to simplify the design of XKMS services [97]. For instance, the Locate and Validate services in the XML Key Information Services specifications may not be needed since identity-based public keys are predictable and can be used on-the-fly.

[11] As with data presented in XML format, a simple text editor is sufficient for the manipulation of an identifier. In the case of a conventional public key such as RSA, it is a common practice to present the key in a certificate with PEM encoded format.

4.8.2 Implementation Issues

Here, we identify some implementation issues that need to be addressed in our IKIG proposal.

GSS-API. Currently, the GSI is built on top of GSS-API [119]. This allows security services to be easily added into grid applications through a set of callers in a generic fashion [69]. GSS-API, which is both transport (communication protocol) and mechanism (cryptographic scheme) independent, provides functions for obtaining credentials, performing authentication, encrypting and signing messages and so on. However, extensions to the standard GSS-API are needed to meet some of the requirements of GSI, such as handling of credential extensions and delegation at any time [126].

To implement our IKIG proposal, a few additional functions will need to be included in GSS-API. Firstly, when a client calls `GSS_Init_sec_context()` to establish a security token with a server, the server should be able to extract the client's identifier rather than obtaining a X.509 certificate in the credential acquisition step. Secondly, we require a delegation credential handle which can manage the use of signed delegation tokens in Protocol 3. Current standard GSS-API only uses a simple Boolean parameter for handling a delegation credential. This does not provide enough control over a delegation target. Since we have replaced proxy certificates with signed delegation tokens, the delegation handle should be able to support mechanisms that offer better control including duration of delegation, constraints on the tasks that may be performed by the delegation target and so on. Also, it is desirable to have a function which collects and coordinates parameters for aggregation of signatures. When the client wishes to aggregate a series of signed messages, `GSS_Wrap()`, which normally deals with message signing, should be able to handle and facilitate the aggregation.

OpenSSL. The authentication library of the GSI is based on OpenSSL [140] with some modification for supporting the use of X.509 proxy certificates. This library, in turn, makes use of GSS-API (and some extensions) to provide security mechanisms required by the standard TLS handshake protocol.

In the IKIG setting, we must define and include the `TLS_HIBE_WITH_DES_CBC_SHA` cipher suite that we proposed in Section 4.4.3 in an extension of the TLS specification. A version of the OpenSSL library that supports IKIG is needed. This library should be able to interpret system parameters used for the HIBE and HIBS schemes and allow creation of identity-based proxy credentials. A client and a server that are engaged in a run of Protocol 1 must also be provided with the ability to construct public keys based on identifiers extracted through the IKIG-enabled GSS-API. Moreover, it is clear that the OpenSSL library will need to support message encryption/decryption and signature generation/verification using HIBC in order to implement Protocol 1.

XML Signature & Encryption. In GT4, the optional message-level security provided by the GSI is based on the WS-Security standard and various specifications which make use of XML signatures and encryption as the building blocks. The current XML Signature and Encryption specifications cover only RSA and DSA based algorithms [53, 54]. Recently, there has been a proposal to use an elliptic curve based signature algorithm for XML signatures [24].

In order to support web services security using HIBC, we need to define the syntax and processing of XML signatures and encryption for the HIBS and HIBE schemes. The HIBS and HIBE XML schema definition should include syntax used for parameters for key generation, system parameters, group and element sizes, key values, and so on. The facts that the identity-based approach is certificate-free and that public keys can be constructed easily mean that the `<KeyInfo>` element in the new XML schemas should be simplified. This should in turn result in shorter schemas than those using standard certificates.

4.8.3 Inter-TA Operation

If our IKIG proposal is to be deployed roughly at the scale of one TA for each country (as with the European grid projects that we described in Section 2.3.2), we expect that the TAs' system parameters would be bootstrapped in the grid system and updated by the users through their GT clients. As with current grid deployment, trust relationships between TAs can be established through the European Grid PMA without a root TA. System parameters of the TAs are then assumed to be trusted by all users and recognised by the grid system. When a user communicates with a remote resource registered with a foreign TA, the user's grid client will select the foreign TA's system parameters for data encryption and signature verification. This is because the Gentry-Silverberg HIBE and HIBS schemes defined in Section 3.4.3 require the use of public parameters of the TA. On the other hand, if the user decrypts a ciphertext encrypted using her public key or signs a message with her private key, the grid client should take its own TA's system parameters as input for the decryption and signature generation algorithms.

Alternatively, we envisage that the European Grid PMA could serve as a root TA. This can be achieved by adding an extra level to the hierarchy that we illustrated in Figure 4.1. In this setting, all users of the grid system share the same root TA system parameters. With that, the user and the user proxy in Figure 4.1 are relegated to levels 2 and 3, respectively. These changes, however, would increase the computational costs of Protocol 1, 2 and 3 because of the extra pairing computations.

4.8.4 Limitations

Our IKIG proposal has several limitations. Being based on IBC, IKIG inevitably inherits an escrow facility. Although a TA may not be able to forge a signature using exactly the same private components (short-term credential) chosen by a user proxy, the TA can always impersonate a user by selecting a different set of secret values and producing a valid signature that is verifiable using the user's public key. Nevertheless, we have explained in Section 4.5.5 that integration of the MyProxy system into the GSI has introduced a similar property. This seems to be acceptable in most ongoing grid projects since there is only one or very few CAs/TAs in each nation that everyone is expected to trust. This "limited" key escrow issue will become one of the motivations for our study of dynamic key infrastructure for grid (DKIG) in the subsequent chapter.

We have shown earlier that the identity-based techniques offer a more flexible and lightweight approach in creating and using public keys. However, it is worth noting that the identity-based key revocation method does not seem to offer any clear advantage over conventional public key revocation techniques. Fine-grained identifiers which make use of dates and short time periods require issuance of the matching private components regularly. Grid applications with high security requirements will still need to rely on traditional means of revoking public key certificates such as CRLs and OCSP.

Another drawback of employing IBC in our proposal is the cost of pairing computations. Even though IKIG is far more lightweight than the GSI in terms of communication overhead, the relatively slow pairing computations in the HIBE and HIBS schemes constrain the advantages that the proposed identity-based techniques can offer. As mentioned before, pairing-based cryptography is still relatively new compared to RSA. We believe that the performance of pairing computations will continue to improve with further research.

4.9 Summary

We have discussed at length how identity-based techniques can replace conventional PKI and be used to offer an alternative security infrastructure for grid. We proposed a TLS-supported identity-based authenticated key agreement protocol which uses only short-term keys. Our infrastructure also supports single sign-on and delegation in a very natural way. The overall computational overheads for our proposed identity-based key infrastructure seem to be comparable to PKI. Interestingly, the computational costs that would be incurred at the user's client in our proposal is roughly a few times less than it is with PKI, at the expense of increased computation at the server side. This aligns well with the whole idea of grid computing to allow the user with an average- or low-end platform to "outsource" her computational tasks or operations to more powerful and high-performance servers. In terms of communication costs, our proposal appears to be significantly more lightweight and less bandwidth-consuming than PKI because of its certificate-free nature and small key sizes. This may enable the expansion of grids to service users with bandwidth-limited or low memory platforms.

Chapter 5

Dynamic Key Infrastructure for Grid

Contents

This chapter proposes an identity-based and escrow-free key infrastructure for grid applications. We introduce the concept of a dynamic key infrastructure for grid. In this approach, each entity in the system acts as his own PKG and obtains a certificate from a traditional Grid CA for the corresponding public parameters. This allows support for proxying and single sign-on but removes key escrow inherent in a pure identity-based approach. We also present TLS-like authenticated key agreement and delegation protocols for our dynamic key infrastructure, and consider their performance in comparison with the corresponding protocols in the GSI and IKIG.

5.1 Motivation

In the previous chapter, we learned that key escrow is inevitable in IBC. Despite that, key escrow seems to be acceptable for most current grid applications since the use of MyProxy also involves the same issue. However, we envisage that when computational grids become commercialised and payment is involved, key escrow that prevents strong non-repudiation[1] may become a major issue. Since IBS schemes generally do not provide the property of strong non-repudiation, users can, for example, fraudulently deny any charges for grid resource usage.

Even though the issue of key escrow can be circumvented by introducing multiple TAs and the use of threshold cryptography, it may be unrealistic to deploy such an approach within a grid environment. In order to enable secret sharing among multiple TAs, a high degree of co-operation between these TAs is needed. They must agree on common policy and standardised mechanisms to manage the shared secret. In addition, the communicating parties must also find out in advance the set of TAs that their system uses for encryption and decryption. These and other potential restrictions seem to reduce the suitability of the secret sharing approach for a heterogenous environment such as grid. This set of problems may become harder to solve if TAs in a number of different countries are to be employed.

The main objective of this chapter is to investigate a means of resolving the key escrow problem while preserving, as much as possible, the advantages that identity-based techniques offer, in particular the benefits of our IKIG proposal. Our focus, as in the previous chapter, will be on simplifying key management aspects of grid applications that rely heavily on both long-term and short-term entity credentials.

5.2 Overview of Dynamic Key Infrastructure for Grid

We propose a dynamic key infrastructure for grid (DKIG). This term is intended to capture the notion that a user proxy credential can be created dynamically and on-the-fly based on a static long-term credential. DKIG is a hybrid approach com-

[1] Here, strong non-repudiation refers to the inability of any party, including a malicious CA/TA, to impersonate a user by producing a valid signature as if it were generated by the actual user.

bining identity-based techniques at the user level and traditional PKI to support key management above the user level. In this hybrid setting, each user publishes a fixed parameter set through a standard X.509 certificate; this parameter set then allows users to act as their own PKGs for the purpose of managing short-term keys which will, in turn, be used for single sign-on and delegation.

We envisage that DKIG is suitable for grid applications because of the following properties, of which some can be difficult to emulate with conventional PKI:

- While we maintain the hierarchical structure of PKI above the user level, our proposal provides improved flexibility at the user level (ground level), where peer entities freely create and manage their own proxy credentials without any intervention from a TA/CA.
- The fixed parameter set that each user possesses is different from a proxy certificate. The parameter set can be seen as a long-term *public-key mould* from which other entities can compute short-lived public keys instantly. The corresponding private keys can be generated directly by the entity himself without interacting with the TA, a feature that IKIG also possesses.
- Our DKIG framework not only solves the key escrow problem in IBC, it also remove the requirement for short-term private key distribution between the entities and their respective TAs. The latter is an important benefit as distributing short-term private keys using secure channels can be a tedious and expensive operation within a grid environment.
- The replacement of the conventional contents of standard X.509 certificates with identity-based cryptographic system parameters implies minimal changes to the current overall pure PKI-based GSI framework.
- Most of the attractive properties of IBC, such as using identity-based and small-size public keys, can still be preserved in the DKIG setting. Even though each entity needs to publish their system parameters through a certificate, proxy credential management can be done without any use of proxy certificates.

As with Chapter 4 where we introduced IKIG, we will explain how our hybrid DKIG approach can be used to support single sign-on, mutual authentication and key agreement, and delegation. In general, DKIG works in a very similar way to IKIG, but without the key escrow issue. However, each entity in the DKIG setting will possess an authentic and valid certificate that needs to be transmitted to other parties when performing security services such as mutual authentication and delegation. In short, we can regard IKIG as a lightweight solution for grid applications that tolerate key escrow, while we can think of DKIG as a fix to remove the key escrow problem of IKIG at the expense of an acceptable increase in bandwidth requirement and computational cost.

5.3 Related Work

Not long after the publication of Boneh and Franklin's work in [25], Chen *et al.* [41] proposed a hybrid certificate-based and identity-based approach related to but different from our work. In Chen *et al.*'s proposal, they presented a certificate-based approach for TAs above the domain level, e.g. between a domain TA and a root TA, and identity-based techniques for entities below the domain TA. It is assumed that all TAs in this hybrid architecture use non-identity-based public/private keys, while the users have identity-based key pairs. The root TA will verify the public keys or system parameters of the domain TAs and issue certificates accordingly. The domain TAs, in turn, extract identity-based private keys for their respective users using the certified system parameters. Hence, when a user wants to encrypt a message for another user belonging to a different domain, the sender must obtain the system parameters for the domain TA of the receiver and its root TA. Although the proposal of [41] provides a certificate-free approach at the user level, key escrow remains possible since the domain TA possesses a master secret which is used to extract the users' private keys. We note that the security architecture proposed by Smetters and Durfee in [164] for a secure email application (also for secure IP-based communications) using IBC is closely related to Chen *et al.*'s approach. As we have already discussed in Section 3.6, the architecture proposed by Smetters and Durfee uses conventional certificate-based PKI above the domain level, while key management within the domain is identity-based. However, private key distribution from a PKG to its users requires the standard TLS protocol which is still certificate-based. In our DKIG, this limitation is removed and short-term key management (in the context of a grid application) is fully identity-based.

On the other hand, the key escrow issue of the Boneh-Franklin IBE scheme partly drove the proposals of certificate-based encryption [79] and certificateless public key cryptography [5]. As we discussed in Section 3.2, the authors of both proposals split the private key of a user into two components: one computed by the user himself, and another by the TA. Because of this requirement, the user's public key is no longer identity-based. To encrypt a message, the sender must obtain the intended recipient's public key which is an arbitrary string like a conventional public key. Hence, the benefits that identity-based techniques can bring to grid applications, as we have discussed in the previous chapter, may no longer be applicable. Although the certificateless approach of [5] can remove the extensive use of certificates in grid applications, it is not clear if the use of two-component-based private keys will actually simplify current PKI-based public key management in grid applications. In summary, even though the key escrow issue can be resolved using the proposals of [5, 79], the suitability of these approaches as alternatives to the current PKI-based GSI requires further exploration.

Parts of the research findings presented in this chapter appear in [118]. In addition, further aspects of the hybrid DKIG approach are explored in [43], in which we revisited the GSI in GT2 and presented some improvements to the security architecture.

5.4 Design of DKIG

As with IKIG, we envisage that the TAs' system parameters in the DKIG setting are boot-strapped into the grid system and can be updated by the users though their GT clients. Also, trust relationships between TAs[2] are established through a Grid PMA.

In Figure 5.1, we show the hierarchy of entities in our DKIG proposal. Traditional certificate-based PKI is used by users and resource providers to manage their long-term credentials. The TA issues standard certificates that contain system parameters chosen by users and resources, in a similar way as a typical CA certifies user-chosen

[2] Note that the TA here is in fact a traditional CA. For the purpose of consistency in using terminology in our proposals, and to avoid confusion between the 'CA' in our proposals and the CA in the GSI, we use the term TA in all our security architectures.

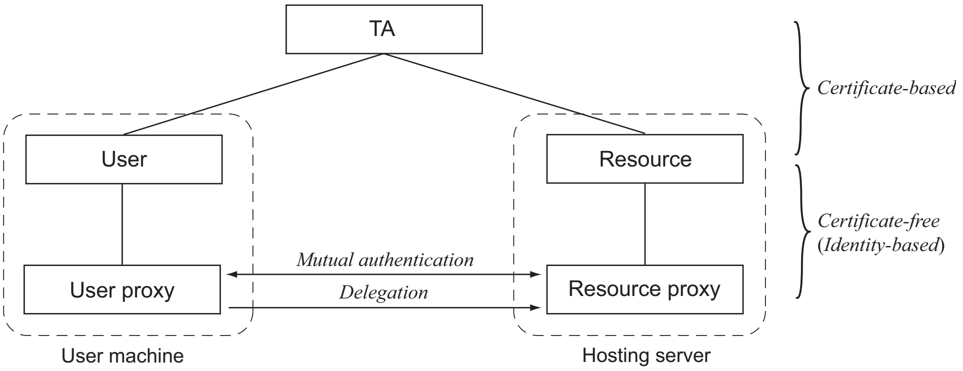

Figure 5.1: The hierarchical structure of entities in the DKIG setting.

Table 5.1: A's long-term and proxy credentials.

Credential Type	Public Key	Private Key
Long-term	$P_A = s_A P_0$	$S_A = s_A$
Short-term	$P_{\bar{A}} = H_1(\mathrm{ID}_A \| \mathrm{LT}_A)$	$S_{\bar{A}} = s_A P_{\bar{A}}$

public keys. We remark that some of these parameters can be fixed to reduce communication costs of the system. Further details of parameter selection, key generation and certification will be given in Section 5.5.

At the user level, the certified system parameters are used by the users (and resources) to extract short-term private keys corresponding to some fixed formatted identifiers. These short-term private keys and identity-based public keys are then taken as input to IBE and IBS schemes for providing security services, such as single sign-on, mutual authentication and delegation. Here, we will use Boneh and Franklin's full IBE scheme [25], and Cha and Cheon's IBS scheme [39] for data encryption and signing, respectively. Note that we are not using the Gentry-Silverberg HIBE and HIBS schemes any longer because in our hybrid approach, identity-based techniques are used starting at the user level. Since the next level is already the user proxy level, non-hierarchical-based IBE and IBS schemes seem to be sufficient. Nevertheless, the hierarchical identity-based approach can be used as an alternative delegation technique. This will be discussed in Section 5.4.4.

We will again consider a simple scenario where A wants to submit a job request `J_req` to a target resource X with long-term and short-term credentials as shown in Table 5.1. We assume A possesses a certificate Cert_A which contains her system parameter set, including $s_A P_0$, and other information such as issuer (A's TA), subject (A's identifier) and validity period. In order to keep the size of the certificate minimal, we propose the use of the BLS short signature scheme [29] for signing the certificate by the TA. As with the IKIG setting, A's short-term public key $P_{\bar{A}}$ is computed based on her identifier and a lifetime (LT_A). The matching private component is obtained by A using her master secret s_A.

5.4.1 Single Sign-On

A can store her certificate and short-term private key $S_{\bar{A}}$ in a local file system accessible by her GT client so that the client can use them when needed for single sign-on. Since A's short-term public key is identity-based, a proxy certificate is not needed. Should A's client be challenged for proof of possession of its short-term private key, it could produce a signature using $S_{\bar{A}}$ from the local file system and forward it to the challenger, along with A's certificate. The challenger verifies the authenticity of the system parameters contained in the certificate, and the parameters together with A's short-term public key (which can be computed by any entity) are then used to verify the signature.

5.4.2 Authorization

A generates an identity-based signature using the Cha-Cheon IBS scheme (which we described in Section 3.4.2) on `J_req` with her short-term private key $S_{\bar{A}}$. She then submits the signed request to a Managed Job Factory (MJF) that resides on X through a GRAM service.

$$A \rightarrow X : \mathrm{Cert}_A, \mathrm{LT}_A, \texttt{J\_req}, Sig_{\bar{A}}(\texttt{J\_req})$$

The signed request is of the form (U, V), where $U = rP_{\bar{A}}$, $V = (r + h)S_{\bar{A}}$. Here, r is a random element of $\mathbb{Z}_q^*$ and $h = H_6(\texttt{J\_req}, U)$. Note that the MJF can verify $Sig_{\bar{A}}(\texttt{J\_req})$ by computing $P_{\bar{A}}$ as a function of A's identifier and the lifetime LT_A, and using the system parameters from Cert_A. The signature verification can be done by checking if:

$$\hat{e}(P_0, V) = \hat{e}(s_A P_0, U + hP_{\bar{A}})$$

where P_0 and $s_A P_0$ are obtained from A's system parameters. Subsequently, as with the GSI and IKIG, the MJF maps A's identifier to its grid-map file before granting access to A. When the check succeeds, the MJF instantiates a managed job service for the job request and returns an endpoint reference to A. Clearly, this technique is very similar to IKIG except that it requires the transmission of a set of certified system parameters. However, the benefit of doing this is the removal of key escrow from the system.

5.4.3 Mutual Authentication and Key Agreement

Here, we present a hybrid identity/certificate-based authenticated key agreement protocol based on the TLS handshake protocol. Our protocol, Protocol 4, is needed for mutual authentication between A and the created managed job service before A's job can be started.

Protocol 4 Identity/certificate-based authenticated key agreement based on the RSA TLS handshake

(1)	$A \rightarrow X$:	`ClientHello` = n_A`, session_id, cipher_suite`
(2)	$X \rightarrow A$:	`ServerHello` = n_X`, session_id, cipher_suite`
		`ServerCertificate` = Cert_X, LT_X
		`CertificateRequest`
		`ServerHelloDone`
(3)	$A \rightarrow X$:	`ClientCertificate` = Cert_A, LT_A
		`ClientKeyExchange` = $Enc_{\bar{X}}$(`pre_master_secret`)
		`CertificateVerify` = $Sig_{\bar{A}}$(`handshake_messages`)
		`ClientFinished`
(4)	$X \rightarrow A$:	`ServerFinished`

Protocol 4 is basically an enhanced version of Protocol 1 from Chapter 4 with the use of certificates. While Protocol 1 makes use of the Gentry-Silverberg HIBE and HIBS schemes for encryption and signing operations, respectively, we use the Boneh-Franklin full IBE and Cha-Cheon IBS schemes for Protocol 4. Note that Protocol 4 is different from the current TLS protocol used in the GSI in terms of number of certificates used in the protocol. In the GSI, the client and the server would exchange both standard and proxy certificates. In the case where the client has been delegated another user's credential, the client must also forward a chain of proxy certificates to the server. On the other hand, in Protocol 4, the client and the server only exchange standard certificates, regardless of the length of the delegation. We will discuss more about delegation in DKIG in the subsequent section.

As with the current TLS specification, Protocol 4 begins with A (the client) sending X (the server) a `ClientHello` message. Here, we assume `cipher_suite` contains a cipher specification that supports the IBE and IBS schemes, for example `TLS_IBE_IBS_WITH_DES_CBC_SHA`.

In step (2), X responds with the `ServerHello` and `ServerCertificate` messages. It is worth noting that Cert_X contains X's identifier ID_X which will be used later on for identity-based encryption and signing. The `CertificateRequest` message is also sent immediately after the transmission of Cert_X, requesting A's certificate. Step (2) ends with the `ServerHelloDone` message.

A replies to X's request for her certificate in step (3). She also chooses a pre-master key and encrypts it with X's short-term public key which can be calculated as $P_{\bar{X}} = H_1(\mathrm{ID}_X \| \mathrm{LT}_X)$. The encrypted pre-master key is transmitted to X as `ClientKeyExchange`. Details of the rest of the protocol messages are similar to Protocol 1 and will not be repeated here.

If A and X execute the protocol properly, they should share the same master secret at the end of the protocol run. The master secret can be calculated as:

$$K_{AX} = \mathrm{PRF}(\texttt{pre\_master\_secret}, \text{``}\texttt{master secret}\text{''}, n_A, n_X),$$

where PRF is a pseudo-random function specified for the TLS protocol in [50].

We remark that neither A nor X are required to generate and sign proxy certificates when they authenticate each other using Protocol 4. This should be compared with the authentication protocol used in the GSI. In our Protocol 4, short-term public keys for both the client and the server can be computed on-the-fly and used without any authenticity check. This is a major advantage of the identity-based techniques. Obviously, Cert_A and Cert_X must be checked by X and A, respectively.

Alternative. As with Protocol 1, Protocol 4 can be easily transformed into a protocol that supports Diffie-Hellman key exchange by using the relevant `cipher_suite`, for example `TLS_DH_IBS_WITH_DES_CBC_SHA`. As in the Diffie-Hellman key exchange based standard TLS protocol, both the client and the server in this alternative approach must agree to the same parameters for the Diffie-Hellman operations during the TLS handshake. We omit the details of this alternative protocol since the construction of the protocol is a straightforward exercise.

5.4.4 Delegation

For delegation, we can still obtain a one-pass protocol as we did for IKIG, even though DKIG now involves the use of certificates. As with IKIG, we assume the delegation token is a 5-tuple: $\texttt{DelegationToken}_X = (\mathrm{ID}_A, \mathrm{ID}_X, \texttt{J\_req}, \texttt{Policy}, \mathrm{LT}_{AX})$. Here LT_{AX} is a lifetime which A imposes on X. Protocol 5 then shows our one-pass identity/certificate-based delegation protocol.

Protocol 5 Identity/certificate-based credential delegation based on the IBS scheme

$A \rightarrow X: \quad \mathrm{Cert}_A, \texttt{DelegationToken}_X,\ Sig_{\bar{A}}(\texttt{DelegationToken}_X)$

As with Protocol 3 that we proposed for IKIG, A can also naturally bind X's public key information (ID_X and LT_{AX}) to the delegation token without acquiring the key from X in Protocol 5. The signed delegation token is of the form of (U, V), where $U = rP_{\bar{A}}$, $V = (r + h)S_{\bar{A}}$ and $h = H_6(\texttt{DelegationToken}_X, U)$.

X's status as the delegation target can be confirmed by a third party by: (i) checking the validity of Cert_A and the system parameters obtained from Cert_A, (ii) verifying the signed delegation token using $P_{\bar{A}}$, and (iii) challenging X for a proof of possession of the private component associated to $P_{\bar{X}}$ via the TLS handshake in Protocol 4 (which involves checking of X's certificate). We note that unlike our IKIG proposal in the previous chapter where we proposed partial aggregation between $Sig_{\bar{A}}(\texttt{DelegationToken}_X)$ and $Sig_{\bar{X}}(\texttt{handshake\_messages})$, it seems that aggregation of signatures is not viable in the DKIG setting. Although Yoon *et al.* [189]

have shown that the Cha-Cheon IBS scheme can be easily extended (with slight modification) to support what they termed as batch verification, all users of the modified Cha-Cheon IBS scheme must possess private keys obtained from the same TA. These private keys are generated by the TA using the same master secret. This contrasts with our DKIG setting, because all users have their own respective master secrets which can be used to derive (short-term) private keys.

If X wants to further delegate A's credential to another resource provider Y, X can repeat A's actions with X becoming the delegator and Y the delegation target. X must sign a delegation token for Y where $\texttt{DelegationToken}_Y = (\mathrm{ID}_X, \mathrm{ID}_Y,$ `J_req`, `Policy`, $\mathrm{LT}_{XY})$. A verifier will have no problem constructing the necessary short-term public keys $(P_{\bar{A}}, P_{\bar{X}})$ for verifying $\texttt{DelegationToken}_X$ and $\texttt{DelegationToken}_Y$, respectively, provided he has obtained the correct system parameter sets for both A and X from their respective certificates.

Alternative I. In Section 4.4.4, we discussed that as an alternative approach in the IKIG setting, A's proxy can delegate her credential to X by issuing a private key to X. The private key is one associated with the hierarchical structure of HIBC and thus this technique is quite natural in the IKIG setting. In DKIG, we envisage that a similar approach can be realised by adopting HIBC in place of IBC at the user level. Let us imagine that we now include additional parameters needed for HIBE/HIBS schemes in the grid entities' existing parameter sets associated with IBE/IBS schemes. Then so long as the delegator forwards his HIBC related system parameters in a certificate signed by the TA to the delegation target, the established delegation chain will be aligned with a hierarchical tree beginning with the delegator at level 0, the first delegation target at level 1 and so on. One limitation of this technique is the requirement for a privacy and integrity protected channel to transmit a private key.

Alternative II. There is also another delegation technique which requires more radical modification to our Protocol 5 than the previous alternative. This approach can result in a substantial saving in the protocol's network bandwidth requirement by using short signatures [28, 29] and full aggregate signatures [27] (as discussed in Section 3.4.5). However, the resulting delegation protocol is no longer identity-based. The details are as follows.

A's long-term public/private key pair are of similar form as in the DKIG setting, i.e. $(s_A P, s_A)$, where s_A is a secret random integer and P is now a fixed point on an elliptic curve agreed by all system users. Note that this was not a requirement previously: up till now, each user could have a different parameter set. In this alternative approach, the public key $s_A P$ can be published in a certificate signed by a TA, as before, using the BLS short signature scheme [28, 29]. To take full advantage of the BGLS aggregate signature scheme of [27], A must create a proxy credential of the form (aP, a), where a is a new random integer used as a short-lived private key of A. However, since the short-term public key aP is not identity-based and is now unpredictable, A must attach with it a proxy certificate signed using her long-term private key. Therefore, despite the fact that the BGLS aggregate signature scheme can compress all signatures (signed delegation tokens by different parties) into a single signature whose size is that of a single group element, the saving in communication costs due to the aggregation of signatures may be offset by the increased bandwidth requirement for transmitting proxy certificates.

5.5 Key Management in DKIG

In this section, we will discuss various key management issues arising in DKIG. In general, we envisage that management of long-term public keys (in the form of user-selected parameters) would be rather similar to the GSI, while short-term credential management may well be more lightweight than the GSI, because of DKIG's avoidance of proxy certificates.

5.5.1 Parameter Generation and TA Initialization

Our DKIG proposal makes use of two different sets of system parameters: (i) root system parameters based on MNT curves for the TA, and (ii) normal system parameters based on supersingular curves for the users.

During the initial system setup phase, the TA selects the root system parameters which are needed for signing certificates using the BLS short signature scheme of [29]. This short signature scheme uses MNT curves in order to obtain short signatures. It is worth noting that we do not use supersingular curves for the root system parameters because otherwise the BLS short signature scheme may well not provide the expected levels of security (as we explained in Section 3.4.5). Hence, we propose that the TA in our DKIG setting chooses an elliptic curve defined over $\mathbb{F}_p$, where p has 168 bits and the embedding degree is 6, as in [29]. Further details of certification will be discussed shortly in the following section.

Each user makes use of system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, sP_0, H_1, H_2, H_3, H_4, H_6 \rangle$[3] for the Boneh-Franklin full IBE and the Cha-Cheon IBS schemes. In our DKIG proposal, all these parameters except sP_0 are selected by the TA and bootstrapped in the grid system so that only user-specific values sP_0 are transmitted across the network. This means that in practice, each user is only required to select a unique sP_0 as the system parameter that will be certified by the TA. This represents a trade-off between savings in communication costs and lack of flexibility in supporting various groups derived from different elliptic curves. In contrast, we could allow users to select all the components that form their system parameters, giving more flexibility to the system and users, but this would require higher bandwidth because then a full set of system parameters would need to appear in each certificate issued by the TA.

In IKIG, we proposed the use of keys with 272 bits for both long-term and short-term credentials, which offer approximately similar security level to 1024-bit RSA keys. This is an advantage over the GSI because short-term keys in IKIG have roughly double the security level of 512 bit short-term RSA keys. At the user level of our DKIG setting, we propose to use the same elliptic curve (of embedding degree 4) and field ($\mathbb{F}_{2^{271}}$) to obtain a similar group with size approximately 2^{252}. We assume that the size of output of H_1 is equivalent to 272 bits, the size of an element of $\mathbb{G}_1$. The hash function H_1 is required in all three Boneh-Franklin full IBE, Cha-Cheon IBS and BLS short signature schemes. In the Boneh-Franklin full IBE scheme, we assume that H_2 and H_4 each has an output size 256 bits, while the output size for H_3 is approximately 252 bits. Also, H_6, used in the Cha-Cheon IBS scheme, has an

[3] For the sake of consistency, as in the previous chapter, we use P_0, a generator, as part of a parameter set, P_A as a long-term public key of user A and $P_{\bar{A}}$ as A's short-term public key.

output size similar to that of H_3.

5.5.2 User Registration

When a new grid user Alice (A) goes to a nearby RA with her generated system parameter, the RA performs the following steps:

1. The RA verifies A's identity by checking her passport (or an acceptable ID-card). Once the check succeeds, the RA compares A's identity with its global identity list and subsequently assigns her a distinguished identifier $\mathrm{ID}_A \in \{0,1\}^*$. For example,

$$\mathrm{ID}_A = \text{“}\texttt{/C=UK/O=eScience/OU=RHUL/CN=Alice}\text{”},$$

2. The RA submits A's application to the TA through its website using the HTTPS protocol. The application includes A's system parameter $s_A P_0$ and a signed request (as a proof of possession) by A using her master secret s_A. This can be done using the BLS short signature scheme defined in Section 3.4.5. If approved, the TA generates a certificate Cert_A (which is signed using the BLS short signature scheme of [29]) and informs A through e-mail that the certificate is ready. A can then download her certificate through an authenticated and integrity protected channel from the TA's server. We envisage that the certificate has the following structure:

Issuer:	/C=UK/O=eScience/OU=Authority/CN=TA
Subject (ID):	/C=UK/O=eScience/OU=RHUL/CN=Alice
Validity:	Not Before Oct 6 15:20:05 2005 GMT Not After Oct 6 15:20:05 2006 GMT
Param:	$s_A P_0$
Signature:	94:33:6c:f3:58:e2:18:87:7f:9a:59:d1:23:0c:66: 31:fb:92:3a:63:a2:67:0d:ec:3d:f0:c7:b4:09:15

   A can verify the signature on her certificate by using the TA's public key which is assumed to be bootstrapped in the grid system.

Before A submits a job, she can create a short-term public/private key pair straightforwardly using the Extract algorithm from the Boneh-Franklin IBE scheme. As we have shown in Table 5.1, A's short-term public key is $P_{\bar{A}} = H_1(\mathrm{ID}_A \| \mathrm{LT}_A)$ and the corresponding private component is $s_A P_{\bar{A}}$. This private component is stored temporarily in a local file system in unencrypted form for the purpose of single sign-on.

5.5.3 Key Update

Although DKIG is based on a combination of identity-based and certificate-based approaches, we expect that the usual annual key update practice in the GSI would be adopted in our proposal. This is so because certificates are used to carry users' system parameters. Barring any exposure of the user's master secret s_A, the TA can renew the user's current certificate by updating the validity period and serial number of the certificate. Alternatively, the user can pick a new master secret and request a new certificate that would contain fresh system parameters, before the current certificate expires.

For short-term identity-based public/private key pairs, as with IKIG, we assume that the lifetime of the keys is 12 hours. Each time the user creates a new proxy credential based on her identifier ID_A, a unique LT_A and her master secret s_A, a new short-term public/private key pair that is fresh and different from the past public/private keys is obtained.

5.5.4 Key Revocation

Since the user's proxy credentials will expire automatically within a short time, we only concern ourselves with the user's certificate. Since the user's long-term credential is managed in a very similar way as in the GSI, we envisage that a CRL-based method can also be used in DKIG. Clearly, the efficiency of this technique relies on the timeliness of the revocation list update, as well as the willingness of the users to update their local copies of CRLs.

5.6 Security Analysis

In many ways, the heuristic security analyses of Protocols 4 and 5 are rather similar to the analyses of Protocols 1 and 3 in Section 4.4. So, our focus in this section will be to pinpoint the differences between the authenticated key agreement and delegation protocols in IKIG and DKIG, from a security perspective.

5.6.1 Mutual Authentication and Key Agreement

The security of our Protocol 4 is different from the long-discussed security of the TLS protocol even though Protocol 4 also makes use of certificates. This is because, as we mentioned earlier, the actual session key establishment method is through identity-based techniques, as with Protocol 1. However, in comparison with Protocol 1, Protocol 4 requires both the client and the server to verify the authenticity of each other's system parameters contained in their respective certificates. Only then is it safe to use identity-based keys for message encryption and signing. For instance, when A receives Cert_X and LT_X from X in the `ServerCertificate` message of Protocol 4, she must verify the signature on Cert_X. Only then, should she use X's identifier and LT_X to construct an identity-based public key. In turn, this key is used to encrypt a pre-master secret for X.

So long as both the client and the server have an authentic copy of the TA's public key, the anticipated mutual authentication and key agreement between the two parties can be achieved if the protocol is executed properly. X is authenticated to A when A receives a correct verification value in the `ServerFinished` message from X, which shows that X has retrieved the pre-master secret chosen by A. On the other hand, A is authenticated to X if X can successfully verify A's signature on the `CertificateVerify` message using public key $P_{\bar{A}}$.

It is also worth noting that, even with the use of certificates in DKIG, it is still possible for the TA to actively impersonate a user by issuing a certificate using its own chosen master secret and system parameters. However, the TA would run the risk of leaving cryptographic evidence, in the form of a new certificate, which might identify its attack. Hence, the likelihood of an undetected active impersonation attack seems to be far lower than the corresponding attack executed by a TA empowered with a key escrow facility, as in, for example, security infrastructures based on IBC.

5.6.2 Delegation

Protocol 5 in our DKIG is even more similar to the delegation protocol that we proposed for IKIG than are the TLS-based key agreement protocols in the two infrastructures. Since certificates are used to confirm the validity of an entity's system parameter, a third party must perform additional signature verification on the certificates to check the status of delegation targets. As in IKIG, our one-pass delegation protocol message does not require an authenticated and integrity protected channel. The security of the Protocol 5 seems to rely only on the security of the BLS short signature and the Cha-Cheon IBS schemes.

5.7 Performance Analysis

In this section, we will discuss the computational and communication overheads of DKIG and compare them with those incurred in the GSI and in our IKIG proposal.

Computational Costs. Here, we examine the dominant cryptographic operations involved in delivering the security services provided by DKIG.

- Generation of proxy credentials: In our proposed DKIG setting, generation of a short-term public/private key pair is very efficient. It involves execution of a cryptographic hash function when computing the public key, while computation of the private key requires only one elliptic curve point multiplication.
- Authenticated key agreement: Protocol 4 in DKIG requires more cryptographic operations than Protocol 1 in IKIG since the former makes use of certificates and the latter is certificate-free. In Protocol 4 the user performs an encryption on a pre-master secret and produce a signature on handshake messages using the Boneh-Franklin full IBE and the Cha-Cheon IBS schemes,

respectively. In addition, the user must verify the TA's signature on Cert_X. At the server side, the resource has to decrypt the encrypted pre-master secret, and verify the signed handshake messages and the user's certificate.

- Delegation: Our delegation protocol for DKIG has computational complexity similar to Protocol 3. The delegator is required to perform signing on a delegation token. However, the verifier has to perform two signature verifications (rather than one in Protocol 3) assuming the length of the delegation chain is one. These signatures include the signed delegation token and the delegator's certificate.

Table 5.2 gives more details of the computational costs in both GSI and DKIG. As with Table 4.3, we obtained the computation times of the IBE and IBS schemes by using programs written based on the MIRACL library with the curves and parame-

Table 5.2: Performance trade-offs in computation times (in milliseconds) between the GSI and the DKIG settings on a Pentium IV 2.4 GHz machine.

	GSI		**DKIG**	
Operation	**RSA**	**Time**	**IBE/IBS**	**Time**
Key generation				
(a.) Long-term	1 GEN	149.90	1 EXT	1.69
(b.) Short-term	1 GEN	34.85	1 EXT	1.69
Authenticated key agreement				
(a.) Requestor	1 1024-bit VER 1 512-bit ENC 1 512-bit SIG 1 512-bit VER	2.67	1 ENC, 1 SIG 1 VER	18.17
(b.) Resource	1 1024-bit VER 1 512-bit DEC 2 512-bit VERs	2.67	1 DEC, 2 VER	29.54
Delegation				
(a.) Delegator	1 512-bit SIG 1 512-bit VER	1.86	1 SIG	3.35
(b.) Delegation target	1 GEN 1 512-bit SIG	35.63	1 EXT	1.69
(c.) Verifier	3 512-bit VERs	0.84	2 VER	16.84

GEN = RSA parameter generation
EXT = IBE/IBS private key extraction
ENC = Encryption
DEC = Decryption
SIG = Signing
VER = Verification

ters specified in Section 5.5.1. As before, we adopted known optimisation techniques wherever possible, such as using a small RSA public exponent for encryptions and the CRT method for RSA decryptions. We also allow pre-computation of pairing values. For simplicity, the length of the delegation chain is limited to one.

From Table 5.2, we can see that key generation at the user side of the GSI is significantly slower than DKIG. However, Protocol 4 is significantly slower than the TLS protocol used in the GSI, mainly because of the pairing computations. For credential delegation, the GSI is more computationally expensive than DKIG because of the RSA parameter generation. Despite that, DKIG is more costly than the GSI for delegation verification.

Both settings are expensive in certain operations, for example RSA key generation and pairing computation, but lightweight in others. Overall, DKIG appears to be slightly more computationally expensive than the GSI because of the high number of pairing computations. Despite this, we believe that pairing computation, being a relatively new cryptographic operation, is likely to get faster as further optimisations are discovered.

Communication Costs. Table 5.3 compares the communication costs of the standard GSI approach and our DKIG proposal. As with Table 5.2, we only consider the dominant communication costs between the job requestor and the resource, i.e. signed or encrypted messages and certificates, which have the biggest contributions to the network bandwidth.

In Section 5.5.2, we showed that a certificate in the DKIG environment contains a 272-bit parameter $s_A P_0$. If the BLS short signature scheme is used by the TA to sign the certificate using the parameters of [29], the size of the signature will be about 170 bits. Hence, by considering only the fields with dominant sizes, we estimate that the size of a customised certificate in binary form is roughly 272 + 170 = 442 bits, almost half of the size of a proxy certificate (we explained in Section 4.7 that a standard proxy certificate in the GSI has a binary size of roughly 1024 bits). Therefore, we assume that the size of a certificate in PEM format for our DKIG setting is 3.2 kilobits.

Table 5.3: Performance trade-offs in communication costs (in kilobits) between the GSI and DKIG.

Operation	**GSI**	**DKIG**
Authenticated key agreement	37.8	7.7
Delegation	7.4	0.5

On the other hand, the size of a ciphertext produced by the Boneh-Franklin full IBE scheme of [25] in our DKIG setting is: $|\text{element of } \mathbb{G}_1| + 2 \times 256$-bit hash value $= 784$ bits. Also, the size of a signature on a message using the Cha-Cheon IBS scheme of [39] is: $2 \times |\text{element of } \mathbb{G}_1| = 544$ bits.

Since we have discussed the communication costs for the GSI in the previous chapter, we now only focus on the communication costs for DKIG. In Table 5.3, the figure of 7.7 kilobits refers to the two certificates, one encrypted pre-master secret and one signed message in Protocol 4: $2(3.2) + 0.784 + 0.544 = 7.7$. This shows that by avoiding the use of proxy certificates in our DKIG approach, a total saving of approximately 80% can be made as compared to the authenticated key agreement protocol in the GSI.

The communication overhead for our delegation protocol in DKIG is estimated to be 0.5 kilobits, based on the size of a signed delegation token. Note that we do not include the communication cost for transmitting (long-term) certificates in both the GSI and DKIG in Table 5.3, because we assume that the certificates for the two communicating parties have already been exchanged during mutual authentication through the TLS protocol or Protocol 4.

Figure 5.2 shows that the network bandwidth needed for key agreement and delegation in DKIG is approximately 70% to 80% less than is required in the GSI. (We assume that two grid entities authenticate each other and establish a session key before delegation takes place.) However, the communication costs in DKIG are still higher than in IKIG due to the use of certificates for long-term credentials. This may be an acceptable price to pay to remove the key escrow facility from a grid application.

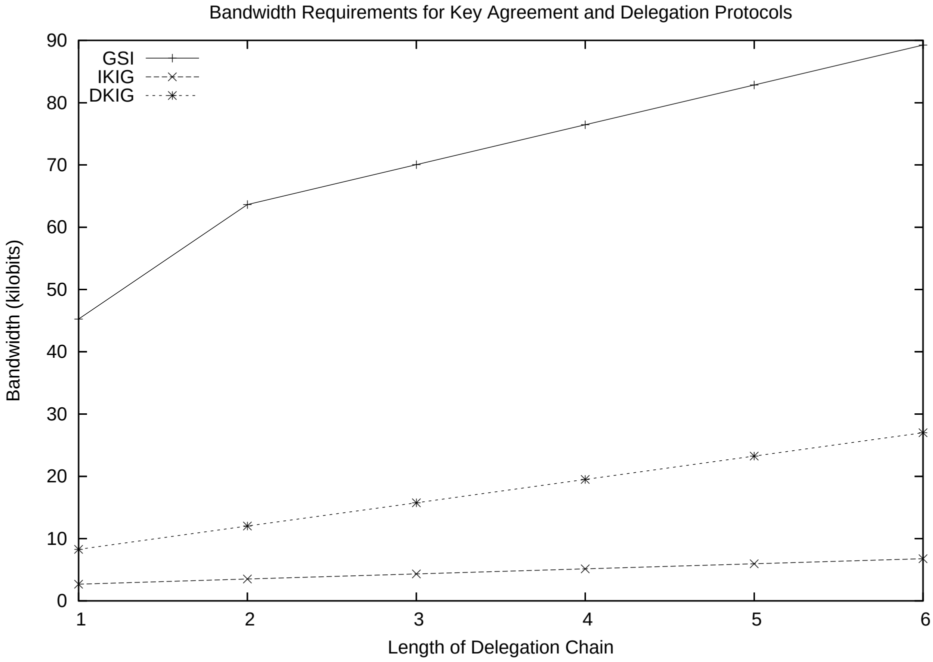

Figure 5.2: A performance graph which measures the network bandwidth required for authenticated key agreement and delegation in the GSI, IKIG and DKIG.

5.8 Discussion

In our description of DKIG in the previous sections, we highlighted that the main difference between DKIG and the PKI-based GSI is in terms of proxy credential management. The fact that users' fixed parameter sets can be published through certificates implies easier integration of DKIG with existing technologies. In this section, we discuss some practical issues related to DKIG.

Impact on Web Services Security. The reduced SOAP header sizes in web services security through identity-based techniques that we discussed in the previous chapter still holds in DKIG, though with less significant savings than with IKIG. Since the size of a DKIG-supported certificate is roughly half the size of a proxy certificate in the GSI, SOAP headers used for Protocol 4 would still be smaller than those required for the TLS protocol (via WS-SecureConversation) which makes use of both standard X.509 and proxy certificates.

Implementation Issues. In order to implement DKIG, the first essential step is to extend the current X.509 certificate profile specified in [101] so that system parameters for identity-based cryptographic schemes can be presented through a standard certificate. The extensions should allow the use of certificate fields described in Section 5.5.2 so that DKIG can inter-operate with PKI. We envisage that current data fields such as issuer, subject and validity can be re-used; while new fields such as param, signature algorithm and (short) signature need to be defined.

From the perspective of GSS-API, the credential acquisition function must be able to support extraction of data fields from DKIG-supported certificates. This is important so that an entity's system parameter can be verified and his identity-based public key can be constructed with ease. In the DKIG setting, we must also define and include a `TLS_IBE_WITH_DES_CBC_SHA` cipher suite in the current TLS specification to support our Protocol 4. In addition, the OpenSSL library must support message encryption/decryption and signature generation/verification through IBE and IBS schemes.

Inter-TA Operation. If the grid user and resource domains are expanded and span various countries, it is natural to have at least one TA in each nation. In this case, we envisage that in our DKIG setting, the European Grid PMA can select and decide on the common system parameters (except for the user-specific component) which these users and resources would use for the IBE and IBS schemes. At the TA level, all the TAs (assuming there is no single root TA) could employ different parameter sets which would be available to all the system users.

Limitations. Our DKIG proposal has two drawbacks in comparison with IKIG. Firstly, it is obvious that the introduction of certificates inevitably increases the communication and computational costs in the DKIG setting. The second limitation is the partial loss of the benefits that identity-based techniques could bring to grid applications. With the use of certificates to verify entities' parameters sets, long-term key management between the entities and their TAs becomes subject to the conventional limitations of certificate-based PKI.

5.9 Summary

We presented a different security infrastructure called a dynamic key infrastructure for grid (DKIG) in this chapter. This infrastructure has solved the key escrow issue encountered by the fully identity-based approach proposed in the previous chapter. We described an interesting way of managing users' proxy credentials whereby the users act as their own PKGs and publish their individual parameter set through a certificate. Through this approach, single sign-on and rights delegation can be achieved without the need for creating proxy certificates. We proposed a TLS-supported authenticated key agreement protocol using a combination of identity- and certificate-based techniques. Even though the authenticated key agreement protocol seems to be slightly more expensive than the TLS protocol in terms of computational costs, its network bandwidth requirement is still considerably less than that of TLS within the GSI. In summary, our dynamic key infrastructure for grid offers the benefits of identity-based techniques at the user level without the key escrow issue, but at the expense of somewhat increased bandwidth requirements and computational costs as compared to the fully identity-based approach.

Chapter 6

Identity-Based Secret Public Keys

Contents

The concept of secret public keys was proposed more than a decade ago as a means of securing password-based authentication protocols against off-line password guessing attacks, but was later found vulnerable to various attacks. In this chapter, we revisit the concept and introduce the notion of identity-based secret public keys. Our new identity-based approach allows secret public keys to be constructed in a very natural way using arbitrary random strings, eliminating the structure found in, for example, RSA or Diffie-Hellman keys. We examine identity-based secret public key protocols and give informal security analyses which show that they may well be secure against off-line password guessing and other attacks. More importantly, we present an identity-based secret public key version of the standard TLS protocol. Our new protocol allows passwords to be tied directly to the establishment of secure TLS chan-

nels. This, in turn, appears to provide an interesting way for grid users to access a MyProxy server merely based on simple user-chosen passwords.

6.1 Overview

The use of *secret public keys* in password-based authentication protocols was first proposed by Gong *et al.* [86] in 1993. As implied by its name, a secret public key is a standard public key which can be generated by a user or a server, and is known only to them but is kept secret from third parties. A secret public key within a password-based protocol, when encrypted with a user's password, should serve as an unverifiable text[1]. This may significantly increase the difficulty of password guessing even if it is a poorly chosen password as an attacker has no way to verify if he has made the correct guess. The secret public key can then be used by the user for encrypting protocol messages. However, it may not be easy to achieve unverifiability of text by simply performing naive symmetric encryption on public keys of standard types such as RSA or Diffie-Hellman. This was overlooked in [86] and other variants of secret public key protocols in [85, 174], but later found to be the main culprit in various attacks on the protocols. These include undetectable on-line password guessing attacks of Ding and Horster [52] and number theoretic attacks due to Patel [141]. It is worth noting that the attacks discovered in [52] may not work against a secret public key protocol which uses a secure public key encryption scheme such as RSA-OAEP [18]. Nevertheless, Patel's attacks seem to be one of the crucial factors that caused diversion of interest away from using secret public keys in password-based protocols. The concept of secret public keys, therefore, was thought to be obsolescent. For example, in more recent work on password-based protocols that requires servers' public keys[2] (e.g. [30, 96]), it is assumed that the public keys are fixed and known to all users.

Independent of the previous work on secret public key protocols, Steiner *et al.* [168] proposed a method for integrating password-based key exchange protocols with the

[1] Verifiable text/plaintext is a term popularised by Lomas *et al.* in [120]. It refers to a message that contains information that is recognisable when decrypted, whether or not it was predictable in advance.

[2] We classify password-based authentication protocols into two categories: (i) those which require the usage of the server's (or the user's) public key, and sometimes together with the user's password, as a key-encrypting key; and (ii) those which only require the user's password for key transport.

TLS protocol. Simple password authentication through a secure TLS channel is widely used in email and e-commerce applications between a user and a web server. It can be achieved by first establishing a secrecy and integrity protected channel between the user's client (e.g. a web browser) and a remote server owned by a service provider, normally through the server-authenticated TLS handshake protocol. The user then submits his user name and password through the secure channel to the server. Subsequently, the server authenticates the user by verifying the submitted user name and password. Note that users within a grid environment make use of exactly the same technique when accessing the MyProxy server to request their proxy credentials. In such a set-up where the user enters his password only after the secure channel is established, the authentication of the user (through his password) is not directly tied to the secure channel. This provided the motivation for the password-based TLS protocol proposed in [168]. However, the proposal of Steiner *et al.* requires significant alterations to the structure of the TLS handshake protocol, an undesirable property which limits the acceptability of their proposal.

The aims of this chapter are twofold: (i) revisit the notion of secret public keys and uncover some unexplored potential benefits of using identity-based secret public keys, through IBC, in password-based protocols; and (ii) show how identity-based secret public keys can support the use of passwords in the TLS protocol in a more natural and less disruptive way than was proposed in [168].

In our quest to revive the notion, we introduce some new properties for secret public keys. In the IBC setting, we show that an identity-based secret public key can offer more flexibility in terms of key distribution. For example, an identity-based secret public key can be computed by a user on-the-fly without needing his authentication server to transport the key to him. More importantly, a random string can be used as the identifier for constructing a secret public key. This technique can offer a clean and natural way of eliminating any predictable structure in the secret public key. Through this, the number theoretic attacks that plague existing secret public key protocols can easily be prevented.

Since both public and private keys in the IBC setting are kept secret, we also propose the notion of *secret signatures* which seem to provide data confidentiality in addition to their original cryptographic use, i.e. authentication and non-repudiation. This appears to provide additional properties in conventional secret public key protocols and in password-based authentication protocols in general.

The TLS protocol is becoming increasingly ubiquitous for web-based applications that require secure authentication and key establishment. Our identity-based approach to the concept of secret public keys may well be of significant practical value when it is integrated with the standard TLS handshake protocol. We design a TLS-compatible identity-based secret public key protocol which requires no structural or message flow modification but only minimal changes or extensions to the message contents.

Parts of the research findings presented in this chapter appear in [116].

6.2 Related Work

Extensive work on password-based key exchange protocols (which rely on user passwords only) has already been carried out. See for example [1, 2, 3, 17, 33, 35, 105], which all originate from [21, 22]. In order to circumvent off-line password guessing attacks, Bellare *et al.* [17, 19] proposed the use of a mask generation function $\mathcal{E}(\cdot)$ as an instantiation of the encryption primitive for encrypting a Diffie-Hellman component, rather than using a standard block (or stream) cipher. For instance, a user with his password PW can encrypt a Diffie-Hellman component g^x by calculating $g^x \cdot H(PW)$, where H is a hash function mapping onto the Diffie-Hellman group and which is modelled as a random oracle in security proofs. Thus the result of the encryption is a group element. This special encryption primitive, which needs to be carefully implemented, is crucial in preventing any leakage of information about the password when an attacker mounts a guessing attack. To decrypt and recover g^x, one can simply divide the ciphertext by $H(PW)$. All recent work, such as [1, 2, 3], utilises this encryption primitive for their password-based key exchange protocols. Our proposal using identity-based techniques can be seen as a novel alternative to these current protocols. In addition, the identity-based techniques can be integrated naturally with the TLS handshake protocol, which seems to be difficult to achieve using current Diffie-Hellman encrypted key exchange techniques without more radical modification of the TLS handshake.

The use of algorithms from a public key encryption scheme in a secret/symmetric key setting is not new. In 1978, Hellman and Pohlig [98] introduced the Pohlig-Hellman symmetric key cipher based on exponentiation. Two different keys are involved in the symmetric key cipher, namely, a secret encrypting key e for the sender and a secret decrypting key d for the receiver, where $e \neq d$. Obviously, the communicating parties must agree in advance to share these two symmetric keys. In more recent work, Brincat [36] investigated how shorter RSA public/private key pairs can be used securely in the secret key world. This is slightly different from [98], as each user has his own secret public/private key pair in [36]. Another related concept is that of public key privacy from Bellare *et al.* [16]. The notion of indistinguishability of keys in public key privacy is an extension of the ciphertext privacy concept: given a set of public keys and a ciphertext generated by using one of the keys, the adversary cannot tell which public key was used to generate the ciphertext. In this chapter, we will make use of identity-based (secret) public keys in the secret key setting. These public keys are known only to the senders and receivers, and thus indistinguishability of encryptions and keys somewhat similar to [16] can be achieved. Moreover, in such a setting, a signature can be made verifiable to only a specific recipient, hence the moniker *secret signature*. In many ways, the concepts of secret public key encryption and signatures seem to be closely related to the notion of signcryption with key privacy from Libert and Quisquater [113]. The proposal of [113] combined Zheng's work on signcryption [191] and the key privacy concept of [16]. Our concept of secret signatures is also related to the strongest security notion for undeniable and confirmer signatures called invisibility in [40].

6.3 Secret Public Key Protocols and Attacks

In this section, we revisit the first secret public key protocol proposed in the literature [86]. We will explain what the problems are with the protocol. This will motivate our introduction of identity-based techniques to this area.

Notation. We use $\hat{PK}$ and $\hat{SK}$ to represent a secret public key (SPK henceforth) and its matching private key, respectively. These are no different from conventional asymmetric keys except that they are *both* kept secret. PW denotes a password-derived symmetric key which is shared between a user and an authentication server. A nonce and a random number are represented by n and r, respectively. We use the notation $Enc_{\hat{PK}}(\cdot)$ to indicate asymmetric encryption using a secret public key $\hat{PK}$ and $\{\cdot\}_K$ for symmetric encryption under a symmetric key K. In the three-party scenarios that we will discuss in this section, we use A and B to denote two communicating parties, while S denotes a trusted authentication server whose role is to distribute a copy of a randomly generated session key to both A and B. Other notations will be introduced as they are needed.

The GLNS SPK Protocol. Gong *et al.* [86] envisaged that using secret public keys in a password-based protocol may be useful in a situation where the public keys are needed for certain protocol messages but the protocol participants do not know in advance the public key of their authentication server. In addition, they implicitly assumed that a secret public key could be viewed as a nonce which, when encrypted with a password, offers unverifiability of text. Assuming A and B share their respective passwords with the authentication server S, the server can distribute fresh copies of public keys to A and B encrypted using their respective passwords as symmetric keys at the beginning of each protocol run. Each public key is only known between the server and the relevant participant. This seems to make traditional chosen plaintext attacks more difficult, as the encryption keys are not known to the attacker. The details of the SPK protocol of [86] are depicted in Protocol 6.

Protocol 6 *The GLNS SPK Protocol*

$$\begin{array}{lll}
(1). & A \rightarrow S: & A,\ B \\
(2). & S \rightarrow A: & A,\ B,\ n_S,\ \{\hat{PK}_{SA}\}_{PW_A},\ \{\hat{PK}_{SB}\}_{PW_B} \\
(3). & A \rightarrow B: & Enc_{\hat{PK}_{SA}}(A, B, n_{A1}, n_{A2}, c_A, \{n_S\}_{PW_A}),\ n_S, r_A,\ \{\hat{PK}_{SB}\}_{PW_B} \\
(4). & B \rightarrow S: & Enc_{\hat{PK}_{SA}}(A, B, n_{A1}, n_{A2}, c_A, \{n_S\}_{PW_A}), \\
 & & Enc_{\hat{PK}_{SB}}(B, A, n_{B1}, n_{B2}, c_B, \{n_S\}_{PW_B}) \\
(5). & S \rightarrow B: & \{n_{A1},\ K_{AB} \oplus n_{A2}\}_{PW_A},\ \{n_{B1},\ K_{AB} \oplus n_{B2}\}_{PW_B} \\
(6). & B \rightarrow A: & \{n_{A1},\ K_{AB} \oplus n_{A2}\}_{PW_A},\ \{H(r_A),\ r_B\}_{K_{AB}} \\
(7). & A \rightarrow B: & \{H(r_B)\}_{K_{AB}}
\end{array}$$

As shown in Protocol 6, S generates two new sets of secret public/private key pairs $(\hat{PK}_{SA}, \hat{SK}_{SA})$, $(\hat{PK}_{SB}, \hat{SK}_{SB})$ and distributes the public components to A in encrypted form whenever A initiates the protocol run. Here, c_A and c_B are sufficiently large random numbers known as confounders. They serve no purpose other than to confound guessing attacks based on some verifiable texts. Also, H is assumed to be a well-designed hash function.

In [86], the authors assumed that so long as the secret public keys $\hat{PK}_{SA}$ and $\hat{PK}_{SB}$ are randomly generated, it will be difficult for the attacker to verify if his password guesses on $\{\hat{PK}_{SA}\}_{PW_A}$ or $\{\hat{PK}_{SB}\}_{PW_B}$ are correct. In reality, however, this is not completely true. When using conventional public keys such as RSA exponents or Diffie-Hellman components, the keys contain certain number theoretic structure even though they are randomly generated. This, in turn, may allow the attacker to verify his guessed passwords efficiently by predicting and checking the outcome of the decryption. For example, if $\hat{PK}_{SA}$ is an RSA public key of the form $N = pq$, then the attacker could expect the decryption of $\{\hat{PK}_{SA}\}_{PW_A}$ under a guess PW'_A for A's password to be an odd integer. This allows the elimination of half of all passwords in a simple off-line guessing attack. It is this observation that led to Patel's study on various number theoretic attacks on secret public key protocols [141]. It is also worth noting that $\{\cdot\}_K$ must not represent the action of an authenticated encryption algorithm as this would also leak information that could be used to verify the correctness or otherwise of password guesses.

Patel's Attacks. As we have just seen, it can be dangerous to transmit an RSA modulus in encrypted form in an SPK protocol. Even if the ciphertext contains only an RSA exponent, e.g. $\{e\}_{PW}$, there are various number theoretic attacks that would reveal the password PW. For example, the attacker could expect the decryption of $\{e\}_{PW}$ under a guess PW'_A to be an odd integer; an even result would eliminate PW'_A as a possible password. Thus, some countermeasures against these number theoretic attacks such as padding or randomisation of the RSA exponent are inevitably required.

Patel [141] showed that even when moduli N are sent in clear, and e are randomised and padded, there is still a lethal off-line guessing attack. Protocol 7 illustrates Patel's RSA version of the SPK protocol. We only show the first 3 out of 7 protocol messages as this is sufficient to describe Patel's attack.

Protocol 7 *The RSA SPK Protocol*

$$\begin{array}{lll}
(1). & A \rightarrow S: & A,\ B \\
(2). & S \rightarrow A: & A,\ B,\ n_S,\ \{e_{SA}\}_{PW_A},\ N_A,\ \{e_{SB}\}_{PW_B},\ N_B \\
(3). & A \rightarrow B: & Enc_{e_{SA}}(A,\ B,\ n_{A1},\ n_{A2},\ c_A,\ \{n_S\}_{PW_A}),\ n_S,\ r_A,\ \{e_{SB}\}_{PW_B} \\
 & \vdots & \vdots
\end{array}$$

An attacker can impersonate S and block A's communication with the real authentication server to mount the following attack.

1. When the attacker E detects A is sending message (1) to S, he blocks S's response from reaching A. E intercepts message (2) and replaces N_A with his own N'_A whose prime factors he knows. Also, since E does not know PW_A, he simply replaces $\{e_{SA}\}_{PW_A}$ with a random string R_A.

2. A unwittingly decrypts R_A with her password-derived key PW_A and obtains e'_{SA} which A believes was generated by S. Subsequently in message (3), A forwards $Enc_{e'_{SA}}(A,\ B,\ \dots)$ to B.

3. E intercepts message (3) and can now perform off-line password guessing on R_A. For each possible PW'_A, E decrypts R_A and retrieves a possible value for e'_{SA}. Since E knows the prime factors of N'_A, he has no problem computing the decryption exponent d'_{SA} for each value of e'_{SA}. By decrypting $Enc_{e'_{SA}}(A,\ B,\ \dots)$ with d'_{SA} and checking if the plaintext is of the form $(A,\ B,\ \dots)$, E can test if PW'_A is the correct password.

It was pointed out in [141] that the above attack on the RSA-based SPK protocol is unavoidable unless all protocol participants use an agreed-upon RSA modulus, or unless the protocol is radically modified.

Even supposing a discrete logarithm based SPK protocol was used, and the ciphertext (which contains a secret public key) transmitted to A was then of the form $\{g^x\}_{PW}$, where g is a generator of a subgroup of $\mathbb{Z}_p^*$ of prime order q and x is a random integer, the password can still be discovered. If a naive encryption of elements in the subgroup is performed with a standard block (or stream) cipher, then there is an off-line password guessing attack. The attacker simply decrypts $\{g^x\}_{PW}$

with a guessed password and observes if the resulting plaintext is an element of the subgroup. If it was an incorrect guess, the likelihood that g^x is not an element of the subgroup is at least $(p - q)/p > 1/2$. This attack can only be prevented by ensuring that decryption of $\{g^x\}_{PW}$ with a guessed password PW' always results in an element of the subgroup. Furthermore, it is also essential that public parameters such as g, p and q have been agreed *a priori* among the users. More examples and discussion on this subject can be found in [141, 168]. Notice that this kind of attack is prevented using mask generation functions of the type discussed in Section 6.2.

From the above descriptions of various number theoretic attacks, it should be evident that designing a SPK protocol can be difficult and not without some extra costs in ensuring the predictable number theoretic structure within public keys is eliminated. These observations are crucial for motivating our identity-based approach. We will show that the aforementioned problems can be prevented easily and naturally, using identity-based techniques.

6.4 New Properties from Identity-Based Secret Public Keys

We now present properties from identity-based SPKs by using the Boneh-Franklin IBE and Zhang-Susilo-Mu IBS schemes of Sections 3.4.1 and 3.4.4. Pre-distribution or fixing of some public/system parameters is common in password-based protocols. In the following sections, we assume that the system parameters for the Boneh-Franklin IBE and the Zhang-Susilo-Mu IBS schemes can be distributed by the server to all its users during the user registration phase using an out-of-band mechanism. This is important as failure to use an authentic set of system parameters would allow the attacker to inject his own chosen parameters. Also, during the registration phase between a user and the server, the user will pick a password pwd and send an image PW of the password to the server. Typically, one might set $PW = H_0(pwd) \cdot P$, where $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $\mathbb{G}_1$ is a group of prime order q used elsewhere in the protocol, and P generates $\mathbb{G}_1$. Note then that the server only knows PW and not pwd. The actual password pwd still remains private to the user only. In some cases where pwd and PW are used together, stronger authentication can be provided in the sense that the user's authenticity can still be guaranteed even if the string PW stored in the server is revealed. This technique of using an image of the actual user-selected password is common to many password-based protocols, for example [1, 17, 19, 22].

Here, we present and discuss some interesting properties of identity-based SPKs (ID-SPKs henceforth) which are new as compared to conventional SPKs based on RSA or Diffie-Hellman primitives. These properties can be obtained from using the Boneh-Franklin and Zhang-Susilo-Mu schemes, and they form the basis and motivation for the ID-SPK protocols that we will discuss in Section 6.5.

6.4.1 ID-SPK as Secret Identifier

In the conventional IBC setting, an identifier refers to some public information which represents a user and is known to all parties. Here, however, we work with secret identifiers, that is, identifiers only known to the user A (or B) and the server S. These can be obtained by binding a secret value such as a password to an identifier. Such an ID-SPK of the form $\hat{PK} = H_1(\texttt{user} \,\|\, \texttt{password} \,\|\, \texttt{policy})$ can be generated by both the user and the server on-the-fly. Here `policy` denotes constraints that can be included in the ID-SPK such as a date, nonces, or roles. In other words, the server does not need to distribute a fresh secret public key to its users, in contrast to [86, 174]. Here we assume the users have access to the server's fixed system parameters. For example, referring back to Protocol 6, when A initiates the protocol she could, in principle, skip messages (1) & (2) and transmit message (3) to B as follows:

$$(3).\ A \rightarrow B: \quad Enc_{\hat{PK}_{AS}}(A, B, \dots)$$

where $\hat{PK}_{AS} = H_1(A\|S\|PW_A\|\text{"10102005"})$ denotes a public key in the IBE scheme of [25]. Here "10102005" represents a date. A date with more granularity (e.g. concatenated with time) or a nonce may well be needed to ensure freshness of $\hat{PK}_{AS}$. We remark that the Boneh-Franklin IBE scheme is IND-ID-CPA secure and thus the attacker cannot use a guessed password PW'_A to verify his guess by generating $Enc_{\hat{PK}_{AS}}(A, B, \dots)$ and comparing it with the actual ciphertext produced by A, even if he knows all the plaintext components. We also assume that given the ciphertext that A produced, the attacker should learn nothing about the encryption key, i.e. $\hat{PK}_{AS}$.

On the server side, the server can extract the matching private key for $\hat{PK}_{AS}$ using its master secret. Unless the attacker can break the IBE scheme or recover the master secret, the above ciphertext is resistant to password guessing attacks. This identity-based technique offers a form of non-interactive distribution of secret public keys from the server to its users.

In the above example, A uses an ID-SPK encryption scheme which is adapted from the full version of the Boneh-Franklin IBE scheme [25] with the encryption key only known to the user and the server. Formal security definitions and proofs of security for ID-SPK encryption schemes are beyond the scope of this thesis and will be addressed in our future work.

6.4.2 Random String as ID-SPK

We have explained earlier in Section 6.3 that a naive encryption of an RSA exponent or a group element with a standard block cipher would lead to effective off-line password guessing attacks. Therefore, some form of padding or randomisation of the keys is needed. In the IBC setting, we note that a random string with arbitrary length without any predictable structure can also be used as an identifier. The corresponding public key is usually derived from this identifier by hashing. Since now only a random string needs to be encrypted under the user password, the possibility of using a standard block cipher for the encryption is opened up[3]. For example, in Protocol 6, the server can transport random strings ST_A and ST_B to A and B, respectively, in message (2) as follows:

$$\begin{array}{ll}
(2).\ S \rightarrow A: & A,\ B,\ n_S,\ \{ST_A\}_{PW_A},\ \{ST_B\}_{PW_B} \\
(3).\ A \rightarrow B: & Enc_{\hat{PK}_{SA}}(A,\ B,\ n_{A1},\ n_{A2},\ n_S),\ n_S,\ r_A,\ \{ST_B\}_{PW_B} \\
(4).\ B \rightarrow S: & Enc_{\hat{PK}_{SA}}(A,\ B,\ n_{A1},\ n_{A2},\ n_S), Enc_{\hat{PK}_{SB}}(B,\ A,\ n_{B1},\ n_{B2},\ n_S)
\end{array}$$

Since ST_A and ST_B are just random strings, they do not contain any predictable structure which could leak some information to the attacker as in the case of RSA or Diffie-Hellman keys. Subsequently, users A and B can derive their ID-SPKs $\hat{PK}_{SA} = H_1(A\|S\|ST_A)$ and $\hat{PK}_{SB} = H_1(B\|S\|ST_B)$, respectively, and respond

[3] However, it is still necessary to take care to avoid attacks based on the introduction of redundancy, for example padding, in the block cipher encryption.

to the server via messages (3) and (4). If the server can decrypt B's reply and recover n_S from both the ciphertexts produced with $\hat{PK}_{SA}$ and $\hat{PK}_{SB}$, it can be assured that the users have received the correct random strings. Thus, A and B are authenticated to S. The use of random strings as identifiers is a key property from our identity-based approach which may give the concept of SPK protocols new life.

We remark that to prevent off-line attacks, ciphertexts obtained by encryption under the keys $\hat{PK}_{SA}$ and $\hat{PK}_{SB}$ must not leak useful information about ST_A and ST_B, respectively. This is not a traditional requirement of a public key encryption scheme (it is related to the public key privacy concept in [16]). Also note that since we use a probabilistic encryption scheme here, we have removed the use of confounders c_A and c_B originally proposed in Protocol 6 in messages (3) and (4). Furthermore, users A and B no longer need to encrypt n_S with their respective passwords in their replies to S, in messages (3) and (4). This is because users A and B can demonstrate knowledge of their respective passwords by their ability to construct correct keys from ST_A and ST_B.

6.4.3 Secret Signatures

In what follows, we show some extended properties that an ID-SPK can offer as compared to a conventional SPK. Again, referring to Protocol 6, if in the protocol A (and B) selects and sends ST_A (and ST_B) to the server (rather than the server sending it to the user), we can, in principle, remove messages (1) & (2) and modify messages (3) – (5) as follows:

$$\begin{array}{ll}
(3).\ A \rightarrow B: & Enc_{\hat{PK}_{SA1}}(A,\ B,\ ST_A),\ r_A \\
(4).\ B \rightarrow S: & Enc_{\hat{PK}_{SA1}}(A,\ B,\ ST_A),\ r_A, \\
 & Enc_{\hat{PK}_{SB1}}(A,\ B,\ ST_B),\ r_B \\
(5).\ S \rightarrow B: & Sig_{\hat{SK}_{SA2}}(K_{AB}),\ Sig_{\hat{SK}_{SB2}}(K_{AB})
\end{array}$$

Note that we have replaced nonces n_{A1}, n_{A2}, n_{B1} and n_{B2} in Protocol 6 by random strings ST_A and ST_B. For ease of exposition, we concentrate on the interaction between A and S. In message (3), A encrypts a random string ST_A with an ID-SPK $\hat{PK}_{SA1} = H_1(A\|S\|PW_A)$. It is obvious that a symmetric encryption of the form $\{A,\ B, \ldots,\ ST_A\}_{PW_A}$ cannot be used in message (3) because the identities of A and B are verifiable texts. The server responds with a signature generated with a private key associated with the public key $\hat{PK}_{SA2} = H_1(S\|A\|PW_A\|ST_A)$. The reason for doing this will be clear when we look at the motivation for using $Sig_{\hat{SK}}(\cdot)$, a signature scheme with a private key $\hat{SK}$, in message (5). As compared to the modification of Protocol 6 given in Section 6.4.2, the server cannot reply to A with an encrypted message using an ID-SPK constructed from $H_1(S\|A\|PW_A\|ST_A)$. This is mainly because in such an asymmetric model (where the user only knows an easy-to-remember password and the server has access to the secret public/private key pairs), only the server itself can extract the corresponding private key. This prompts the requirement to use a secret signature which not only provides non-repudiation of the signed message and message recovery, but also preserves message confidentiality. This last property is needed because the server wants only A and B to be able to verify the signatures and recover the signed messages. This, in turn, leads us to the use of an ID-SPK signature scheme with message recovery which can be adapted from [190]. So long as the verification keys used in the scheme of [190] are kept secret between the intended parties, our concept of secret signatures can be used. However, we remark that the IBS scheme with message recovery must be used carefully because the scheme provides message integrity. In other words, a simple off-line password guessing attack would be enabled if a secret signature was created based on a private key corresponding to $\hat{PK}_{SA2} = H_1(A\|S\|PW_A)$. For instance, the attacker could construct an ID-SPK $\hat{PK'}_{SA2} = H_1(A\|S\|PW'_A)$ using a guessed password PW'_A and then attempt to verify the signature. If he used the wrong password, the VERIFY algorithm would return an error message. Because of that, the identifier from which the verifying key is derived must contain a secret value chosen from a space much larger than the password space. We achieve this by including ST_A (or ST_B) in the identifier. It is also worth mentioning that a secret signature should not leak information about the signing key, the verifying key, or the plaintext that has been signed.

As we have explained earlier, a secret identifier can bind a user's password naturally to a secret public/private key pair. As such, secret signatures may be beneficial in a password-based protocol when one or both of the following conditions apply:

(i). Non-repudiation, confidentiality and integrity of a signed message are required.

(ii). An additional line of defence is desirable (e.g. assuming the server keeps its master secret in a tamper-resistant hardware token or smartcard, the attacker cannot impersonate the server to any of its users even if the users' passwords are exposed).

Formal security definitions and proofs of security for ID-SPK signature schemes with message recovery are beyond the scope of the thesis. These will be addressed in our subsequent work on ID-SPKs.

6.5 The ID-SPK Protocols

In the previous sections, we learned that to exploit the advantages of using SPKs in a password-based protocol, the keys must not contain any predictable structure, such as that appearing in RSA or discrete logarithm-based systems. This section presents three-party and two-party ID-SPK protocols which can solve this structural issue in a clean and natural way. We assume that all the protocol participants have agreed on some system parameters for the ID-SPK encryption and signature schemes *a priori*.

Before we look at the ID-SPK protocols, it may be useful to classify some common attacks on password-based protocols.

- *On-line password guessing attacks*: The attacker chooses a password from his dictionary and tries to impersonate a user. He verifies the correctness of his guess based on responses from a server. If the impersonation fails, the attacker tries again using a different password from his dictionary. Note that the attacker can also impersonate the server to the user by intercepting and modifying a message originating from the server before forwarding it to the user (assuming the server has used the user's password in some way in creating the message). He can verify his password guesses based on responses from the user.
- *Off-line password guessing attacks*: The attacker records past communication and makes a verifiable guess using a password from his dictionary. If the guess

fails, the attacker tries again with a different password until the correct password is found. No on-line participation of a server (or a user) is required and the attacks take place without the knowledge of the actual protocol participants.

- *Attacks exploiting exposed secrets*: The attacker may occasionally have access to sensitive information such as past session keys or a user's password. This is possible when the user's machine or the server are compromised, or the user's password is revealed through a keystroke logger. It is a desirable security property that exposure of past session keys will not lead to the exposure of the user's password and vice versa.
- *Undetectable on-line password guessing attacks*: The attacker mounts an on-line guessing attack. However, a failed guess cannot be detected and logged by the server (or the user). In other words, the protocol participants cannot distinguish a genuine protocol message from a modified (malicious) message.

Security Model. We sketch here our definition of the security for a password-based ID-SPK protocol, using an informal security model. In the model, there is an adversary E, who is allowed to watch regular runs of the protocol between a user, $U \in \mathcal{U}$, where $\mathcal{U}$ is a set of protocol users, and a server S. E can actively communicate with the user and the server in replay, impersonation, and man-in-the-middle attacks. The adversary can prompt one of the parties to initiate new sessions. In each session, E can see all the messages sent between U and S. Furthermore, he can intercept the messages and modify or delete them. Also, E gets to see whether S accepts the authentication or not. In addition, we allow the adversary to establish as many "accounts" as he wishes with the server using his own chosen passwords. He can then run arbitrarily many authentication sessions using these accounts to obtain information for his attacks.

It is clear that if the user picks a password from his dictionary $\mathcal{D}$, then the adversary that attempts n active impersonation attacks (or on-line guessing attacks) over n distinct sessions with the server can succeed with probability at least $n/|\mathcal{D}|$ by trying a different password from $\mathcal{D}$ in each attempt.

Definition 1 (*Informal*) *We say that the ID-SPK protocol is secure if all the following conditions are satisfied.*

1. *No useful information about a session key is revealed to the adversary during a successful protocol run and the exposure of past session keys does not leak any information about the current session key.*
2. *The adversary cannot discover the correct user password after n active impersonation attempts with probability significantly higher than $n/|\mathcal{D}|$.*
3. *The protocol is resistant to off-line password guessing attacks.*
4. *The protocol is resistant to undetectable on-line password guessing attacks.*
5. *The exposure of the user's past session keys will not lead to the exposure of the user's password and vice versa.*

We remark that a formal security model and definition, such as those used in [17], have not been employed in this thesis. This is mainly because the objectives of the chapter are to explore new ways of using SPKs in the IBC setting and to study their practical application to the TLS protocol. The above informal security model and definition will be used instead in the following sections in describing three-party and two-party ID-SPK protocols.

6.5.1 The Three-Party ID-SPK Protocol

In [85], Gong further optimised the original SPK protocol in [86] by reducing the number of protocol messages to reduce the communication costs incurred by the protocol. We further modify Gong's optimised SPK protocol by building on the example given in Section 6.4.3, as shown in Protocol 8.

Protocol 8 *The Modified Gong SPK Protocol*

$$\begin{array}{ll}
(1).\ A \rightarrow B: & A,\ r_A,\ Enc_{\hat{PK}_{A1}}(A,\ ST_A) \\
(2).\ B \rightarrow S: & B,\ Enc_{\hat{PK}_{B1}}(B,\ ST_B),\ A,\ r_A,\ Enc_{\hat{PK}_{A1}}(A,\ ST_A) \\
(3).\ S \rightarrow B: & Sig_{\hat{SK}_{B2}}(K_{AB}),\ Sig_{\hat{SK}_{A2}}(K_{AB}) \\
(4).\ B \rightarrow A: & Sig_{\hat{SK}_{A2}}(K_{AB}),\ \mathrm{MAC}_{K_{AB}}(r_A),\ r_B \\
(5).\ A \rightarrow B: & \mathrm{MAC}_{K_{AB}}(r_B)
\end{array}$$

In Protocol 8, users A and B select their respective random strings ST_A and ST_B and encrypt them with an ID-SPK. As before, $\hat{PK}_{A1} = H_1(A\|S\|PW_A)$ and $\hat{PK}_{B1} = H_1(B\|S\|PW_B)$. The server recovers ST_A and ST_B, and computes private keys $\hat{SK}_A$ and $\hat{SK}_B$ matching the ID-SPKs constructed from the random strings, where $\hat{PK}_{A2} = H_1(S\|A\|PW_A\|ST_A)$ and $\hat{PK}_{B2} = H_1(S\|B\|PW_B\|ST_B)$. The private keys are then used to sign a session key. We assume that an IBS scheme with message recovery is used, so that the intended recipients are able to recover the session key. These secret signatures also provide non-repudiation. Even though this is rarely a requirement in protocols for authentication and key establishment, it automatically provides the important data integrity and data origin authentication services [31]. Note that $\mathrm{MAC}_{K_{AB}}(r_A)$ and $\mathrm{MAC}_{K_{AB}}(r_B)$ in messages (4) and (5) are used by A and B, respectively, to prove to each other that they are indeed sharing the same session key. This provides key confirmation.

To improve the performance of Protocol 8, $Sig_{\hat{SK}_{A2}}(K_{AB})$ and $Sig_{\hat{SK}_{B2}}(K_{AB})$ in message (3) can be replaced with $\{K_{AB}\}_{K_A}$ and $\{K_{AB}\}_{K_B}$, respectively, where $K_A = F(ST_A)$ and $K_B = F(ST_B)$, with F being a key derivation function. It is worth noting that K_A and K_B must not be derived from user passwords because this would allow the attacker to mount an off-line password guessing attack. The correctness of a candidate password could be verified by comparing $\mathrm{MAC}_{K_{AB}}(r_A)$ from message (4) with $\mathrm{MAC}_{K'_{AB}}(r_A)$, where K'_{AB} is obtained using the guessed password.

Security Analysis. Protocol 8 shows that users A and B communicate with S using secret identifiers $\mathrm{ID}_A = A\|S\|PW_A$ and $\mathrm{ID}_B = B\|S\|PW_B$, respectively. These identifiers involve the users' passwords. Since S is the only party who has knowledge of PW_A and PW_B apart from A and B, the users should receive the same session key created by the server provided the correct private keys are used to transport the session key. If A and B can successfully recover K_{AB} from their respective received secret signatures, they can be assured of the authenticity of the server.

It is clear that requirement 1 of Definition 1 can be satisfied if the session key is randomly generated by the server. Moreover, the session key cannot be computed directly by the adversary E.

By observing a protocol run, E can gather information such as $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$, $Enc_{\hat{PK}_{B1}}(B,\ ST_B)$, $Sig_{\hat{SK}_{A2}}(K_{AB})$ and $Sig_{\hat{SK}_{B2}}(K_{AB})$. However, since we assume that the ID-SPK encryption scheme used in this protocol is IND-ID-CCA secure, E cannot gain any useful information about ST_A and ST_B from the encrypted random strings $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$ and $Enc_{\hat{PK}_{B1}}(B,\ ST_B)$ without knowledge of the master secret held by the server. As for the session key transportation in the form of secret signatures from the server to the users, E can choose his own verification keys in an attempt to recover the session key. However, there seems to be no efficient way for E to predict the correct ID-SPK if the ID-SPK signature scheme used in the protocol offers appropriate security. In particular, we assume that E cannot distinguish a secret signature from a randomly generated string if the identifier is constructed using sufficient randomness. We also assume that the adversary cannot forge valid secret signatures, impersonating the server to users. Apart from that, it is very unlikely that E can impersonate a legitimate user by guessing the user's password. This is so since the adversary's impersonation attack would be detected immediately by the server if the user's chosen random string cannot be recovered successfully from message (2). Note that the number of impersonation attempts can be kept acceptably small by using mechanisms that can log and control the number of failed authentication attempts. A brute force attack on message (3) or (4) to deduce the session key can be easily thwarted by using random strings ST_A and ST_B with entropy significantly larger than the password space of $\mathcal{D}$. Also, so long as ST_A and ST_B are fresh and randomly generated for each protocol run, E would not be able to mount a replay attack. It is thus conjectured that requirement 2 is satisfied.

When E uses a password $PW'_A \in \mathcal{D}$ to mount an off-line password guessing attack on a recorded $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$, there is no way for the adversary to verify the correctness of $\hat{PK'}_{A1} = H_1(A\|S\|PW'_A)$ if the ID-SPK encryption is randomised and IND-ID-CPA secure. If E selects $PW'_A \in \mathcal{D}$ and ST'_A at random, computes $\hat{PK'}_{A2} = H_1(S\|A\|PW'_A\|ST'_A)$, and then attempts to verify $Sig_{\hat{SK}_{A2}}(K_{AB})$, his check will almost certainly fail since the entropy of ST_A is much larger than the entropy of PW_A. Thus this form of off-line guessing attack will not succeed and therefore, Protocol 8 also satisfies requirement 3.

If E has a valid account with S, he may possibly mount an insider attack by impersonating A to S, pretending to be wanting to establish a session key K_{AE} with himself. In the attack, E initiates the protocol by computing $Enc_{\hat{PK'}_{A1}}(A,\ ST'_A)$ with a guessed password PW'_A, and hence $\hat{PK'}_{A1} = H_1(A\|S\|PW'_A)$. However, once this message has reached S, the server should get an error message when decrypting $Enc_{\hat{PK'}_{A1}}(A,\ ST'_A)$ using the decryption key matching $\hat{PK'}_{A1}$. Therefore it is clear that the protocol can detect on-line guessing attacks and thus requirement 4 is satisfied.

On certain rare occasions, E may have access to A's or B's machine and thus the past session keys shared between them are exposed. However, since E has no knowledge of the master secret of S and the matching private component of $\hat{PK}_{A2}$, E still cannot determine PW_A even though he can mount a brute-force attack on $\hat{PK}_{A2}$. On the other hand, if for some reason, E has the correct password for A, he may attempt to find the value of ST_A given A's password and the ciphertext $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$. Since the encryption scheme is IND-ID-CCA secure, E only has a negligible success probability to discover the correct ST_A. Also, since the value of the verification key for $Sig_{\hat{SK}_{A2}}(K_{AB})$ depends on the secret value ST_A, E can only recover the session key with negligible probability and forward secrecy of the protocol is preserved. Hence, requirement 5 is also satisfied and we conclude that Protocol 8 is a secure ID-SPK assuming that the ID-SPK encryption and signature schemes are appropriately secure.

Nevertheless, it is worth noting that if the server's master secret is compromised, the adversary can deduce the users' passwords without much difficulty. For instance, for each candidate password PW'_A, E can extract the private key matching the identifier $\text{ID}'_A = A\|S\|PW'_A$ and use it to attempt to decrypt $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$

from message (1), and check if the decryption unveils A's identity. Hence, it is of the utmost importance that the server's master secret is kept private, for example by using a strong protective mechanism such as storing it in a tamper-resistant device.

6.5.2 The Two-Party ID-SPK Protocol

We now present a Diffie-Hellman type two-party ID-SPK protocol. Our protocol is adapted from [1, 17] which make use of the encrypted Diffie-Hellman ephemeral key exchange technique between two parties. We apply the identity-based techniques that we introduced in Section 6.4 to obtain Protocol 9, as shown below.

Protocol 9 *The Diffie-Hellman ID-SPK Protocol*

$$\begin{array}{ll} (1).\ A \rightarrow S: & A,\ Enc_{\hat{PK}_{A1}}(aP) \\ (2).\ S \rightarrow A: & S,\ Sig_{\hat{SK}_{A2}}(xP) \end{array}$$

In Protocol 9, the user randomly selects $a \in \mathbb{Z}_q^*$ and computes aP, where $P \in \mathbb{G}_1$ is part of the system parameters. A then encrypts the Diffie-Hellman component with $\hat{PK}_{A1} = H_1(A\|S\|PW_A)$ and sends message (1) to S. The server extracts the matching private key $\hat{SK}_{A1}$ with its master secret to recover aP. Subsequently, S picks a random number $x \in \mathbb{Z}_q^*$ and calculates xP. The server then extracts another private key which is associated with $\hat{PK}_{A2} = H_1(S\|A\|PW_A\|aP)$, produces $Sig_{\hat{SK}_{A2}}(xP)$, and transmits it to A. After receiving message (2), the user retrieves xP with $\hat{PK}_{A2}$. Both the user and the server calculate a session key as $K_{AS} = F(A\|S\|PW_A\|aP\|xP\|axP)$, where F is a key derivation function. Note that key confirmation can be provided by adding a third message from A to S, in which A provides a MAC computed on all the protocol messages using the session key (derived using a different key derivation function to F).

Security Analysis. As with Protocol 8, user A uses an ID-SPK, but in this case to transport a Diffie-Hellman ephemeral key aP to the server. It is worth noting that message (1) can be replayed but this is not an issue because the purpose of the protocol is to authenticate the session key. If the adversary E has captured message

(1) and replays it, he will not gain any information about the session key, unless he has access to a and to xP in message (2). Also, we note that since only S other than A has access to PW_A, S is authenticated to A when A successfully recovers xP (recall that an ID-SPK signature scheme provides a message integrity check) using $\hat{PK}_{A2} = H_1(S\|A\|PW_A\|aP)$.

Clearly, requirement 1 of Definition 1 can be satisfied if the ephemeral Diffie-Hellman components from A and S are randomly generated and information used to compute the session key including a, aP, x, xP, and PW_A cannot be computed directly by E.

E has access to $Enc_{\hat{PK}_{A1}}(aP)$ and $Sig_{\hat{SK}_{A2}}(xP)$ through watching a protocol run between A and S. However, since we assume that the ID-SPK encryption scheme used in this protocol is IND-ID-CCA secure, E cannot obtain any useful information about aP from $Enc_{\hat{PK}_{A1}}(aP)$ without knowledge of the master secret held by the server. Also, we assume that the ID-SPK signature scheme used in the protocol produces secret signatures $Sig_{\hat{SK}_{A2}}(xP)$ that are indistinguishable from random strings. Hence it is hard for E to deduce any information about the Diffie-Hellman component chosen by the server. Apart from that, as we have discussed when analysing Protocol 8, it appears unlikely that E will successfully impersonate A in n attempts with probability significantly higher than $n/|\mathcal{D}|$ or mount a replay attack, provided aP and xP are fresh and their entropy is significantly higher than the entropy of $\mathcal{D}$. Also, the use of an incorrect password in generating $\hat{PK}_{A1}$ can be easily detected by the server when the server uses the wrong matching private key to recover aP. It is thus conjectured that requirements 2, 3 and 4 are satisfied.

It is possible that E may have access to A's machine and recover the past session keys used by A. In that case, despite the fact that E knows K'_{AS}, he must be able to break the key derivation function F in order to deduce A's password. On the other hand, if for some reason A's password is revealed to E, E may attempt to find the value of aP given A's password and the ciphertext $Enc_{\hat{PK}_{A1}}(aP)$. Since the encryption scheme is IND-ID-CCA secure, E only has a negligible success probability to find the correct aP. Also, since the value of the verification key for $Sig_{\hat{SK}_{A2}}(xP)$ depends on the secret value aP, E can only recover the session key with negligible probability. This is related to the forward secrecy of protocols discussed in [1, 17]. Therefore, requirement 5 is also satisfied. We note that in addition to having met this requirement, even if E knows aP and xP, he has to solve the intractable CDH problem in order to calculate axP and hence the session key. We conclude that Protocol 9 is a secure ID-SPK protocol.

As with the security of Protocol 8, it is essential to have the server's master secret adequately protected to ensure that the aforementioned security conditions hold.

6.6 Integrating ID-SPKs with TLS

Steiner *et al.* [168] first proposed the integration of password-based Diffie-Hellman encrypted key exchange with the TLS handshake protocol (DH-EKE/TLS). Here we propose a server-authenticated, TLS-compatible ID-SPK protocol (ID-SPK/TLS) by building on the ideas developed in the previous sections.

Protocol 10 *The ID-SPK/TLS Protocol*

```
(1)  $A \rightarrow S$ :  ClientHello = $n_A$, session_id, cipher_suite
(2)  $S \rightarrow A$ :  ServerHello = $n_S$, session_id, cipher_suite
                ServerKeyExchange = $\{ST_A\}_{PW_A}$
                ServerHelloDone
(3)  $A \rightarrow S$ :  ClientKeyExchange = $Enc_{\hat{PK}_A}$(pre_master_secret)
                ClientFinished
(4)  $S \rightarrow A$ :  ServerFinished
```

The description of Protocol 10 and comparison with the standard TLS handshake protocol are as follows.

(1) As with the current TLS specification [50], Protocol 10 begins with A sending S a `ClientHello` message. The message contains a fresh nonce, a session identifier and a cipher specification. In the standard TLS protocol, `session_id` contains a value which identifies a previously established session between A and S some or all of whose security parameters A wishes to re-use. Otherwise it is an empty field. In Protocol 10, we envisage that if there is no session identifier to be re-used or resumed, then `session_id` would carry the identity

of A. Meanwhile, `cipher_suite` contains a cipher specification extended from TLS version 1.0 to handle the ID-SPK encryption and signature schemes. For example, `TLS_ID_SPK_WITH_DES_CBC_SHA` would define an ID-SPK-supported cipher specification that uses DES-CBC and SHA as the symmetric encryption algorithm and hash function, respectively.

(2) S responds with a `ServerHello` message which contains an independent nonce and a session identifier whose value depends on the client's input. For instance, if `session_id =` A, S will create a new session identifier. Otherwise, S will search its local cache to check if there is a session identifier which matches the value submitted by A and decide whether or not to resume the session. S then generates a random string ST_A that will be used by A to communicate a pre-master secret. String ST_A is encrypted using DES-CBC with A's password. As in Section 6.4.2, care must be taken to avoid introduction of redundancy that would enable off-line guessing attacks. The encrypted random string is sent to A in `ServerKeyExchange`. The `ServerHelloDone` message is then transmitted to indicate the end of step (2).

It is worth noting that if we replace $\{ST_A\}_{PW_A}$ in the `ServerKeyExchange` message with $Sig_{\hat{SK_A}}(ST_A)$, the protocol will be insecure. For let the correct verification key for $Sig_{\hat{SK_A}}(ST_A)$ be $\hat{PK_A} = H_1(A\|S\|PW_A)$. At first glance, it may seem that the `ServerKeyExchange` message would achieve its intended purpose in Protocol 10, i.e. securely transporting a random string to A. However, the signature integrity check in an IBS scheme with message recovery can be used to mount a simple off-line password guessing attack, as described in Section 6.4.3.

(3) A uses her password to recover ST_A. A then selects a random pre-master secret and encrypts it using an ID-SPK encryption scheme with $\hat{PK_A} = H_1(A\|S\|PW_A\|ST_A)$. It appears that the attacker could not learn any information about PW_A by mounting an off-line password guessing attack on a recorded $Enc_{\hat{PK_A}}$(`pre_master_secret`), since the ID-SPK encryption is assumed to be IND-ID-CPA secure (as discussed in the security analysis of Protocol 8 in Section 6.5.1). For the same reason, the attacker only has a negligible success probability to discover the correct pre-master secret chosen by A even if PW_A is revealed.

A transmits the encrypted pre-master key to S using the `ClientKeyExchange` message. Note that A still cannot authenticate S (and its system parameters) at this stage because A has only decrypted $\{ST_A\}_{PW_A}$ and used the recovered ST_A to perform the ID-SPK encryption. However, S can authenticate A when A completes step (3) as the `ClientFinished` message contains a verification value:

$$\mathrm{PRF}(\texttt{master\_secret}, \text{“}\texttt{client finished}\text{”}, \texttt{h\_messages\_1}, \texttt{h\_messages\_2}),$$

where PRF is a pseudo-random function, and

$$\texttt{master\_secret} = \mathrm{PRF}(\texttt{pre\_master\_secret}, \text{“}\texttt{master secret}\text{”}, n_A, n_S).$$

Here, `h_messages_1` and `h_messages_2` represent hash values of all handshake messages up to but not including this message, using different hash functions. If A has used the correct password in deriving the key $\hat{PK}_A$ used to encrypt the pre-master secret and S has retrieved this value, then the verification value computed by S matches the value sent by A in `ClientFinished`, and thus A is authenticated.

(4) S calculates a verification value as above with “`server finished`” replacing “`client finished`” and transmits it to A in `ServerFinished`. A checks the verification value received from S. If it was correctly calculated, only now has S been authenticated by A, since only S could have derived the appropriate decryption key using the shared password PW_A and its master secret to retrieve the correct pre-master secret.

Subsequently, `master_secret` will be used to derive further keys for protecting application data between A and S.

As with Protocols 8 and 9, Protocol 10 also achieves forward secrecy. The exposure of the user's long-term credential, i.e. password, does not compromise the user's past session keys since the encryption scheme used is IND-ID-CPA secure. The master secret of the server must not be compromised in order for this condition to hold.

Discussion. In [168], the authors claimed that five protocol flows are the best possible design to prevent dictionary attacks in the DH-EKE/TLS protocol. Also, swapping the order of `ClientFinished` and `ServerFinished` was necessary. Our Protocol 10 shows that no structural changes are required but only minimal alterations to the contents of the standard TLS protocol messages: (i) inclusion of the client's identity in `session_id`; and (ii) replacement of a signed temporary RSA or a Diffie-Hellman ephemeral key with an encrypted random string in `ServerKeyExchange`. These modifications only require small adjustments to data fields of the current TLS protocol specification. Hence, the advantages that changes (i) and (ii) could bring seem to outweigh any implementation issues that they may cause.

One limitation of our proposed protocol is the need for pre-distribution of the server's system parameters. Otherwise, a man-in-the-middle attack similar to Patel's attack on Protocol 7 is possible. In this attack, the attacker can impersonate S to A by inserting his own set of system parameters and substituting $\{ST_A\}_{PW_A}$ with a random string R_A. A would then use her password to recover a value ST'_A from R_A. When A replies with his chosen pre-master secret encrypted under the identifier $\mathrm{ID}'_A = A\|S\|PW_A\|ST'_A$, for each candidate password PW'_A, the attacker now decrypts R_A using PW'_A to obtain a value ST'_A, and then uses ST'_A to derive a private key corresponding to the identifier $\mathrm{ID}'_A = A\|S\|PW'_A\|ST'_A$. Subsequently, the server recovers a pre-master secret and computes a `ClientFinished` value. The guessed password can then be verified by comparing the `ClientFinished` value that S computed with the `ClientFinished` value that A sent to S. If they match, then the guessed password is a correct one.

6.7 Application in MyProxy

It is not uncommon that many users within the grid community fail to take the necessary precautions to protect their machines such as installing the latest vulnerability patches and updating the virus definition lists on their machines. This may lead to partial or complete control of the machines by a remote attacker who exploits vulnerabilities found on the machines. Even if the users do care about patching vulnerabilities, it may not be difficult for the attacker to install trojan horses on the users' computers. This may well explain why many grid implementations have incorporated the MyProxy system to securely store and protect users' long-term credentials.

Our ID-SPK/TLS protocol can fit nicely into our IKIG proposal since all the required identity-based cryptographic primitives will already be in place. A user can access the MyProxy server to retrieve her long-term credential by using a password-based TLS protocol. We note that there also exist trojans or keystroke loggers that can track and steal user passwords. Luckily, there are also countermeasures against this kind of attack such as the use of one-time passwords and virtual keyboards[4].

From a more general perspective, our ID-SPK/TLS protocol seems to be a good candidate for a password-based, authenticated key agreement protocol used in identity-based cryptographic schemes for delivering private keys. The protocol aligns nicely with the certificate-free property of the schemes, unlike the common proposal of using the certificate-based TLS protocol for private key distribution, such as in [164].

6.8 Summary

We studied the history of secret public key protocols and discussed some known problems with these protocols. We explored some interesting properties of identity-based cryptography which form the basis of our proposed identity-based secret public key protocols. These properties also allow us to convert a conventional identity-based encryption scheme and a standard identity-based signature scheme (with message recovery) into their secret public key equivalents.

We presented three-party and two-party identity-based secret public key protocols for key exchange. Our heuristic security analyses show that the protocols appear to be secure against off-line password guessing attacks and undetectable on-line password guessing attacks, and provide forward secrecy. Then we combined the new properties from identity-based secret public keys and the techniques used in constructing the identity-based secret public key protocols, and showed that secret public keys can support the use of passwords in the TLS handshake protocol in a very natural way.

[4] A virtual keyboard is one of the latest technologies used to combat stealing of passwords through keystroke loggers. It is a keyboard displayed as a pop-up window on the user's desktop. The user must use his mouse to input his password. The order of the characters on the virtual keyboard is normally unpredictable.

Chapter 7

Conclusions

Contents

This chapter summarises the thesis and gives some concluding remarks which reflect the problems that we have studied and the results that we have achieved. We also give suggestions for future work in this area.

7.1 Concluding Remarks

The development of grid computing and identity-based cryptography are amongst today's most important technical innovations in the field of computer science and cryptology. As we have described in Chapter 2, security issues in grid applications are numerous due to complex grid properties such as heterogeneity, scalability and adaptability. One of the unique security requirements for grid applications is the use of short-term or proxy credentials to achieve single sign-on, delegation and other security services. These are made possible through the combined use of standard X.509 and proxy certificates, supported by PKI.

In this thesis, we studied the application of some identity-based cryptographic schemes presented in Chapter 3, in designing security infrastructures for grid applications. The main focus has been on simplifying current PKI-based security architectures which make extensive use of certificates for supporting grid security services. We addressed issues related to certificate and public key management, such as certification and verification of public keys, and distribution of certificates, which cause extra overheads to and potentially limit the scalability of grid applications. It is natural to consider the application of IBC to grid security because of its attractive properties, such as being certificate-free and using small key sizes, which may well match the requirements of grid computing. The properties of IBC, in turn, are likely to result in a more lightweight security architecture than the certificate-based PKI approach. We presented our findings that pertain to the use of IBC for constructing IKIG in Chapter 4. Our results show that even though the PKI-based GSI is workable, it is still far from lightweight in terms of the network bandwidth requirement. On the other hand, IKIG, which makes use of identity-based techniques, consumes minimal communication bandwidth. The significant saving in message sizes in IKIG augurs well for the on-going transition from transport-level to message-level security based on web services. In addition, we observed that identity-based public keys can be used in a very natural way to support various grid security services, such as mutual authentication and delegation. However, one limitation is that IKIG inherits the key escrow property that plagues identity-based cryptographic schemes. Despite that, the drawback may not pose a major problem since the use of the MyProxy system technically also introduces a key escrow facility. Both IKIG and GSI with the MyProxy plug-in require strong trust relationships to be in place between users and the relevant trusted third parties.

There may be circumstances where key escrow is not desirable for grid applications. In Chapter 5, we proposed DKIG as a solution that not only removes key escrow, but that also eliminates the requirement for short-term private key distribution from a PKG/TA to its users. In our DKIG proposal, each user publishes a fixed IBC parameter set through a standard X.509 certificate. The parameter set can be used by the user to manage his proxy credentials by acting as his own PKG. This simple technique appears to be cleaner than the use of threshold cryptographic or secret sharing techniques that we highlighted in Chapter 5. Our research findings show that, even though the communication costs in DKIG are higher than in IKIG, they are still relatively low compared to those of the GSI. We have also shown that the computational overhead in DKIG is increased as compared to IKIG, due to the increased number of pairing computations. We foresee that further improvements in the speed of pairing computations are likely to be made with the discovery of new algorithmic techniques. This is especially so, considering that pairing-based cryptography is still at a relatively young age. On the other hand, we remark that since certificates are used for users' long-term credentials in DKIG, identity-based techniques are applicable only to the users' proxy credentials. The benefits that the identity-based techniques could offer are therefore limited to the user level.

In Chapter 6, we extended our study of the application aspects of IBC to secret public keys. The concept of secret public keys has historically been employed in password-based authentication protocols. However, the use of more conventional secret public keys such as RSA and Diffie-Hellman keys can allow various number theoretic attacks and this concept was thus thought to be unworkable. In this thesis, we explored and introduced new properties of identity-based secret public keys. In the IBC setting, a secret public key can be computed based on a random string. This technique appears to offer a clean and natural way of removing any predictable structure in the secret public key. By using identity-based techniques, we designed a TLS-like identity-based secret public key protocol. This protocol allows passwords to be tied directly to the establishment of secure TLS channels. Furthermore, our protocol requires only relatively small changes to the message contents of the current TLS handshake protocol, and is only based on the use of easy-to-remember user passwords. These advantages seem to make our approach a sensible and practical improvement over the design of the current authentication protocol that MyProxy employs.

We have highlighted some of the fundamental issues in the PKI-based GSI. Some identity-based solutions aimed at resolving these issues have been proposed. In conclusion, we believe that a security infrastructure designed using identity-based techniques has more advantages than disadvantages as compared to the PKI-based GSI. Our identity-based approach offers more flexibility in terms of key usage and management than the more conventional PKI approach. There are some useful security features that seem to be provided only by identity-based cryptographic schemes, for example, generation of public keys on-the-fly and binding of a policy or password to a public key. More importantly, identity-based techniques offer a more natural and clean way of delivering various grid security services. We expect that the identity-based concepts and techniques presented in this thesis may well be useful in other applications. These may include, for example, P2P systems, ad hoc network environments, and distributed systems in which it is desirable to use and manage public keys in a lightweight, natural and flexible manner.

7.2 Suggestions for Future Work

At the time of writing, no implementation of an identity-based grid security architecture has been carried out either by grid or IBC research communities. This is partly because the study of the application of IBC in grid security at the architectural level is still an ongoing research avenue. This study is essential to understand and answer various fundamental issues and questions in regards to the suitability of IBC in grid applications, for example, could an identity-based security architecture support the use of short-term keys in a better way than using the conventional RSA-based PKI? In the medium-term, however, the development of prototypes that implement our IKIG and DKIG proposals would seem to be a sensible and natural follow-on. Nevertheless, constructing prototypes may not be straightforward and could be time-consuming. This is so because it is believed that all the currently available tools and mechanisms used to construct grid systems do not yet have the capability of supporting IBC. As we have discussed in Sections 4.8.2 and 5.8, GSS-API, OpenSSL, various web services security standards, and other grid-related tools must be modified/extended to support the adoption of identity-based cryptographic schemes. If prototypes of IKIG and DKIG can be developed successfully, actual performance figures could be obtained by testing the prototypes. These would be very useful in further assessing the efficiency and suitability of our IKIG and DKIG proposals.

The concept of identity-based secret public keys appears to be new. In this thesis, we focussed on investigating new properties possessed by identity-based secret public keys and how these properties can be used to design password-based protocols. We also explored the practical use of the concept, aligning with the theme of this thesis. However, our proposed identity-based secret public key protocols require formal security analyses to prove the security of the protocols. Before we can do that, we must also develop security models and proofs for identity-based secret public key encryption and signature schemes. It is not clear if this is a straightforward exercise since these schemes have unusual properties compared to standard schemes. For example, secret signatures produced by an identity-based secret public key signature scheme should not only provide authentication and non-repudiation, but also data confidentiality. In summary, studying security models and proofs for identity-based secret public key protocols and schemes seems to be a natural and important step to further develop the concept of identity-based secret public keys. This may, in turn, stimulate new research on the concept.

Bibliography

[1] M. Abdalla, O. Chevassut, and D. Pointcheval. One-time verifier-based encrypted key exchange. In S. Vaudenay, editor, *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2005*, pages 47–64. Springer-Verlag LNCS 3386, 2005.

[2] M. Abdalla, P. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In S. Vaudenay, editor, *Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2005*, pages 65–84. Springer-Verlag LNCS 3386, 2005.

[3] M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In A. Menezes, editor, *Proceedings of the RSA Conference: Topics in Cryptology - the Cryptographers' Track (CT-RSA 2005)*, pages 191–208. Springer-Verlag LNCS 3376, 2005.

[4] M. Abe and T. Okamoto. A signature scheme with message recovery as secure as discrete logarithm. In K. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology - Proceedings of ASIACRYPT 1999*, pages 378–389. Springer-Verlag LNCS 1716, 1999.

[5] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In C.S. Laih, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2003*, pages 452–473. Springer-Verlag LNCS 2894, 2003.

[6] S.S. Al-Riyami and K.G. Paterson. Tripartite authenticated key agreement protocols from pairings. In K.G. Paterson, editor, *Proceedings of the 9th IMA International Conference on Cryptography and Coding*, pages 332–359. Springer-Verlag LNCS 2898, 2003.

[7] J. Almond and D. Snelling. UNICORE: Uniform access to supercomputing as an element of electronic commerce. *Future Generation Computer Systems*, 15(5-6):539–548, October 1999.

[8] J. Astalos, R. Cecchini, B. Coghlan, R. Cowles, U. Epting, T. Genovese, J. Gomes, D. Groep, M. Gug, A. Hanushevsky, M. Helm, J. Jensen, C. Kanellopoulos, D. Kelsey, R. Marco, I. Neilson, S. Nicoud, D. O'Callaghan, D. Quesnel, I. Schaeffner, L. Shamardin, D. Skow, M. Sova, A. Wäänänen, and P. Wolniewiczand W. Xing. International Grid CA interworking, peer review and policy manangement through the European DataGrid Certification Authority Coordination Group. In P.M.A. Sloot, A.G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Proceedings of the European Grid Conference (EGC 2005)*, pages 285–294. Springer-Verlag LNCS 3470, 2005.

[9] E. Barker, W. Barker, W. Burr, W. Polk, and Miles Smid, editors. *Recomendation for Key Management Part 1: General.* NIST Special Publication 800-57, August 2005. Available at http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf, last accessed in January 2006.

[10] P. S. L. M. Barreto, S. D. Galbraith, C. Ó h́Eigeartaigh, and M. Scott. *Efficient Pairing Computation on Supersingular Abelian Varieties.* Cryptology ePrint Archive, Report 2004/375, September 2005. Available at http://eprint.iacr.org/2004/375.

[11] P.S.L.M. Barreto. *The Pairing-Based Crypto Lounge.* Available at http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html, last accessed in November 2005.

[12] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology - Proceedings of CRYPTO 2002*, pages 354–368. Springer-Verlag LNCS 2442, 2002.

[13] P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, C. Galdi, and G. Persiano, editors, *Proceedings of the 3rd International Conference on Security in Communication Networks (SCN 2002)*, pages 263–273. Springer-Verlag LNCS 2576, 2002.

[14] P.S.L.M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In M. Matsui and R. Zuccherato, editors, *Proceedings of the 10th International Workshop on Selected Areas in Cryptography(SAC 2003)*, pages 17–25. Springer-Verlag LNCS 3006, 2004.

[15] J. Basney, M. Humphrey, and V. Welch. The MyProxy online credential repository. *Journal of Software: Practice and Experience*, 35(9):817–826, July 2005.

[16] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2001*, pages 566–582. Springer-Verlag LNCS 2248, 2001.

[17] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2000*, pages 139–155. Springer-Verlag LNCS 1807, 2000.

[18] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In A.D. Santis, editor, *Advances in Cryptology - Proceedings of EUROCRYPT '94*, pages 92–111. Springer-Verlag LNCS 950, 1995.

[19] M. Bellare and P. Rogaway. *The AuthA Protocol for Password-Based Authenticated Key Exchange.* Contribution to IEEE P1363, March 2000.

[20] M. Bellare and M. Yung. Certifying permutations. *Journal of Cryptology*, 9(1):149–166, 1996.

[21] S.M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, 1992.

[22] S.M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the 1st ACM Computer and Communications Security Conference*, pages 244–250. ACM Press, 1993.

[23] I.F. Blake, G. Seroussi, and N.P. Smart, editors. *Elliptic Curve Cryptography.* Cambridge University Press, LMS 265, Cambridge, 1999.

[24] S. Blake-Wilson, G. Karlinger, T. Kobayashi, and Y. Wang. Using the elliptic curve signature algorithm (ECDSA) for XML digital signatures. *The Internet Engineering Task Force (IETF)*, RFC 4050, April 2005.

[25] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology - Proceedings of CRYPTO 2001*, pages 213–229. Springer-Verlag LNCS 2139, 2001.

[26] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[27] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2003*, pages 416–432. Springer-Verlag LNCS 2656, 2003.

[28] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In P. Gaudry and N. Gurel, editors, *Advances in Cryptology - Proceedings of ASIACRYPT 2001*, pages 514–532. Springer-Verlag LNCS 2248, 2001.

[29] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[30] M.K. Boyarsky. Public-key cryptography and password protocols: The multi-user case. In *Proceedings of the 6th ACM Computer and Communications Security Conference*, pages 63–72. ACM Press, 1999.

[31] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, Berlin, 2003.

[32] X. Boyen. Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. In D. Boneh, editor, *Advances in Cryptology - Proceedings of CRYPTO 2003*, pages 383–399. Springer-Verlag LNCS 2729, 2003.

[33] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2000*, pages 156–171. Springer-Verlag LNCS 1807, 2000.

[34] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau,editors. *eXtensible Markup Language Version 1.0 (Third Edition)*, February 2004. Available at http://www.w3.org/TR/REC-xml/, last accessed in November 2005.

[35] E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *Proceedings of the 10th ACM Computer and Communications Security Conference*, pages 241–250. ACM Press, 2003.

[36] K. Brincat. On the use of RSA as a secret key cryptosystem. *Designs, Codes, and Cryptography*, 22(3):317–329, 2001.

[37] S. Cantor, J. Kemp, R. Philpott, and E. Maler, editors. *Assertions and Protocols for the OASIS Security Assertion Markup Language(SAML) Version 2.0*. OASIS Standard 200503, March 2005.

[38] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password interception in a SSL/TLS channel. In D. Boneh, editor, *Advances in Cryptology - Proceedings of CRYPTO 2003*, pages 583–599. Springer-Verlag LNCS 2729, 2003.

[39] J.C. Cha and J.H. Cheon. An identity-based signature from Gap Diffie-Hellman groups. In Y.G. Desmedt, editor, *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2003*, pages 18–30. Springer-Verlag LNCS 2567, 2003.

[40] D. Chaum, E.v. Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In J. Feigenbaum, editor, *Advances in Cryptology - Proceedings of CRYPTO'91*, pages 470–484. Springer-Verlag LNCS 576, 1992.

[41] L. Chen, K. Harrison, A. Moss, D. Soldera, and N.P. Smart. Certification of public keys within an identity based system. In A.H. Chan and V. Gligor, editors, *Proceedings of the 5th International Information Security Conference (ISC2002)*, pages 322–333. Springer-Verlag LNCS 2433, 2002.

[42] L. Chen and C. Kudla. Identity-based authenticated key agreement protocols from pairings. In *Proceedings of 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 219–233. IEEE Computer Society Press, 2003.

[43] L. Chen, H.W. Lim, and W. Mao. User-friendly grid security architecture and protocols. In *Proceedings of the 13th International Workshop on Security Protocols 2005*, to appear.

[44] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana, editors. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, May 2005. Available at http://www.w3.org/TR/2005/WD-wsdl20-20050510/, last accessed in November 2005.

[45] K. Chiu, M. Govindaraju, and R. Bramley. Investigating the limits of SOAP performance for scientific computing. In *Proceedings of 11th IEEE Symposium on High Performance Distributed Computing*, pages 246–254. IEEE Computer Society Press, 2002.

[46] D. Clark. Face-to-face with peer-to-peer networking. *IEEE Computer*, 34(1):18–21, January 2001.

[47] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363. Springer-Verlag LNCS 2260, 2001.

[48] C.R. Dalton. The NHS as a proving ground for cryptosystems. *Information Security Technical Report*, 8(3):73–88, 2003.

[49] Y. Desmedt and J. Quisquater. Public-key systems based on the difficulty of tampering. In A.M. Odlyzko, editor, *Advances in Cryptology - Proceedings of CRYPTO'86*, pages 111–117. Springer-Verlag LNCS 263, 1987.

[50] T. Dierks and C. Allen. The TLS protocol version 1.0. *The Internet Engineering Task Force (IETF)*, RFC 2246, January 1999.

[51] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[52] Y. Ding and P. Horster. Undetectable on-line password guessing attacks. *ACM Operating Systems Review*, 29(4):77–86, 1995.

[53] D. Eastlake, J.M. Reagle, and D. Solo. (Extensible Markup Language) XML-Signature syntax and processing. *The Internet Engineering Task Force (IETF)*, RFC 3275, March 2002.

[54] D. Eastlake and J.M. Reagle, editors. *XML Encryption Syntax and Processing*, December 2002. Available at http://www.w3.org/TR/xmlenc-core/, last accessed in November 2005.

[55] The Enabling Grids for E-SciencE Project. *EGEE*. Available at http://public.eu-egee.org/, last accessed in November 2005.

[56] The European DataGrid Project. *DataGrid*. Available at http://eu-datagrid.web.cern.ch/eu-datagrid/, last accessed in November 2005.

[57] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology - Proceedings of CRYPTO '86*, pages 186–194. Springer-Verlag LNCS 263, 1987.

[58] I. Foster. The Grid: A new infrastructure for 21st century science. *Physics Today*, 55(2):42–47, February 2002.

[59] I. Foster. The Grid: Computing without bounds. *Scientific American*, 288(4):78–85, April 2003.

[60] I. Foster, J. Geisler, W. Nickless, W. Smith, and S. Tuecke. Software infrastructure for the I-WAY high performance distributed computing experiment. In *Proceedings of 5th IEEE Symposium on High Performance Distributed Computing*, pages 562–571. IEEE Computer Society Press, 1997.

[61] I. Foster and A. Iamnitchi. On death, taxes, and the convergence of Peer-to-Peer and Grid computing. In F. Kaashoek and I. Stoica, editors, *Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pages 118–128. Springer-Verlag LNCS 2735, 2003.

[62] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputing Applications*, 11(2):115–128, 1997.

[63] I. Foster and C. Kesselman. Computational grids. In I. Foster and C. Kesselman, editors, *Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure*, pages 15–51, San Francisco, 1999. Morgan Kaufmann.

[64] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, 1999.

[65] I. Foster and C. Kesselman. The grid in a nutshell. In J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski, editors, *Chapter 1 of Grid Resource Management: State of the Art and Future Trends*, pages 3–13, Boston, 2003. Kluwer Academic.

[66] I. Foster and C. Kesselman. Concepts and architecture. In I. Foster and C. Kesselman, editors, *Chapter 4 of The Grid: Blueprint for a New Computing Infrastructure*, pages 37–63, San Francisco, 2004. Elsevier.

[67] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure.* Elsevier, San Francisco, 2004.

[68] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.* Open Grid Service Infrastructure Working Group, Global Grid Forum, June 2002.

[69] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational Grids. In *Proceedings of the 5th ACM Computer and Communications Security Conference*, pages 83–92. ACM Press, 1998.

[70] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[71] I. Foster, C. Kesselman, and S. Tuecke. The open grid services architecture. In I. Foster and C. Kesselman, editors, *Chapter 17 of The Grid: Blueprint for a New Computing Infrastructure*, pages 215–257, San Francisco, 2004. Elsevier.

[72] A.O. Freier, P. Karlton, and P.C. Kocher. *Internet Draft: The SSL Protocol Version 3.0.* The Internet Engineering Task Force (IETF), November 1996 (expired). Available at http://wp.netscape.com/eng/ssl3/draft302.txt, last accessed in November 2005.

[73] G. Frey, M. Müller, and H. Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, July 1999.

[74] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology - Proceedings of CRYPTO'99*, pages 537–554. Springer-Verlag LNCS 1666, 1999.

[75] S.D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2001*, pages 495–513. Springer-Verlag LNCS 2248, 2001.

[76] S.D. Galbraith. Pairings. In I.F. Blake, G. Seroussi, and N.P. Smart, editors, *Chapter 9 of Advances in Elliptic Curve Cryptography*, pages 183–213, Cambridge, 2005. Cambridge University Press, LMS 317.

[77] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D.R. Kohel, editors, *Proceedings of the 5th International Symposium on Algorithmic Number Theory (ANTS-V)*, pages 324–337. Springer-Verlag LNCS 2369, 2002.

[78] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancbek, and V.S. Sunderam. *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing.* MIT Press, Cambridge, MA, 1994.

[79] C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2003*, pages 272–293. Springer-Verlag LNCS 2656, 2003.

[80] C. Gentry and A. Silverberg. Hierarchical ID-Based cryptography. In Y. Zheng, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2002*, pages 548–566. Springer-Verlag LNCS 2501, 2002.

[81] The Globus Alliance. *Globus Toolkit.* Available at http://www.globus.org/toolkit/, last accessed in November 2005.

[82] The Globus Alliance. *GT 4.0 Security Features.* Available at http://www.globus.org/toolkit/docs/4.0/security/WS_AA_Features.html, last accessed in November 2005.

[83] The Globus Alliance. *GT 4.0 WS_GRAM.* Available at http://www.globus.org/toolkit/docs/4.0/execution/wsgram/, last accessed in November 2005.

[84] The Globus Alliance. *The WS-Resource Framework.* Available at http://www.globus.org/wsrf/, last accessed in November 2005.

[85] L. Gong. Optimal authentication protocols resistant to password guessing attacks. In *Proceedings of 8th IEEE Computer Security Foundations Workshop (CSFW'95)*, pages 24–29. IEEE Computer Society Press, 1995.

[86] 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*, 11(5):648–656, 1993.

[87] M. Govindaraju, A. Slominski, V. Choppella, R. Bramley, and D. Gannon. Requirements for and evaluation of RMI protocols for scientific computing. In *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (SC2000),CD-ROM*. ACM Press, November 2000.

[88] G. Graham, R. Cavanaugh, P. Couvares, A.D. Smet, and M. Livny. Distributed data analysis - federated computing for high-energy physics. In I. Foster and C. Kesselman, editors, *Chapter 10 of The Grid: Blueprint for a New Computing Infrastructure*, pages 135–145, San Francisco, 2004. Elsevier.

[89] GridCafé. *Grid Projects in the World*. Available at http://gridcafe.web.cern.ch/gridcafe/gridprojects/grid-tech.html, last accessed in November 2005.

[90] GRIDtoday. *Revolutionary Grid Offers Glimpse into Future*, September 2003. Available at http://www.gridtoday.com/03/0929/102012.html, last accessed in November 2005.

[91] A.S. Grimshaw, W.A. Wulf, and the Legion Team. The Legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, January 1997.

[92] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H.F. Nielsen. *Simple Object Access Protocol (SOAP) Version 1.2*, June 2003. Available at http://www.w3.org/TR/soap/, last accessed in November 2005.

[93] M. Gudgin and A. Nadalin, editors. *Web Services Secure Conversation Language (WS-SecureConversation) Version1.1*, February 2005. Available at http://www-106.ibm.com/developerworks/library/specification/ws-secon/, last accessed in November 2005.

[94] M. Gudgin and A. Nadalin, editors. *Web Services Trust Language (WS-Trust) Version 1.1*, February 2005. Available at http://www-106.ibm.com/developerworks/library/specification/ws-trust/, last accessed in November 2005.

[95] L.C. Guillou and J-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology - Proceedings of CRYPTO '88*, pages 216–231. Springer-Verlag LNCS 403, 1990.

[96] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and System Security*, 2(3):230–268, August 1999.

[97] P. Hallam-Baker and S.H. Mysore, editors. *XML Key Management Specification (XKMS 2.0)*, June 2005. Available at http://www.w3.org/TR/xkms2/, last accessed in November 2005.

[98] M.E. Hellman and S.C. Pohlig. *Exponentiation Cryptographic Apparatus and Method*. U.S. Patent #4,424,414, 3 January 1984 (expired).

[99] F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. Heys, editors, *Proceedings of the 9th International Workshop on Selected Areas in Cryptography (SAC 2002)*, pages 310–324. Springer-Verlag LNCS 2593, 2003.

[100] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In L.R. Knudsen, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2002*, pages 466–481. Springer-Verlag LNCS 2332, 2002.

[101] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *The Internet Engineering Task Force (IETF)*, RFC 3280, April 2002.

[102] X. Huang, L. Chen, L. Huang, and M. Li. An identity-based grid security infrastructure model. In R.H. Deng, F. Bao, H. Pang, and J. Zhou, editors, *Proceedings of the 1st International Conference on Information Security Practice and Experience (ISPEC 2005)*, pages 314–325. Springer-Verlag LNCS 3439, 2005.

[103] M. Humphrey and M.R. Thompson. Security implications of typical grid computing usage scenarios. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 2001)*, pages 95–103. IEEE Computer Society Press, August 2001.

[104] M. Humphrey, M.R. Thompson, and K.R. Jackson. Security for grids. *Proceedings of the IEEE*, 93(3):644–652, 2005.

[105] D.P. Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, October 1996.

[106] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of 4th Algorithmic Number Theory Symposium (ANTS-IV)*, pages 385–394. Springer-Verlag LNCS 1838, 2000.

[107] B. Kaliski. PKCS #10: Certification request syntax version 1.5. *The Internet Engineering Task Force (IETF)*, RFC 2314, March 1998.

[108] A. Khalili, J. Katz, and W.A. Arbaugh. Toward secure key distribution in truly ad-hoc networks. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03)*, pages 342–346. IEEE Computer Society Press, 2003.

[109] O. Kornievskaia, P. Honeyman, B. Doster, and K. Coffman. Kerberized credential translation: A solution to web access control. In *Proceedings of 10th USENIX Security Symposium*, pages 235–250, August 2001.

[110] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In J. Kilian, editor, *Advances in Cryptology - Proceedings of CRYPTO 2001*, pages 310–331. Springer-Verlag LNCS 2139, 2001.

[111] LHC Computing Grid Project. *LHC Computing Grid: Distributed Production Environment for Physics Data Processing.* Available at http://lcg.web.cern.ch/LCG/, last accessed in November 2005.

[112] B. Libert and J-J. Quisquater. *New Identity Based Signcryption Schemes from Pairings.* Cryptology ePrint Archive, Report 2003/023, February 2003. Available at http://eprint.iacr.org/2003/023.

[113] B. Libert and J-J. Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In F. Bao, R.H. Deng, and J. Zhou, editors, *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2004*, pages 187–200. Springer-Verlag LNCS 2947, 2004.

[114] J.C.R. Licklider and R.W. Taylor. The computer as a communication device. *Science and Technology*, April 1968. Reprint is available at http://memex.org/licklider.pdf, last accessed in November 2005.

[115] H.W. Lim and K.G. Paterson. Identity-based cryptography for grid security. In H. Stockinger, R. Buyya, and R. Perrott, editors, *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005)*, pages 395–404. IEEE Computer Society Press, 2005.

[116] H.W. Lim and K.G. Paterson. Secret public key protocols revisited. In *Proceedings of the 14th International Workshop on Security Protocols 2006*, to appear.

[117] H.W. Lim and M.J.B. Robshaw. On identity-based cryptography and Grid computing. In M. Bubak, G.D.v. Albada, P.M.A. Sloot, and J.J. Dongarra, editors, *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, pages 474–477. Springer-Verlag LNCS 3036, 2004.

[118] H.W. Lim and M.J.B. Robshaw. A dynamic key infrastructure for Grid. In P.M.A. Sloot, A.G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Proceedings of the European Grid Conference (EGC 2005)*, pages 255–264. Springer-Verlag LNCS 3470, 2005.

[119] J. Linn. Generic security service application program interface version 2, update1. *The Internet Engineering Task Force (IETF)*, RFC 2743, January 2000.

[120] T.M.A. Lomas, L. Gong, J.H. Saltzer, and R.M. Needham. Reducing risks from poorly chosen keys. *ACM Operating Systems Review*, 23(5):14–18, 1989.

[121] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - Proceedings of EUROCRYPT 2004*, pages 74–90. Springer-Verlag LNCS 3027, 2004.

[122] J. Malone-Lee. *Identity-Based Signcryption.* Cryptology ePrint Archive, Report 2002/098, July 2002. Available at http://eprint.iacr.org/2002/098.

[123] W. Mao. *An Identity-based Non-interactive Authentication Framework for Computational Grids.* HP Lab, Technical Report HPL-2004-96, June 2004. Available at http://www.hpl.hp.com/techreports/2004/HPL-2004-96.pdf.

[124] U.M. Maurer and Y. Yacobi. A non-interactive public-key distribution system. *Designs, Codes, and Cryptography*, 9(3):305–316, 1996.

[125] N. McCullagh. Securing e-mail with identity-based encryption. *IT Professional*, 7(3):61–64, May/June 2005.

[126] S. Meder, V. Welch, S. Tuecke, and D. Engert. *GSS-API Extensions.* Global Grid Forum (GGF) Grid Security Infrastructure Working Group, June 2004. Available at http://www.ggf.org/documents/GFD.24.pdf, last accessed in November 2005.

[127] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Florida, 1997.

[128] P.C. Moore, W.R. Johnson, and R.J. Detry. Adapting Globus and Kerberos for a secure ASCI Grid. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (SC2001), CD-ROM*, page 21. ACM Press, November 2001.

[129] T. Moses, editor. *eXtensible Access Control Markup Language (XACML) 2.0.* OASIS Standard 200502, February 2005.

[130] MPI Forum. MPI: A message-passing interface standard. *International Journal of Supercomputer Applications*, 8(3-4):165–414, 1994.

[131] MPI Forum. MPI2: A message-passing interface standard. *International Journal of High Performance Computing Applications*, 12(1-2):1–299, 1998.

[132] A. Nadalin, C. Kaler, P. Hallam-Baker, and R. Monzillo, editors. *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004).* OASIS Standard 200401, March 2004.

[133] The National e-Science Center. *National e-Science.* Available at http://www.nesc.ac.uk/, last accessed in November 2005.

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

[135] B.C. Neuman. Proxy-based authorization and accounting for distributed systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[136] B.C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.

[137] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the Grid: MyProxy. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 2001)*, pages 104 –111. IEEE Computer Society Press, August 2001.

[138] Object Management Group. *CORBA/IIOP Specification.* Available at http://www.omg.org/technology/documents/formal/corba_iiop.htm, last accessed in November 2005.

[139] E. Okamoto. Key distribution systems based on identification information. In C. Pomerance, editor, *Advances in Cryptology - Proceedings of CRYPTO'87*, pages 194–202. Springer-Verlag LNCS 293, 1988.

[140] The OpenSSL Project. *OpenSSL: The Open Source Toolkit for SSL/TLS*, 2005. Available at http://www.openssl.org/, last accessed in November 2005.

[141] S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 236–247. IEEE Computer Society Press, 1997.

[142] K.G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[143] K.G. Paterson. Cryptography from pairings. In I.F. Blake, G. Seroussi, and N.P. Smart, editors, *Chapter 10 of Advances in Elliptic Curve Cryptography*, pages 215–251, Cambridge, 2005. Cambridge University Press, LMS 317.

[144] K.G. Paterson and G. Price. A comparison between traditional public key infrastructures and identity-based cryptography. *Information Security Technical Report*, 8(3):57–72, 2003.

[145] L.C. Paulson. Inductive analysis of the Internet protocol TLS. *ACM Transactions on Information and System Security*, 2(3):332–351, August 1999.

[146] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 50–59. IEEE Computer Society Press, June 2002.

[147] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. The community authorization service: Status and future. In *Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), eConf*, March 2003.

[148] G. Price and C.J. Mitchell. Interoperation between a conventional PKI and an ID-based infrastructure. In D. Chadwick and G. Zhao, editors, *Proceedings of the 2nd European Public Key Infrastructure Workshop (EuroPKI 2005)*, pages 73–85. Springer-Verlag LNCS 3545, 2005.

[149] A. Rajasekar and R. Moore. Data and metadata collections for scientific applications. In L.O. Hertzberger, A.G. Hoekstra, and R. Williams, editors, *Proceedings of the 9th International Conference on High-Performance Computing and Networking*, pages 72–80. Springer-Verlag LNCS 2110, 2001.

[150] B. Ramsdell, editor. S/MIME version 3 message specification. *The Internet Engineering Task Force (IETF)*, RFC 2633, June 1999.

[151] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[152] J. Rosenberg and D. Remy. *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption.* Sams, Indiana, 2004.

[153] D.D. Roure, M.A. Baker, N.R. Jennings, and N.R. Shadbolt. *Grid Computing: Making the Global Infrastructure a Reality*, chapter 3: The Evolution of the Grid, pages 65–100. John Wiley and Sons, West Sussex, 2003.

[154] RSA Security. *How fast is the RSA algorithm?*, 2004. Available at http://www.rsasecurity.com/rsalabs/node.asp?id=2215, last accessed in November 2005.

[155] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the 2000 Symposium on Cryptography and Information Security (SCIS 2000)*, January 2000.

[156] J. Schlimmer, editor. *Web Services Policy Framework (WS-Security Policy)*, September 2004. Available at http://www-128.ibm.com/developerworks/webservices/library/specification/ws-polfram/, last accessed in August 2005.

[157] M. Scott. Computing the Tate pairing. In A. Menezes, editor, *Proceedings of the RSA Conference: Topics in Cryptology - the Cryptographers' Track (CT-RSA 2005)*, pages 293–304. Springer-Verlag LNCS 3376, 2005.

[158] M. Scott and P.S.L.M. Barreto. Compressed pairings. In M. Franklin, editor, *Advances in Cryptology - Proceedings of CRYPTO 2004*, pages 140–156. Springer-Verlag LNCS 3152, 2004.

[159] A. Shamir. Identity-based cryptosystems and signature schemes. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology - Proceedings of CRYPTO'84*, pages 47–53. Springer-Verlag LNCS 196, 1985.

[160] Shamus Software Ltd. *MIRACL*. Available at http://indigo.ie/∼mscott/, last accessed in November 2005.

[161] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon. Performance comparison of security mechanisms for grid services. In *Proceedings of 5th IEEE/ACM International Workshop on Grid Computing (GRID2004)*, pages 360–364. IEEE Computer Society Press, 2004.

[162] F. Siebenlist, N. Nagaratnam, V. Welch, and C. Neuman. Security for virtual organizations - federating trust and policy domains. In I. Foster and C. Kesselman, editors, *Chapter 21 of The Grid: Blueprint for a New Computing Infrastructure*, pages 353–387, San Francisco, 2004. Elsevier.

[163] N.P. Smart. An identity-based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.

[164] D.K. Smetters and G. Durfee. Domain-based administration of identity-based cryptosystems for secure email and IPSEC. In *Proceedings of 12th USENIX Security Symposium*, pages 215–229, August 2003.

[165] B. Sotomayor. *The Globus Toolkit 3 Programmer's Tutorial*, 2004. Available at http://gdp.globus.org/gt3-tutorial/, last accessed in November 2005.

[166] T. Stading. Secure communication in a distributed system using identity based encryption. In *Proceedings of 3rd IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2003)*, pages 414–420. IEEE Computer Society Press, May 2003.

[167] Stanford University. *IBE Secure Email.* Available at http://crypto.stanford.edu/ibe/, last accessed in November 2005.

[168] M. Steiner, P. Buhler, T. Eirich, and M. Waidner. Secure password-based cipher suite for TLS. *ACM Transactions on Information and System Security*, 4(2):134–157, May 2001.

[169] D.R. Stinson. *Cryptography: Theory and Practice.* Chapman & Hall/CRC, Florida, 2002.

[170] I. Stoica, R. Morris, D.R. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 149–160. ACM Press, 2001.

[171] H. Tanaka. A realization scheme for the identity-based cryptosystem. In C. Pomerance, editor, *Advances in Cryptology - Proceedings of CRYPTO'87*, pages 340–349. Springer-Verlag LNCS 293, 1988.

[172] The TeraGrid Project. *TeraGrid.* Available at http://www.teragrid.org/, last accessed in November 2005.

[173] M.R. Thompson and K.R. Jackson. Security issues of grid resource management. In J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski, editors, *Chapter 5 of Grid Resource Management: State of the Art and Future Trends*, pages 53–69, Boston, 2003. Kluwer Academic.

[174] G. Tsudik and E.v. Herreweghen. Some remarks on protecting weak keys and poorly chosen secrets from guessing attacks. In *Proceedings of the 12th IEEE Symposium on Reliable Distributed Systems (SRDS'93)*, pages 136–141. IEEE Computer Society Press, 1993.

[175] S. Tsuji and T. Itoh. An ID-based cryptosystem based on the discrete logarithm problem. *IEEE Journal on Selected Areas in Communications*, 7(4):467–473, 1989.

[176] S. Tuecke, V. Welch, D. Engert, L. Pearman, and M. Thompson. Internet X.509 public key infrastructure proxy certificate profile. *The Internet Engineering Task Force (IETF)*, RFC 3820, June 2004.

[177] University of Wisconsin-Madison. *Condor Project.* Available at http://www.cs.wisc.edu/condor/, last accessed in November 2005.

[178] S.A. Vanstone and R.J. Zuccherato. Elliptic curve cryptosystems using curves of smooth order over the ring z_n. *IEEE Transactions on Information Theory*, 43(4):1231–1237, July 1997.

[179] S. Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In L.R. Knudsen, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2002*, pages 534–546. Springer-Verlag LNCS 2332, 2002.

[180] Voltage Security. *The Voltage IBE Toolkit Overview.* Available at http://www.voltage.com/ibe_dev/about_ibe/overview.htm, last accessed in November 2005.

[181] Voltage Security. *Email Security – The IBE Advantage*, white paper, July 2004. Available at http://www.voltage.com/whitepaper/index.htm, last accessed in November 2005.

[182] Voltage Security. *Voltage Security Platform Architecture*, white paper, June 2004. Available at http://www.voltage.com/whitepaper/index.htm, last accessed in November 2005.

[183] V.A. Vyssotsky, F.J. Corbató, and R.M. Graham. Structure of the Multics supervisor. In *Proceedings of AFIPS Fall Joint Computer Conference*, pages 203–212. Spartan Books, 1965.

[184] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *Proceedings of 2nd USENIX Workshop on Electronic Commerce*, pages 29–40, November 1996.

[185] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). *The Internet Engineering Task Force (IETF)*, RFC 2251, December 1997.

[186] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 proxy certificates for dynamic delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*, pages 42–58, 2004.

[187] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 2003)*, pages 48–61. IEEE Computer Society Press, June 2003.

[188] V. Welch, editor. *Globus Toolkit version 4 Grid Security Infrastructure: A Standards Perspective.* The Globus Security Team, September 2005. Available at http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf, last accessed in November 2005.

[189] H. Yoon, J.H. Cheon, and Y. Kim. Batch verifications with ID-based signatures. In C. Park and S. Chee, editors, *Proceedings of the 7th International Conference on Information Security and Cryptology (ICISC 2004)*, pages 233–248. Springer-Verlag LNCS 3506, 2005.

[190] F. Zhang, W. Susilo, and Y. Mu. Identity-based partial message recovery signatures (or how to shorten ID-based signatures). In A.S. Patrick and M. Yung, editors, *Proceedings of the 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, pages 45–56. Springer-Verlag LNCS 3570, 2005.

[191] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In B.S. Kaliski Jr., editor, *Advances in Cryptology - Proceedings of CRYPTO'97*, pages 165–179. Springer-Verlag LNCS 1294, 1997.