Identity-Based Cryptography for Grid Security

Hoon Wei Lim and Kenneth G. Paterson
Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
{h.lim, kenny.paterson}@rhul.ac.uk

Abstract

The majority of current security architectures for grid systems use public key infrastructure (PKI) to authenticate identities of grid members and to secure resource allocation to these members. Identity-based cryptography (IBC) has some attractive properties which seem to align well with the demands of grid computing. This paper 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 Grid Security Infrastructure (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.

1 Introduction

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 [9, 11] has been proposed as a mechanism to provide 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 which 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.

Public key infrastructure (PKI) 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 Grid Security Infrastructure (GSI) [10] for the Globus Toolkit (GT) [8], the leading toolkit used in developing grid applications, proxy certificates [27] have been designed and deployed in addition to standard X.509 public key certificate [16], to compensate some of the shortcomings in the conventional PKI setting and to provide additional properties that align with the requirements for secure communications among grid entities within a dynamically changing environment. The motivations for the proxy certificates which carry short-term public keys 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 within a dynamic grid environment offers the best possible solution for public key management.

Independent of grid computing, a variant of traditional public key technologies called identity-based cryptography (IBC) [2, 25] 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 greater 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 traditional PKI does. The aim of this paper is to propose and investigate an identity-based grid security architecture that aligns with the security services provided by the GSI adopted in the GT version 4 (GT4). Our proposal make uses of both long-term and short-term identity-based keys, by exploiting some properties from hierarchical identity-based cryptography (HIBC) [14, 15]. The proposal incorporates a certificate-free key agreement protocol based on the TLS handshake [5] and a lightweight delegation protocol.

Related Work. The idea of applying IBC within a grid security infrastructure was initially explored in [17, 20]. However, the supposedly dynamic use of identity-based keys has been hindered by some traditional limitations of IBC such as key escrow and the need to distribute private keys through secure channels. In addition, the proposals in [17, 20] do not address some of the essential security requirements desired in the GT such as using proxy credentials for single sign-on and delegation. In [18], Lim and Robshaw proposed a hybrid approach combining 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 Trusted Authorities for the purposes of delegation and single sign-on. This framework solves the two issues of key escrow and distribution of private keys in IBC, but has the limitation that the original dynamic and lightweight qualities that IBC offers are partially lost, because users now need to authenticate and verify other parties' parameter sets before they can be used.

Recently, Chen *et al.* [3] revisited the GSI in the GT version 2 (GT2) and presented some improvements to the security architecture. Their work is related to [18] in which each user has a static long-term credential which can be used by other parties to derive dynamic public keys on-the-fly. Chen *et al.* modified the security protocols in [10] and the improved protocols seem to offer better performance. In addition, they also proposed an interesting application of aggregate signature to save computational costs in verifying chained signatures. As with [18], however, each user is required to get hold of the intended communicating party's authentic certificate before a dynamic public key can be computed and used.

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. Here we examine what can be achieved in a fully identity-based approach. Our contributions can be summarised as follows.

- We propose a fully identity-based key infrastructure for grid (IKIG). It inherits attractive properties from IBC such as being certificate-free and having small key sizes. This potentially offers a more lightweight key management approach.
- We present a customised identity-based authenticated key agreement protocol for grid environments. This protocol can support single sign-on and credential delegation and has a restricted escrow-free[1] property. We build our protocol using some properties from HIBC and the protocol is integrated with the widely used TLS protocol so that it can be implemented in existing grid infrastructures.
- We design a lightweight one-pass delegation protocol that supports short-term identity-based keys. This protocol does not require transmission of a short-term public key from a delegation target to a delegator, yet the public key can be bound to the delegator's credential.
- We identify and discuss the performance trade-offs in terms of computational and communication overheads of IKIG as compared to the GSI.

[1] A malicious trusted authority can always impersonate a user to other users. We define restricted escrow-free protocol as a protocol which resists passive but not active attacks from the trusted authority.

Paper Outline. The remainder of this paper is organised as follows. Section 2 explains some fundamental concepts of identity-based cryptography. In Section 3, we present in-depth identity-based key management and usage methods for the GSI. Section 4 discusses the performance analyses of our proposal. In Section 5, we consider the impact of IBC on web services security and some integration issues with existing technologies. Lastly, Section 6 presents our concluding remarks.

2 Identity-Based Cryptography

Identity-based cryptography (IBC) was first introduced by Shamir [25] in 1984. Recently, there has been an increased intensity in research on IBC. This was mainly due to the seminal discovery of a practical and secure identity-based encryption (IBE) scheme which uses pairings over elliptic curves by Boneh and Franklin [2] in 2001. Following the publication of [2], many proposals for identity-based schemes related to signatures, signcryption, key agreement, and so on have been proposed.

The main stimulus for this trend is the perceived problem of managing certificates and associated keys within PKI. In an identity-based cryptosystem, public keys can be derived from arbitrary strings, e.g. a user's identity, whilst the corresponding private keys are generated and distributed by a Trusted Authority (TA) in possession of a system master secret. This TA roughly corresponds to the Certificate Authority/Registration Authority (CA/RA) combination in a traditional certificated-based PKI. An identity-based cryptosystem enjoys most of the functionality of public key cryptography without the need for certificates and the problems that these bring. Thus it promises a more lightweight approach to deploying public key cryptography. A comparison of the certificate-based PKI and the identity-based PKI can be found in [23]. In the remainder of this section, we recap some basic concepts of pairings and outline the Gentry-

Silverberg hierarchical identity-based encryption and signature schemes (HIBE and HIBS) [14] which we will apply in subsequent sections.

2.1 Overview of Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q for some large prime q, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ denotes a related multiplicative group. A pairing, which can be either a modified Weil pairing [2] or a Tate pairing [12] is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties.

- *Bilinear*: Given $P, Q, R \in \mathbb{G}_1$, we have $\hat{e}(P, Q+R) = \hat{e}(P, Q)\cdot\hat{e}(P, R)$ 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 a $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1$.
- *Computable*: If $P, Q \in \mathbb{G}_1$, $\hat{e}(P, Q)$ can be efficiently computed.

For any $a \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, we write aP as the scalar multiplication of group element P by integer a. Typically, $\mathbb{G}_1$ is obtained as a subgroup of the group of points on a suitable elliptic curve over a finite field, and $\mathbb{G}_2$ is obtained from a related finite field.

2.2 The Gentry-Silverberg HIBE & HIBS

Gentry and Silverberg proposed hierarchical identity-based cryptography (HIBC) in [14] to ease the private key distribution problem and improve the scalability of the original IBE scheme proposed in [2]. In the HIBC setting, a root Private Key Generator (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 identity-based encryption (HIBE)[2] and signature (HIBS) schemes as follows.

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 key 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.

[2]We only present a basic HIBE scheme. See [14] for the full and appropriately secure HIBE scheme.

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 $(1 \leq i \leq t)$, 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 $\sum_{i=1}^{t} s_{i-1} P_i$, and defines Q-values as $Q_i = s_i P_0$ for $1 \leq i \leq t-1$.

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 $r \in \mathbb{Z}_q^*$; and the ciphertext is set to:

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

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

DECRYPT**:** Given a ciphertext $\langle U_0, U_2, \ldots, U_t, V \rangle$ 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 $S_t + s_t h$. The algorithm outputs $\langle \sigma = S_t + s_t h, Q_1, \ldots, Q_t \rangle$ as the signature.

VERIFY**:** Given a signature $\langle \sigma, Q_1, \ldots, Q_t \rangle$ of a message m signed by an entity who has a public key $P_t = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_t) \in \mathbb{G}_1$, 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, $\hat{e}(Q_0, P_1)$ and $\prod_{i=2}^{t} \hat{e}(Q_{i-1}, P_i)$ can be pre-computed. That means there will be only two pairing computations when the verification is performed.

It is worth noting that other HIBE and HIBS schemes are available. We have selected the schemes from [14] because they are efficient and their security is based on reasonable computational assumptions.

3 Identity-Based Key Infrastructure for Grid

We now apply both the HIBE and the HIBS schemes in the context of the GSI and examine how identity-based keys can be used and managed in different ways from conventional public key management.

3.1 Key Generation

In our identity-based key infrastructure for grid (IKIG), we propose to replace the CA in the current certificate-based grid systems with a TA. The TA's roles including acting as the PKG and supporting other user related administration and management. It is a common proposal (see for example the European DataGrid (EDG) project[3]) that each nation has one CA to serve members within the country. If IKIG is to be deployed at such a scale, we expect that the root TA's system parameters will be bootstrapped in the grid system, while the other TAs' public information can be downloaded by the users through their grid clients.

During the initial system setup phase, each TA runs a parameter generator on input k to generate groups $\mathbb{G}_1$, $\mathbb{G}_2$ of large prime order q and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where k is a security parameter. It then performs the ROOT SETUP described in Section 2.2 to produce a master secret key s_0 and its system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3 \rangle$.

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 certificate mainly due to its frequent use for job submissions and also because of the computationally intensive RSA key pair generation [28]. 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 as 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 finite field $\mathbb{F}_{2^{271}}$ [13]. 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 very efficiently [1, 24].

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

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

   `/C=UK/O=eScience/OU=RHUL/CN=Alice/Y=2005`

   (using the syntax from [16]).

2. The RA 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. The long-term credential for A and her TA's system parameters which have been signed by the root TA are distributed to her through a temporary storage medium such as a floppy disk.

3. A performs the LOWER-LEVEL SETUP algorithm to pick a random $s_A \in \mathbb{Z}_q^*$ which she will keep secret. She then defines her Q-values as $(Q_0 = s_0 P_0, Q_A = s_A P_0)$.

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

1. A runs the EXTRACT algorithm to generate a short-term private key $S_{\bar{A}} = s_0 P_A + s_A P_{\bar{A}}$, where $P_{\bar{A}} = H_1(\mathrm{ID}_A \| \mathrm{LT}_A)$ is the corresponding short-term public key. LT_A denotes the lifetime of A's key in some fixed format.

2. She also performs the LOWER-LEVEL SETUP algorithm, picking a random $s_{\bar{A}} \in \mathbb{Z}_q^*$ which will be kept secret. The Q-values for A's proxy are $(Q_0 = s_0 P_0, Q_A = s_A P_0, Q_{\bar{A}} = s_{\bar{A}} P_0)$. 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.

Only then is A ready and allowed to submit job requests. We remark that in our approach, the keys are allocated as they would be in HIBE/HIBS schemes with the TA at level 0, the user at level 1 and the user's proxy at level 2.

3.2 Key Usage

Consider a simple scenario where A wants to submit a job request `J_req` to a target resource X with identity-based key sets as shown in Table 1.

Table 1. A's long-term and proxy credentials.

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

We now illustrate how the grid security services are performed via IKIG to secure a job invocation process.

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

[3] The European DataGrid Project, `http://www.edg.org/`

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 when challenged by other entities during execution of key agreement and delegation protocols.

Authorization A generates an identity-based signature using the Gentry-Silverberg IBS scheme on `J_req` with her short-term private key $S_{\bar{A}}$. She then submits the signed request to a Managed Job Factory (MJF) that resides on X through a Grid Resource Allocation and Management (GRAM) service.

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

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

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

Subsequently, the MJF maps A's identifier to its grid-map file before granting access to A. When the check succeeds, the MJF instanstiates 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 which 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 and used many times for other purposes such as key agreement and delegation. Thus, our approach can achieve some real-time computation savings.

Mutual Authentication and Key Agreement Before the job can be started, A must perform mutual authentication with the created managed job service. We present an identity-based authenticated key agreement protocol based on the TLS handshake protocol [5]. Protocol 1 shows our key agreement protocol which uses the Gentry-Silverberg HIBE and HIBS schemes for encryption and signing operations, respectively.

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

(1) $A \rightarrow X$: `ClientHello`
(2) $X \rightarrow A$: `ServerHello`
`ServerIdentifier`
`ServerHelloDone`
(3) $A \rightarrow X$: `ClientIdentifier`
`ClientKeyExchange`
`IdentityVerify`
`ClientFinished`
(4) $X \rightarrow A$: `ServerFinished`

Protocol 1 is analogous to the TLS protocol which uses the RSA key exchange algorithm as specified in [5]. As with the current TLS specification, Protocol 1 begins with A (the client) sending X (the server) a `ClientHello` message. The message contains a fresh random number, n_A, a session identifier, and a cipher specification extended from TLS version 1.0 to handle the HIBE and HIBS schemes, e.g. `TLS_HIBC_WITH_DES_CBC_SHA` which defines a HIBC-supported cipher specification that uses DES-CBC and SHA as the symmetric encryption algorithm and hash function, respectively. It also includes information such as an elliptic curve selection and other security parameters that we have discussed in Section 3.1.

X responds with a `ServerHello` message which contains a new random number, n_X, a session identifier, and its supported ciphersuite. X also forwards `ServerIdentifier` $= (\mathrm{ID}_X, \mathrm{LT}_X)$ to A. The `ServerHelloDone` message is sent to indicate the end of step (2).

In step (3), A first forwards her short-term public key information, e.g. $(\mathrm{ID}_A, \mathrm{LT}_A)$, in `ClientIdentifier` to X. She then chooses a pre-master secret and encrypts it with X's short-term public key using the HIBE scheme, where 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 signed `handshake_messages` is generated for `IdentityVerify` to provide explicit verification of A's identifier. 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.

As with the `ClientFinished`, X should compute the exact verification value as A in

ServerFinished. A master secret K_{AX}, the shared secret key between A and X which will eventually be used for protecting application data can be calculated as:

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

where PRF is a pseudo-random function.

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 certificate-based TLS protocol.

Delegation A may, at her discretion, delegate her credential to X for later use when necessary. Currently, the GSI delegation protocol [27, 28] requires a round-trip between a delegator and a delegation target. We propose a one-pass delegation protocol which works in the same way as the 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 predict and compute the server's short-term public key and vice versa. With the same reasoning, the delegator can straightforwardly predict and 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 $\texttt{DelegationToken}_X = (\text{ID}_A, \text{ID}_X, \texttt{J\_req}, \texttt{Policy}, \text{LT}_{AX})$, where LT_{AX} is a lifetime which A imposes on X. Protocol 2 then shows our one-pass delegation protocol.

Protocol 2 Credential delegation

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

Note that A can naturally bind X's short-term public key $P_{\bar{X}} = H_1(\text{ID}_X \| \text{LT}_{AX})$ to the delegation token without acquiring the key from X. The signed delegation token is of the form of $\langle \sigma, Q_A, Q_{\bar{A}} \rangle$, where $\sigma = S_{\bar{A}} + s_{\bar{A}} h$, $h = H_3(\text{ID}_A \| \text{LT}_{AX} \| \texttt{DelegationToken}_X)$, $Q_A = s_A P_0$, and $Q_{\bar{A}} = s_{\bar{A}} P_0$.

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 $P_{\bar{X}}$, normally via a TLS handshake.

3.3 Key Update

We expect that the user's long-term public key is fixed as $P_A = H_1(\text{ID}_A)$, where its 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 a 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 $s_0 P'_A$ (we have discussed in Section 3.1 that the user's identifier includes the year, 'Y' field) directly to the user through a secure channel which can be established via Protocol 1. 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. This once-a-year key update method seems to be secure since Protocol 1 achieves forward-secrecy as with the standard TLS protocol. That means compromise of the user's long-term private key does not allow the adversary to recover any past session keys. 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 long-term 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 the 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 due to an unforeseen longer than 12 hours job execution, the user needs to be notified before the credential expires so that it can be renewed by the user.

3.4 Key Revocation

For key revocation in the GSI, the user is expected to check a certificate revocation list (CRL) stored in a trusted directory or the CA's web site periodically depending on the policy enforced by his local administrator. 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. In the IBC setting, a

number of revocation mechanisms are possible. We could use a more fine-grained identifier [2] such as extending the user's identifier to another level which specifies a month, 'M' field such as "/C/O/OU/CN/Y/M". This allows automated expiry of public keys after one month (hence window of exposure is also limited to a month). However, should this approach prove insufficient, e.g. in some high security applications, then existing PKI revocation mechanisms can easily be adapted to the IBC setting. See [23] for a longer discussion of key revocation in identity-based and certificate-based PKIs.

Revocation of short-term keys is a minor concern here as these keys will be destroyed upon expiry of their validity periods.

4 Performance Analysis

We have seen how conventional public key usage and management can be replaced by identity-based techniques. Here we discuss the performance trade-offs between the GSI and our proposed IKIG.

Communication Costs. In terms of communication overhead, we summarise the performance trade-offs between the two different settings in Table 2. Note that we only consider the dominant communication costs between the job requestor and the resource, i.e. signed or encrypted messages, which may have the biggest contributions to the network bandwidth. 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 (12 kilobits), and 0.8 kilobytes (6.4 kilobits) for a proxy certificate. Here, we ignore small fields in a certificate such as issuer, subject, and validity period. An encrypted or a signed message with a short-term RSA key is 512 bits in size assuming the length of the message is smaller than the 512-bit block size. On the other hand, the size of a ciphertext and a signature produced by the HIBE and HIBS schemes of [14] in our IKIG setting are 1056 bits and 816 bits, respectively.

Table 2. Performance trade-offs in communication costs (in kilobits) between the GSI and IKIG.

Operation	GSI	IKIG
Key agreement	37.8	1.9
Delegation	7.4	0.8

From Table 2, the communication cost for the TLS protocol in the GSI is estimated to be $2(12)+2(6.4)+2(0.512) = 37.8$ kilobits, since there are two public key certificates, two proxy certificate, one encrypted pre-master secret (for `ClientKeyExchange`), and one signed message (for `CertificateVerify`) being transmitted over the network. 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. This clearly shows that an execution of the TLS protocol in the GSI is significantly more costly than our Protocol 1 in terms of bandwidth requirements. We remark that IKIG uses short-term key sizes which offer roughly double the security level of 512 bit RSA keys, an additional benefit of our approach. The communication overhead for delegation between the delegator and the delegation target can be estimated straightforwardly from our discussion on delegation protocols in Section 3.2.

We have shown that, in general, the resulting communication overheads from a job submission through IKIG is significantly lower than the 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.

Computational Costs. We break down the computational complexity in both the GSI and our proposed IKIG in Table 3. Note that hashing is not a dominant operation and only adds minimal overheads to the overall computational costs. Thus it is excluded from the table. We assume that small public exponent is used for RSA encryption and precomputation of pairing values is possible. Also, for simplicity, we limit the length of the delegation chain to one.

From Table 3, we can see that key generation and delegation in the GSI are considerably slower than IKIG because of the dominant computational costs in generating fresh RSA key parameters. Significant computational savings can be achieved by the resource provider in the IKIG setting when it is needed to create n distinct proxy credentials (used by the managed job services) for performing n mutual authentications with n different users at the same time. Nevertheless, it seems that Protocol 1 is slower than the actual TLS protocol mainly because of the pairing computations. Fortunately in our proposed IKIG, the computationally intensive pairing computations are performed at the high-performance resource's side. Operations or computations at the user's side appear to be lightweight, e.g. a HIBE encryption requires two MULs and a HIBS signing needs one ADD and one MUL. We note that RSA and HIBE/HIBS schemes are very different from each other in terms of their mathematical properties and in some ways cannot be fairly compared. Therefore, we believe that it is fair to conclude that the overall computational costs for the IKIG setting is comparable to the GSI. Recent results (see for example [24]) 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

Table 3. Performance trade-offs in computational costs between the GSI and the IKIG settings in terms of cryptographic operations performed by the job requestor and the resource provider.

Operation	GSI	IKIG	Remark
Key generation			
(a.) Long-term	1 GEN	1 MUL	RSA key generation is significantly slower than ADD/MUL.
(b.) Short-term	1 GEN	1 ADD, 1 MUL	
Key agreement			
(a.) Requestor	3 ENCs, 1 DEC	3 MULs, 1 ADD	A MUL and an ADD are a few times faster than a DEC [1].
(b.) Resource	3 ENCs, 1 DEC	4 PAIs	An ENC is many times faster than a PAI depending on the key size [1].
Delegation			
(a.) Delegator	1 ENC, 1 DEC	1 MUL	The speed of a MUL is comparable to an ENC depending on the size of the RSA exponent.
(b.) Delegation target	1 GEN, 1 DEC		
(c.) Verifier	3 ENCs	2 PAIs	

GEN = RSA parameter generation
ENC = RSA encryption (modular exponentiation)
DEC = RSA decryption (modular exponentiation)
ADD = Elliptic curve point addition
MUL = Elliptic curve point multiplication
PAI = Pairing computation

the near future.

5 Discussion

In this section, we look into some practical issues surrounding the use of our IKIG approach.

5.1 Impact on Web Services Security

In the 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 which is clearly shown in [26]. 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 [4]. This can substantially increase the communication costs, the latency of sending/receiving SOAP messages, and the time needed for parsing the XML data.

By looking at using IBC for supporting the GSI as an alternative approach to the recommended RSA public key encryption and signature schemes, we envisage that the sizes of SOAP headers may well be reduced substantially. This is especially true considering the fact that two types of certificates are used within a grid environment. Moreover, an identity-based public key is predictable, self-describing, and human-readable. Thus no tools are needed to parse and render the key information. This is indeed a very desirable property and it matches a fundamental objective of XML.

5.2 Implementation Issues

Here, we identify some implementation issues that need to be addressed in IKIG. The features or functions available from current technologies and mechanisms used to support and implement the GSI need to be modified and extended to support identity-based techniques.

GSS-API. Currently, the GSI is built on top of Generic Security Service Application Program Interface (GSS-API) [19] that allows security services to be easily added into grid applications through a set of callers in a generic fashion [10]. GSS-API, which is both transport (communication protocols) and mechanism (cryptographic schemes) 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 GSI requirements, such as credential extensions handling and delegation at any time [21].

To implement IKIG, we envisage that a couple of additional functions may need to be included in GSS-API to support the proposed Protocols 1 and 2. One is 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 2. 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.

OpenSSL. The authentication library of the GSI is based on OpenSSL [22] with some modification for supporting the use of X.509 proxy certificates. This library, in turn, make uses of GSS-API (and some extensions) to perform mutual authentication and key agreement between two parties complying with the standard TLS handshake protocol of [5].

In the IKIG setting, we must define and include the `TLS_HIBC_WITH_DES_CBC_SHA` cipher suite that we proposed for Protocol 1 to the current TLS specification. The IKIG-enabled OpenSSL 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 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 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 [6, 7].

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 paramaters for key generation, system parameters, group and element sizes, key values, and so on.

6 Conclusions

The development of grid computing and identity-based cryptography are amongst today's most important technical problems in the field of computer science and cryptology. We have discussed at length how identity-based techniques can replace conventional PKI and be used to offer an alternative security infrastructure for grid. 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.

Acknowledgements

The authors would like to thank Steven Galbraith for helpful discussions about parameters selection in IKIG and anonymous referees for their useful feedback.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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 June 2005.

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In L. Knudsen, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2002*, pages 466–481. Springer-Verlag LNCS 2332, 2002.

[16] 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.

[17] H.W. Lim and M.J.B. Robshaw. On identity-based cryptography and GRID computing. In M. Bubak, G. Albada, P. Sloot, and J. Dongarra, editors, *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, pages 474–477. Springer-Verlag LNCS 3036, 2004.

[18] H.W. Lim and M.J.B. Robshaw. A dynamic key infrastructure for GRID. In P. Sloot, A. 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.

[19] J. Linn. Generic security service application program interface version 2, update 1. *The Internet Engineering Task Force (IETF)*, RFC 2743, January 2000.

[20] 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.

[21] 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 September 2005.

[22] The OpenSSL Project. *OpenSSL: The Open Source Toolkit for SSL/TLS*, 2005. Available at http://www.openssl.org/, last accessed in September 2005.

[23] 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.

[24] 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.

[25] A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editors, *Advances in Cryptology - Proceedings of CRYPTO'84*, pages 47–53. Springer-Verlag LNCS 196, 1985.

[26] 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 (GRID 2004)*, pages 360–364. IEEE Computer Society Press, 2004.

[27] 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.

[28] 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*, 2004.