

# A Certificate-Free Grid Security Infrastructure Supporting Password-Based User Authentication\*

Jason Crampton, Hoon Wei Lim, Kenneth G. Paterson, and Geraint Price  
Information Security Group  
Royal Holloway, University of London  
Egham, Surrey TW20 0EX, UK

{jason.crampton, h.lim, kenny.paterson, geraint.price}@rhul.ac.uk

## Abstract

*Password-based authentication is still the most widely-used authentication mechanism, largely because of the ease with which it can be understood by end users and implemented. In this paper, we propose a security infrastructure for grid applications, in which users are authenticated using passwords. Our infrastructure allows users to perform single sign-on based only on passwords, without requiring a public key infrastructure. Nevertheless, our infrastructure supports essential grid security services, such as mutual authentication and delegation, using public key cryptographic techniques. Moreover, hosting servers in our infrastructure are not required to have public key certificates, meaning mutual authentication and delegation of proxy credentials can be performed in a lightweight and efficient manner.*

## 1 Introduction

The vision of grid computing [17, 19] is to provide easy access to “unlimited” resources, thereby enabling computationally complex tasks to be performed and huge amounts of data to be stored and shared. There has been some suspicion, since the term *grid* was first used about a decade ago, that grid computing might be another technological vision that turns out to be more hype than substance. Despite that, the vision prevails and the gap between vision and reality is narrowing quickly. This is evident from the large and growing number of grid projects and testbeds worldwide [25]. TeraGrid [47], one of the pioneering grid projects, which was completed in 2004, is currently capable of providing 102 teraflops<sup>1</sup> of computing power and more than 15

petabytes<sup>2</sup> of online data storage. Despite the promising signs, many believe that a lot still needs to be done in order to realise computational grids analogous to the pervasive electrical power grid. In particular, as commercial interest grows in grid computing, grid security is an issue that will become increasingly important.

Currently, the grid security infrastructure (GSI) of the Globus Toolkit (GT) [16], proposed by Foster *et al.* [18], plays an essential role in supporting various grid security services, such as single sign-on, mutual authentication and delegation. Being based on public key infrastructure (PKI),<sup>3</sup> GSI users are required to possess and manage long-term credentials (typically RSA public/private key pairs), which are usually renewed yearly. Inevitably, some machines within the scope of a grid community may lack up-to-date protection in the form of the latest vulnerability patches and virus definitions. This may lead to such machines falling under the partial or complete control of attackers who are able to remotely exploit vulnerabilities and hence obtain long-term user credentials. To minimise the risk of compromise, many recent grid implementations make use of the MyProxy system [8, 37] to securely store and protect long-term user credentials. MyProxy also offers the benefit of “credential mobility”, enabling users to access their credentials from any machine through, for example, a web browser.

**Motivations.** It is desirable that remote computing resources be accessible from various platforms, including handheld devices, such as personal digital assistants (PDAs) and mobile phones, as well as desktop and laptop machines. In fact, due to the increasing availability of wireless devices in recent years, *wireless grids* [2, 34, 41] can offer additional untapped resources to existing wired computing re-

\*The research in this paper was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through Grant EP/D051878/1.

<sup>1</sup>A teraflop is one trillion floating-point operations per second.

<sup>2</sup>1 petabyte =  $10^3$  terabytes =  $10^6$  gigabytes.

<sup>3</sup>Here, we assume that existing PKIs make use of certificates. However, we will later show that a PKI can be certificate-free.

sources. In particular, wireless devices can offer different types of resources through their embedded objects, such as cameras, microphones, sensors and global positioning system (GPS) receivers. More importantly, wireless devices can supply information – on temperature, health, pollution levels, etc. – from geographic locations and social settings that are difficult to access through conventional wired networks [34].

The PKI-based GSI is a rather heavyweight apparatus, mainly because of the extensive use of public key certificates [28] and proxy certificates [48]. Generation, certification and verification of public keys, distribution of certificates, and other aspects of traditional public key management using PKIs incur non-trivial overheads. Wireless devices are often battery-powered and the energy required for transmission of a single bit of data is over 1000 times greater than that required by a single 32-bit computation [6]. Therefore, it is necessary to minimise the communication overheads of any grid security infrastructure if we are to exploit the full potential of wireless grids. In short, the emergence of wireless grids has prompted the need for a more lightweight architecture.

Lim and Paterson recently proposed a fully identity-based security infrastructure for grid applications [32] using identity-based public key cryptography (ID-PKC) [14, 44]. Key management in this approach is simpler than in the PKI-based GSI because it does not use certificates and key sizes are relatively small. For instance, the communication bandwidth requirement for mutual authentication and delegation between two entities can be reduced by up to 90%, when appropriately chosen elliptic curves and system parameters are used [32].

Nevertheless, *key revocation* in the identity-based setting can be complicated. Boneh and Franklin [14] proposed the use of a date concatenated with a user’s identifier (the construction of a public key from an identifier will be explained in Section 2.1.1) to achieve automated key expiry. However, this approach has the disadvantage of increasing the workload of a Private Key Generator (PKG), since the PKG is required to regularly issue private keys to its users. Alternatively, the PKG could issue private keys less frequently, for example monthly or yearly. In this case, however, it would be necessary to adapt conventional key revocation mechanisms, such as Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP), to the identity-based setting, in order to provide timely revocation of an identifier and its associated public key.

Moreover, *key escrow* is inevitable in the identity-based setting because the PKG uses a master secret to extract private keys of its users. This may not be desirable in some grid applications.

MyProxy continues to play a major role in the GSI by offering better credential protection and mobility to grid users.

However, its architecture has a subtle but crucial drawback. In the MyProxy protocol [8], although users are authenticated to their respective MyProxy servers using conventional username/password techniques, server authentication is achieved using the server-authenticated version of the TLS handshake protocol [15]. This implies the need to protect the root Certificate Authority’s public key certificates on the users’ machines. There are ways for an attacker to install a bogus root key in the user’s browser [5, 27]. Hence, if a desktop is vulnerable to stealing of a private key, then the desktop may also be at risk from replacement of the associated Certificate Authority’s certificate by the attacker.

The above issues and observations have led us to our investigation of a grid security infrastructure which is not only certificate-free, but also “PKI-free” from the user perspective.

**Contributions.** In this paper, we propose a password-enabled and certificate-free grid security infrastructure (PECF-GSI). Briefly, our proposal enhances the earlier work of Lim and Paterson [32] so that users are *only* authenticated using passwords, with the authentication taking place between users and a centralised authentication server. This server plays a similar, but not identical, role to the MyProxy server in the PKI-based GSI. Our approach has the benefit that neither client nor server certificates are required during user authentication. Our proposal also completely removes the need for long-term user public keys, and hence the need for a revocation mechanism for these public keys too.<sup>4</sup> Instead, users are given short-lived, identity-based credentials by the authentication server upon successful authentication. All subsequent security services are carried out using these credentials on behalf of users, without requiring direct user intervention. Thus our proposal separates security functions into two “zones”: a user-friendly zone, where only passwords are involved, and a certificate-free zone, which is hidden from the users’ view, and makes use of full-strength public key techniques. In addition, we show how to solve the key escrow issue by adopting certificateless public key cryptography (CL-PKC) [3]. Our contributions can be summarised as follows:

- We design a lightweight and user-friendly grid security infrastructure. Our proposal inherits attractive properties of the identity-based approach, in particular being certificate-free and using small key sizes. Mutual authentication of a user and a server is based only on a provably secure password-based authentication protocol. Yet, our architecture still supports various grid security services, such as single sign-on, mutual authentication and delegation.

---

<sup>4</sup>We still require mechanisms for handling revocation of server public keys, however.

- Key revocation in the identity-based setting is a well-known issue [14, 40]. We employ “just-in-time” issuance of short-lived keys to avoid any complications related to revoking users’ long-term public keys. Our approach is rather similar to the use of short-lived symmetric keys in Kerberos [36]. This, and the fact that system parameters of the identity-based primitives do not necessarily need to be pre-distributed or bootstrapped, gives rise to easy, flexible and user-friendly deployment of ID-PKC. We also show how timely revocation for hosting servers’ long-term public keys can be carried out very simply in our architecture, by pushing up-to-date revocation information to users during authentication.
- We develop an escrow-free grid security infrastructure using CL-PKC. Key escrow is inevitable in the identity-based setting because the PKG extracts private keys on behalf of its users, and is a feature of the identity-based grid security architecture of Lim and Paterson [32]. In applications where high-value or commercially sensitive resources are to be shared, an escrow capability at the Certificate Authority (CA) or MyProxy level is unlikely to be acceptable.
- We devise a more efficient and natural delegation protocol than the current technique used in the GSI [49] and the original approach of Lim and Paterson [32]. This can be achieved by exploiting the properties of hierarchical ID-PKC [24] and CL-PKC [4]. The mathematical properties of hierarchical ID-PKC and CL-PKC allow very efficient credential verification of a delegatee for a particular delegation. A verifier needs *only* to check the credential of the delegatee, instead of having to verify the credentials of the delegatee *and* all of his ancestors along the delegation chain, as in existing proposals [32, 49].

**Organisation.** The remainder of this paper is organised as follows. In Section 2, we introduce background material relevant to this paper. In Section 3, we present our proposal for a password-enabled and certificate-free grid security infrastructure. This includes a description of the architecture and its underlying protocols. In Section 4, we explain how key escrow can be removed from the security architecture proposed in the previous section. Section 5 discusses some performance issues of our proposal. Finally, we conclude in Section 6.

## 2 Background

In this section, we first give a brief introduction to pairings, which are bilinear maps fundamental to identity-based public key cryptography (ID-PKC) and certificateless pub-

lic key cryptography (CL-PKC). We then describe a provably secure password-based authentication protocol due to Abdalla *et al.* [1]. We also give a brief overview of the existing grid security infrastructure (GSI) of the Globus Toolkit (GT), and review some related work.

### 2.1 Cryptographic Preliminaries

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.

An *admissible pairing* in the context of identity-based and certificateless public key cryptography is a function  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  with the following properties:

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

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

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

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

- *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.

Typically,  $\mathbb{G}_1$  is a subgroup of the group of points on a suitable elliptic curve over a finite field,  $\mathbb{G}_2$  is obtained from a related finite field, and  $\hat{e}$  is obtained from the Weil or Tate pairing on the curve. Given  $P, Q \in \mathbb{G}_1$  and  $a \in \mathbb{Z}_q^*$ ,  $P + Q$  denotes elliptic curve point addition, and  $aP$  denotes elliptic curve point (or scalar) multiplication. Note that  $aP$  can be computed very efficiently. However, the problem of finding  $a$  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 reader is referred to [21] for more mathematical background on pairings.

#### 2.1.1 Identity-Based Public Key Cryptography

In 1984, Shamir [44] proposed the idea of identity-based public key cryptography (ID-PKC). Instead of generating and using a random public/private key pair in a public key cryptosystem such as RSA or ElGamal, Shamir proposed using a user’s name or other unique identifier (such as an email address) as a public key, with the corresponding private component being generated by a trusted Private Key Generator (PKG). Since a user’s public key is based on

some publicly available information that uniquely identifies the user, an identity-based cryptosystem does not require a mechanism for authenticating public keys. However, Shamir was only able to develop an identity-based signature (IBS) scheme based on the RSA primitive.

Only in the early 2000s did the emergence of cryptographic schemes based on pairings on elliptic curves result in the construction of a feasible and secure IBE scheme [14, 29, 42]. Further details can be found in [39].

Gentry and Silverberg [24] proposed hierarchical identity-based encryption (HIBE) and hierarchical identity-based signature (HIBS) schemes with total collusion resistance, regardless of the number of levels in the hierarchy. In the hierarchical setting, a root PKG produces private keys for PKGs in the next level of the tree, who in turn generate private keys for PKGs or users in the next level (and so on). It is this scheme on which our proposal is based.<sup>5</sup> We now sketch Gentry and Silverberg’s HIBE and HIBS schemes (see [24] for full details).

**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  $n$ ,  $H_3 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ ,  $H_4 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*$  and  $H_5 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . The root PKG’s master secret is  $s_0$  and the system parameters are  $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$ .

**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  $\langle \text{ID}_1, \dots, \text{ID}_t \rangle$ , where  $\langle \text{ID}_1, \dots, \text{ID}_i \rangle$  is the ID-tuple of the entity’s ancestor at level  $i$  ( $1 \leq i \leq t-1$ ), the entity’s parent computes  $P_t = H_1(\text{ID}_1, \dots, \text{ID}_t) \in \mathbb{G}_1$ , 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  $S_t$ , as his private key, and the Q-values by its parent.

**ENCRYPT:** Given a message  $m$  and ID-tuple  $\langle \text{ID}_1, \dots, \text{ID}_t \rangle$ , this algorithm computes the ciphertext  $\langle rP_0, rP_2, \dots, rP_t, z \oplus H_2(\hat{e}(Q_0, P_1)^r), m \oplus H_5(z) \rangle$ , where  $z \in \{0, 1\}^n$  and  $r = H_4(z, m)$ . Note that  $rP_0, rP_2, \dots, rP_t$  are all elements of  $\mathbb{G}_1$  and the sizes of the last two components of the ciphertext are dependent on  $n$ .

<sup>5</sup>It is worth noting that other HIBE and HIBS schemes are available. We chose the Gentry/Silverberg schemes because they are efficient and their security is based on reasonable computational assumptions.

**DECRYPT:** Given a ciphertext  $\langle U_0, U_2, \dots, U_t, V, W \rangle$ , this algorithm takes as input the associated private key  $S_t$  and recovers  $m$ . It also checks if  $U_0, U_2, \dots, U_t$  have the correct structure. Otherwise, the recovered  $m$  is rejected.

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

**VERIFY:** Given a signature  $\langle \sigma, Q_1, \dots, Q_t \rangle$  of a message  $m$ , this algorithm takes as input the associated public key, computed from ID-tuple  $\langle \text{ID}_1, \dots, \text{ID}_t \rangle$ , and returns a message indicating the success or failure of the verification.

## 2.1.2 Certificateless Public Key Cryptography

There are applications which do not tolerate key escrow, which is a feature of identity-based cryptosystems. This has led to the development of escrow-free variants of pairing-based public key cryptography, including Al-Riyami and Paterson’s certificateless public key cryptography (CL-PKC) [3] and the certificate-based encryption (CBE) concept of Gentry [23]. It is the former approach that we use in this paper.

A user’s private key in the certificateless setting consists of two components: (i) an identity-dependent partial private key (generated in the same way as in the normal identity-based approach); and (ii) a full private key which can be produced using the partial private key and some secret known only to the user. Succinctly, this approach uses input from the PKG and the user to generate a private key, thereby eliminating key escrow. We now briefly describe a hierarchical certificateless encryption (HCLE) scheme and a hierarchical certificateless signature (HCLS) scheme [4].

**ROOT SETUP:** As in Section 2.1.1.

**LOWER-LEVEL SETUP:** As in Section 2.1.1.

**PARTIAL-PRIVATE-KEY EXTRACT:** As with the EXTRACT algorithm in Section 2.1.1, except that this algorithm sets the entity’s partial private key  $D_t$  to be  $\sum_{i=1}^t s_{i-1} P_i = D_{t-1} + s_{t-1} P_t$ , where  $D_{t-1}$  is the parent’s partial private key and  $s_{t-1}$  is a secret value only known to the parent. The entity’s parent also defines Q-values by setting  $\langle Q_{X_i}, Q_{Y_i} \rangle = \langle s_i P_0, s_i Q_0 \rangle$  for  $1 \leq i \leq t-1$ .

**SET-PRIVATE-KEY:** This algorithm transforms a partial private key  $D_t$  of an entity at level  $t$  with ID-tuple  $\langle \text{ID}_1, \dots, \text{ID}_t \rangle$  into a private key  $S_t = s_t D_t$ , where  $s_t$  is the secret value that the entity has chosen in LOWER-LEVEL SETUP.

**SET-PUBLIC-KEY:** This algorithm sets a public key of an entity at level  $t$  with ID-tuple  $\langle \text{ID}_1, \dots, \text{ID}_t \rangle$  as  $\langle s_t P_0, s_t Q_0 \rangle$ .

**ENCRYPT:** This algorithm first checks if the Q-values have the correct structure. It then performs similar steps to the ENCRYPT algorithm in Section 2.1.1, except that this algorithm takes as input the ID-tuple  $\langle \text{ID}_1, \dots, \text{ID}_t \rangle$ , the public key  $\langle s_t P_0, s_t Q_0 \rangle$  and the related Q-values to compute the ciphertext.

**DECRYPT:** As with the DECRYPT algorithm in Section 2.1.1, except that this algorithm takes as input a different set of Q-values.

**SIGN:** As with the SIGN algorithm in Section 2.1.1, except that this algorithm uses different Q-values.

**VERIFY:** This algorithm first validates the format of the Q-values. It then performs the similar steps as with the VERIFY algorithm in Section 2.1.1, except that this algorithm uses a different public key and Q-values.

We remark that the above schemes do not have formal security models and proofs. Nevertheless, they are straightforward adaptations of the provably secure HIBE and HIBS schemes of Gentry and Silverberg described in Section 2.1.1.

### 2.1.3 A Password-Based TLS Protocol

Abdalla *et al.* [1] recently proposed a provably secure password-based TLS protocol, based on earlier work of Steiner *et al.* [46]. The protocol makes use of a discrete logarithm based mask generation function to instantiate a symmetric encryption primitive, as suggested by Bellare *et al.* [10, 11]. The protocol also makes use of a hash function  $H$ , mapping onto a Diffie-Hellman group generated by  $g$ . Then,  $A$  with password  $PW_A$  encrypts a Diffie-Hellman component  $g^a$  by calculating  $\{g^a\}_{\pi_A}$ , where  $\pi_A = H(PW_A)$  and  $\{g^a\}_{\pi_A} = g^a \cdot \pi_A$ . Thus the result of the encryption is a group element. To decrypt and recover  $g^a$ , one can simply divide the ciphertext by  $\pi_A$ . We describe a modified version of this protocol and explain how it is used to support single sign-on in Section 3.3.1.

The important point about this protocol is that dictionary attacks are of little value to an adversary. If the adversary guesses a password and uses it to decrypt  $g^a \cdot \pi_A$ , he simply obtains a group element. In effect, the group element  $g^a$  masks the (hash of the) password.

## 2.2 The Grid Security Infrastructure

The PKI-based GSI focuses on authentication, message protection, and the use of proxy credentials to support single sign-on and credential delegation [18, 49, 50]. In grid applications that employ the GSI, each entity is assigned a

unique identity or distinguished name and given a public key certificate signed by a Grid CA. Public key certificates are used to support authentication and key agreement protocols, such as the TLS protocol. Proxy certificates are 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. The user's long-term private key does not need to be accessed again until the expiry of the proxy certificate. For rights delegation from a user  $A$  to a target service provider  $X$ , three steps are required [49]:

1.  $X$  generates a new public/private key pair and sends a request (that is signed with the new private key) to  $A$ ;
2.  $A$  verifies the request using the new public key, creates a new proxy certificate, and signs it with her current proxy credential (short-lived private key);
3.  $A$  forwards the new proxy certificate to  $X$ .

Note that  $A$  can impose some constraints on what  $X$  can and can't do, using the `ProxyPolicy` field of the proxy certificate.  $A$  has to trust that an entity to which  $X$  presents this proxy certificate will impose the constraints specified.

The GSI has been built on the Generic Security Service Application Program Interface (GSS-API) [33] and incorporates GSI-enabled OpenSSL [38] to support proxy certificates. Examples of the RSA-based cipher suites include `TLS_RSA_WITH_RC4_128_MD5` and `TLS_RSA_WITH_DES_CBC_SHA`.

In the GSI setting, each user has a long-term RSA public/private key pair with a 1024-bit modulus. 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 generating an RSA key pair is a computationally expensive operation. It has been shown 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 [49]. Since the proxy credential has a relatively short lifetime, it is currently believed that the reduction in security implied by using only 512-bit moduli poses an acceptably low risk in grid systems.

There are a small number of grid projects that use Kerberos [36] as the backbone of their security infrastructures. It is generally believed that Kerberos, being based on symmetric key cryptography, is more efficient than PKI-based approaches. However, Kerberos is unlikely to be a suitable long-term solution because many computational grids have a dynamic entity population, and the establishment and management of shared symmetric keys will be impractical. Furthermore, it is not clear how the dynamic delegation mechanism of [49] can be supported using Ker-

beros. 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, some 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 [30, 35].

## 2.3 Related Work

MyProxy [8, 37] is an online credential repository that implements the virtual smart card concept [43]. As with storing keys in a smart card, a MyProxy server is expected to provide better protection for long-term user private keys than desktop computing environments.

To create a proxy credential, a user authenticates himself to the MyProxy server using a password which he shares with the MyProxy server by performing the following steps [8].

1. The user establishes a TCP connection to the server and initiates a server-authenticated TLS handshake protocol.
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 information, such as a username, a password 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.
4. The user then generates a new public/private key pair and forwards the public key to the server through the established secure channel.
5. Subsequently, the server creates a new proxy certificate signed with the user's stored private key and returns it to the user.

This approach relies on a certificate-based PKI and the user must ensure that the associated certificates bootstrapped in his machine are trustworthy and have not been replaced.

Recently, Beckles *et al.* [9] considered issues related to the usability of the PKI-based GSI, noting that managing certificates can be burdensome and tedious for general grid users. In an effort to improve the usability of the PKI-based GSI, they adopted Gutmann's plug-and-play PKI concept [26], which emphasises automated and transparent setup of PKI for the end user. In so doing, Beckles *et al.* make use of the PKIBoot service of [26] to allow a user to authenticate himself to a PKIBoot server with the standard username/password method. Subsequently, the user can securely retrieve his public key certificate (and optionally his private key) and/or CAs' certificates. This approach

can eliminate the difficult tasks involved in correctly establishing trust roots of CAs from the user side. It can also minimise the user's direct involvement in certificate management. Our proposal for a user-friendly and certificate-free security architecture is influenced by Beckles *et al.*'s work.

Although the plug-and-play PKI concept seems to make PKI more usable for the users, there are still many aspects of PKI that need to be addressed. For example, how can we improve the effectiveness of current key revocation mechanisms, such as CRLs, by exploiting the advantages that the plug-and-play PKI concept could bring? Furthermore, the application of the plug-and-play PKI to the GSI does not reduce the extensive use of certificates, and certificate chain verification is still required for all the grid security services which involve certificates.

## 3 Our Proposal

Here, we propose a password-enabled and certificate-free GSI (PECF-GSI). We begin by giving a conceptual view of the PECF-GSI design. We explain how PECF-GSI can support various grid security services. Then, we provide details of the protocols which underpin PECF-GSI.

### 3.1 Architectural Overview

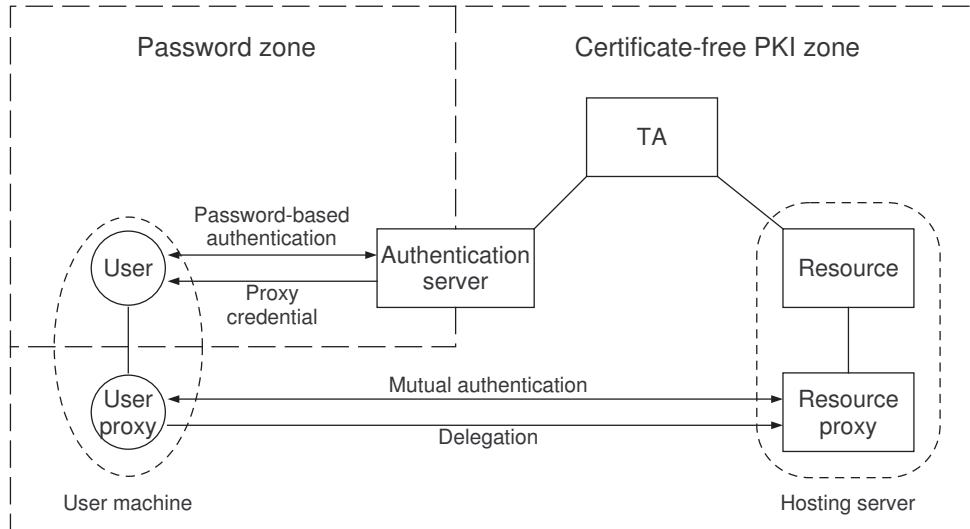
PECF-GSI employs a Trusted Authority (TA), instead of a CA, as the root of trust within a grid environment. The TA's roles include acting as the PKG in the identity-based setting and providing a key management service. In PECF-GSI, a user's long-term credential is simply a password, which he shares with an authentication server. We assume that the user delivers his password to the authentication server during a one-off user registration phase.<sup>6</sup>

The authentication server is assumed to be accredited by the TA and hosting servers (or resource providers) within the grid environment. Unlike the user, who only has to remember a password, the authentication and hosting servers must obtain the TA's authenticated parameter set through out-of-band mechanisms. This hybrid approach divides our architecture into two zones: (i) a user-centric zone which employs password-based authentication, and (ii) a server-centric zone which makes use of identity-based PKI (non-certificate-based PKI). This is illustrated in Figure 1.

As with the current GSI, we make use of proxy credentials when providing security services such as mutual authentication and delegation. In Figure 1, a user proxy is a short-lived agent created by the user to perform security services on the user's behalf. Similarly, a resource proxy is

---

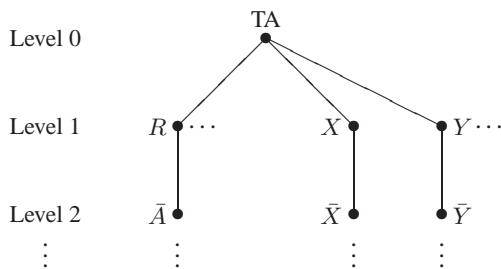
<sup>6</sup>Note that user registration in PECF-GSI setting may well be much simpler than applying for an X.509 certificate in the GSI setting. This is because the user does not have to obtain a certified public key from a CA.



**Figure 1. A conceptual view of PECF-GSI.**

created by a resource provider to help manage a job submission from a user.

We map the entities in Figure 1, each of which will require some form of credentials to interact with another entity, into the hierarchical setting of the Gentry–Silverberg HIBE and HIBS schemes (as introduced in Section 2.1.1). Let  $R$  be an authentication server,  $A$  be a user, and  $X$  and  $Y$  be hosting servers. We write  $\bar{A}$  and  $\bar{X}$  to denote proxies for  $A$  and  $X$ , respectively. Then, the TA is a level 0 entity in the hierarchy, and issues private keys to  $R$ ,  $X$  and  $Y$  at level 1. These entities, in turn, issue private keys to their respective children at level 2, as shown in Figure 2. Note that  $A$  does not possess any long-term credential issued by the TA; instead she obtains proxy credentials from  $R$ . Hence,  $\bar{A}$ , a user proxy for  $A$ , becomes a child of  $R$ .



**Figure 2. The hierarchical relationships between entities in PECF-GSI.**

Using the above notation, we now briefly explain how

PECF-GSI can be used to support essential security services for grid applications. Further details of the underlying protocols will be provided in Section 3.3.

Before user  $A$  submits a job to resource  $X$ , for example, through the Grid Resource Allocation and Management (GRAM) module of the GT [50], she authenticates herself to  $R$  using a secure username/password mechanism. When the password-based authentication is successful and a secure channel between  $A$  and  $R$  is established,  $R$  extracts a proxy credential for use by  $\bar{A}$ . The proxy credential, which comprises a short-term public/private key pair, is transmitted to  $A$ , along with other required information, such as the TA's system parameters, through the secure channel.

Subsequently,  $\bar{A}$  signs her job request (with the new proxy private key), which is then submitted to  $X$ .  $X$  verifies  $\bar{A}$ 's signed request and checks if  $A$  is an authorized user. If the checks are successful,  $X$  creates  $\bar{X}$  and the associated managed job service [50]. This is followed by the mutual authentication of  $\bar{A}$  and  $\bar{X}$  through an identity-based and certificate-free TLS handshake protocol.

$A$  may, at her discretion, delegate her credential through  $\bar{A}$  to  $\bar{X}$  for later use [50].  $\bar{A}$  can achieve this in PECF-GSI by simply issuing a new proxy private key to  $\bar{X}$  through the established TLS channel. This short-lived private key is generated based on the relevant delegation information. Now the delegatee  $\bar{X}$  effectively becomes a child of the delegator  $\bar{A}$  in the hierarchy depicted in Figure 2. Similarly, if  $\bar{X}$  further delegates some rights to  $Y$ , then  $\bar{Y}$  will become a child of  $\bar{X}$ . The resulting delegation chain rooted at the TA is  $TA \rightarrow R \rightarrow \bar{A} \rightarrow \bar{X} \rightarrow \bar{Y}$ .

In this paper, we use a hierarchy with a single TA for ease

of exposition. In actual implementations, we can expand the hierarchy of Figure 2 to support multiple TAs. This can be achieved by adding a root TA at the top of the hierarchy with the TAs becoming level 1 entities. Similarly, lower-level entities are moved down to the next level in the hierarchy.

## 3.2 On System Parameters and Keys

We now describe the system parameters and keys that will be used in our protocols, which are described in Section 3.3.

### 3.2.1 Parameter Generation and Distribution

During the system setup phase, the TA runs a Bilinear Diffie-Hellman (BDH) parameter generator to obtain 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  $s_0$ . The system parameters are  $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$ . We remark that an authentic set of the TA parameters must be made available to the authentication and hosting servers. One way to achieve this is by bootstrapping these parameters into the grid system. Alternatively, distribution of the parameters is also possible through the use of a certificate obtained from a conventional CA that certifies the parameters. We will discuss concrete parameter choices in Section 5.

### 3.2.2 Key Generation

Once the system parameters have been set up, the TA can issue private keys to its subordinates at level 1 (see Figure 2) using its master secret  $s_0$  computed by the ROOT SETUP algorithm. For example, authentication server  $R$ 's long-term private key is  $S_R = s_0 P_R$ , where  $P_R = H_1(\text{ID}_R)$  is the matching public key. Hosting servers' long-term public/private keys are generated in a similar way. A proxy's public key at level 2 can be computed based on its ancestor's identifier and its own identifier concatenated with a lifetime  $LT$  in some fixed format. For example, user  $A$ 's proxy public key would be  $P_{\bar{A}} = H_1(\text{ID}_R, \text{ID}_A \| \text{LT}_A)$ , and the corresponding private key can be obtained from  $R$ , who will run the EXTRACT algorithm of the Gentry–Silverberg HIBE (or HIBS) scheme to generate  $S_{\bar{A}} = S_R + s_R P_{\bar{A}}$ . Here,  $s_R$  is a secret value chosen by  $R$  when it performs the LOWER-LEVEL SETUP algorithm. The upper part of Table 1 summarises the credentials possessed by the authentication server  $R$ , user  $A$  and hosting server  $X$ .

It is worth noting that  $A$  does not possess any long-term credential, except a password which she shares with  $R$ . In fact,  $R$ 's proxies are proxies of the users with whom it shares passwords.

### 3.2.3 Key Revocation

Our proposed design deals with revocation of user keys and of hosting server keys in different ways. The users are never given long-term public keys, instead they are only ever provided with proxy keys. As with proxy certificates [48] and Kerberos tickets [36], these proxy credentials in our PECF-GSI setting have a short lifetime, typically less than 12 hours. As the window of exposure to compromise is minimised, there is no need for an explicit revocation mechanism for user keys. In this case, the hosting servers are trusting  $R$  to only distribute fresh keys to the users if the users' privileges are still valid.

Conversely, hosting servers are issued with long term public keys. In this case, there is a requirement for an explicit revocation mechanism. To allow for the revocation of servers' public keys, we introduce the notion of an Identity Revocation List (IRL), where an IRL is analogous to a CRL in a certificate-based environment. The IRL includes the identity of any server whose key has been revoked. This allows users to verify the validity of a particular hosting server's public key prior to submitting a job to that server. IRLs are distributed to users by  $R$  via the secure channel established at authentication time. From the user's perspective, this "push" method of distribution simplifies the process of verifying whether a hosting server has had its public key revoked. We discuss this issue in more detail in Section 3.3.1.

## 3.3 Protocols

We now describe the protocols that we employ in PECF-GSI to provide single sign-on, mutual authentication and delegation for grid applications.

### 3.3.1 Single Sign-on

The MyProxy system makes use of the existing standard TLS protocol [15] to provide mutual authentication between a user and a MyProxy server. This is typically based on the MyProxy server's public key certificate and a password shared by the server and the user. However, in such a setup, 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 [46]. This may give a false sense of security if management of certificates of the relevant parties (e.g. the server and its CA) are not handled properly. This prompted the study of password-based TLS protocols [1, 46].

We use a modified version of the protocol described in Section 2.1.3 for mutual authentication between a user and the authentication server in our PECF-GSI setting. This is because the protocol is provably secure and it can be im-

**Table 1. Credentials and keys in PECF-GSI.**

| Scheme                                | Entity | Long-term Credential               |                     | Proxy Credential   |  |
|---------------------------------------|--------|------------------------------------|---------------------|--|--|
|                                       |        | Public Key                         | Private Key         | Public Key   | Private Key  |
| <b>Gentry–Silverberg (HIBE/HIBS)</b>  | $R$    | $P_R = H_1(\text{ID}_R)$           | $S_R = s_0 P_R$     | –  | –  |
|                                       | $A$    | –                                  | –                   | $P_{\bar{A}} = H_1(\text{ID}_R, \text{ID}_A \  \text{LT}_A)$ | $S_{\bar{A}} = S_R + s_R P_{\bar{A}}$              |
|                                       | $X$    | $P_X = H_1(\text{ID}_X)$           | $S_X = s_0 P_X$     | $P_{\bar{X}} = H_1(\text{ID}_X, \text{ID}_X \  \text{LT}_X)$ | $S_{\bar{X}} = S_X + s_X P_{\bar{X}}$              |
| <b>Al-Riyami–Paterson (HCLE/HCLS)</b> | $R$    | $\langle s_R P_0, s_R Q_0 \rangle$ | $S_R = s_R s_0 P_R$ | –  | –  |
|                                       | $A$    | –                                  | –                   | $\langle s_{\bar{A}} P_0, s_{\bar{A}} Q_0 \rangle$           | $S_{\bar{A}} = s_{\bar{A}}(S_R + s_R P_{\bar{A}})$ |
|                                       | $X$    | $\langle s_X P_0, s_X Q_0 \rangle$ | $S_X = s_X s_0 P_X$ | $\langle s_{\bar{X}} P_0, s_{\bar{X}} Q_0 \rangle$           | $S_{\bar{X}} = s_{\bar{X}}(S_X + s_X P_{\bar{X}})$ |

plemented by modifying existing implementations of the widespread standard TLS protocol.

In PECF-GSI, we translate the discrete logarithm based approach of Abdalla *et al.* to the elliptic curve setting. We make use of the hash function  $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$  from the HIBE scheme.  $A$ 's password  $PW_A$  is mapped to  $\pi_A = H_1(PW_A) \in \mathbb{G}_1^*$ . The Diffie-Hellman component  $g^a$  is replaced by  $aP_0$ , where  $P_0$  generates  $\mathbb{G}_1$ , and  $\{aP_0\}_{\pi_A}$  is defined to be  $aP_0 + \pi_A$ . To recover  $aP_0$ , we simply subtract  $\pi_A$  from  $\{aP_0\}_{\pi_A}$ . Based on this mask generation function, which makes use of the system parameters in our PECF-GSI setting, we can derive a password-based TLS protocol analogous to the protocol of [1].<sup>7</sup> Further details of this protocol are given in Appendix A.

Steiner *et al.* and Abdalla *et al.* suggest that parameters such as  $\mathbb{G}_1$ ,  $P_0$  and  $H_1$  should be fixed or form part of a standardised ciphersuite. This obviates the need for the user  $A$  to verify the number-theoretic appropriateness of these parameters.

Once  $A$  and  $R$  have been mutually authenticated and established a secure session,  $R$  extracts a short-lived public/private key pair  $(P_{\bar{A}}, S_{\bar{A}})$ , shown in Table 1. Subsequently,  $R$  sends the following information to  $A$  through the secure channel:

1. the newly created proxy credential  $(P_{\bar{A}}, S_{\bar{A}})$ ;
2. an authenticated copy of the TA system parameters  $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$ ;<sup>8</sup>
3. an up-to-date IRL.

Upon receiving the proxy credential,  $A$  stores the private key  $S_{\bar{A}}$  in a local file system accessible by her proxy  $\bar{A}$  when necessary. This completes the process of single sign-on by  $A$ . The system parameters that  $A$  receives from  $R$  are needed to run the Gentry–Silverberg HIBE and HIBS

<sup>7</sup>In order to optimise the efficiency of our proposal, we re-use some of the components of the system parameters  $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$  that  $R$  obtained from the TA.

<sup>8</sup>We note that the parameters  $\mathbb{G}_1$ ,  $P_0$  and  $H_1$  are already in use by  $A$  to run the password-based TLS protocol. Hence, in an actual implementation,  $R$  only needs to transmit the remaining system parameters to  $A$ .

schemes when performing mutual authentication and delegation (see Sections 3.3.2 and 3.3.3).

The IRL is used by  $\bar{A}$  to check the continuing validity of the identifier of the hosting server to which  $\bar{A}$  is going to submit her job. It is worth noting that this approach “forces” the user into receiving an up-to-date IRL. Additionally, the user does not have to check the authenticity of the IRL, assuming  $R$  behaves in an honest manner. Upon expiry of the proxy credential, all the information that  $\bar{A}$  obtained from  $R$  can be destroyed.

We remark that the whole process of single sign-on does not involve any kind of certificate or parameter verification from the user’s perspective (recall that users are in a password-based zone in our PECF-GSI setting). Here, we regard verification of parameters as checking the authenticity of the parameters and not validating their number-theoretic structure.

### 3.3.2 Mutual Authentication and Key Agreement

While a password-based TLS protocol is a convenient mechanism for the authentication server and its (known) users, the standard PKI-based TLS protocol is clearly more suitable for mutual authentication and key agreement between two entities who have not previously communicated. We now leave the password-based zone and enter the certificate-free PKI zone, where we explain how two entities within a grid environment can authenticate each other and share a session key.

Figure 3 shows a certificate-free authenticated key agreement protocol, adapted from Lim and Paterson’s identity-based TLS protocol [32], with minor modifications to the `ServerIdentifier` and `ClientIdentifier` messages. The protocol employs the Gentry–Silverberg HIBE and HIBS schemes.

In the first message of the protocol,  $n_{\bar{A}}$  denotes a nonce chosen by  $\bar{A}$ , `session_id` is self-explanatory, and `cipher_suite` contains a cipher specification that handles the HIBE and HIBS schemes, e.g. `TLS_HIBE_HIBS_WITH_DES_CBC_SHA`. Here,  $Enc_{\bar{X}}(\cdot)$  denotes an encryption using the HIBE scheme

|     |                                 |  |
|-----|---------------------------------|--|
| (1) | $\bar{A} \rightarrow \bar{X}$ : | ClientHello = $n_{\bar{A}}$ , session_id, cipher_suite   |
| (2) | $\bar{X} \rightarrow \bar{A}$ : | ServerHello = $n_{\bar{X}}$ , session_id, cipher_suite,<br>ServerIdentifier = $ID_X, ID_X    LT_X$ ,<br>ServerHelloDone  |
| (3) | $\bar{A} \rightarrow \bar{X}$ : | ClientIdentifier = $ID_R, ID_A    LT_A$ ,<br>ClientKeyExchange = $Enc_{\bar{X}}(\text{pre\_master\_secret})$ ,<br>IdentityVerify = $Sig_{\bar{A}}(\text{handshake\_messages})$ ,<br>ClientFinished |
| (4) | $\bar{X} \rightarrow \bar{A}$ : | ServerFinished   |

**Figure 3. A certificate-free authenticated key agreement protocol**

with  $X$ 's proxy public key  $P_{\bar{X}}$ , while  $Sig_{\bar{A}}(\cdot)$  represents a signing operation in the HIBS scheme using  $A$ 's proxy private key  $S_{\bar{A}}$ .

When  $\bar{A}$  (playing the role of client) performs mutual authentication with  $\bar{X}$  (playing the role of server), she needs to forward her public key information, i.e.  $ID_R, ID_A || LT_A$  to  $\bar{X}$  as part of the protocol handshake, and vice versa. Note that since  $\bar{X}$  includes its long-term identifier in the `ServerIdentifier` message,  $\bar{A}$  must check if the identifier is still valid using the IRL that she received from  $R$ . Similarly, since  $\bar{A}$  uses  $R$ 's long-term public key information,  $\bar{X}$  must validate  $R$ 's public key before using it.

Space constraints preclude a more detailed description of the TLS protocol and the above protocol. The interested reader is referred to the literature for further details [15, 32]. We also note that our certificate-free authenticated key agreement protocol can be adapted straightforwardly to support user-to-user authentication.

### 3.3.3 Delegation

The current delegation technique used in the GSI requires a round-trip interaction between a delegator (typically a grid user) and a delegatee (typically a hosting server or resource provider), as described in Section 2.2. Also, verification of a delegatee's credential requires validating both long-term and proxy certificates of all the parties involved along the delegation chain.

Lim and Paterson proposed an identity-based one-pass delegation protocol [32], in which the delegator signs a delegation token and forwards it to the delegatee. One advantage of this approach is that the delegator can bind the delegatee's public key information to the delegation token without acquiring the delegatee's proxy public key, thus requiring only one-pass protocol message. However, verification of a delegatee's status as the delegation target requires validation of all the signed delegation tokens (analogous to validation of certificates in the GSI) issued by all the delegators along the delegation chain.

Here, we propose a delegation protocol which is not only

certificate-free and is a one-pass protocol message, but also has a very efficient verification mechanism in the sense that a delegatee's delegated credential can be checked by performing only one signature verification, regardless of the length of the delegation chain. This is a significant improvement on the two aforementioned delegation methods.

We now explain the details of the delegation technique that we employ in PECF-GSI by using an example between the delegator  $A$  (through her proxy  $\bar{A}$ ) and the delegatee  $\bar{X}$ . In the delegation process,  $\bar{A}$  performs the following steps:

1. compute a proxy public key  $P_{\bar{A}/\bar{X}}$  of the form

$$H_1(ID_R, ID_A || LT_A, ID_X || LT_{\bar{X}} || Job_{\bar{X}} || Policy_{\bar{X}}),$$

where  $LT_{\bar{X}}$  is the lifetime that  $\bar{A}$  decides for  $\bar{X}$ ,  $Job_{\bar{X}}$  describes  $A$ 's job request, and  $Policy_{\bar{X}}$  indicates the policy that  $A$  wishes to enforce on  $\bar{X}$ ;

2. extract a proxy private key  $S_{\bar{A}/\bar{X}} = S_{\bar{A}} + s_{\bar{A}}P_{\bar{A}/\bar{X}}$  with her secret value  $s_{\bar{A}}$ ;
3. transmit  $\langle ID_X || LT_{\bar{X}} || Job_{\bar{X}} || Policy_{\bar{X}}, S_{\bar{A}/\bar{X}} \rangle$  to  $\bar{X}$  through a secrecy and integrity protected TLS channel.<sup>9</sup>

In this case,  $\bar{A}$  actually acts as a PKG and issues a private key to  $\bar{X}$ , which becomes the entity below  $\bar{A}$  at level 3 in Figure 2. This can be seen in the ID-tuple used to construct  $P_{\bar{A}/\bar{X}}$ , in which the first two parts are identifiers of  $\bar{X}$ 's ancestors, i.e.  $R$  and  $\bar{A}$ .

It is worth noting that here,  $\bar{X}$ 's delegated proxy credential is different from  $\bar{X}$ 's proxy credential shown in Table 1. The latter is usually used when  $\bar{X}$  performs mutual authentication with users.

If a third party, for example  $\bar{Y}$ , wants to verify that  $\bar{X}$  indeed is acting on  $\bar{A}$ 's behalf, then  $\bar{Y}$  must: (i) authenticate

<sup>9</sup>It might be thought that the need for the secure channel to transport the proxy private key from  $\bar{A}$  to  $\bar{X}$  is a limitation of this approach. In fact, the secure channel between these two parties will exist anyway; the parties have to authenticate each other using the TLS handshake protocol, before the delegation can take place. This is to ensure that the delegation is targeted at the right entity and that the delegation target is convinced of the identity of the delegator.

$\bar{X}$  and (ii) check that  $\bar{X}$  is in possession of  $S_{\bar{A}/\bar{X}}$ . These two checks will be carried out as part of the TLS handshake that takes place between  $\bar{X}$  and  $\bar{Y}$ .

When  $\bar{X}$  further delegates  $\bar{A}$ 's credential to another hosting server  $Y$ ,  $\bar{X}$  can construct a new proxy public key

$$P_{\bar{A}/\bar{X}/\bar{Y}} = H_1(\text{ID}_R, \text{ID}_A \parallel \text{LT}_A, \text{ID}_X \parallel \text{LT}_{\bar{X}} \parallel \text{Job}_{\bar{X}} \parallel \text{Policy}_{\bar{X}}, \text{ID}_Y \parallel \text{LT}_{\bar{Y}} \parallel \text{Job}_{\bar{Y}} \parallel \text{Policy}_{\bar{Y}}),$$

where  $\text{Job}_{\bar{Y}}$  refers to the job (potentially sub-tasks of  $\text{Job}_{\bar{X}}$ ) that  $\bar{X}$  wants  $\bar{Y}$  to execute and  $\text{Policy}_{\bar{Y}}$  refers to the policy that  $\bar{X}$  imposes on  $\bar{Y}$ , respectively. The matching private key is  $S_{\bar{A}/\bar{X}/\bar{Y}} = S_{\bar{A}/\bar{X}} + s_{\bar{X}} P_{\bar{A}/\bar{X}/\bar{Y}}$ . This private key and the relevant information can then be forwarded to  $\bar{Y}$ , which subsequently becomes subordinate to  $\bar{X}$  at level 4 of the hierarchy.

To verify  $\bar{Y}$ 's delegated proxy credential, the verifier only needs to authenticate  $\bar{Y}$  and check whether  $\bar{Y}$  knows the private key corresponding to  $P_{\bar{A}/\bar{X}/\bar{Y}}$ , even though the delegation chain now has two delegates ( $\bar{X}$  and  $\bar{Y}$ ). This can be done, in principle, by verifying a signature produced by  $\bar{Y}$  using  $S_{\bar{A}/\bar{X}/\bar{Y}}$ .

We remark that the use of the Gentry–Silverberg HIBS scheme for this purpose would result in the size of the signature increasing as the delegation chain grows and verification of the signature becoming slower. Nevertheless, there exist improved HIBE schemes in the literature, from which we can derive a more efficient HIBS scheme. Boneh *et al.*, for example, recently proposed a HIBE scheme in which the size of the ciphertext is constant and decryption requires only two pairing computations, regardless of the hierarchy depth [13].

### 3.4 Security Considerations

In our single sign-on approach, we assume that  $R$  is a party trusted to issue the correct system parameters, most importantly  $Q_0$ , and up-to-date IRLs to its users through secure channels. Therefore, no additional infrastructure is required to verify the authenticity of the parameters and IRLs. Note that most of the components of the system parameters discussed in Section 3.2 can be fixed and made public, except  $Q_0 = s_0 P_0$ , where  $s_0$  is the TA's master secret. A failure to obtain  $Q_0$  from a trusted source would allow a trivial man-in-the-middle attack. Our single sign-on protocol is secure against such an attack, assuming  $R$  behaves honestly. Also, we assume that hosting servers always trust  $R$  in issuing proxy credentials to the correct users. These assumptions are essential for the protocol in Figure 3 and our delegation protocol to work as intended.

We now consider the possibility of an adversary attacking the delegation protocol. Using our example from the previous section, we need to consider the possibility that

an adversary  $E$ , who is also a valid user under the same TA, intercepts the proxy private key  $S_{\bar{A}/\bar{X}}$  that  $\bar{A}$  created for  $\bar{X}$ , and replaces it with  $E$ 's self-computed private key  $S'_{\bar{A}/\bar{X}} = S_{\bar{E}} + s_{\bar{E}} P_{\bar{A}/\bar{X}}$ . Superficially, this appears to be a feasible attack, since  $P_{\bar{A}/\bar{X}}$  is public and  $E$  knows the system parameters. However, the HIBE and HIBS schemes have the property that the private key corresponding to  $P_{\bar{A}/\bar{X}}$  can only be computed by the owner of the identity “ $\text{ID}_A \parallel \text{LT}_A$ ”, which is the immediate ancestor of the next level identity “ $\text{ID}_X \parallel \text{LT}_{\bar{X}} \parallel \text{Job}_{\bar{X}} \parallel \text{Policy}_{\bar{X}}$ ” in the same hierarchy. Gentry and Silverberg's security model does model an adversary that obtains identifiers to which it is not entitled, and their HIBE and HIBS schemes are provably secure in such an attack model [24].

## 4 Removing Key Escrow

Key escrow is a feature of the HIBE and HIBS schemes used in PECF-GSI, as with other standard identity-based cryptographic schemes. This may not be acceptable for certain grid applications. One way of solving the key escrow problem is to apply the Al-Riyami–Paterson HCLE and HCLS schemes to PECF-GSI.

In order to use the Al-Riyami–Paterson HCLE and HCLS schemes (see Section 2.1.2), we need to modify the keys used in the HIBE/HIBS schemes. An entity's private key in the HCLE and HCLS schemes is the product of the entity's private key and its chosen secret value for the Gentry–Silverberg HIBE and HIBS schemes. For instance,  $R$ 's long-term private key in the HIBE/HIBS schemes is  $s_0 P_R$ ; hence  $R$ 's new private key for the HCLE/HCLS schemes would be  $s_R s_0 P_R$ , where  $s_R$  is a secret value that  $R$  uses to extract private keys for its immediate lower-level entities. The public key of  $R$ , which comprises two components, is now  $\langle s_R P_0, s_R Q_0 \rangle$ . The lower section of Table 1 summarises the new key sets required for the Al-Riyami–Paterson HCLE and HCLS schemes.

### 4.1 Single Sign-on Without Key Escrow

The changes that we have to make to PECF-GSI in order to remove the key escrow issue are rather trivial. When  $A$  performs a single sign-on, she and her authentication server  $R$  first perform mutual authentication using a shared password, and then establish a secure channel (as before). Subsequently,  $R$  runs the PARTIAL-PRIVATE-KEY EXTRACT algorithm and issues a partial private key  $(S_R + s_R P_{\bar{A}})$  to  $A$ , along with other information such as the system parameters and an updated IRL.

When  $A$  receives the partial private key, she runs the LOWER-LEVEL SETUP algorithm to randomly pick a secret value  $s_{\bar{A}}$ . This secret value, in turn, is used to compute her proxy public/private key pair.  $\bar{A}$  can then use the proxy

credential to perform mutual authentication and delegation with a hosting server. Since the value  $s_{\bar{A}}$  is unknown to  $R$ ,  $A$ 's new proxy private key is kept secret from  $R$ , which is not the case in the protocols described in Section 3.3.

## 4.2 Security Concerns

The cryptographic set-up in CL-PKC allows users to create more than one public key for the same partial private key [3]. For example,  $A$  can randomly select two different secret values  $s_{\bar{A}}$  and  $s'_{\bar{A}}$ , and compute two sets of distinct public/private key pairs. However, we believe that this property would not cause any major issues within a grid environment. This is because partial private keys produced by the authentication server are short-lived. In fact, the users can take advantage of this property by extracting different proxy key pairs for different job submissions to increase key freshness, before the expiry of their respective partial private keys.

In the context of CL-PKC, we must trust the authentication server not to mount active impersonation attacks against its users. Such attacks are possible because the authentication server can always select a secret value (for some ‘‘victim’’) and calculate a private key based on the victim’s partial private key to which it necessarily has access. However, these attacks would leave behind cryptographic evidence which may reveal the authentication server’s actions. We also note that traditional PKIs have an analogous problem: we have to trust a CA not to illegally sign user certificates, enabling the CA to impersonate these users to other parties.

## 5 Performance

In this section, we compare the communication costs of the protocols used in GSI and PECF-GSI for key agreement and delegation. We then compare the computational costs of long-term and proxy key generation, key agreement and delegation.

In the GSI, we assume the size of a 1024-bit RSA public key certificate is 1.5 kilobytes (ignoring small fields, such as subject and validity period). Similarly, a 512-bit RSA proxy certificate is 0.8 kilobytes.<sup>10</sup> Ciphertexts and signatures generated using a short-term RSA key are 512 bits.

For PECF-GSI, we work with a supersingular elliptic curve of embedding degree 4 over  $\mathbb{F}_{2^{271}}$  [20, 22] to obtain the system parameters described in Section 3.2.<sup>11</sup> This

<sup>10</sup>It is worth mentioning that RSA keys can be replaced by much shorter keys, which are based on elliptic curve cryptography (ECC) [12]. However, this does not eliminate the fact that certificates will still be in use, and hence, the associated limitations of certificated-based architectures.

<sup>11</sup>We note that this curve is only chosen so that concrete timings and bit counts can be given. A wide variety of other choice of curves and their associated parameters are available.

choice results in a corresponding group of prime order  $q$  approximately equal to  $2^{252}$ , and gives roughly the same security level as 1024-bit RSA. Using the point compression technique, 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 [7].

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 252 bits. Note that the size of outputs of  $H_2$  and  $H_5$  are dependent on  $n$ , the bit length of plaintexts. We assume that  $n = 256$ , since this is sufficient for our protocol messages (see Section 3.3.2). Hence, the size of ciphertexts and signatures produced by the Gentry–Silverberg HIBE and HIBS schemes (or the Al-Riyami–Paterson HCLE and HCLS schemes) can be computed, and are 1056 bits and 816 bits, respectively.

The estimated communication costs for the protocols that underpin the GSI and PECF-GSI are summarised in Table 2. The architecture based on the Gentry–Silverberg schemes is denoted by PECF-GSI-I, while the architecture based on the Al-Riyami–Paterson schemes is denoted by PECF-GSI-II. Actual computational costs in milliseconds are also summarized in this table. These timings were obtained by implementing the key generation algorithms in RSA and the Gentry–Silverberg HIBE/HIBS schemes using the MIRACL library [45]. The experiments were performed on a Pentium IV 2.4 GHz processor. For simplicity, we limit the length of the delegation chain to one. Computational costs are not currently available for PECF-GSI-II.

### 5.1 Communication Costs

In the GSI, the communication cost of the key agreement protocol through the standard TLS handshake is approximately 37.8 kilobits; the corresponding cost in PECF-GSI-I (with key escrow) is approximately 1.9 kilobits. Note that for simplicity, we ignore small components in both the TLS protocols, such as the `ClientHello` and `ClientFinished` messages. Key agreement in PECF-GSI-II, which does not have key escrow and so makes use of additional public key components, has a slightly higher communication cost compared to key agreement in PECF-GSI-I.

The communication costs for delegation in the GSI can be estimated straightforwardly from the protocol described in Section 2.2. Delegation in PECF-GSI-I is very lightweight because it only involves issuance of a private key. In PECF-GSI-II, additional public key components are included as well, and hence extra bandwidth is required. It is obvious that our certificate-free approach suits wire-

**Table 2. A comparison of performance characteristics.**

| Type of Cost (units)  | Operation                | GSI    | PECF-GSI-I | PECF-GSI-II |
|-----------------------|--------------------------|--------|------------|-------------|
| Communication (KB)    | Key agreement            | 37.8   | 1.9        | 2.4         |
|                       | Delegation               | 7.8    | 0.3        | 0.8         |
| Computation time (ms) | Long-term key generation | 149.90 | 1.69       |             |
|                       | Proxy key generation     | 34.85  | 1.74       |             |
|                       | Key agreement            | 5.34   | 28.95      |             |
|                       | Delegation               | 38.33  | 10.16      |             |

less environments well, in which transmission of data using battery-powered mobile devices is a relatively expensive operation.

## 5.2 Computational Costs

We can see from Table 2 that key generation in PECF-GSI is significantly more efficient than RSA key generation in the GSI.<sup>12</sup> This is, to some extent, an unfair comparison, because of the completely different mathematical properties behind these two approaches, but it does suggest that the single sign-on protocol in our proposal is less computationally expensive than in the GSI incorporating MyProxy.

The figures for key agreement (including mutual authentication) are obtained by summing the times taken for the user and authentication server to perform their respective parts of the protocol. A similar method is used to obtain a single figure for the computational costs of delegation [31, Table 4.3]. Key agreement in the GSI (using the standard TLS protocol) is computationally less expensive than the corresponding operations in PECF-GSI (using the modified identity-based TLS protocol). In contrast, delegation in PECF-GSI is almost four times faster than in the GSI.

It is unfortunate that key agreement is slower in PECF-GSI, but we note that the cumulative time for key agreement and delegation is still lower in PECF-GSI. Overall, it can be seen that the computational costs of PECF-GSI are comparable to those of the GSI. Our approach allows a different trade-off to be struck between the computational costs at the user side and at the server side.

## 6 Conclusions and Future Work

We have proposed a grid security infrastructure which is password-enabled and certificate-free. Our infrastructure offers three distinct advantages.

<sup>12</sup>Note that we only compare private key extraction in PECF-GSI to RSA public/private key pair generation in the GSI, because the time taken to compute a public key using the hash function  $H_1$  in PECF-GSI is negligible given the parameters we have chosen. Furthermore, construction of a public key by hashing an identifier occurs as part of the associated encryption/decryption scheme.

- The only long-term secret required by users is a password. This is likely to improve usability and accessibility of grid applications considerably. Moreover, we do not need to worry about revocation of users' public keys, a considerable problem in certificate-based architectures.
- Key agreement and delegation in our approach requires much less bandwidth than the GSI. In addition, our delegation technique requires only a single verification.
- The computational effort required by the key generation algorithms that we employ is considerably lower than in the GSI.

Our lightweight security architecture is more suitable than the GSI for use by devices with limited resources, thereby significantly extending the number of devices that can interact with computational grids and going some way to realising the potential of wireless grids. That said, identity-based and certificateless public key cryptography are relatively new, and thus lack support from standardisation bodies. This may hinder early adoption of our proposal.

To meet the requirement of grid applications which do not tolerate key escrow, we proposed the use of certificateless public key cryptography, which enables users to select their own private components. This only resulted in minor changes, in terms of key set-up, to our original architecture.

An important security aspect of grid applications which we have not considered in this paper is authorization. A natural extension of this work is to develop novel authorization techniques using properties of identity-based cryptography.

We are also aware that there are other aspects of performance that ought to be considered, such as fault tolerance and availability of our architecture in comparison with MyProxy. Since MyProxy still continues to evolve, it will be appropriate to make such comparisons in the near future.

**Acknowledgements** The authors would like to thank Jim Basney for helpful discussions and the anonymous referees for useful comments.

## References

- [1] M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval. Provably secure password-based authentication in TLS. In *Proceedings of the 1st ACM Symposium on InformAtion, Computer and Communications Security (ASI-ACCS 2006)*, pages 35–45. ACM Press, March 2006.
- [2] S.P. Ahuja and J.R. Myers. A survey on wireless grid computing. *The Journal of Supercomputing*, 37(1):3–21, 2006.
- [3] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In C.S. Lai, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2003*, pages 452–473. Springer-Verlag LNCS 2894, November 2003.
- [4] S.S. Al-Riyami and K.G. Paterson. *Certificateless Public Key Cryptography*. Cryptology ePrint Archive, Report 2003/126, October 2003. Available at <http://eprint.iacr.org/2003/126>.
- [5] A. Alsaid and C.J. Mitchell. Installing fake root keys in a PC. In D. Chadwick and G. Zhao, editors, *Proceedings of the 2nd European Public Key Infrastructure Workshop (EuroPKI 2005)*, pages 227–239. Springer-Verlag LNCS 3545, 2005.
- [6] K. Barr and K. Asanović. Energy aware lossless data compression. *ACM Transactions on Computer Systems*, 24(3):250–291, August 2006.
- [7] 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.
- [8] J. Basney, M. Humphrey, and V. Welch. The MyProxy online credential repository. *Journal of Software: Practice and Experience*, 35(9):817–826, July 2005.
- [9] B. Beckles, V. Welch, and J. Basney. Mechanisms for increasing the usability of grid security. *International Journal of Human-Computer Studies*, 63(1-2):74–101, July 2005.
- [10] 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.
- [11] M. Bellare and P. Rogaway. *The AuthA Protocol for Password-Based Authenticated Key Exchange*. Contribution to IEEE P1363, March 2000.
- [12] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Möller. Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS). *The Internet Engineering Task Force (IETF)*, RFC 4492, May 2006.
- [13] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2005*, pages 440–456. Springer-Verlag LNCS 3494, 2005.
- [14] 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, August 2001.
- [15] T. Dierks and C. Allen. The TLS protocol version 1.0. *The Internet Engineering Task Force (IETF)*, RFC 2246, January 1999.
- [16] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputing Applications*, 11(2):115–128, 1997.
- [17] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, San Francisco, 2004.
- [18] 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 (CCS '98)*, pages 83–92. ACM Press, November 1998.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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, May 2003.
- [24] 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, December 2002.
- [25] GridCafé. *Grid Projects in the World*. Available at <http://gridcafe.web.cern.ch/>, last accessed in January 2007.
- [26] P. Gutmann. Plug-and-play PKI: A PKI your mother can use. In *Proceedings of 12th USENIX Security Symposium*, pages 45–58, 2003.
- [27] J.M. Hayes. The problem with multiple roots in web browsers - certificate masquerading. In *Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 306–311. IEEE Computer Society Press, 1998.
- [28] 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.
- [29] 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.

- [30] 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.
- [31] H.W. Lim. *On the Application of Identity-Based Cryptography in Grid Security*. Ph.D thesis, University of London, 2006.
- [32] 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.
- [33] J. Linn. Generic security service application program interface version 2, update1. *The Internet Engineering Task Force (IETF)*, RFC 2743, January 2000.
- [34] L.W. McKnight, J. Howison, and S. Bradner. Wireless grids: Distributed resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, 8(4):24–31, July/August 2004.
- [35] 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.
- [36] B.C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.
- [37] 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.
- [38] The OpenSSL Project. *OpenSSL: The Open Source Toolkit for SSL/TLS*, 2006. Available at <http://www.openssl.org/>, last accessed in September 2006.
- [39] 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.
- [40] 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.
- [41] T. Phan, L. Huang, and C. Dulan. Challenge: Integrating mobile wireless devices into the computational grid. In *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MOBICOM 2002)*, pages 271–278. ACM Press, 2002.
- [42] 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.
- [43] R.S. Sandhu, M. Bellare, and R. Ganesan. Password-enabled PKI: Virtual smartcards versus virtual soft tokens. In *Proceedings of the 1st Annual PKI R&D Workshop*, pages 89–96, 2002.
- [44] 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, August 1985.
- [45] Shamus Software Ltd. *MIRACL*. Available at <http://indigo.ie/~mscott/>, last accessed in January 2007.
- [46] 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.
- [47] The TeraGrid Project. *TeraGrid*. Available at <http://www.teragrid.org/>, last accessed in January 2007.
- [48] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M.R. Thompson. Internet X.509 public key infrastructure proxy certificate profile. *The Internet Engineering Task Force (IETF)*, RFC 3820, June 2004.
- [49] 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, April 2004.
- [50] 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.

## A A Password-Based TLS Protocol

Figure 4 shows the EC-SOKE-TLS protocol, an adaptation of the Simple Open Key Exchange for TLS (SOKE-TLS) [1] to the elliptic curve setting.

|     |                     |  |
|-----|---------------------|--|
| (1) | $A \rightarrow R$ : | ClientHello = $n_A$ ,<br>session_id,<br>cipher_suite   |
| (2) | $R \rightarrow A$ : | ServerHello = $n_R$ ,<br>session_id,<br>cipher_suite,<br>ServerKeyExchange = $ID_R$ ,<br>$rP_0$ ,<br>ServerHelloDone |
| (3) | $A \rightarrow R$ : | ClientKeyExchange = $ID_A$ ,<br>$\{aP_0\}_{\pi_A}$   |
| (4) | $R \rightarrow A$ : | ServerFinished   |
| (5) | $A \rightarrow R$ : | ClientFinished   |

**Figure 4. EC-SOKE-TLS protocol**

We assume user  $A$  (playing the role of client) and authentication server  $R$  (playing the role of server) share a password  $\pi_A$  and some parameters required for the protocol, such as  $\mathbb{G}_1$ ,  $P_0$  and  $H_1$ . We use  $\{\cdot\}_\pi$  to denote a mask generation function which maps an element of  $\mathbb{G}_1$  into another element of  $\mathbb{G}_1$  by using the password  $\pi$ . In our case, the mask generation function is defined to be the addition of a group element and a password (as described in Section 3.3.1).

In step (1),  $A$  sends  $R$  a standard ClientHello message as in the standard TLS protocol. Here,  $n_A$  is a random number generated by  $A$ .

In step (2),  $R$  responds with a ServerHello message which contains a different random number  $n_R$  and other associated information.  $R$  also randomly picks  $r \in \mathbb{Z}_q^*$ , calculates its Diffie-Hellman value as  $rP_0$  and forwards the ServerKeyExchange message to  $A$ . The ServerHelloDone message is sent to indicate the end of step (2).

In step (3),  $A$  randomly selects  $a \in \mathbb{Z}_q^*$  and computes  $aP_0$ . This Diffie-Hellman value is then encrypted (or masked) using  $A$ 's password  $\pi_A$  and forwarded to  $R$ , along with her identity.

$A$  and  $R$  compute a pre-master secret

$$pms = H_0(ID_A, ID_R, \pi_A, \{aP_0\}_{\pi_A}, yP_0, ayP_0),$$

where  $H_0$  is a secure hash function; the associated master secret is

$$ms = \text{PRF}(pms, \text{"master secret"}, n_A, n_X),$$

where PRF is a pseudo-random function specified for the standard TLS protocol [15].

Note that  $R$  can compute  $pms$  if and only if it can recover  $aP$  from  $A$  and compute the composite Diffie-Hellman value  $ayP_0$ . On the other hand,  $A$  must know the value  $a$  of  $aP$  that  $R$  would recover from her password  $\pi_A$ . This is to prevent an adversary from impersonating  $A$  to  $R$  using a guessed password  $\pi'_A$  by computing  $\{a'P_0\}_{\pi'_A}$ , where  $a'$  is a value which is known to the adversary. This impersonation would fail unless the adversary could predict the value  $a''$  of  $a''P_0$  that  $R$  recovers using the correct password  $\pi_A$  from  $\{a'P_0\}_{\pi'_A}$ . The difficulty of finding  $a''$  is believed to be as hard as solving the ECDL problem.

In step (4),  $R$  produces the ServerFinished message, which contains the verification value:

$$\text{PRF}(ms, \text{"server finished"}, h_1, h_2)$$

where  $h_1$  and  $h_2$  represent hash values of all handshake messages up to but not including this message, using different hash functions.

Note that  $A$  must verify if  $R$  has computed the correct value in the ServerFinished message before calculating her corresponding verification value in the ClientFinished message in the last step. This is to prevent a bogus server from impersonating  $R$  to  $A$  and mounting an offline dictionary attack. If  $A$  sends the ClientFinished message immediately after sending the ClientKeyExchange message, the bogus server  $R'$  can test if a candidate password  $\pi'_A$  is correct by performing the following steps:

1. decrypt  $\{aP_0\}_{\pi_A}$  from  $A$  using its guessed password  $\pi'_A$  and obtain  $a'P$ ;
2. compute the corresponding pre-master secret  $pms'$  and master secret  $ms'$  using  $\{aP_0\}_{\pi_A}$ ,  $yP_0$ ,  $a'yP$ , assuming  $R'$  knows the value  $y$ ;
3. compute the verification value of the ClientFinished message using  $pms'$  and  $ms'$ ;
4. compare the verification value obtained in step (3) with the value that  $R'$  received from  $A$ . A match between these two values indicates a correct guess. Otherwise,  $R'$  repeats steps (1) to (4) using a different candidate password  $\pi''_A$ .

$A$  is authenticated to  $R$  if the last message from  $A$  contains the correct verification value. The master\_secret is subsequently used to derive further keys for the TLS record layer.