Towards Unified Public Key Management through Dynamic Key Infrastructure (Position Paper)

Hoon Wei Lim
Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
h.lim@rhul.ac.uk

31 March 2005

Abstract

In this paper, we introduce the concept of unified public key management. Increasing number of new emerging computing technologies with widening applications in more dynamic environments have prompted a need for more uniform and flexible public key management techniques, which may complement a simpler and more dynamic security infrastructure. Through the use of a static certificate which contains a fixed parameters set, lightweight dynamic key generation and non-interactive key update of a user can be achieved. This paper proposes a dynamic key infrastructure (DKI) that lays the foundation and framework for a simpler and unified public key management approach.

1 Introduction

We are now living in a digital age where human beings benefit greatly from fast evolving computing technologies. From Peer-to-Peer (P2P) computing for home users' file sharing, to Grid computing for advanced science and engineering applications, to pervasive computing for mobile and ubiquitous systems, and other new emerging technologies, the communication models adopted are becoming rather complex, dynamic, and decentralised. It is natural that we have been relying heavily on security architecture built during the early development of traditional distributed computing technologies which has been proven to be sufficient for most conventional security requirements. As human endeavour seeks for continuous improvement in communication technologies, the needed ideal security infrastructure seems to have been changing to align with the loosely structured, flexible, and globalised environments.

Public key infrastructure (PKI) was invented to facilitate the growth of practical use of public key cryptosystems. A typical convention use of a public key is for secure transportation of a session key from one user to another intended recipient. Also, the corresponding private key can be used by the user to generate a digital signature on a message. It is a common case in which the public key is long-term and carried by a standard X.509 public key certificate [12], and it can be renewed typically once a year by contacting the Certificate Authority (CA) who issued the certificate. The renewal here essentially involves re-signing of the certificate by the CA with a new validity period and updating of the long-term public/private key pair. While this appears to work perfectly in most existing systems, it may not be the case when it is deployed in some new emerging dynamic distributed systems such as pervasive and Grid systems which tend to require extensive use of certificate, regular key generation and update, interactive key distribution, and so on. The computational and communication overheads incurred may add extra burden to the security infrastructure and result a less scalable system.

In this paper, we attempt to unify the management of public key and yet allow diversification of public key through a concept which we term it as *master-public-key mould* based on some properties from identity-based cryptography (IBC) [1, 18]. As with a normal public key, the master-public-key mould is published and carried by a certificate. However, it allows a user who is in possession of the master-public-key mould to create different kinds of unique dynamic public keys without interacting with the CA. Any user can verify the authenticity of the dynamic public keys based on the certificate which contains the relevant master-public-key mould. We propose a dynamic key infrastructure (DKI), adopted from the dynamic key infrastructure for GRID (DKIG) [13], to facilitate distribution, management, and usage of master-public-key mould. Our approach seems to offer simpler and lightweight public key management, and it may fit nicely into dynamically changing environments which require heavyweight key management through the PKI setting.

Related Work. In [13], Lim and Robshaw proposed a dynamic key infrastructure for GRID (DKIG) which seems to offer a more lightweight and dynamic key management approach within a GRID environment. They proposed the use of fixed parameters set for each user which allows computation of ephemeral public key on-the-fly by other entities to achieve single sign-on and delegation. This is made possible through some properties from IBC. Recently, Chen *et al.* [3] presented their work on user-friendly GRID protocols which is based on the use of dynamic public/private key pair as a function of a time period, a concept related to [13].

Contributions. In this paper, we generalise and build on the ideas from [3, 13], and provide a proposal emphasising on the shift of paradigm in public key management, which may be useful in other applications. We highlight our contributions as follows.

- Proposal of a unified public key management approach through the use of master-public-key mould.
- A generic framework of a dynamic key infrastructure.
- Classification of dynamic public key including a new technique of deriving a secret public key.
- Some applications of dynamic key infrastructure in pervasive and GRID computing.

Organisation. The remainder of this paper is organised as follows. In Section 2, we briefly review some background information on pairings and Boneh and Franklin's identity-based encryption scheme. Section 3 presents our proposal on a dynamic key infrastructure which includes definition of a master-public-key mould and methods of unifying public key management. In Section 4, we classify different roles for a dynamic public key which can be derived from a master-public-key mould. In Section 5, we give some examples on the applications of our proposal in pervasive and GRID distributed systems. Section 6 concludes the paper.

2 Background on 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 (e.g. a subgroup of the group of points on an elliptic curve over a finite field), and $\mathbb{G}_2$ denotes a related multiplicative group (e.g. a subgroup of the multiplicative group of a related finite field). A pairing, which can be either a modified Weil pairing [1] or a Tate pairing [8, 9] is a bilinear 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

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

  Also, 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 computed in polynomial time.

For any $a \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, we write aP as an elliptic curve point multiplication which can be computed efficiently. However, the problem of finding a when given aP is believed to be intractable. This is known as the *elliptic curve discrete logarithm* (ECDL) problem.

The other relevant mathematical problem which may be used later on is called the *Computational Diffie-Hellman* (CDH) problem: given $\langle P, aP, bP \rangle \in \mathbb{G}_1$ for any $a, b \in \mathbb{Z}_q^*$, compute abP. We say that the CDH assumption is satisfied if there exists no probabilistic algorithm which can solve the CDH problem with non-negligible advantage within polynomial time.

2.1 The Boneh-Franklin Identity-Based Encryption Scheme

In 2001, Boneh and Franklin [1] successfully proposed a fully functional and practical Identity-based Encryption (IBE) scheme, answered Shamir's call back in 1984 on development of an encryption scheme for its novel design of an identity-based cryptosystem [18]. In an identity-based cryptosystem, a user's public key can be generated from some public information such as e-mail address and IP address, which can be used to represent the identity of the user. The corresponding private key is generated and maintained by a trusted third party called Private Key Generator (PKG). The following four algorithms underpin Boneh and Franklin's IBE scheme.

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

(i). specifies two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$;

(ii). chooses an arbitrary generator $P \in \mathbb{G}_1$;

(iii). defines two cryptographic hash functions, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_1^* \rightarrow \{0,1\}^n$ for some n; and

(iv). picks a master secret key $s \in \mathbb{Z}_q^*$ at random and computes the matching public key as sP.

The system or public parameters are $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, sP, H_1, H_2 \rangle$. This algorithm is normally run by the PKG whereby (sP, s) becomes its public/private key pair.

EXTRACT: This algorithm is usually run by the PKG to extract a private key when given an arbitrary identity-based public key Q as sQ, where $Q = H_1(\mathrm{ID}) \in \mathbb{G}_1^*$ and ID is an arbitrary identifier.

ENCRYPT: To encrypt a message m under an identifier ID, the public key is $Q = H_1(\mathrm{ID})$. The resulting cipertext is $c = (U, V) \in \mathcal{C}$, where $U = rP$, $V = m \oplus H_2(\hat{e}\,(Q, sP)^r)$, and $r \in \mathbb{Z}_q^*$ is a random integer.

DECRYPT: To decrypt a ciphertext c encrypted using the identifier ID with its corresponding private key $sQ \in \mathbb{G}_1^*$, the plaintext is $m = V \oplus H_2(\hat{e}\,(sQ, U))$.

Although IBE seems to eliminate certificate management issues, the trade-off is that IBC-based systems inherit key escrow and private key distribution problems. The authors of [3, 13] attempt to solve these problems by proposing the idea of each entity acts as his PKG. In this paper, we build on the similar approach and extend it to a broader and generic context in an attempt to achieve simplification and unification of public key management.

3 Dynamic Key Infrastructure

We propose a user-centric unified public key management approach which seems to provide users with more flexible and easier key generation and update techniques. They can be useful in some dynamically changing environments which require constant key management from the

users. Dynamic Key Infrastructure (DKI) captures the notion of dynamic and diversified usage of public key, e.g. a dynamic public key can be a long-term or short-term key, or it can also has other roles, which is difficult to emulate with current PKI technologies.

3.1 The Notion of Master-Public-Key Mould

We envisage that heavyweight public key management for some distributed environments in the conventional PKI setting can be made simpler and more lightweight through a more unified approach, which may be achieved through the use of a master-public-key mould (we refer as key mould henceforth). A key mould which can be published in the form of a certificate encompasses some fixed public information of a user. It will not be used for message encryption but rather to derive a public key associated with the owner of the key mould on-the-fly. We envisage that a key mould has the following properties.

- *Long-lived*: The key mould is permanent and fixed unless its associated master secret key is compromised. So long as the owner keeps his master secret key in a secure platform such as a smart card and a highly secure centralised server, its exposure may well be minimal as it is only used for extraction of private key and will not be used in message decryption or signing.
- *Multipurpose*: The key mould can be used to derive as many distinguished public keys as the user wishes without getting them certified by the CA before they are used in various applications.

Besides that, the key mould also contains public information that assists a message encryption operation.

3.2 Key Setup

In the DKI setting, initially each user runs the Setup algorithm as described in the IBE scheme. This includes choosing a generator $P \in \mathbb{G}_1$ of order q and selecting a random integer $s \in \mathbb{Z}_q^*$. The parameters set of $\langle \hat{e}, \mathbb{G}_1, \mathbb{G}_2, H_1, H_2, P, sP \rangle$ becomes the key mould and the associated master secret key is s.

3.3 Key Mould Certification and Renewal

As with conventional verification and certification of a public key by a CA, we assume that a key mould can be published through a standard X.509 certificate. The steps involved in a key mould validation and certification by the CA are as follows.

(i). The user submits his master-public-key mould to the CA through an integrity protected channel.

(ii). The CA verifies the user's identity by checking his passport (or national ID-card). The user must also show his proof of possession of the matching master secret key. Once the checks succeed, the CA issues a signed certificate that embeds the key mould.

(iii). The new certificate is forwarded to the user.

Depending on the security policy of an organisation, the user may need to renew his certificate every year. Despite that, we expect that the key mould is permanent and will not be altered unless such a need arises, e.g. master secret key compromise or personal preference. This makes certificate renewal in the DKI setting as a matter of signing of the certificate (with new serial number) without updating the key mould itself. This seems to be an easier task in comparison with renewal of current public key as it is a common case that the user's public/private key pair themselves need to be updated.

3.4 Dynamic Key Derivation

We now illustrate how a dynamic public key can be computed in the DKI setting. Generally speaking, a user's identifier is in the form of

$$\mathrm{ID} = \texttt{Name} \parallel \texttt{App_ID} \parallel \langle \mathrm{Type} \rangle$$

where `Name` refers to the entity name of the user; `App_ID` represents the unique identity number of an application where the public key will be used; and $\langle \mathrm{Type} \rangle$ indicates a field which contains information such as date, nonce, secret value, or a counter, depending on the role of the dynamic public key (this will be discussed in Section 4). Based on the input for the user's identifier, his dynamic public key can be computed as $Q \leftarrow H_1(\mathrm{ID}) \in \mathbb{G}_1$, where H_1 is a collision resistant hash function: $\{0,1\}^* \rightarrow \mathbb{G}_1$.

The corresponding dynamic private key is sQ, with s is the associated master secret key. Note that finding s when given P and sP is the ECDH problem, which is intractable. Therefore, even though P, sP, and Q are publicly known to all users, computing sQ is believed to be equally difficult. Even if sQ is compromised, the adversary will not be able to compute other private keys $sQ_1, sQ_2, \ldots, sQ_n$ corresponding to different dynamic public key strings $Q_1, Q_2, \ldots, Q_n$, where $Q \neq Q_1 \neq Q_2 \cdots \neq Q_n$ assuming n is sufficiently large so that the key space is big enough to avoid any repeated key. This is due to the computationally infeasible CDH problem in polynomial time.

We also remark that computation of new public/private key pair in our approach may well be significantly more efficient than the RSA key pair generation. This is mainly because finding two large primes for the RSA keys is a rather costly operation while computation of a private key $sQ \in \mathbb{G}_1$ is merely a scalar point multiplication on elliptic curve.

3.5 Certificate Revocation

We adopt the certificate revocation method defined for X.509 public key certificate through a signed data structure called a *Certificate Revocation List* (CRL) [11]. When the user verifies a signature of another party using a dynamic public key of that party, he must not only check the signed certificate that carries the associated master-public-key mould, but also acquire a recently updated CRL and check if the certificate serial number is on the CRL. One limitation of this approach is that the time granularity of revocation is limited to the CRL issue period. Depending on the local policy, should the system does not tolerate some latency in notification of the up-to-date CRL, an online method of notification called *Online Certificate Status Protocol* (OCSP) [14] can be used instead. The trade-off is that an additional trusted online service has to be integrated into the system as well.

4 Different Roles of Dynamic Public Key

In this section, we define different possible types and roles of dynamic public keys. The potential usage of these keys will be described in Section 5.

Long-term Key Pair It is envisaged that a long-term key pair in the DKI setting is analogous to the long-term keys in the current PKI setting which typically can be valid up to a year. Essentially a long-term public key comprises the user's identity concatenated with the serial number of his certificate which carries the key mould, as follows.

$$PK = H_1(\texttt{Name} \parallel \texttt{App_ID} \parallel \texttt{Serial_No})$$

Any other entity who possesses a copy of the user's certificate can compute the public key with ease. Just before the end of current validity period, the user can request his CA to re-sign his certificate with new expiry date and serial number.

The usage of long-term keys is common to most current applications for: (i) entity authentication through digital signature using the long-term private key; and (ii) a session key transportation by encrypting of the session key using the long-term public key.

Short-term Key Pair There are two common practice in protecting a long-term private key. First it is typically protected by storing it in a file with restricted access, i.e. storing it in an encrypted file with a decryption passphrase. Alternatively, the key can be stored on a hardware token (e.g. a smart card) that is protected by a password. The second approach, which can be performed in addition of the first method, is to limit the exposure of the private key by regularly issuing a new short-lived public/private key, usually on the order of hours or days. It is a convention in cryptography and computer security that if a cryptographic key is short-lived, then the level of protection can be eased. For example, the temporary private key can be stored unencrypted on a local file system, protected only by file system permissions. The introduction of *X.509 proxy certificate* for short-lived public key by Tuecke *et al.* [19, 20] shows the increasing popularity of this approach.

An ephemeral public key can be calculated with an identifier that includes a current date such as the following.

$$\bar{PK} = H_1(\texttt{Name} \parallel \texttt{App_ID} \parallel \texttt{Date})$$

Again, a short-term public key which has only one-day lifetime, can be computed by anyone without interacting with the key's owner provided the entity who computes the key has a copy of the appropriate key mould.

Secret Public Key A public key can be turned into secret public key by keeping it secret and only known between the sender and the receiver. Here, we demonstrate that a secret public key can be generated immediately based on a shared secret between the two communicating parties as follows.

$$\hat{PK} = H_1(\texttt{Name} \parallel \texttt{App_ID} \parallel \texttt{Secret})$$

In a secret public key protocol proposed by Gong *et al.* [10], the two communicating parties can exchange session key establishment messages using a secret public key so that the encrypted messages are difficult to be guessed by the adversary. Our technique in generating a secret public key seems to be more elegant as compared with the method in [10] as the latter requires distribution of a new secret public key to the other party through the shared secret key before the secret public key can be used.

Multipurpose Key It seems to be a common practice for a user to own several sets of public/private key pairs for different applications or devices. Even though it is believed that this approach may be a good practice, managing many key pairs would seem to be rather tedious, in particular in key update and renewal. We envisage that the use of a key mould can be viewed as using a multipurpose smart card which incorporates several functions or roles such as identity card, driving license, and credit card. Assuming the user has many personal computing devices, he may only need to certify his key mould through the CA and subsequently, create a public key for each device by concatenating the media access control (MAC) address of the device as follows.

$$PK = H_1(\texttt{Name} \parallel \texttt{App_ID} \parallel \texttt{MAC_Address} \parallel \langle \text{Type} \rangle)$$

Note that `MAC_Address` is public information and thus other entities can compute the public key in which its matching private key can only be extracted by the key mould owner. Depending on the system requirements, two communicating parties can add in more granularity into the public key string by using nonce, date, secret value, and so forth in $\langle \text{Type} \rangle$.

5 Applications

The DKI setting seems to be suitable for environments which require frequent generation, update, distribution, or revocation of key. We identify two emerging distributed computing technologies which may gain some benefits through our approach.

5.1 Pervasive Computing

One of the most important current emerging technologies is pervasive computing [2] which gives a new dimension of personal computing that integrates mobile communication, ubiquitous embedded computer systems, consumer electronics, and the global reach Internet.

Generally speaking, one way of securing communication between a user (potentially from his mobile device) and a content (or application) provider within a Wireless Application Protocol (WAP) based pervasive computing environment is through WAP PKI (WPKI) [16]. WPKI adopts similar technologies for PKI but it is used for wireless environments and it supports standard X.509 public key certificate. Figure 1 shows a secure WAP session between a user and a content provider. A WAP session security can be provided through a Wireless TLS (WTLS) or a TLS session between the user and the WAP gateway which is supported by WPKI, and a standard TLS session in PKI between the gateway and the content server.

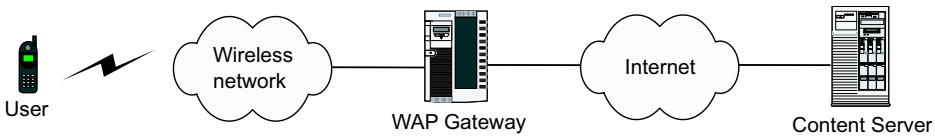

Figure 1: A WAP session through WPKI.

We identify some possible advantages of using DKI in such an environment.

- *Key Revocation*: In the event of the long-term private key of either the gateway or the server being compromised, conventional means of notifying the user through CRL or OCSP are not feasible as the mobile device does not normally have the local resource nor the communication bandwidth to perform such tasks as in the wired PKI-based setting. To satisfy this revocation requirement, therefore, Open Mobile Alliance [16] proposed the use of short-lived certificate. In their proposal, the server (or the gateway) is authenticated during the certification of its long-term public key (typically once a year). However, rather than signing the certificate with one year validity period, the CA will issue a new certificate with a very short lifetime (e.g. 48 hours) everyday which carries the same public key. When the CA wants to revoke the server's public key, it simply stops issuing further short-lived certificates. This approach seems to impose considerably high computational costs on the CA if it is required to sign a large number of certificates everyday.

  We propose a method which appears to be more efficient and elegant than the one in [16]. As with [16], the CA is required to certify the server's key mould once a year. However, no daily issuance of certificate is needed as each entity in Figure 1 can compute short-term public keys of the other entities on-the-fly, assuming each entity has possession of the certificates of the others. For instance, when the user is engaged with the gateway in a secure WAP session through the WTLS (or TLS) protocol, he can derive a short-term public key of the gateway (G) by using current date and the gateway's key mould as follows.

  $$\bar{PK}_G = H_1(\texttt{Name} = G \parallel \texttt{App_ID} = \text{wap12345} \parallel \texttt{Date} = 01012005)$$

  Then, as part of the WTLS handshake protocol, the user can encrypt a pre-master secret with $\bar{PK}_G$ and send it to the gateway. We remark that in the encryption process, $s_G P$ (where s_G is G's master secret key and P is its chosen generator) which can be extracted from the gateway's key mould, is used through the IBE scheme described earlier. Therefore, only the gateway could have been able to obtain the matching private key and decrypt the message. By using only short-term keys, the compromise of a short-term private key will not expose other future or past short-term private keys. Thus, even if the gateway's short-term private key was compromised the day before, the user can still check the authenticity of the gateway through a fresh short-term public key because only the holder of the appropriate master secret key can derive the matching short-term private key.

- *Multiple Key Management*: The other potential benefit of implementing DKI within a pervasive computing environment is that it seems to offer a more user-centric key management approach. A home user who has many mobile computing devices can essentially obtain only one certificate for his key mould and subsequently use it to manage the public keys for those devices by himself. Assuming that a user A has m computing devices $D_1, D_2, \ldots, D_m$. She can compute each device a public key PK_i, where $1 \leq i \leq m$, in the following form.

$$PK_i = H_1(\texttt{Name} = A \parallel \texttt{App_ID} = \text{app999} \parallel \texttt{MAC_Address}_i \parallel \langle \text{Type} \rangle)$$

  When A's key mould is expiring, all she needs to do is to get it recertified by the CA without updating or revoking each individual public key for all her devices.

5.2 Grid Computing

Grid computing [5, 7] has emerged as a fast-evolving and yet another important field which has gained substantial attentions from multidisciplinary researchers worldwide due to its broad applicability. Grid differs from conventional distributed computing in its focus on large-scale and dynamic resource sharing that provides an unheard level of massive resource including processing power, storage capacity, network band-width, databases, applications, and any other devices (hardware) or components (software) which are required to complete resource demanding tasks submitted by the users (or subscribers).

A job submission session from a user to a remote resource can be secured through PKI-based Grid security infrastructure [6]. Currently, proxy certificate [19, 20] is used to facilitate single sign-on and delegation, two essential security requirements within a Grid environment. Figure 2 shows a job submission from user A to resource X. Potentially, X may approach Y if it requires additional resource to complete A's job request.

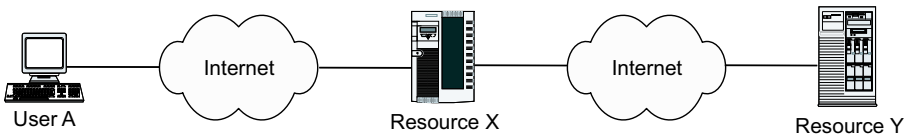

Figure 2: A job submission session through PKI.

Here, we list some potential benefits that DKI can offer.

- *Single Sign-on and Delegation*: Before A submit a job to X, she can, using her key mould, extract a short-term private key which corresponds to the following ephemeral public key.

$$\bar{PK}_A = H_1(\texttt{Name} = A \parallel \texttt{App_ID} = \text{grid101010} \parallel \texttt{Date} = 01012005 \parallel r_A)$$

  Note that r_A is a fresh random number that A picks for that particular job session so that each job request can be tied down to a unique short-term public key. By storing the short-term private key in a local file system, single sign-on and delegation can be achieved without any physical intervention from the user. This technique has been proposed in [13] and will not be further discussed here. We just want to point out that the advantage of this technique over current method is that it does not require: (i). creation of proxy certificate, (ii). certification of a new short-term public key, and (iii). distribution of the new short-term public key, before the key can be used. This can cut down the computational and communication costs considerably between the user and the resource provider.

- *Distribution of Proxy Credentials*: We also envisage that the secret public key protocol proposed in [10] can be integrated with MyProxy system [15] within a Grid environment. MyProxy server is a so-called high-end secure web-based Grid portal which stores the users' long-term credentials. It allows its users to access Grid resource from anywhere by generating proxy (short-term) credentials on users' behalf. Assuming A has her long-term

private key securely kept in the MyProxy server. Every time before she submits a job, she has to log-on to the MyProxy server using a shared password K_{AS} between her and the server, to request for generation of proxy credentials. Using a shared password for authentication, however, may be vulnerable to online or offline password guessing attacks such as those found in [4, 10, 17]. Furthermore, an exposed password can be difficult to detect. Rather than using a password-derived key for the authentication protocol between A and the server, we propose the use of a one-time secret public key in encrypting protocol messages. The secret public key is in the following form.

$$\hat{PK} = H_1(\texttt{Name} = A \parallel \texttt{App_ID} = \text{grid101010} \parallel \texttt{Secret} = K_{AS} \parallel \texttt{Date} = 01012005)$$

The MyProxy server can compute a fresh secret public key with one-day lifetime on-the-fly and it has high confidence that the associated secret private key can only be extracted by A using her key mould. Through this method, even if either the one-day secret public or the one-day secret private key is compromised, the potential damage can be limited to a day or less.

6 Conclusions

Currently, PKI is the most widely used security infrastructure in facilitating the use of public-key cryptosystems such as RSA encryption and signature schemes. As such, it is natural that some new emerging distributed computing technologies such as pervasive and GRID computing have adopted PKI in providing the underlying security mechanisms/services. Despite that, due to some new and unique security requirements in those systems, PKI may seem relatively rigid, heavyweight, and less scalable because of the extensive use of certificate, frequent key update, and other key management concerns in such dynamically evolving environments.

We propose a more lightweight and user-centric unified public key management method through a dynamic key infrastructure (DKI), which seems can reduce the computational and communication costs that the systems may cause in the PKI setting. We expect that the concepts presented in this paper may well be useful in other applications which are complex and tedious in managing public key.

References

[1] 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.

[2] J. Burkhardt, H. Henn, S. Hepper, K. Rindtorff, and T. Schäck. *Pervasive Computing: Technology and Architecture of Mobile Internet Applications.* Addison-Wesley, London, 2002.

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

[5] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter 2: omputational Grids, pages 15–51. Morgan Kaufmann, San Francisco, 1999.

[6] 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.

[7] 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.

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

[9] 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.

[10] L. Gong, T.M.A. Lomas, R.M. Needham, and J.H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[11] 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.

[12] ITU-T Rec. X.509 (Revised). *The Directory - Authentication Framework*. International Telecommunication Union, Geneva, Switzerland, 1993 (equivalent to ISO/IEC 9594-8:1994).

[13] H.W. Lim and M.J.B. Robshaw. A dynamic key infrastructure for GRID. In *Proceedings of the European Grid Conference (EGC 2005)*, to appear.

[14] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. Internet X.509 public key infrastructure online certificate status protocol (OCSP). *The Internet Engineering Task Force (IETF)*, RFC 2560, June 1999.

[15] 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.

[16] Open Mobile Alliance. *Wireless Application Protocol Public Key Infrastructure Definition*, February 2005. Available at http://member.openmobilealliance.org/ftp/public_documents/sec/Permanent_documents/, last accessed in March 2005.

[17] S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 236–247. IEEE Computer Society Press, 1997.

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

[19] 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.

[20] 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.