Countering Digital Forensics

An Identity Based Ephemerizer Cryptosystem

Srijith K. Nair[1], Chandana Gamage[1], Mohammad Torabi Dasti[2], Bruno Crispo[1], and Andrew S. Tanenbaum[1]

[1] Dept. Computer Science, Vrije Universiteit, Amsterdam, The Netherlands,
{srijith,chandag,crispo,ast}@few.vu.nl,
[2] CWI, Amsterdam, The Netherlands
dashti@cwi.nl

Abstract. Digital forensics rely on the ability to recover data stored in persistent storage of a machine, even if it has been tagged as being deleted. We consider one of the solutions proposed to counter such analysis and recovery - storing the data in an encrypted form in the user's machine and the key to decrypt the data in a physically separate machine. We discuss a scheme proposed to implement this and identify flaws and constraints associated with it. We then propose an alternate scheme based on identity-based cryptography that overcomes these constraints.

1 Introduction

One of the most effective tools used in investigation of crimes involving computers and privacy invasion is data recovery from storage devices with the aim of discovering incriminating digital information. The field of digital forensics in large part consists of recovery of files that have been hidden, encrypted or deleted from a rewritable persistent storage device like a hard disk. While other main activities of digital forensics, such as analysis of activity logs, are useful and important, the real smoking gun is to find the tracks of a music album ripped from a copyright protected disk, the source file and object code for the latest computer virus or files pertaining to a whistleblower case. While a judge and a jury may not understand arcane entries in a access log showing login attempts, directory changes, file copy commands, etc., everybody understands pictures that one should not have taken or music that you did not compose or buy. But this paper is not about helping digital forensics or helping the crime fighter. It is about protecting your privacy. Towards this end, in this paper we explore one of the techniques used to safely manage files while preventing their access or recovery by third parties.

To formulate the problem more precisely, assume that Alice wants to send a plaintext message M to Bob at time t_0 with following expectations: (1) only Bob will be able to read the plaintext and (2) after time t_1 ($t_0 < t_1$), no one including Bob can read the plaintext. We assume that an attacker can break into the computer system of any ordinary user and can seize all equipments, extract

any data from persistent storage that is presently stored or previously deleted or kept encrypted, including forcing the user to disclose passwords used to secure decryption keys. We also assume an open communication medium in which all transmissions can be intercepted, recorded, modified, and retransmitted. In other words, the attacker has powers similar to that of a totalitarian government agency.

In the next section we discuss the *Ephemerizer* scheme of Perlman [1] which has been presented as a solution to the above problem and point out some shortcomings associated with it. We then provide a brief introduction to Identity Based public key cryptography, a primitive used in our proposed solution, in Section 3. In Section 4 we describe our proposed scheme and discuss the security of the system in detail and suggest possible extensions to the system in Section 5. We conclude in Section 6.

2 The Ephemerizer System

Perlman [1] presented the concept of an *Ephemerizer* as a central server that allows parties to keep data for a finite time period and then makes it unrecoverable after that. Alice and Bob executes the protocol steps with the active participation of this trusted third party. This concept was later used in a system designed to provide assured delete [2].

The original paper presented two versions of the protocol - one using triple decryption and the other using blind decryption. We discuss the triple decryption method in detail here since an attack against this version is identified and presented in later section. Interested readers are referred to [1] for more details on the blind decryption scheme.

2.1 Triple Encryption Method

Throughout the rest of the paper, we use $\{M\}_{K_A}$ to denote asymmetric key encryption of M with public key of entity A and $[M]_K$ to denote symmetric key encryption of M with symmetric key K.

Step 1 - The Ephemerizer E creates sets of asymmetric key pairs, associate them with different expiry dates and advertises tuples - public key, key ID, expiry date.

Step 2 - Alice chooses one of the keys, K_{eph}, based on the expiration time she requires, and encrypts the data M with a random secret key S to obtain $[M]_S$. She then encrypts S with Bob's long-term public key (K_{bob}) and the result with K_{eph}.The resulting value is encrypted again with a random session key T to get $[\{\{S\}_{K_{bob}}\}_{K_{eph}}]_T$. T is then encrypted with Bob's public key and an integrity check value [3] HMAC(T,$\{\{S\}_{K_{bob}}\}_{K_{eph}}$) is calculated. Finally, Alice sends the following to Bob:

$$A \rightarrow B : \{T\}_{K_{bob}}, [\{\{S\}_{K_{bob}}\}_{K_{eph}}]_T, [M]_S, keyID, K_{eph}, HMAC(T, \{\{S\}_{K_{bob}}\}_{K_{eph}})$$

Step 3 - Bob, on receiving this message, decrypts the first part of the ciphertext using his long term private key to obtain T and uses T to decrypt and obtain $\{\{S\}_{K_{bob}}\}_{K_{eph}}$. He then verifies the HMAC value and if this check is successful, chooses a random secret key J to secure his communication with the Ephemerizer, encrypts J using K_{eph} and sends the following to the Ephemerizer:

$$B \rightarrow E : keyID, \{J\}_{K_{eph}}, [\{\{S\}_{K_{bob}}\}_{K_{eph}}]_J$$

Step 4 - The Ephemerizer identifies K_{eph} using $keyID$ and decrypts J using the private key corresponding to K_{eph}, if the key has not expired. Using K_{eph}'s private key and J, the Ephemerizer then decrypts the third part of the message to obtain $\{S\}_{K_{bob}}$ and uses J to re-encrypt the decrypted part and send it back to Bob.

$$E \rightarrow B : [\{S\}_{K_{bob}}]_J$$

If K_{eph} has expired, Ephemerizer sends back an error message to Bob indicating the unavailability of the key.

Step 5 - Since Bob knows the value of J and his own long-term private key, he can then decrypt the message from E and retrieve the value of S which is then used to decrypt M_E to obtain M.

The Ephemerizer periodically scans its database of asymmetric key pairs and securely delete all key pairs that have an expired time value. Therefore, after the expiry of time t_1, no one will be able to recover the plaintext M if all the participants (Alice, Bob, and the Ephemerizer) have truthfully executed the protocol.

The design of the above protocol is based on two elements: (1) creation of temporary tunnels when transmitting data from one location to another and destroying the tunnel after the completion of the data transfer so that an observer of the activity does not learn anything useful and (2) keeping the ciphertext and the key to recover the plaintext at separate locations so that both locations need to be compromised for an attacker to obtain the plaintext. The key is fetched every time the user needs to obtain the plaintext and securely deleted, along with the plaintext, once the temporary use is over. The assumption is that the retrieved decryption key and the plaintext are stored and the decryption process is performed within the volatile memory of the user's machine, which is easier to delete securely than data stored in persistent storage.

2.2 Attack Against Triple Encryption Scheme

The triple encryption Ephemerizer scheme as presented in [1] suffers from a serious flaw whereby the attacker can get hold of $\{S\}_{K_{bob}}$, in effect nullifying the ephemeral property of the system. The attack plays out as follows.

The attacker captures the message sent between Bob and Ephemerizer in *Step 3* and *Step 4*. It then generates a random key X and encrypts it with K_{eph}. The attacker then encrypts the second part of the original message in *Step 3* with X and sends the whole message as shown below, to the Ephemerizer.

$$Att. \rightarrow E : keyID, \{X\}_{K_{eph}}, [\{J\}_{K_{eph}}]_X$$

As long as this attack message is sent before the expiry of keyID, the Ephemerizer cannot differentiate it from a genuine message from a user. Hence, it decrypts X using the private key of K_{eph} and from that it decrypts the third part of the attack message to obtain J. This value of the random session key, used by Bob to encrypt his dialog with the Epehemerizer, is then sent back to the attacker.

$$E \rightarrow Att. : [J]_X$$

Since X is known to the attacker, he can decrypt this message from the Epehemerizer and obtain J, which in turn can be used to decrypt the message sent to Bob in *Step 4* to get $\{S\}_{K_{bob}}$. Thus, the purpose served by the ephemeral key K_{eph} is nullified, since $\{S\}_{K_{bob}}$ is now known to the attacker and is not protected by K_{eph} in a timebound manner. Knowing $\{S\}_{K_{bob}}$, the attacker can wait as long as needed to break into Bob and retrieve the long-term private key of Bob, and decrypts the value of S, even after K_{eph} has been deleted from the Ephemerizer's database.

2.3 Other Issues

One of the major hurdles in implementing the Ephemerizer system proposed in [1] is its reliance on the existence of a reliable public key infrastructure (PKI) to obtain the certified long term public key of the parties involved. Such an infrastructure is still missing in a real-world non-organizational setup.

Another limitation of the scheme is that Alice does not have the flexibility to define her own expiry dates for the data. Instead she has to choose an expiry date advertised by the Ephemerizer, which may not have the granularity that she desires. The scheme also does not allow Alice to specify additional restrictions on the access of the decryption key, like for example that Bob is allowed access the data only from within a specified network, or only while using a certified secure setup that is guaranteed to delete the intermediate key J after every invocation of decryption steps etc.

In summary, one of the method proposed in [1] is shown to be fataly flawed. The scheme also relies on the existence of a working PKI, which sadly has not be realised as of yet. In addition, the scheme proposed is not flexible with respect

to the policies that a user can associate the release of the data encrypted with the ephemeral key.

In the rest of the paper we present an alternative scheme to provide the envisioned ephemeral service, without the identified flaws and limitations, using Identity Based cryptography primitive as the underlying basis.

3 Identity Based Cryptography

Shamir [9] first proposed the use of an encryption scheme in which an arbitrary string can be used as a public key as early as in 1984. However, it was only in 2001 that a mathematically sound identity-based public key cryptosystem (IB-PKC) was proposed by Boneh and Franklin [10]. An IB-PKC system, based on bilinear pairing, is used as a foundation of our proposed scheme. In this section we provide a brief introduction to the mathematics and the steps involved in using the crypto-primitive.

Let P denote a generator of $\mathbb{G}_1$, an additive group of some large prime order q. Let $\mathbb{G}_2$ be a multiplicative group of the same order. A pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1. Bilinear: $e(aQ, bR) = e(Q, R)^{ab} = e(abQ, R) = e(bQ, R)^a$, where $Q, R \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$.
2. Non-degenerate: $e(P, P) \neq 1_{\mathbb{G}_2}$, where $1_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$.
3. Computable: There exists an efficient algorithm to compute $e(Q, R)$ for all $Q, R \in \mathbb{G}_1$

It is believed that bilinear Diffie-Hellman problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ (given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, compute $e(P, P)^{abc} \in \mathbb{G}_2$) is hard. Typically the map e is derived from either the Weil [11] or Tate [12] pairing on an elliptic curve.

An IB-PKC scheme consists of four main steps (1) **setup** in which a Key Generation Center (KGC) generates global system parameters and a system secret key, (2) **encrypt** where a message is encrypted using an arbitrary public key, (3) **extract** during which the system secret key is used by the KGC to generate the private key corresponding to the arbitrary public key chosen in the step earlier and (4) **decrypt** where the private key generated is used to decrypt the encrypted message.

Interested readers are referred to [10] for a more rigorous explanation of the mathematics, protocol steps and related proof of security behind the IB-PKC cryptosystem.

4 Proposed System

In this section we describe our proposed scheme that uses IB-PKC to overcome the deficiencies found in the *Ephemerizer* system.

As in the *Ephemerizer* scheme, our approach is to keep only the encrypted version of the data on the persistent storage of the user and to 'store' the key needed to decrypt the data on a different machine, the Ephemerizer server. When the user needs to access the plaintext data, he retrieves the decryption key from the server and uses it to decrypt the data.

Note that for the ease of explanation, the scheme described here uses the *BasicIdent* version of the IB-PKC scheme proposed in [10]. This version is not secure against an adaptive chosen ciphertext attack and hence an actual implementation of our scheme would need to use the secure *FullIdent* version of the scheme [10].

We divide our scheme into four steps:

Step 1 - The KGC generates two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ for the groups and choose an arbitrary generator $P \in \mathbb{G}_1$. It also specifies two hash functions H_1, H_2 as:

- $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$
- $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$, n being the bit-length of plaintexts

Message space $\mathcal{M} = \{0,1\}^n$, ciphertext $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$

KGC also computes a system-wide master-key s_0 uniformly at random from $\mathbb{Z}_q^*$ and public key $P_0 = s_0 P$ and publishes the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2 \rangle$. This is equivalent to the **setup** phase of a generic IB-PKC.

Every user who wants to use the Ephemerizer system registers with the KGC and negotites a shared secret for use in authentication in later steps. A similar registration is performed at the Ephemerizer server.

Step 2 - Alice chooses a random symmetric key K and encrypts the data M with it: $M_E = [M]_K$. She then chooses an arbitary string as Bob's public key ID_b, and computes $Q_{ID_b} = H_1(ID_b)$. Alice then chooses $r_b \in \mathbb{Z}_q^*$ and encrypts K by computing:

$C_b = \langle U_b, V_b \rangle = \langle r_b P, K \oplus H_2(g_b^{r_b}) \rangle$
where $g_b = e(Q_{ID_b}, P_0) \in \mathbb{G}_2$

V_b is then encrypted using a public key of the Ephemerizer, E. For this, Alice chooses a public key for E of the form $ID_e =$ '$Eph|Expiry : 2006-3-10-2359$' (where '2006-3-10-2359' denotes the expiry date of the key) and computes $Q_{ID_e} = H_1(ID_e)$. She then chooses $r_e \in \mathbb{Z}_q^*$ and performs the encryption by computing:

$C_e = \langle U_e, V_e \rangle = \langle r_e P, V_b \oplus H_2(g_e^{r_e}) \rangle$
where $g_e = e(Q_{ID_e}, P_0) \in \mathbb{G}_2$

Alice then sends the following to B:

$A \rightarrow B : ID_b, ID_e, U_b, C_e, M_E$

Step 3 - When Bob needs to decrypt and obtain M, he first contacts the KGC to get his private key corresponding to ID_b. However, for reasons explained later in Section 5.2, he first blinds Q_{ID_b} by computing $Q'_{ID_b} = r_{blind}Q_{ID_b}$, $r_{blind} \in \mathbb{Z}_q^*$ and sends this Q'_{ID_b} to the KGC. The KGC authenticates Bob using a scheme like [14] and computes his blinded private key $d'_b = s_0 Q'_{ID_b}$ and sends it back to Bob. Bob, knowing the blinding factor, unblinds d'_b to retrieve his private key d_b.

Step 4 - Even though Bob has obtained his private key d_b, he still does not know the value of V_b. To obtains V_b Bob contacts E, authenticates himself, again using a scheme like [14], and sends:

$B \rightarrow E : C_e, ID_e$

E checks the expiry date specified in ID_e and if valid, asks KGC for the private key corresponding to ID_e. After authenticating E, KGC also verifies the expiry date and if valid, computes and sends the private key $d_e = s_o Q_{ID_e}$ back to E. E then computes $V_e \oplus H_2(e(d_e, U_e))$ which returns V_b (due to the bilinear pairing properties discussed earlier) and send V_b back to Bob

$E \rightarrow B : V_b$

Bob then uses the value of d_b obtained in Step 3 to compute $V_b \oplus H_2(e(d_b, U_b))$ which returns the value of the symmetric key K. K is then used to decrypt M_E to obtain the cleartext M. All the calculations are performed in the volatile meory of Bob's machine and at the end of the protocol run d'_b, d_b, V_b, K and M are securely deleted.

5 Discussion

5.1 Comparison with Ephemerizer scheme

The use of ID-PKC eliminates the flaws identified with the original Ephemerizser scheme. As can be seen from the format of ID_e, Alice can specify her own expiry date, to whatever granularity she desires, without having to rely on that provided by E, as was the case in [1]. By extending the format, Alice can put additonal restrictions on the release of the secret by the Ephemerizer server. For example by creating $ID_e = `Eph|Expiry : 2006-3-10-2359|IP : 130.45.6.*'$, Alice can tell the Ephemerizer to release V_b only if Bob's request originates from an IP address of the 130.45.6 range. Or similarly, using the concept of remote attestation [15], Alice could put additional restriction on say, the version of document reader or a specific operating system that Bob needs to be using when requesting for the decrypted data. Using semantic remote attestation [16] techniques that allows for the attestation of dynamic, high-level program properties, Alice can specify even more complex restrictions on Bob's environment.

Since our scheme relies on IB-PKC, a PKI is not required for its working. However, to provide an authenticated priviate channel betweek Bob and KGC, Bob and E and E and KGC, a scheme similar to [14] needs to be in place using the shared secret setup in *Step 1*. While it may be thought that the shared secret setup phase is similar in complexity to a PKI setup, we argue otherwise. In a PKI setup, every user needs to know any other user's public key before a message can be shared with that user. In a shared secret scheme, the user needs to register only with the KGC and the Ephemerizer.

As the expiry date is embedded within ID_e, E can ensure that it will not respond to any query from Bob or any other external parties after the expiry of the time frame. Thus a compromise of Bob after time t_1 does not compromise the system. Even if the intruder is able to compromise Bob *and* E after the expiry date of the data, it will still not be able to decrypt M because the KGC will refuse to provide the compromised E with its private key d_e. An intruder who has captured all the data exchanged between Alice and Bob does not have enough knowledge to obtain M, as only Bob knows the value of his private key, he alone can decrypt C_b to obtain M_E and M.

It is to be noted that while we have kept the identity of KGC and the Ephemerizer seperate to mirror the entities of the Certificate Authority and the Ephemerizer in the original PKI based scheme, it is possible to combine the functionalities of the KGC and the Ephemerizer into one trusted third party.

All in all, we claim that our scheme provides additional advantage over the *Ephemerizer* scheme [1], without lowering the security of the system.

5.2 On Key Escrow and Blinding

A typical IB-PKC cryptosystem suffers from the key escrow problem since the KGC of such a system, knowing the system secret s_0, can compute the private keys of all its users. Though several solutions have been proposed to counter the inherent key escrow problem, including threshold key issuing using multiple KGCs [10], generating private key using multiple independent private keys issued by multiple KGCs [17], certificate-based encryption [18] and certificateless public key encryption [19], none of them can be directly used in our scheme. On one hand the schemes proposed in [18] and [19] breaks the IB-PKC's independence of public key chosen for encryption, and hence Alice's ability to specify her own expiry date, while on the other, proposals similar to [17] require Bob to authenticate with multiple KGCs every time he needs to decrypt the data.

However, to make our scheme equivalent in security to the original Ephemerizer scheme, the key escrow has to be removed and for this we perform the additional step of blinding the communication between Bob and KGC. The process of blinding prevents the KGC from knowing the actual private key as it can only calculate the blinded private key. While it may be argued that even with the blinding a malicious KGC can still eavesdrop on the communication between Alice and Bob, find out the value of ID_b and compute the private key d_b from it, this can be prevented by the use of a three-party password based authenticated exchange scheme like [20]. On a more fundamental level, our counter-arguement

is also that a threat model in which the KGC is malicious is equivalent to a PKI threat model with a malicious Certificate Authority (CA). In such a model, the malicious CA would be in the position to associate Bob's identity to an public key whose corresponding private key is not under Bob's control but under the CA's. Since such an intruder model is never associated with a PKI setup, a malicious KGC is excluded from our intruder model too.

A special case of our scheme is the one in which Alice and Bob are physically the same entity. In fact such a setup is the subject of [2]. It is worth pointing out that in such a scenario, the process of blinding alone prevents a malicious KGC from compromising the scheme since the KGC cannot know the value of ID_b without compromising the user.

It is worth mentioning that one has to be careful in selecting the random values b_j to be used for blinding, to prevent an attacker from collecting a large number of blinded public values b_jQ transmitted between an unknown party (say, Q_a) and the KGC and then attempting to determine the *suspected identity* of that unknown source by computing the value $W = \sum b_jQ$ and testing $W \stackrel{?}{=} Q_a$ for selected subsets of blinded values. The attacker will succeed if our selection of blinding values create subsets for which the value $\sum b_j$ is equal to the unit value.

5.3 Possible extensions

The proposed scheme can be extended in a couple of ways to increase its security. During the registration phase, every user could negotiate a secret 'destruct trigger' with the KGC, which the user can use in an authenticated manner if he suspects that his machine has been compromised. Once the trigger is received by the KGC, it will not honor any more requests for d_b, even if the (possibly compromised) user authenticates correctly with the KGC and asks for an unexpired key.

Anonymity can be added into the scheme by using a variation of the scheme proposed by Sur et. al. in [21]. This or any similar scheme would however increase the number of steps required to setup the system.

6 Conclusion and Future Work

In this paper we analyzed the *Ephemerizer* system proposed by Perlman [1] and used in [2] as a system that allows parties to share data while avoiding digital forensics, by keeping the encrypted data and the decryption key in physically separate entities, for a finite time period and then making it unrecoverable after that. We presented a fatal flaw in one of the methods proposed in [1] and also identified some constraints associated with the original scheme. We then proposed and discussed an alternate scheme to implement the Ephemerizer system with none of the identified flaws and functional constraints.

Some possible extensions to the scheme were also suggested in this paper and we plan to followup on some of them. A formal proof of correctness and

an implementation will also be worked on to prove that the scheme is indeed as secure as we claim it to be.

References

1. R. Perlman, *The Ephemerizer: Making Data Disappear*, Sun Microsystems Technical Report SMLI TR-2005-140, 2005.
2. R. Perlman, *File System Design with Assured Delete*, Third IEEE International Security in Storage Workshop (SISW'05), pp. 83-88, 2005.
3. M. Bellare, R. Canetti, and H. Krawczyk, *Keying hash functions for message authentication*, Advances in Cryptology - Crypto 96, LNCS 1109, Springer-Verlag, 1996.
4. K. Sakurai, Y.Yamane, *Blind decoding, Blind Undeniable Signatures, and their Applications to Privacy Protection*, Information Hiding : First International Workshop, Lecture Notes in Computer Science 1174, Springer-Verlag, pp. 257-264, England, May 1996.
5. R. Anderson, and R. Needham, *Robustness Principles for Public Key Protocols*, Lecture Notes in Computer Science 963, Advances in Cryptology - Crypto '95, Spring-Verlag, pp. 236-247, 1995.
6. M. Mambo, K. Sakurai, and E. Okamoto, *How to Utilize the Transformability of Digital Signatures for Solving the Oracle Problem*, Advances in Cryptology - Asiacrypt'96, Lecture Notes in Computer Science 1163, Springer-Verlag, pp. 322-333, 1996.
7. I.Damgard, M.Mambo and E.Okamoto, *Further study on the transformability of digital signatures and the blind decryption*, The 1997 Symposium on Cryptography and Information Security, 1997.
8. K. Ohta, *Remarks on Blind Decryption*, Proc. of Information Security Workshop, Lecture Notes in Computer Science 1396, Springer-Verlag, pp. 273 - 281, Japan, September 17-19, 1997.
9. A. Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology - Crypto 84, Lecture Notes in Computer Science 196, Springer-Verlag, pp. 47-53, 1984.
10. D. Boneh, and F. Franklin, *Identity-based Encryption from Weil Pairing*, Advances in Cryptology - Crypto 2001, Lecture Notes in Computer Science 2139, Springer-Verlag, pp. 213-229, 2001
11. S. Lang, *Elliptic Functions*, Addision-Wesley, pp. 243-245, 1973.
12. G. Frey, M. Muller, amd H. Ruck, *The Tate Pairing and the Descrete Logarithm Applied to Elliptic Curve Cryptosystems*, IEEE Transactions on Information Theory, 45(5)L1717-9,1999.
13. W. Diffie and M.E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory 22 (1976), pp. 644-654, 1976.
14. P. MacKenzie, S. Patel, and R. Swaminathan, *Password-authenticated key exchange based on RSA*, ASIACRYPT 2000, Lecture Notes in Computer Science 1976 , pp. 599-613. Springer-Verlag, Dec. 2000.
15. R. Sailer, X. Zhang, T. Jaeger, L. Van Doorn, *Design and Implementation of a TCG-Based Integrity Measurement Architecture*, Proc. of 13th Usenix Security Symposium, USENIX, San Diego, CA, August 2004.
16. V. Haldar, D. Chandra, and M. Franz, *Semantic Remote Attestation: A Virtual Machine Directed Approach to Trusted Computing*, USENIX Virtual Machine Research and Technology Symposium, May 2004.

17. L. Chen, K. Harrison, N.P. Smart, and D. Soldera, *Applications of multiple trust authorities in pairing based cryptosystems*, InfraSec 2002, LNCS 2437, Springer-Verlag, pp. 260-275, 2002.
18. C. Gentry, *Certificate-based encryption and the certificate revocation problem*, Advances in Cryptology - EUROCRYPT 2003, pp. 490-497, 2003
19. S. Al-Riyani, and K. Paterson, *Cerfificateless public key cryptography*, Advances in Cryptology - Asiacrypt 2003, pp. 452-473, 2003.
20. M. Abdalla, P. Fouque and D. Pointcheval, *Password-Based Authenticated Key Exchange in the Three-Party Setting*, IACR eprint 2004/233, Available at http://eprint.iacr.org/2004/233/, 2004
21. Ai-fen Sui, Sherman S.M. Chow, Lucas C.K. Hui, S.M. Yiu, K.P. Chow, W.W. Tsang, C.F. Chong, K.H. Pun, H.W. Chan., *Separable and Anonymous Identity-Based Key Issuing*, 11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS'05), pp. 275-279, 2005.