A DRM System Protecting Consumer Privacy

Min FENG
Microsoft Research Asia
Beijing, P. R. China
Email: minfeng@microsoft.com

Bin ZHU
Microsoft Research Asia
Beijing, P. R. China
Email: binzhu@microsoft.com

***Abstract*— Digital Rights Management (DRM) is widely used to protect intellectual property for content owners but consumer privacy is sacrificed. A user's playing statistics can be collected by the client DRM module and the license server. In this paper, we propose a DRM system in which the license server can generate the content decryption key for a user to play an encrypted content object without gaining any information to link to the specific content object encrypted by the content encryption key. This is achieved by applying a (partially) blind signature primitive in the license acquisition protocol and by adopting a key scheme that a content encryption key depends on the information retrieved from the content object and a secret that only the license server knows. By requesting that the client DRM module does not send any information about a user's playing statistics and all the messages the client DRM module sends out are in plain text for easy checking by a user if the client DRM module abides by this rule, consumer privacy is fully protected in our DRM system.**

I. Introduction

Digital content has been increasingly used in our daily life due to its high quality and efficiency in storage and distribution. Protection against piracy is needed. Digital Rights Management (DRM) is technologies that provide persistent rights management for digital contents [1]. In addition to DRM technologies offered by the standards organizations such as MPEG [2], [3], the Digital Media Project (DMP) [4], and the Open Mobile Alliance (OMA) [5], there also exist several proprietary DRM systems on the market. A typical commercial DRM system is the Windows Media Rights Manager (WMRM) from Microsoft [6]. In a typical DRM system, content is encrypted and packaged into a content object to distribute. A rights object, also called a license, is needed to play the protected content. A license contains the decryption key as well as a specification of rights a user has acquired. Licenses are usually distributed and stored separately from the corresponding contents to make it easier to manage the entire system. A license is acquired from a license server. It is usually locked to a user or a user's computer to prevent illegal sharing with other people or computers. A DRM system enforces the acquired rights through trusted DRM modules on the client side.

The technologies employed by a DRM system to enforce intellectual property protection implicate sacrifice of consumer privacy: the DRM client module knows what a user plays, and the license server knows what contents a user has acquired licenses for. The latter case, which is the focus of this paper, can be explained as follows. In a DRM system such as Microsoft's WMRM, a license and the decryption key are associated with a protected content through a key ID. A key ID is used instead of a content ID since it enables a content to be packaged into different content objects by encrypting it with different encryption keys. When a user acquires a license from a license server, the key ID is retrieved from the protected content and sent to the license server which generates or retrieves the corresponding content decryption key and sends to the user in a license. More precisely, the decryption key in the license is encrypted by the public key bound to the user's device so that only the targeted device can play the protected content. By searching a database or protected content objects, it is not difficult to find the content associated with the key ID. As a result, the submitted key ID enables a license server to link a user with the contents associated with the licenses acquired from the license server. This is an intrusion to consumer privacy. We need to strike a balance between protection of intellectual property for content owners and protection of privacy for consumers. A question arises naturally: is it possible to let a license server send a user the correct decryption key without knowing the key ID (or content ID)? This seems to be a hard problem for a DRM system in which, for the sake of security, no encryption key is shared by different content objects, i.e, each content object is encrypted with a different encryption key. Without knowing the key ID, a license server does not know which decryption key a consumer wants for the content. We are going to address this problem in this paper. We present a DRM system based on (partially) blind signatures that enables a license server to generate decryption keys and deliver to users without knowing the corresponding key IDs or the contents that users play. Therefore consumer privacy is protected during license acquisition. If we request that the client DRM module in our DRM system does not disclose any information about a user's playing statistics and all the messages that the client DRM module sends out must be in plain text for easy checking by a user if the client DRM module abides by this request, then consumer privacy is fully protected by our DRM system.

The remaining of this paper is organized as follows. In Section II blind and partially blind signatures are introduced. Our proposed scheme is described in Section III. Two examples of our proposed scheme are presented in Sections IV and V, respectively. One uses RSA and the other uses bilinear pairings. Discussions of the proposed scheme and its variations for more application scenarios are presented in Section VI. The

1-4244-1457-1/08/$25.00 © IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

paper concludes in Section VII.

II. Blind and Partially blind Signatures

Blind signatures were first proposed by Chaum [7]. A blind signature plays an important role in cryptographic protocols such as e-cash and e-voting which require user anonymity. In an e-cash system, neither the merchant nor the bank can identify the owner of e-cash. In an e-voting system, nobody except the voter himself/herself knows the owner of a vote. In a blind signature scheme, a requester usually uses a random number to blind the message to be signed, and then submits to the signer to sign. Since the blinded message that the signer signs cannot be differentiated from an arbitrary message, the signer does not know the actual message he/she has signed. The requester, knowing the random number on the other hand, can unblind the received signature to obtain a valid signature for the original message.

Unlike blind signatures in which the signer knows nothing about the message he/she signs, partially blind signatures, introduced by Abe and Fujisaki [8], allow the signer to explicitly include some agreed information in the blind signature. This partial knowledge of the message sent to the signer to sign enables a requester to send a specific nominal e-cash value as the explicit information together with the blinded information to a bank to sign. When the agreed information is empty, a partially blind signature can be regarded as a normal blind signature. Security of blind signatures was formalized in [9], [10], while security of partially blind signatures was formalized in [11]. Many blind signature schemes can be found in [9], [12], [14]–[19], [22] and partially blind signature schemes in [8], [11], [19]–[21].

In this paper, we classify (partially) blind signatures into two types: stable (partially) blind signatures and unstable (partially) blind signatures. In a stable (partially) blind signature, there is only one valid signature generated for each original message. No random value is included in the final signature. In an unstable (partially) blind signature, the final signature still includes a random number. As a result, many signatures can be generated for an original message when an unstable (partially) blind signature scheme is applied.

In our solution, a stable (partially) blind signature protocol is used to generate a signature for a content's key ID, which is used as the content encryption key. During license acquisition, the key ID is first blinded by the client DRM module. After the license server signs the blinded key ID, the client DRM module unblinds the received blind signature to obtain the signature of the key ID, i.e. the content encryption key in our solution. A symmetric encryption scheme is used to encrypt the content in our content packaging stage. Since the content has already been encrypted when a user plays a protected content, the (partially) blind signature used in our scheme must be stable. Otherwise a decryption key different from the encryption key used in the content packaging stage is generated, and fails to decrypt the protected content.

III. Our Solution

For the applications that our DRM system is applied to, we assume for the time being that all the contents are classified into different groups according to their prices and the contents in the same group are charged with the same price. Other applications scenarios will be discussed in Section VI. Suppose that the content object that a consumer C wants to acquire a license for is in a group g. C pays for certain rights for a content in the group g with a payment server and obtains a payment token from the payment server. C then submits the payment token together with a blinded version of the key ID extracted from the content object to the license server S. S returns a license containing the acquired rights and the content decryption key encrypted with C's public key. When the content object is played, C's DRM module checks against the acquired rights to see if the user has the rights to play, and then decrypts the encrypted content encryption key with C's private key to obtain the decryption key to decrypt the content. In this process, the payment server and the license server have only the knowledge that the consumer has acquired a license with certain rights for a content object in a group g, but cannot find out which content object in the group that the consumer has obtained a license for, although the license server has generated the decryption key for that content object, and sent to C in the license. If group g contains enough number of contents, consumer privacy is well protected.

Before describing the detail of our solution, we define the following symbols:

C	A consumer.
S	The license server.
K_{ID}	Key ID, a unique identifier to a content encryption key.
m	A message to be signed.
g	Group ID the content belongs to.
PT_g	Payment token as a proof of payment for a content object in group g.
r	A random number.
$\mathcal{B}_g(r, m)$	Blinded message of m by r with group ID g as the agreement.
$\mathcal{S}_g(.)$	A stable (partially) blind signature algorithm.
$\mathcal{B}_g^{-1}(r, \mathcal{S}_g(.))$	Unblinded signature signed by $\mathcal{S}_g(.)$ with r.
$Sgn(m)$	Signature of message m.
$Sgn_g(m)$	Resulting signature of message m with a parameter g.
$\mathcal{E}_C$	Public key encryption with C's public key.
$\mathcal{D}_C$	Public key decryption with C's private key.
k	Content encryption key corresponding to K_{ID}.

In this paper, we don't differentiate a consumer and the device a consumer uses to play content. As a result, C also means the device a consumer uses to play. The public and private keys used in $\mathcal{E}_C$ and $\mathcal{D}_C$ can be locked to either a device that plays contents or a person, depending on an application's needs.

In our DRM system, symmetric encryption is used to encrypt a content object during content packaging. Therefore the encryption and decryption keys are the same. A content object's encryption key is identified by its key ID K_{ID}. The encryption key k is related to K_{ID} with following equation: $k = Sgn_g(K_{ID})$, where g is the ID of the group that the content belongs to. As described later, the signature $Sgn_g(K_{ID})$ generated from a stable (partially) blind signature is a deterministic mapping from K_{ID} and g. The result does not depend on the random number used to blind K_{ID} during the license acquisition. This implies that the exactly same key can be regenerated once K_{ID} and g are given. This property is used by the license server to generate the decryption key without the knowledge of K_{ID} in our solution.

Both K_{ID} and g that are inserted into an unencrypted header of a content object. They are extracted from an content object and sent to the license server to obtain the decryption key in license acquisition. K_{ID} is also used to find the acquired license(s) associated with a content object. All the acquired licenses are stored locally in a license store. The license acquisition protocol is as follows.

Protocol 1: **License Acquisition Protocol**

1) $C \rightarrow S$: $g, m_b = \mathcal{B}_g(r, K_{ID}), PT_g$.
2) $S \rightarrow C$: verifies PT_g, if passes, $Rights, \mathcal{E}_C(s_b)$, $Sgn(Rights||\mathcal{E}_C(s_b))$, where $s_b = \mathcal{S}_g(m_b)$.
3) C: verifies $Rights$, $\mathcal{E}_C(s_b)$ against the signature $Sgn(Rights||\mathcal{E}_C(s_b))$, and stores them together with r in the local license storage, identifiable by K_{ID}, if the verification passes.

In this protocol, a consumer C sends the content object's group ID g, the blinded K_{ID} with a random number r, $m_b = \mathcal{B}_g(r, K_{ID})$, and a payment token PT_g, a proof of payment for a content object in group g, to the license server S in Step 1. In Step 2, S verifies the payment. If the verification fails, it refuses to issue a license to C. Otherwise S signs m_b with a stable (partially) blind signature algorithm: $s_b = \mathcal{S}_g(m_b)$, and then encrypts it with C's public key: $\mathcal{E}_C(s_b)$. The result is sent back to C together with the acquired rights $Rights$, and the signature $Sgn(Rights||\mathcal{E}_C(s_b))$. When C receives the license from the license server S, it verifies the integrity and authenticity of the received license by checking $Rights$ and $\mathcal{E}_C(s_b)$ against the signature $Sgn(Rights||\mathcal{E}_C(s_b))$. If the verification passes, C stores the license together with the random number r in the local license store. The license and the corresponding r can be retrieved from the license store with K_{ID}.

In this scheme, s_b is encrypted by C's public key so that only C can decrypt and retrieve s_b even if the license and the random number r are shared with other people or devices by the consumer. Once s_B and r are known, the content encryption key k can be calculated, as described in the next paragraph. C's private key is protected by the client DRM module so that it cannot be shared with others.

When a protected content object is played, the player extracts K_{ID} from the object and searches the local license store for associated licenses. If there is no valid license matching the requested action, the client DRM module acquires a license by using the license acquisition protocol 1 and stores in the license store. Once the relevant license is found, authenticity of the license is verified by checking against the license server's signature $Sgn(Rights||\mathcal{E}_C(s_b))$. If the authenticity is verified and the acquired rights agree with the action, the client DRM module uses its private key to decrypt $\mathcal{E}_C(s_b)$ to extract s_b, and then uses the random number r to unblind s_b to get $Sgn_g(K_{ID}) = \mathcal{B}_g^{-1}(r, s_b)$. Since $\mathcal{S}_g(.)$ is a stable (partially) blind signature algorithm, the resulting signature $\mathcal{S}_g(m)$ is the encryption key $k = Sgn_g(K_{ID})$ used to encrypt the content object during content packaging. This key is then used to decrypt the protected content object since the content decryption key is the same as the content encryption key.

As we can see from the license acquisition protocol 1, the license server receives only the information of the group g that the content belongs to. The payment token PT_g does not carry any more information. K_{ID} is blinded with a random number r, and the result $m_b = \mathcal{B}_g(r, K_{ID})$ received by the license server appears as an arbitrary message. After it generates the requested decryption key and sends the license to the consumer, the license server knows only the content's group but cannot identify which content in that group. As a result, consumer privacy is protected if the group has enough number of different contents.

In our DRM system, the content encryption key $k = Sgn_g(K_{ID})$ depends on the key ID K_{ID}, the group ID g, and the secret in the (partially) blind signature algorithm that only the license server knows. Both K_{ID} and g are inserted into a content object in the content packaging stage, and extracted by the client module and sent to the license server during license acquisition. In this way, the license server can regenerate the content encryption key for a user without needing a database to store encryption keys for all the released protected content objects. Therefore the license server in our DRM solution is a light server which requires only limited computing resources.

In our DRM system, a content encryption key used to encrypt content in the content packaging stage has to be acquired from the license server since the key depends on a secret known only to the license server. To package a content, the content packaging server generates an unique key ID K_{ID} for the content and finds the group ID g the content belongs to. K_{ID} and g are sent to the license server to generate a content encryption key which is then sent back to the packaging server. The key is used to encrypt the content. Unlike the license acquisition process in which K_{ID} is blinded before sending to the license server, K_{ID} in this stage is sent to the license server in its original form. Even if the packaging server and the license server collude, i.e., it is known which content corresponds to a key ID, the license server still cannot gain any information about the content corresponding to the license it issues to a consumer since the key ID submitted to the license server is blinded, i.e., no difference from a random sequence of bits.

With the scheme described above, the client DRM module can still collect a consumer's playing statistics. To fill that gap, we require that the client DRM module in our DRM system does not send out any information about a consumer's playing statistics, and all messages the client DRM module sends out must be in plain text[1] for easy checking by a user if the client DRM module abides by this rule, then consumer privacy is fully protected.

Two practical blind signature schemes, one based on RSA and the other based on pairing, can be used in our general solution described in this section. Details of the resulting two solutions are described in the next two sections, respectively.

IV. RSA-Based Solution

We can use a partial blind signature based on Chaum's RSA blind signature [7] as the blind signature primitive in our solution described in Section III. Security of Chaum's algorithm is based on the One-More-RSA-Inversion problem, as analyzed in [22]. In this solution, S generates an integer $n = pq$, where p and q are two large prime numbers, and releases n as a public key. Let $\varphi(n) = (p-1)(q-1)$. Group ID g belongs to a set of prime numbers co-prime with $\varphi(n)$. The license acquisition protocol 1 becomes:

Protocol 2: **RSA-Based License Acquisition Protocol**

1) C: generates a random number r, where $1 < r < n-1$, and computes $R = r^g \mod n$.
2) $C \to S$: g, $m_b = K_{ID} \cdot R \mod n$, PT_g.
3) S: checks validity of g and PT_g, and computes $g^{-1} \mod \varphi(n)$ and $s_b = {m_b}^{g^{-1}} \mod n$ if checking passes.
4) $S \to C$: $Rights$, $\mathcal{E}_C(s_b)$, and $Sgn(Rights||\mathcal{E}_C(s_b))$
5) Same as Step 3 in Procotol 1.

When it calculates the content decryption key, the client DRM module uses its private key to decrypt $\mathcal{E}_C(s_b)$ to obtain $s_b = {m_b}^{g^{-1}} \mod n$. It then uses the random number r to calculate the content decryption key $k = {m_b}^{g^{-1}}/r = K_{ID}^{g^{-1}} \mod n$. The result is what we expected: the resulting content decryption key k depends on only K_{ID} and g. It does not depend on the random number r used to blind K_{ID} during license acquisition.

In this solution, the public key encryption $\mathcal{E}_\mathcal{C}/\mathcal{D}_\mathcal{C}$ can be and any public key encryption primitive and the signature algorithm $Sgn(.)$ can be and any signature primitive.

V. Pairing-Based Solution

Bilinear pairings are used widely in constructing cryptographic primitives. A brief summary is given here. Let G_1 and G_2 be two cyclic groups with the same order q. Let $e : G_1 \times G_1 \to G_2$ be a bilinear pairing with the following properties.

1) **Bilinearity:** $e(aP, bQ) = e(P,Q)^{ab}$ for all $P, Q \in G_1, a, b \in Z_q$.
2) **Non-degeneracy:** There exist $P, Q \in G_1$ such that $e(P,Q) \neq 1$.
3) **Computability:** There is an efficient algorithm to compute $e(P,Q)$.

Security of bilinear pairings-based cryptographic primitives is built on some well-known hard problems of pairing. These hard problems include the Discrete Logarithm Problem (DLP) and the Computational Diffie-Hellman Problem (CDHP) and its variations. An example of CDHP variations is the Inverse Computational Diffie-Hellman Problem (inv-CDHP) which is to compute $a^{-1}P$ for given P and aP, where $a \in Z_q^*$.

The pairing-based partially blind signature algorithm proposed in [20] can be used as the blind signature primitive in our solution. Security of this partially blind signature algorithm is based on inv-CDHP: breaking this algorithm is equivalent to breaking inv-CDHP [20]. In this solution, S picks up a random number $x \in Z_q^*$, and computes $P_{pub} = xP$. The public key is P_{pub}. The secret key is x. $H_0(\cdot)$ is a hash function mapping values into G_1, and $H(\cdot)$ is a hash function mapping values into Z_q. The license acquisition protocol becomes:

Protocol 3: **Pairing-Based License Acquisition Protocol**

1) C: generates a random number $r \in Z_q^*$, and computes $m_b = H_0(K_{ID}||g) + r(H(g)P + P_{pub})$.
2) $C \to S$: g, m_b, and PT_g.
3) S: checks whether PT_g is valid, and computes $s_b = (H(g) + x)^{-1} m_b$ if checking passes.
4) $S \to C$: $Rights$, $\mathcal{E}_C(s_b)$, and $Sgn(Rights||\mathcal{E}_C(s_b))$.
5) Same as Step 3 in Protocol 1.

In calculating the content decryption key before playing, the client DRM module uses its private key to decrypt $\mathcal{E}_C(s_b)$ to get $s_b = (H(g)+x)^{-1} m_b$. The content encryption key is then

$$\begin{aligned}
k &= s_b - rP \\
&= \frac{m_b}{H(g)+x} - rP \\
&= \frac{H_0(K_{ID}||g) + r(H(g)P + P_{pub})}{H(g)+x} - rP \\
&= \frac{H_0(K_{ID}||g)}{H(g)+x} + \frac{r(H(g)P + xP)}{H(g)+x} - rP \\
&= \frac{H_0(K_{ID}||g)}{H(g)+x} + rP - rP \\
&= \frac{H_0(K_{ID}||g)}{H(g)+x}
\end{aligned}$$

As we expected, the resulting key k depends on only K_{ID} and g. It does not depend on the random number r used in blinding K_{ID} during license acquisition. This is exactly what we want: the license server regenerates the encryption key that a consumer requests and sends to the consumer without any knowledge of the corresponding content object.

Like the RSA-based solution, we can use any public key encryption primitive and signature primitive as $\mathcal{E}_\mathcal{C}/\mathcal{D}_\mathcal{C}$ and $Sgn(.)$, respectively, in this solution.

[1]Encryption at a later stage such as SSL applied at the transport layer is allowed since it does not affect a user's capability to check the messages sent out by the machine's client DRM module in his/her machine.

VI. Discussion

A content encryption key in our solution depends on the following three parameters: key ID K_{ID}, group ID g, and the license server's secret key. Both K_{ID} and g are publicly known. The only secret value is the license server's secret key that only the license server knows. If the secret key is compromised, all the encryption keys are compromised. Therefore the license server's secret key must be tightly guarded to prevent any compromise. To alleviate this problem, we can proactively update the license server's secret key periodically so that a secret key is used only for certain period of time. When a content is encrypted in a content packaging stage, the license server's secret key currently valid is used to generate the content encryption key. A version number indicating which secret key is used in generating the content encryption key is packaged into a content object. It is retrieved and sent to the license server together with K_{ID} and g by the client DRM module during license acquisition to regenerate the content encryption key. In this scheme, the license server needs to store all the secret keys it has used.

In the solutions described in Sections III-V, a group ID g is used to indicate which group a content object belongs to. Its purpose is to allow the license server to know that the consumer has paid for a content object in a group so that a license for a content object in that group can be issued to the consumer without disclosing the actual content object. We want to prevent the license server from knowing the content object associated with the license it issues by linking to the price the consumer pays. The proposed solutions are still applicable if there is another method to allow the license server to make a decision if a license should be issued to a consumer without providing any information to link to the content object associated with the license. In this case, g may be set to be empty in our solutions, i.e., a blind signature primitive is used in our solutions rather than a partially blind signature primitive. For example, if an application uses monthly subscription and all the contents available for each month are encrypted with the content encryption keys generated from the license server's secret key valid for that month to avoid expired members from accessing current contents, our solutions are applicable if g is set to be empty. In acquiring a license from the license server, a consumer needs to show the evidence of valid subscription for the current month. As a special case, if no payment is needed in acquiring a license, our solutions are applicable by setting g to be empty.

VII. Conclusion

In this paper, we have proposed a DRM system which protects consumer privacy during license acquisition. The license server generates the content decryption key that a consumer requests for a content object and sends to the consumer in a license without knowing the content encrypted by the encryption key. This is achieved by using a stable (partially) blind signature primitive in our license acquisition protocol and by adopting a key scheme that a content object's encryption key depends on its key ID and group ID as well as the secret of a blind signature primitive that only the license server knows. If we add a request in our DRM system that a client DRM module does not send any information about a consumer's playing statistics and all messages a client DRM module sends out must be in plain text, then consumer privacy is fully protected.

The license server in our DRM system is a light server which does not require any database to store the encryption keys for protected content objects. An encryption key can be regenerated from the information sent by a consumer as well as the secret that the license server knows.

References

[1] W. Zeng, H. Yu, and C.-Y. Lin, eds, Multimedia Security Technologies for Digital Rights Management, Elsevier, 2006.

[2] ISO/IEC JTC1/SC29/WG11 14496-13:2004(E), *Information Technology C Coding of Audio-Visual Object C Part 13: Intellectual Property Management and Protection (IPMP) Extensions*, 2004.

[3] MPEG, *MPEG-21 Part 5 – Rights Expression Language*, *MPEG-21 Part 6 – Rights Data Dictionary*, http://www.chiariglione.org/mpeg/.

[4] *Digital Media Project*, http://www.dmpf.org/.

[5] Open Mobile Alliance, *OMA DRM Specification Draft Version 2.0*, March 2004. http://www.openmobilealliance.org.

[6] Microsoft, *Microsoft Windows Media Digital Rights Management*, http://www.microsoft.com/windows/windowsmedia/drm/default.aspx.

[7] D. Chaum, *Blind Signatures for Untraceable Payments*, Advances in Cryptology - Crypto 82, Plenum, NY, pp.199-203, 1983.

[8] M. Abe and E. Fujisaki, *How to date blind signatures*, Advances in Cryptology - Asiacrypt 1996, LNCS 1163, pp. 244-251, Springer-Verlag, 2002.

[9] A. Juels, M. Luby and R. Ostrovsky, *Security of Blind Signature*, Crypto'97, LNCS 1294, pp. 150-164, Springer-Verlag, 1997.

[10] D. Pointcheval and J. Stern, *Provalable Secure Blind Signature Schemes*, Asiacrypt'96, LNCS, Springer-Verlag, 1996.

[11] M. Abe and T. Okamoto, *Provable Secure Partially Blind Signatures*, Crypto'00, LNCS 1880, pp. 271-286, Springer-Verlag, 2000.

[12] M. Abe, *A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures*, Rutocrypt'01, LNCS 2045, pp. 136-151, Springer-Verlag, 2001.

[13] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko, *The Power of RSA Inversion Oracles and the Security of Chaum's RSA-based blind signature scheme*, Financial Cryptography'01, LNCS, Springer-Verlag, 2001.

[14] J. Camenisch, M. Koprowski and B. Warinschi, *Efficient Blind Signatures without Random Oracles*, SCN'04, LNCS, Springer-Verlag, 2004.

[15] A. Kiayias and H. Zhou, *Two-Round Concurrent Blind Signatures without Random Oracles*, IACR Cryptology ePrint Archive, 2005/435, 2005.

[16] D. Pointcheval, *Strengthened Security for Blind Singature* Eurocrypt'98, LNCS, pp. 391-405, Springer-Verlag, 1998.

[17] D. Pointcheval and J. Stern, *New Blind Signatures Equivalent to Factorization*, ACM CCS, pp. 92-99, ACM Press 1997.

[18] D. Pointcheval and J. Stern, *Security Auguments for Digital Signatures and Blind Signatures*, Journal of Cryptology, 13, 3, pp. 361-396, Springer-Verlag, 2000.

[19] T. Okamoto, *Efficient Blind and Partially Blind Signatures without Random Oracle*, IACR Cryptology ePrint Archive, 2006/102, 2006.

[20] F. Zhang, R. Safavi-Naini and W. Susilo, *Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairing*, Indocrypt'03, LNCS, Springer-Verlag, 2003.

[21] C.I. Fan and C.L. Lei, *Low-Computation Partially Blind Signatures for Electronic Cash*, IEICE transactions on Fundamentals of Electronics, Communications and Computer Sciences, E81-A(5), pp. 818-824, 1998.

[22] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko, *The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme*, Journal of Cryptology, 16, 3, pp. 185-215, Springer-Verlag, 2003.