International Symposium on Information Theory and Its Applications
Xi'an, PRC, October 7–11, 2002

A Model For Signature Revocation

Rui ZHANG,[†] Michiharu KUDO,[‡] Kanta MATSUURA[†] and Hideki IMAI[†]

[†]Institute of Industrial Science, University of Tokyo,
4-6-1, Komaba, Meguro-ku, Tokyo, 153-8505 Japan
zhang@imailab.iis.u-tokyo.ac.jp
{kanta,imai}@iis.u-tokyo.ac.jp

[‡]Tokyo Research Laboratory, IBM Japan Ltd.
1623-14 Shimotsuruma, Yamato-shi,
Kanagawa, 242-8502 Japan
kudo@jp.ibm.com

Abstract

This paper introduces the important concept of signature revocation. An important property of digital signature is non-repudiatability that a signer can't deny his legal signature. However, in some occasion a signer wants to take back his previously singed signature in a proper legal way, so that after certain interaction a signature can be regarded as invalidate. We originate a new study of instance management regarding digital signature and develop the definition of security requirements. We show under some careful treatments the non-repudiatability can be removed from the original signature. Our scheme is simple and effective. We also give a proof that our scheme is as secure as the underlying signature scheme.

1. Introduction

Digital signature is a very important tool in modern cryptography. Two most important properties of signatures are unforgeability, by which means another person cannot create a valid signature of the signer, and non-repudiatability, by which means the sender cannot deny his valid signature. Any receiver of a digital signature can verify the origin of the source, the identity of the sender, the time of the issue and distribution.

However, there are some occasions when a distributed signature should be "destroyed". For example, when a customer does shopping in a net store, within limited time, he is able to cancel the bargain: by returning the merchandise, he is supposed to get his money back. This will be extremely easy in the paper world when shopping in a department store with a credit card despite of the somehow electronic transaction. By giving back the item to the shop assistant who will tear both receipts of the credit card payment, the customer can be convinced with his own eyes that his hand-writing signature has been destroyed and no longer exists. The shop assistant may proceed the purchase process by mistake, but they cannot prove the validity of his purchase because there is no customer signature in the world.

Difficulties of achieving the same for the digital signature are obvious: digital data can be easily duplicated, thus in the shopping scenario stated just now, the shop can keep a copy of receipt of the transaction, and later request the customer to execute the contract to make more sales. Though this is somehow fair, for the shop will transfer the goods to the customer and take the money, the customer will have to pay for the goods he does not want!

While a lot of research has been done to maintain its security under various kinds of attack, an entire management regarding each signature instance has not been thoroughly studied yet. We point out that how and when to revoke a valid signature of a signer as important as its generation. We shall investigate the management of digital signature.

1.1. Overview

Digital signatures are generated by a secret *signing key* that is only known by the signer and verified by a public *verification key*. To prevent impersonation, the public verification key is often attached with a certificate signed by a *Certificate Authority*, assigning the time of issue, the expiration of the certificate, the owner of the public key and etc.. There are several methods regarding revocation of certificate, all based on some chain structure, if we can think the tree structure as an extension of complicate chain. *Certificate Revocation List* (CRL), which has been adopted as RFC 2495, however, has the shortage as not scalable, slow validation. *Certificate Revocation Directory*: public bulletin board system, one or more non-trusted parties get updated certificate revocation information from the CA that serves as a certificate. *Certificate Revocation System* [4] the underlying idea is to sign a message for every certificate stating whether it was revoked or not, and an off-line/on-line signature scheme is used to reduce the communication cost.

However, we point out there are essential differences from revocation of each instance signature, which leads to the main point of this work. Certificate revocation is to paralyze the long term signing key and verification key pair but the revocation of signature instance is invalidate a certain

The first author was sponsored by Panasonic Scholarship.

signature while does not affect other valid signatures. Furthermore, it is desirable to have a scheme that revokes only part of the issued signatures in the effective period of the certificate.

1.2. Our contribution

There has been no known research in the literature on the revocation of signature. We formalize this problem, and define necessary security notions. Furthermore, we give an implementation based on the normal signature schemes. Our scheme is provably as secure as the underlying signature schemes.

The rest part of this paper will be organized as follows: we shall clean the basic definitions for our discussion in section 2, and go on to discuss the revocation problem and define the security requirements in section 3. In section 4, we give an implementation meeting these security requirements. In section 5, we evaluate our scheme and extend it to chain (tree) structured revocable signature.

2. Preliminary

2.1. Digital signature scheme

A digital signature scheme is a 5-tuple algorithm (pk, sk, Gen, $Sign$, $VerSign$), in which:

1. pk is the public verification key and sk is the secret signing key.
2. Generation algorithm Gen: a polynomial probabilistic algorithm, with the input $\{0,1\}^s$ and internal random coins r, where k is the system security parameter, generates the public verification key and the secret signing key pair $(pk, sk) \in \{0,1\}^k$.
3. Signing algorithm: $Sign$, maybe a polynomial probabilistic algorithm, for an input message M, the Signer using the signing algorithm outputs $Sign(M)$, where $Sig(M)$ will be called the signature on M.
4. Verification algorithm: $VerSign$, a deterministic algorithm, given the input a message M, the corresponding verification key pk and a signature Sig(M), $VerSign(pk, M, Sig(M))$ outputs a boolean value $(\mathsf{TRUE}, \mathsf{FALSE})$ that either rejects or accepts that $Sig_A(M)$ is a valid signature on M.

2.2. Hash function

Instead of signing on the original message $Sign(M)$, usually a signature is signed on the hash of the message,

$$Sign(h(M))$$

where the hash function is believed to be secure:

- (Inversion) Given a hashed value $H(M)$ it is hard to find the image M.
- (Collision) It is hard to find two messages satisfying:

$$h(M_1) = h(M_2) \quad M_1 \neq M_2$$

It is easy to see that a hash chain will be as secure as a single hashed value on the assumption of onewayness of hash function. For the simplicity, hereafter we will still write a signature as $Sig(M)$.

3. Security Notions

Security of signature scheme are usually defined in the attack model. If the scheme is secure in a specific model, then it will be called *secure*, yet only in the limit sense. The strongest security notion recognized publicly for digital signature is *IND-CCA2*, *semantic security* under *adaptive chosen ciphertext attack*, which was defined in [5]. This was proved in [2] equivalent with another notion *non-malleability* [3] under the adaptive chosen ciphertext attack: *NM-CCA2*. So if a signature scheme is secure under IND-CCA2 model, it is secure in other known attack model. Without loss of generality, we assume the underlying signature scheme is secure.

Revocation of a digital signature leads to two effects. First, within the authorized period of the certificate, it should be able for any third party other than the signer and the designated receiver to have the knowledge of this revocation. Second, even after the expiration of the certificate, it should be never the case that a malicious party can prove to a third party that a revoked signature is valid, but an honest signer or receiver cannot prove the truth. We show that the revocation of digital signature is in fact to remove the non-repudiatability for a valid signature under mutual agreements. Typical flow for signature revocation may be conducted as:

1. A signer issued a signature to a receiver.
2. They negotiate to revoke the signature.
3. They exchange some information.
4. The signature is revoked.

Remark 1 *The negotiation phase cannot be omitted for in most of the applications, the signature represents signer's commitment to the receiver, and this should not be able to change freely. Reasonable approach should require that both signer and receiver contribute to the revocation.*

Furthermore, we define the security of signature revocation into two levels:

Universal revocability: For a revoked signature, no one can verify the correctness of this signature.

Provable revocability: After the revocation, though someone can verify the correctness of the original signature, the signer or the receiver can still prove the truth that this signature has been legally properly revoked.

Universal revocablility is hard to achieve, because if the verification key and verification algorithm are not changed with the original data, one can always verify the "correctness" of this data. If both the key and verification algorithm are discarded after the revocation for a single signature, then all other legal signatures will become ineffective. The indeed information to verify for the confirmation is whether a "correct" signature has been revoked or not.

Linkability between the revocation statement and the original signature is very important. Of course, one trivial way is to have the signer and receiver to sign another statement on revocation of the signature. To revoke one signature instance, one have to cooperate with the receiver to sign another statement, which demands at least two signatures on the original signature with proper time linked. To verify this statement, one have to verify three signatures! For a large amount of instances management, we'd like some more intelligent way to solve this problem.

4. An Implementation of Revocable Signature

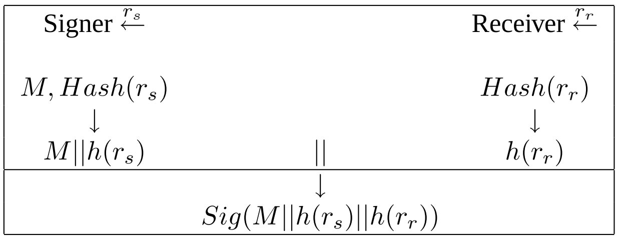
It is obvious that a signature can be revoked only once. We build the implementation on a simple idea: let the signer and receiver embed their secret information into the original signature. This will slightly lengthen the message to be signed, but it will not harm the length of the signature or the signing and verification process. This also links the signature and its disavowal. We point out the merits of this scheme: this contributes a real management on signature instance. For the seed or its hash can be easily stored in a chain or a tree structure, the convenience of which is obvious if the signatures signed are of a large number.

4.1. Signature generation

Recall in the signing phase, we sign on the hash of the message rather than the message itself. The signature is secure if the trapdoor oneway function and the hash function are secure. Let the sender and receiver choose a random number, r_s and r_r respectively, hash it and concatenate them with the original message. Each player must keep his own secret until the time he wants to revoke the signature.

The signature is $Sig(M||h(r_s)||h(r_r))$. The verification will take the input pk and the contents of the signature $(M||h(r_s)||h(r_r))$, verifying if:

$$VeriSign[pk, (M||h(r_s)||h(r_r)), Sig] = \mathsf{TRUE}$$



Signer $\overset{r_s}{\longleftarrow}$
Receiver $\overset{r_r}{\longleftarrow}$
$M, Hash(r_s)$
$Hash(r_r)$
$M||h(r_s)$
$||$
$h(r_r)$
$Sig(M||h(r_s)||h(r_r))$

Figure 1: Signature Generation

4.2. Signature revocation

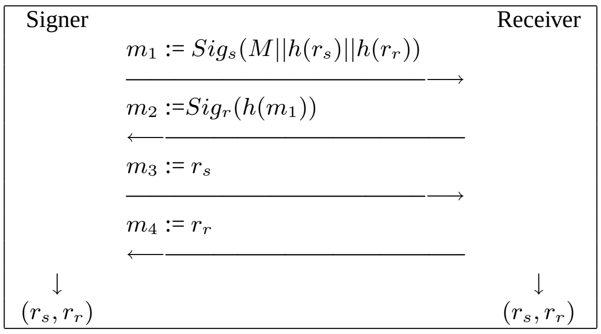
The revocation is straightforward. Just have the signer and receiver exchange r_s and r_r. Everyone can verify

$$Hash(r_s) \overset{?}{=} h(r_s) \quad Hash(r_r) \overset{?}{=} h(r_r)$$

We shall call every qualified pair (r_s, r_r) a Revocation Token for the specific signature instances. Anyone who holds the revocation token can verify the correctness of the revocation. Since the revocation can only be done once, that is no one but the player who embedded the seed of the hash can do this, then no matter whether the signing key is valid now or not, the existence of this revocation action can always be attested with the revocation token correctly constructed. Linkability is also solved for only the related signer and receiver know the seed for the exact singnature instance.

4.3. Construction of revocation token

The signer and receiver must cooperate to correctly construct such a revocation token. It leads to a small negotiation between the Signer and receiver, when they agree to revoke the signature. A typical conversation is shown in Figure 2.



m_1="Let's revoke it."
m_2="I agree."

Figure 2: Negotiation protocol

However, sometimes one of them may be malicious and want to get advantage like this: as soon as he get the desired item, he shut down the communication. Then he has the ability either to revoke or validate the signature, while the other can't. A fair exchange technique [1] can solve this problem, but that will involve a trusted third party, who is originally unnecessary. Techniques in [6] can be used to verifiably exchange a secret verifiably between two parties in a bit-by-bit manner.

5. Security Analysis

Intuitively if r_s and r_r are random chosen, according to the onewayness of hash function, r_s and r_r are hardly computable given from $h(r_s)$ and $h(r_r)$. We build the security of our scheme on the following claim:

Claim 1 *If the underlying signature scheme is secure, then our revocable signature is secure.*

Proof. We give the sketch of the proof by contradiction. If the underlying signature scheme is secure, one can't forge the embedded seed of the hash. Suppose an adversary can find another pre-image of the hash value, such that

$$Hash(r') = Hash(r)$$

However, by assumption we know this is not possible. That completes the proof. ■

5.1. Extension

As we have addressed in section 2.2, hash chains have the same security with a single hash. We further extend our basic scheme into chain revocable signatures (Figure 3).

$$\begin{tikzcd}
 & r_{s1} \arrow[d] & r_{r1} \arrow[d] \\
Sig(M_1||h(r_{s_1})||h(r_{r_1})) \arrow[d] & h(r_{s_1}) \arrow[d] & h(r_{r_1}) \arrow[d] \\
Sig(M_2||h(r_{s_2})||h(r_{r_2})) \arrow[d] & h(r_{s_2}) \arrow[d] & h(r_{r_2}) \arrow[d] \\
\vdots & \vdots & \vdots \\
Sig(M_n||h(r_{s_n})||h(r_{r_n})) & h(r_{s_n}) & h(r_{r_n})
\end{tikzcd}$$

Figure 3: Revocable Signature Chain

Hash chain can be embedded into signatures according to the priority of signatures. Revealing certain level $(h(r_{s_i}), h(r_{r_i}))$ can form the revocation token for all the signature below them. This scheme can be applied to package tour reservation system, where the top level root represents the basic contract and the lower levels stand for the options of the tour. These can be further built into trees, greatly reducing the cost of communication and data storage for structured signature revocation.

6. Conclusion

We have defined requirements of signature revocation and given a robust compact implementation as a generalization of management on signature. The use of hash chain structure extends the basic revokable signatures, which greatly make signature instance management easy. Our scheme enjoys both simplicity and effectiveness. In addition, it is proved as secure as the underlying signature scheme.

References

[1] N. Asokan, Victor Shoup, and Michael Waidner, Optimistic fair exchange of digital signatures, in Advances in Cryptology - EUROCRYPT '98, LNCS vol. 1403, pp. 591-606, SpringerVerlag, 1998.

[2] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In Advances in Cryptology, LNCS 1462, pp.26-45, Springer-Verlag, 1998.

[3] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In 23rd Annual ACM Symposium on Theory of Computing, pp. 542-552, 1991.

[4] S. Micali. Efficient Certificate revocation. Technical Memo MIT/LCS/TM-542b, 1996.

[5] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Advances in Cryptology CRYPTO '91, Lecture Notes in Computer Science Vol. 576, Springer-Verlag, 1991

[6] Andrew C. Yao. How to generate and Exchange secrets. Proceedings of the 27th Symposium on Foundations of Computer Science, pp.162-167, 1986.