Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 1961–1969
computer communications
www.elsevier.com/locate/comcom

Fair exchange protocol of signatures based on aggregate signatures

Zuhua Shao *

Department of Computer and Electronic Engineering, Zhejiang University of Science and Technology, No. 318, LiuHe Road, Hangzhou, Zhejiang 310012, PR China

Received 15 May 2007; received in revised form 21 December 2007; accepted 23 December 2007
Available online 1 January 2008

Abstract

In Eurocrypt 2003, Boneh et al. proposed verifiably encrypted signatures from the concept of aggregate signatures that support aggregation. Such signatures enable verifiers to test that a given ciphertext is the encryption of a signature on a given message. Verifiably encrypted signatures are used in fair exchange protocols of signatures. In this paper, we first show that Boneh et al.'s verifiably encrypted signature is not secure against rogue-key attacks. Moreover, the fairness of fair exchange protocols of signatures with the adjudicator relies on the neutrality of the adjudicator, which has become a major practical hindrance to fair exchange protocols of signatures getting widely deployed. Then we propose a fair exchange protocol of signatures from pairings by using aggregate signatures. We not only enhance the fair exchange protocol of signatures against three types of inside attackers but also relax the need of the trust in the adjudicator so that it only needs to be trusted by the signer.

Keywords: Fair exchange; Aggregate signature; Computational Diffie–Hellman problem

1. Introduction

As electronic commerce is becoming more and more important and popular in the Internet, the problem of fair exchange of signatures has become one of the fundamental problems in secure electronic transactions and digital rights management.

Protocols for fair exchange of signatures attracted much attention in the cryptographic community in the past few years [1–3]. Popular approaches resort to a neutral third party as an adjudicator, who can be called upon to handle possible disputes between the involved parties. In fair exchange protocols of signatures with an off-line adjudicator, the adjudicator is invoked only in case of a network failure or one of the two parties misbehaves. In most normal cases, the protocols can run without any intervention of the adjudicator since the participating parties are honest and the network is well functioning. Hence, those protocols are called optimistic, since they offer a more cost-effective use of the adjudicator.

A practical paradigm for building an optimistic protocol for fair exchange of signatures is based on the concept of verifiably encrypted signatures [4], i.e., a way of encrypting a signature under a designated public key and subsequently proving that the resulting ciphertext indeed contains such a signature. However, many such schemes involve expensive and highly interactive zero-knowledge proofs in the exchange phase [5–7].

The first non-interactive verifiably encrypted signature scheme was constructed by Boneh et al. [8] from aggregate signatures: given n signatures on n distinct messages from n distinct users, it is possible to aggregate all these signatures into a single short signature. This single signature, along with the n original messages will convince verifiers that the n users did indeed sign the n original messages respectively, i.e., user i signed message M_i for $i = 1, \ldots, n$. Later, Zhang et al. [9] proposed a new verifiably encrypted signature scheme from bilinear pairings and further showed that it was more efficient than the previous protocols of this kind. Recently, Lu et al. presented the first verifiably

* Tel.: +86 571 85171332.
E-mail address: zhshao_98@yahoo.com

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.12.030

encrypted signature scheme [10], deriving from a novel application of the Waters's signature scheme [11], which is provably secure without random oracles in the same security model as that of Boneh et al.'s scheme.

A different paradigm for building an optimistic and non-interactive fair exchange protocol of signatures was proposed by Park et al. [12]. They introduced a connection between fair exchange protocols and sequential two-party multi-signature schemes and provided a novel method of constructing fair exchange protocols by distributing the computation of RSA. Their approach avoided the design of verifiably encrypted signature schemes at the cost of having cosigner (adjudicator) store a piece of prime signer's secret key. However, Dodis and Reyzin [13] broke Park et al.'s protocol by pointing out that an honest-but-curious adjudicator can easily derive the private key of the signer after the end of registration phase. Moreover, they proposed a new primitive, called verifiably committed signatures, for constructing fair exchange protocols, and presented a committed signature scheme based on GDH signatures [14] and RSA signatures.

In this paper, we find that the existing verifiably encrypted signature schemes use a weaker security model, in which the key pair of the adjudicator is chosen by the simulator instead of the signature forger. It means that the adjudicator never forges signatures. In the real world, the adjudicator is a potential adversary as Dodis and Reyzin thought [13]. Hence, we propose a rogue-key attack against the Boneh et al.'s scheme, where the signature forger is allowed to choose the key pair of the adjudicator.

Other weakness in the security model of fair exchange protocols is the excessive reliance on the trust in the adjudicator. The fairness of exchange protocols with the adjudicator so far, no matter whether from verifiably encrypted signatures or from verifiable committed signatures, relies on the neutrality of the adjudicator. If he colludes with one party, the other would be duped. Hence both involved mistrusted parties must negotiate to choose a trusted third party as the adjudicator before exchanging. This reliance on the trust in the adjudicator has become an obstacle in the real world, since it is difficult for mistrusted parties to seek unity of thinking on the honesty of a third party over the open Internet, particularly for those parties in different countries with unbalanced information.

Then we propose a fair exchange protocol of signatures from pairings by using the aggregate signatures of Boneh et al. directly. We further present a security model to show that the proposed protocol is existential unforgeable against adaptive chosen message attacks (EUF-CMA) launched by three types of inside attackers in the random oracle model. We not only enhance the fair exchange protocol of signatures in the chosen key security model but also relax the need of trust in the adjudicator so that it only needs to be trusted by the signer. These two contributions make fair exchange protocols of signatures more trustworthy and practical than the previous ones.

The rest of this paper is organized as follows. Section 2 analyses the verifiably encrypted signature schemes of Boneh et al. Then Section 3 describes the new fair exchange protocol of signatures based on aggregate signatures. In Section 4, we provide a formal security proof for the new protocol. Finally, we sum up the merits and shortcomings of the new protocol in Section 5.

2. Aggregate and verifiably encrypted signature from pairings

In this section, we first briefly review the verifiably encrypted signatures from pairings and the associated computation problems proposed by Boneh et al. Then we analyze the weakness in their security model and propose a rogue-key attack launched by a malicious adjudicator.

2.1. Review of pairings

Let G_1 and G_2 be two cyclic groups of the same large prime order q. We write G_1 and G_2 additively and multiplicatively groups, respectively. Let P be a generator of G_1 and H: $\{0,1\}^* \to G_1$ be a full-domain hash function. Assume that the discrete logarithm problems in G_1 and G_2 are hard. Let e: $G_1 \times G_1 \to G_2$ be an admissible pairing which satisfies the following properties:

1. *Bilinear.* $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$, and $a, b \in Z_q$, where Z_q is set of non-negative integers less than q.
2. *Non-degenerate.* There exist $P, Q \in G_1$ such that $e(P, Q) \neq 1$.
3. *The map* f_P. $G_1 \to G_2$ by $f_P(Q) = e(Q, P)$, where $P \in G_1^*$, is believed to be a one-way isomorphic function.

Computational Diffie–Hellman problem: given aP, bP in a group G_1 compute $abP \in G_1$, for any a, $b \in Z_q$.

Definition. The advantage of an algorithm B is solving the computational Diffie–Hellman problem in group G_1 is

$$\text{Adv CDH}_B = \Pr[B(P, aP, bP) = abP : a, b \in_R Z_q, P \in_R G_1]$$

The probability is taken over the choices of P, aP, bP and B's coin tosses.

2.2. Verifiably encrypted signature scheme

Boneh et al. motivated their construction for verifiably encrypted signatures by considering aggregate signatures as a launching point. Their scheme comprises six algorithms:

Key generation. The signer and the adjudicator generate their key pairs $\{x, Y\}$ and $\{x_A, Y_A\}$, respectively.
Sign. For a message $M \in \{0,1\}^*$, a signer with key pair $\{x, Y\}$ computes his individual signature $\sigma = xH(M)$.

Verify. Each individual signature σ can be verified by checking $e(\sigma, P) = e(H(M), Y)$.

VESig Creation. Given the private key x of the signer, the public key Y_A of the adjudicator and a message M in $\{0,1\}^*$, select r at random in Z_q^* and compute $\sigma = xH(M)$, $U = rP$ and $W = \sigma + rY_A$. The verifiably encrypted signature is (U, W).

VESig Verification. Given the pair (U, W) on the public key Y of the signer, the public key Y_A of the adjudicator and a message M, accept if $e(W, P) = e(H(M), Y)e(U, Y_A)$.

Adjudication. Given the key pair $\{x_A, Y_A\}$ of the adjudicator, a certified public key Y, and a verifiably encrypted signature (U, W) on some message M, ensure that the verifiably encrypted signature is valid; then output $\sigma = W - x_A U$.

Remark 1. Boneh et al. stressed the public key Y of the signer is *certified* by a CA, (assumed that the CA, in issuing a certificate on Y, verifies that the user knows the private key for Y), whereas not mentioning if the public key Y_A of the adjudicator is *certified* by the CA.

2.3. The weakness in the security model

The weakness in the verifiably encrypted signature scheme of Boneh et al. comes from the role of the adjudicator. They assumed that the adjudicator was entirely trusted by both the signer and verifier. That is, the adjudicator would not only never forge verifiably encrypted signatures, but also be free from any bias. As a result, the function of the adjudicator is beyond those required of a normal Certification Authority. Lu et al.'s scheme also uses this security model.

2.3.1. The rogue-key attacks

Because, in the real world, the adjudicator is a potential adversary, we propose a rogue-key attack against Boneh et al.'s scheme launched by the adjudicator. In a so-called rogue-key attack, an adversary chooses its public key as a function of other honest users in such a way that it can then easily forge multi-signatures.

1. First, the adversary selects x_A at random in Z_q^* and computes $Y_A = x_A Y$ as his public key, where Y is the public key of a signer.
2. Then, the adversary applies for the certificate of the public key Y_A to a CA. The standards PKCS#10 [15] – used by VeriSign – and RFC 4210/11 [16,17] demand an applicant to send the CA a signature, under the public key it is attempting to get certified, of some message that includes the public key and the identity of the applicant. Suppose that such message for the adversary is PI and the adversary is able to ask the signer to sign the message PI by some ways, for example, blind signatures, fair exchange protocols of signatures and chosen-message attacks. Then the adversary would obtain σ with $e(\sigma, P) = e(H(PI), Y)$. Then the adversary computes $\sigma' = x_A\sigma$. Obviously, $e(\sigma', P) = e(H(PI), x_A Y) = e(H(PI), Y_A)$. Consequently, the adversary gets a certificate of the public key Y_A from the CA.
3. Finally, the adversary can forge a verifiably encrypted signature under the given certified public key Y of the signer and the public key Y_A of the adjudicator. For a message M in $\{0,1\}^*$, the adversary selects a at random in Z_q^* and computes $W = aY$ and $U = x_A^{-1}(aP - H(M))$. Then the adversary claims that the pair (U, W) is the verifiably encrypted signature under the public key Y of the signer and the public key Y_A of the adjudicator.

Because $e(W, P) = e(aY, P) = e(Y, aP) = e(Y, x_A U + H(M)) = e(Y, H(M))e(Y, x_A U) = e(H(M), Y)e(U, Y_A)$, the verifier cannot find that the pair (U, W) is forged by the adjudicator even if the verifier validates the public key Y_A of the adjudicator.

Therefore, if the adversary is able to ask the signer to sign the message PI by some ways, or if the CA does not demand an applicant to demonstrate that he knows the private key corresponding to the public key it is attempted to get certified, or if verifiers do not verify the public key of the adjudicator, the malicious adjudicator can cheat verifiers into accepting forged verifiably encrypted signatures for any message of its choice.

On the other hand, the malicious adjudicator does not know his own private key, which prevents him from playing his role as adjudicator: all he can do is to forge signatures against the signer, nothing more.

2.3.2. The neutrality of the adjudicator

The fairness of signature exchange protocols with the adjudicator, no matter whether from verifiably encrypted signatures or from verifiable committed signatures, relies on the neutrality of the adjudicator. If he colludes with one party, the other would be duped. For example, in a fair exchange protocol based on verifiably encrypted signatures, the signer computes a signature, and then encrypts it under the public key of a designated third party. Though the encrypted signature is undeniable assuming the adjudicator is honest, encrypting means that the signer would not release the signature unless the verifier fulfills its obligation. Though the adjudicator is able to decrypt the encrypted signature, *he does not commit any thing at all*. He would either refuse to extract the signature even if the verifier fulfills its obligation or presumptuously extract the signature without seeing the fulfilling obligation.

3. New fair exchange protocol of signatures

In this section, we describe a fair exchange protocol of signatures from aggregate signature. To turn an aggregate signature into a fair exchange protocol, we combine the message signature of the signer with a valid voucher signature of the adjudicator, rather than with a forged signature of the adjudicator as the verifiably encrypted signatures of Boneh et al. did. Consider the following:

A fair exchange protocol of signatures comprises eight algorithms: *KeyGen*, *Sign*, *Verify*, *VsigCreate*, *AsigCreate*, *AsigVerify*, *Adjudication*, and *Abort*. The algorithms are described below. There are three parties in a fair exchange protocol of signatures: a signer, a verifier and an adjudicator chosen by the signer.

KeyGen. Take as input 1^k, run the randomized algorithm to generate the system parameters $\langle q, G_1, G_2, e, P\rangle$ and a full-domain hash function $H:\{0,1\}^* \to G_1$. The signer picks up at random x in Z_q^* as her private key and computes her public key $Y = xP$. Similarly, the adjudicator chooses his key pair $\{x_A, Y_A\}$.

VsigCreate. The adjudicator composes a voucher V (*affidavit*), which stipulates the obligation the verifier must fulfill to exchange signer's signature in aggregate signatures and explicitly promises that if the signer fails in sending her signature to exchange the verifier's fulfilling obligation, the adjudicator would do.

For a voucher $V \in \{0,1\}^*$, the adjudicator computes his voucher signature $\sigma_A = x_A H(V)$. Then the adjudicator chooses a random integer $r \in Z_q^*$ and computes $S = rP, T = \sigma_A + rY$. Finally, the adjudicator sends (S, T) to the signer over a public channel.

The signer first recovers σ_A by computing $\sigma_A = T - xS$. Then the signer verifies σ_A by checking the following equation:

$$e(\sigma_A, P) = e(H(V), Y_A).$$

AsigCreate. For a message $M \in \{0,1\}^*$ and $M \neq V$, the signer computes her message signature $\sigma = xH(M)$, then computes the aggregate signature $W = \sigma + \sigma_A$, where σ_A is her current voucher signature received from the adjudicator. The message contains a statement: the message signature would go into effect if and only if the message signature is extracted from the aggregate signature by the signer or the adjudicator when the verifier fulfills his obligation. Then signer sends the aggregate signature (W, M, V) to the verifier.

AsigVerify. Each aggregate signature (W, M, V) can be verified by checking $e(W, P) = e(H(M), Y)e(H(V), Y_A)$ and $M \neq V$.

Sign. When the signer receives items from the verifier stipulated in the obligation, after validating them, the signer first generates a message M' that has the same meaning as the message M. For example, a space is added to the original message M or deleted in M. The signer computes her message signature as $\sigma' = xH(M')$ with the key pair $\{x, Y\}$. Finally, the signer sends (σ', M') to the verifier.

Verify. Each message signature (σ', M') can be verified by checking the equation $e(\sigma', P) = e(H(M'), Y)$.

Adjudication. Given an aggregate signature (W, M, V) sent by the verifier, the adjudicator first checks if the verifier has fulfilled his obligation and if the aggregate signature and the voucher are valid. If so, the adjudicator extracts signer's message signature $\sigma = W - x_A H(V)$. If V is not the current voucher, the aggregate signature can serve as an evidence of signer's misbehavior.

After sending σ to the verifier, the adjudicator updates his voucher signature $\sigma'_A = x_A H(V')$ for a new voucher V' with new time information. The adjudicator sends (σ'_A, V') to the signer as her current voucher signature together with verifier's obligation as the *VsigCreate* algorithm does.

Abort. When the signer sends the aggregate signature to the verifier, the reply of the verifier does not arrive to the signer after a reasonable amount of time [18] (the signer chooses herself how long she decides to wait for a given message before reaction), the signer asks the adjudicator to execute an abort algorithm to update his voucher signature $\sigma'_A = x_A H(V')$ for a new voucher V'.

It is easy to see that the output of the *Adjudication* algorithm is a valid ordinary signature σ of the message M under the public Y of the signer. Thus, the normal signature (σ', M') sent from the signer and the abnormal signature (σ, M) sent from the adjudicator are valid signatures on the two messages with the same meaning. Hence, this fair exchange protocol is transparent since at the end of the protocol, it is impossible to decide whether the adjudicator did intervene or not, by only looking at the evidences. This property is useful in electronic commerces.

4. Security of fair exchange protocol based on aggregate signatures

The components of aggregate signatures, i.e., message signatures computed by the signer and voucher signatures computed by the adjudicator, are existential unforgeable against adaptive chosen message attacks (EUF-CMA) [19] since they are the short signature proposed by Boneh et al. [14].

Besides them, we require three more security properties for the fair exchange protocol of signatures: unforgeability, opacity, and extractability.

1. Unforgeability requires that it is difficult for the signer to forge an aggregate signature (W, M, V) without seeing the voucher signature σ_A under the public key Y_A of the adjudicator, and that it is also difficult for the adjudicator to forge an aggregate signature (W, M, V) alone.
2. Opacity requires that it is difficult for the verifier, given an aggregate signature (W, M, V), to extract the ordinary signature σ of the message M under the public key Y of the signer.
3. Extractability requires that it is difficult for the signer to generate aggregate signatures, from which the adjudicator cannot extract signer's message signatures.

We provide security models and proofs for unforgeability, opacity, and extractability, where three types of inside

attackers have more powers than those in Boneh et al.'s security models.

In the security model, three parties in a fair exchange protocol, a singer, a verifier and an adjudicator, are all potential adversaries.

A malicious signer wants to forge an aggregate signature without the corresponding voucher signature of the honest adjudicator. The forged aggregate signature is with respect to the public key of the honest adjudicator besides its public key. He is allowed to query any aggregate signature for any message and voucher of its choice. From aggregate signatures he can extract any voucher signature of the honest adjudicator by using its private key.

A malicious adjudicator wants to forge an aggregate signature without the corresponding message signature of the honest signer. The forged aggregate signature is with respect to the public key of the honest signer besides its public key. He is allowed to query any aggregate signature for any message and voucher of its choice. From aggregate signatures he can extract any message signature of the honest signer by using its private key.

A malicious verifier wants to obtain a signer's message signature without fulfilling his obligation in fair exchanging. He is allowed to query any aggregate signature for any message and voucher of its choice, and then query aggregate signature extraction. From the answers, he would obtain individual message signatures or voucher signatures.

4.1. Security model and proof for unforgeability

We say that the fair exchange protocol of signatures is existential unforgeable against adaptive chosen message attacks (EUF-CMA) launched by an adjudicator if no polynomial bounded adversary A has a non-negligible advantage against the challenger in the following game:

KeyGen. The challenger takes as input 1^k, runs the randomized algorithm to generate the system parameters $\langle q, G_1, G_2, e, P\rangle$ and a full-domain hash function H: $\{0,1\}^* \to G_1$. Then the challenger generates a public key Y of the signer at random. Finally, the challenger gives the results to the adversary.

Queries. A issues queries $q_1, \ldots, q_n$ adaptively, where query q_i is one of:

- Hash function query $\langle M_i\rangle$ or $\langle V_j\rangle$.
- Aggregate signature query $\langle M_i, Y, V_j, Y_j\rangle$, where Y_j is a public key of the adjudicator chosen by A.

Response. Finally, the adversary A outputs an aggregate signature (W, M_i, V_j) under public keys (Y, Y_A), where Y_A is the public key of the adjudicator chosen by A.

The adversary A wins the game if the output signature is non-trivial, i.e., A did not request any aggregate signature on the message M_i. The probability is over the coin tosses of the key generation algorithm and of A.

Remark 2. This is a chosen key security model, in which the adversary is allowed to arbitrarily choose key pairs of the adjudicator. Meanwhile, ordinary message signature queries are not provided since they can be simulated by aggregate signature queries followed by extracting with the private key chosen by A.

Theorem 1. *Suppose that there is an adjudicator adversary A that has advantage ε to forge an aggregate signature in the chosen key security model, and A runs in time at most t. Suppose that A makes at most q_H adaptive queries to the hash function, and at most q_{as} adaptive queries to the aggregate signing oracle. Then there is a CDH algorithm B that has an advantage ε' with running time t', where:*

$$\varepsilon \leqslant e(2q_{\mathrm{as}} + 2)\varepsilon' \tag{1}$$

$$t \approx t' - (q_H + 4q_{\mathrm{as}} + 4)c_{G_1} \tag{2}$$

where e is the base of natural logarithms, and one point scalar on G_1 takes time c_{G_1}.

Proof. We show how to construct a CDH adversary B that uses A as a computer program to gain an advantage ε' with running time t'. The challenger runs a randomized algorithm to obtain $\langle q, G_1, G_2, e, P, aP, bP\rangle$. Its goal is to output $D = abP \in G_1$. Algorithm B simulates the challenger and interacts with A as follows. □

KeyGen. Algorithm B takes as input 1^k, runs the randomized algorithm to generate the system parameters $\langle q, G_1, G_2, e, P\rangle$, a full-domain hash function H: $\{0,1\}^* \to G_1$, $Q_1 = aP$ and $Q_2 = bP$. Algorithm B generates the public key Y of the signer by $Y = rQ_1$, where r is a random in Z_q^*. Finally, Algorithm B gives these results to A.

Answering H-oracle queries $\langle M_i\rangle$ or $\langle V_i\rangle$:

1. B generates a random $\mathrm{coin}_i \in \{0,1\}$ so that $\Pr[\mathrm{coin}_i = 0] = \delta$ for some δ that will be determined later.
2. B picks a random c_i in Z_q^*. If $\mathrm{coin}_i = 0$, B computes $h_i = c_i Q_2$. If $\mathrm{coin}_i = 1$, B computes $h_i = c_i P$.
3. B responds with $h_i = H(M_i)$ and adds the tuple $\langle M_i, h_i, c_i, \mathrm{coin}_i\rangle$ to the H-list.

Obviously, either way, h_i is uniform in G_1 and is independent of A's current view as required.

Answering aggregate signature queries $\langle M_i, Y, V_j, Y_j\rangle$:

1. B runs the above algorithm for responding to H-queries on M_i, obtaining the corresponding tuple $\langle M_i, h_i, c_i, \mathrm{coin}_i\rangle$ on the H-list. If $\mathrm{coin}_i = 0$, then B reports failure and terminates.
2. If $\mathrm{coin}_i = 1$ holds, we know that $h_i = c_i P$. Let $\sigma_i = c_i Y \in G_1$. Observe that $e(\sigma_i, P) = e(c_i Y, P) = e(c_i P, Y) = e(H(M_i), Y)$. Therefore, σ_i is a valid message signature on M_i under the public key Y.
3. Then, Algorithm B computes the voucher signature (σ_j, V_j) under the public key Y_j similarly.

4. Finally, Algorithm B computes $W = \sigma_i + \sigma_j$. If the corresponding $\text{coin}_i = \text{coin}_j = 1$, Algorithm B could generate a valid aggregate signature. Otherwise, B reports failure and terminates. The probability of success is at least $(1-\delta)^2$.

Obviously, the outputs of the simulated oracles are indistinguishable from those in the real attacks.

If B does not report failure, Algorithm A would return a non-trivial aggregate signature (W, M_i, V_j) with probability ε. Thus, $e(W,P) = e(H(M_i), Y)e(H(V_j), Y_A)$ and A did not request any aggregate signature on the message M_i.

Algorithm B runs its hash algorithm at (M_i, V_j), obtaining the corresponding tuples $\langle M_i, h_i, c_i, \text{coin}_i\rangle$ and $\langle V_j, h_j, c_j, \text{coin}_j\rangle$ on the H-list.

Algorithm B now proceeds only if $\text{coin}_i = 0$ and $\text{coin}_j = 1$; otherwise B declares failure and halts. Note that $\text{coin}_j = 1$ implies $H(V_j) = h_j = c_j P$. Algorithm B can compute $\sigma_j = c_j Y_A$ so that $e(\sigma_j, P) = e(c_j Y_A, P) = e(c_j P, Y_A) = e(H(V_j), Y_A)$.

Thus, Algorithm B can compute $\sigma_i = W - \sigma_j$ so that $e(\sigma_i, P) = e(W,P)/e(H(V_j), Y_A) = e(H(M_i), Y)$.

Because $\text{coin}_i = 0$, $H(M_i) = h_i = c_i Q_2$ holds. Hence,

$$e(\sigma_i, P) = e(H(M_i), Y) = e(c_i Q_2, rQ_1) = e(rc_i a Q_2, P)$$

Because the map, f_P: $G_1 \to G_2$ by $f_P(Q) = e(Q,P)$, is an isomorphic map, $\sigma_i = rc_i a Q_2$, which implies $D = aQ_2 = abP = (rc_i)^{-1}\sigma_i$.

Therefore, Algorithm B can derive D if $\text{coin}_i = 0$ and $\text{coin}_j = 1$. Otherwise B declares failure and halts.

Now, it remains to compute the probability ε' that Algorithm B can derive D in the attack game.

First, we compute the probability that B does not abort during the simulation. To respond an aggregate signature query, B runs at most two hash queries, if $\text{coin}_i = \text{coin}_j = 1$, B does not abort. Hence

$$\Pr[B \text{ does not abort during the simulation}] = (1-\delta)^{2q_{\text{as}}}$$

Then, Algorithm A returns a non-trivial aggregate signature (W, M_i, V_j) with probability ε.

Finally, Algorithm B transforms A's forgery into the CDH solution. If A returns a non-trivial aggregate signature (W, M_i, V_j), the probability that Algorithm B can derive D is that of $\text{coin}_i = 0$ and $\text{coin}_j = 1$. Hence, the probability that Algorithm B can derive D from the output of A is at least $\delta(1-\delta)$.

Therefore, the probability ε' that Algorithm B can derive D is at least $\varepsilon\delta(1-\delta)^{2q_{\text{as}}+1}$. This expression is optimal for $\delta = 1/(2q_{\text{as}}+2)$. For a huge value $2q_{as}+2$, the success probability is approximately $\varepsilon/(e(2q_{\text{as}}+2))$.

Therefore, $\varepsilon \leqslant e(2q_{\text{as}}+2)\varepsilon'$.

The running time of Algorithm B is that of Algorithm A plus time taken to respond to q_H hash queries, q_{as} aggregate signature queries and the time taken to transform A's forgery into the CDH solution. Each hash query requires one point scalar in G_1. An aggregate signature query requires at most two hash queries and two point scalars in G_1. To transform A's forgery into the CDH solution, B requires at most two additional hash queries, two point scalars. Hence, $t' \approx t + (q_H + 4q_{\text{as}} + 4)c_{G_1}$.

Notice that we only consider the time taken to compute point scalars in G_1, since it is more time-consuming than point additions in G_1.

By the same reason, we can show that it is also difficult for the malicious signer to forge aggregate signatures alone in the chosen key security model.

4.1.1. Security model and proof for opacity

We say that the fair exchange protocol of signatures is existential unforgeable against single-occurrence adaptive chosen message attacks (SO-CMA) launched by a malicious verifier if no polynomial bounded adversary A has a non-negligible advantage against the challenger in the following game:

KeyGen. The challenger takes as input 1^k, runs the randomized algorithm to generate the system parameters $\langle q, G_1, G_2, e, P\rangle$ and a full-domain hash function H: $\{0,1\}^* \to G_1$. Then the challenger generates, at random, two public key Y and Y_A of the signer and the adjudicator, respectively. Finally, the challenger gives these results to the adversary.

Queries. A issues queries $q_1, \ldots, q_n$ adaptively, where query q_i is one of:

– Aggregate signature query $\langle M_i, V_j\rangle$.

– Aggregate signature extract query $(H(M_i), H(V_j), W_{ij})$.

Only constraint is that the adversary A is allowed to ask at most one aggregate signature query for each message M_i.

Response. Finally, the adversary A extracts a message signature $(\sigma_i, M_i, H(M_i))$ from the answer $(H(M_i), H(V_j), W_{ij})$ of an aggregate signature query $\langle M_i, V_j\rangle$.

The adversary A wins the game if the output message signature is non-trivial, i.e., A did not request any aggregate signature extract query on the voucher $H(V_j)$. The probability is over the coin tosses of the key generation algorithm and of A.

Remark 3. we assume that an adversary A is well behaved in the sense that it never requests extracts on messages on which it had not previously asked for an aggregate signature. It is reasonable since the input to extract oracles in this case would be a valid aggregate signature that can only be generated by using the two private keys. A can also be modified simply to output it and halt [8].

Remark 4. The adversary A does not need the hash queries, since he is required to extract message signature $(\sigma_i, M_i, H(M_i))$ from the answer $(H(M_i), H(V_j), W_{ij})$ of an aggregate signature query $\langle M_i, V_j\rangle$.

Theorem 2. *Suppose that there is an adversary A that has advantage ε to extract a message signature, and A runs in time at most t. Suppose that A makes at most q_{as} adaptive queries to the aggregate signing oracle, and at most q_{ae} adaptive queries to the aggregate signature extract oracle. Then there is a CDH algorithm B that has an advantage ε' with running time t', where*:

$$\varepsilon \leqslant (e(q_{as}+1))\varepsilon' \quad (3)$$

$$t \approx t' - (q_{ae} + 4q_{as} + 1)c_{G_1} \quad (4)$$

where e is the base of natural logarithms, and one point scalar on G_1 *takes time* c_{G_1}.

Proof. We show how to construct a CDH adversary B that uses A as a computer program to gain an advantage ε' with running time t'. The challenger runs a randomized algorithm to obtain $\langle q, G_1, G_2, e, P, aP, bP\rangle$. Its goal is to output $D = abP \in G_1$. Algorithm B simulates the challenger and interacts with A as follows. □

KeyGen. Algorithm B takes as input 1^k, runs the randomized algorithm to generate the system parameters $\langle q, G_1, G_2, e, P\rangle$, a full-domain hash function H: $\{0,1\}^* \to G_1$, $Q_1 = aP$ and $Q_2 = bP$. Algorithm B generates the public keys of the signer and the adjudicator by $Y = rQ_1$, and $Y_A = tQ_1$, respectively, where r, t are chosen at random in Z_q^*. Finally, Algorithm B gives these results to A.

Answering aggregate signature queries $\langle M_i, V_j\rangle$:

1. B looks up the AS-list (a query-answer list, where entry consists of $(M_i, H(M_i), c_i, V_j, H(V_j), d_j, \text{coin}_j, W_{ij})$) to get the corresponding answer.
2. If there exists an entry $(M_i, H(M_i), c_i, V_j, H(V_j), d_j, \text{coin}_j, W_{ij})$ in the AS-list, B answers with $(H(M_i), H(V_j), W_{ij})$.
3. If there exists an entry $(M_i', H(M_i'), c_i', V_j, H(V_j), d_j, \text{coin}_j, W_{ij})$ in the AS-list, B obtains d_j and coin_j. Otherwise, B picks a random d_j in Z_q^* and generates a random $\text{coin}_j \in \{0,1\}$ so that $\Pr[\text{coin}_j = 0] = \delta$ for some δ that will be determined later.
4. B picks a random c_i in Z_q^*.
5. If $\text{coin}_j = 1$, B computes $H(M_i) = c_iP$, $H(V_j) = d_jP$, and $W_{ij} = (c_ir + d_jt)Q_1$; otherwise, B computes $H(M_i) = c_iP - tr^{-1}d_jQ_2$, $H(V_j) = d_jQ_2$, and $W_{ij} = c_irQ_1$.
6. B answers with $(H(M_i), H(V_j), W_{ij})$, and adds $(M_i, H(M_i), c_i, V_j, H(V_j), d_j, \text{coin}_j, W_{ij})$ to the AS-list.

Consistency. If $\text{coin}_j = 1$ holds, then $H(M_i) = c_iP$, $H(V_j) = d_jP$, and $W_{ij} = (c_ir + d_jt)Q_1$. Thus,

$$e(W_{ij},P) = e((c_ir + d_jt)Q_1, P) = e(c_irQ_1, P)e(d_jtQ_1, P) \quad (5)$$

$$= e(c_iP, rQ_1)e(d_jP, tQ_1) = e(H(M_i), Y)e(H(V_j), Y_A). \quad (6)$$

If $\text{coin}_j = 0$, then $H(M_i) = c_iP - tr^{-1}d_jQ_2$, $H(V_j) = d_jQ_2$, and $W_{ij} = c_irQ_1$. Thus

$$\begin{aligned} e(W_{ij},P) &= e(c_irQ_1, P) = e(c_iP, rQ_1) \\ &= e(H(M_i), rQ_1)e(tr^{-1}d_jQ_2, rQ_1) \quad (7) \\ &= e(H(M_i), rQ_1)e(d_jQ_2, tQ_1) \\ &= e(H(M_i), Y)e(H(V_j), Y_A). \quad (8) \end{aligned}$$

Hence, in the two cases, the answer is valid aggregate signature on $\langle M_i, V_j\rangle$.

Answering aggregate signature extract query $\langle H(M_i), H(V_j), W_{ij}\rangle$:

1. B looks up the AS-list to get $(M_i, H(M_i), c_i, V_j, H(V_j), d_j, \text{coin}_j, W_{ij})$.
2. If $\text{coin}_j = 0$, B declares failure and halts.
3. If $\text{coin}_j = 1$ holds, then $H(M_i) = c_iP$. B computes $\sigma_i = c_iY$. Thus, $(\sigma_i, M_i, H(M_i))$ is a valid message signature, since $e(\sigma_i, P) = e(c_iY, P) = e(c_iP, Y) = e(H(M_i), Y)$. The probability of success is at least $(1 - \delta)$.

Obviously, the outputs of the simulated oracles are indistinguishable from those in the real attacks.

If B does not report failure, Algorithm A would return a non-trivial message signature $(\sigma_i, M_i, H(M_i))$ from the answer $\langle H(M_i), H(V_j), W_{ij}\rangle$ of an aggregate signature query. Then $e(\sigma_i, P) = e(H(M_i), Y)$.

B looks up the AS-list to get $(M_i, H(M_i), c_i, V_j, H(V_j), d_j, \text{coin}_j, W_{ij})$. So, $e(W_{ij}, P) = e(H(M_i), Y)e(H(V_j), Y_A)$.

Note that A did not request any aggregate signature extract query $(H(M_i'), H(V_j), W_{ij}')$. If $\text{coin}_j = 0$, $H(V_j) = d_jQ_2$. Thus, $e(W_{ij} - \sigma_i, P) = e(H(V_j), Y_A) = e(d_jQ_2, tQ_1) = e(atd_jQ_2, P)$.

It implies $D = aQ_2 = abP = (td_j)^{-1}(W_{ij} - \sigma_i)$.

Now, it remains to compute the probability ε' that Algorithm B can derive D in the attack game.

First, we compute the probability that B does not abort during the simulation. To respond an aggregate signature extract query, B looks up the AS-list.

If $\text{coin}_j = 1$, B does not abort. Hence,

$$\Pr[B \text{ does not abort during the simulation}] = (1-\delta)^{q_{ae}}$$

Then, Algorithm A returns a non-trivial message signature $(\sigma_i, M_i, H(M_i))$ with probability ε.

Finally, Algorithm B transforms A's forgery into the CDH solution. If A extracts a non-trivial message signature $(\sigma_i, M_i, H(M_i))$ from the answer $\langle H(M_i), H(V_j), W_{ij}\rangle$ of an aggregate signature query, the probability that Algorithm B can derive D is that of $\text{coin}_j = 0$, which is δ.

Therefore, the probability ε' that Algorithm B can derive D is at least $\varepsilon\delta(1-\delta)^{q_{as}}$. This expression is optimal for $\delta = 1/(q_{as} + 1)$. For a huge value $q_{as} + 1$, the success probability is approximately $\varepsilon/(e(q_{as} + 1))$.

Therefore, $\varepsilon \leqslant e(q_{as} + 1)\varepsilon'$.

The running time of Algorithm B is that of Algorithm A plus time taken to respond to q_{as} aggregate signature queries, q_{ae} aggregate signature extract queries and the time taken to transform A's forgery into the CDH solution. An aggregate signature query requires at most four point scalars. An aggregate signature extract query requires at most one point scalar. To transform A's forgery into the CDH solution, B requires at most one point scalar. Hence

$$t' \approx t + (q_{ae} + 4q_{as} + 1)c_{G_1}$$

Notice that we only consider the time taken to compute point scalars in G_1, since it is more time-consuming than point additions in G_1.

4.2. Security proof for extractability

The verification equation is $e(W,P) = e\ (H(M),Y)\ e(H(V),Y_A)$. Thus $e(H(M),Y) = e(W,P)/e(H(V),x_AP) = e(W - x_AH(V),P)$. With his private key x_A, the adjudicator can easily compute signer's message signature $\sigma = W - x_AH(V)$.

Therefore, our fair exchange protocol based on aggregate signatures is secure under the assumption of the CDH problem in the random oracle model.

5. Conclusions

The fair exchange protocols of signatures result from distrust spreading in the Internet. However, the fair exchange protocols of signatures of both Boneh et al. and Lu et al. are built on the trust in the adjudicators. They are entirely honest, neither forging signatures nor colluding with one party. It is very difficult to find such gentlemen in the Internet today. To overcome this difficulty, we have proposed a fair exchange protocol of signatures by directly using the aggregate signatures of Boneh et al. Our idea is to enforce the adjudicator accountability in fair exchanges of signatures. The aggregate signature contains adjudicator's voucher signature, by which the adjudicator commits to extract signer's message signature from the aggregate signature, if the verifier fulfills his obligation, instead of not committing any thing in verifiably encrypted signatures.

By verifying an aggregate signature, the verifier is convinced that the signer and the adjudicator did indeed sign the message and the voucher (*affidavit*), respectively, and the adjudicator can extract signer's message signature from the aggregate signature afterwards. Unlike the verifiably encrypted signatures, in this way, the verifier is guaranteed by the adjudicator to get signer's message signature, even if the signer fails in releasing it, as long as the aggregate signature is valid and the verifier fulfills his obligation since the adjudicator cannot deny his voucher signed by his private key. The adjudicator has no excuse for not to extract the message signature computed by the signer since the communication channel between the verifier and the adjudicator is assumed to be resilient. Hence, the adjudicator does not need to be trusted by the verifier.

Therefore, in the new protocol, the reliance on the trust in the adjudicator is relaxed, which only needs to be trusted by the signer. This advantage over the previous fair exchange protocols would make tradesmen easily exchange their signatures with doubters by alone asking some famous companies, banks or *CA*s as their vouchers in electronic commerce transactions. Therefore, the new fair exchange protocol built on aggregate signatures is more practical in the Internet.

The other contribution is that the new fair exchange protocol of signatures uses a stronger security model, in which the adversary is allowed to choose one public key except the challenged public key. Hence, neither the signer nor the adjudicator can forge aggregate signatures alone.

Because the new protocol directly uses the aggregate signatures of Boneh et al., in normal cases, the overheads of computation and communication required by both the signer and the verifier, as well as the exchanged signature, are the same as those in the fair exchange protocol based on the verifiably encrypted signatures of Boneh et al. Therefore, the new fair exchange protocol based on aggregate signatures is optimistic.

However, in abnormal cases, the two shortcomings of the new protocol would be revealed. One shortcoming is that when invoked in abnormal cases to ensure fairness, the adjudicator would do one more operation than those based on the verifiably encrypted signatures. The adjudicator should update his current voucher signature (σ_A, V) whenever the signer asks for the abort algorithm and whenever the verifier asks for the adjudication algorithm.

The other is that when the verifier asks for the adjudication algorithm, the adjudicator would release his current voucher signature σ_A. Maybe there are several message signatures for some messages M masked by the current voucher signature σ_A. In normal cases, the verifier has already received the corresponding message signature for the message M', which has the same meaning as the original message M. Now, with the knowledge of the current voucher signature σ_A, the verifier would obtain the second message signature. Though the difference between the two messages is only a space, their digests are quite different, so are the two message signatures.

Recently, Wang and Kuo [20] proposed a contract signing protocol to protect signature privacy from TTP, though they wanted to protect contract content secret. Because, they used the (extended) verifiable encrypted signature of Boneh et al., their protocol also suffered from the rogue-key attack, by which not only the adjudicator but also the verifier can forge aggregate signatures.

References

[1] O. Goldreich, A simple protocol for signing contracts, Crypto'83, Plenum Press, Springer-Verlag, 1983, pp. 133–136.
[2] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts, Communications of the ACM 28 (6) (1985) 637–647.
[3] S. Kremer, O. Markowitch, J. Zhou, An intensive survey of non-repudiation protocols, Computer Communications 25 (17) (2002) 1606–1621.
[4] J. Camenisch, I.B. Damgard, Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes, in: Proceeding of Asiacrypt, LNCS, Springer-Verlag, Berlin, vol. 1976, 2000, pp. 331–345.
[5] F. Bao, R. Deng, W. Mao, Efficient and practical fair exchange protocols with off-line TTP, in: Proceedings of the IEEE Symposium on Security and Privacy, 1998, pp. 77–85.

[6] C. Boyd, E. Foo, Off-line fair payment protocols using convertible signature, in: Proceeding of Asiacrypt'98, LNCS, Springer-Verlag, Berlin, vol. 1514, 1998, pp. 271–285.
[7] G. Ateniese, Efficient verifiable encryption (and fair exchange) of digital signatures, in: Sixth ACM Conference on Computer and Communication Security, ACM, 1999, pp. 138–146.
[8] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, in: Proceeding of Eurocrypt, LNCS, Springer-Verlag, Berlin, vol. 2656, 2003, pp. 416–432.
[9] F. Zhang, R. Safavi-Naini, W. Susilo, Efficient verifiably encrypted signature and partially blind signature from bilinear pairings, in: Proceeding of Indocrypt, LNCS, Springer-Verlag, Berlin, vol. 2904, 2003, pp. 191–204.
[10] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, B. Waters, Sequential aggregate signatures and mulitisignatures without random oracles, in: Proceeding of EUROCRYPT'06, LNCS, Springer, Berlin, vol. 4004, 2006, pp. 465–485.
[11] B. Waters, Efficient identity-based encryption without random oracles, in: Proceeding of EUROCRYPT'05, LNCS, Springer, Berlin, vol. 3494, 2005, pp. 114–127.
[12] J.M. Park, E. Chong, H. Siegel, I. Ray. Constructing fair exchange protocols for E-commerce via distributed computation of RSA signatures, in: 22th Annual ACM Symposium on Principles of Distributed Computing, 2003, pp. 172–181.
[13] Y. Dodis, L. Reyzin, Breaking and repairing optimistic fair exchange from PODC2003, in: ACM Workshop on Digital Right Management (DRM), 2003, pp.47–54.
[14] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, in: Proceeding of Asiacrypt 2001, LNCS, Springer-Verlag, Berlin, vol. 2248, 2001, pp. 514–532.
[15] PKCS#10: Certification request syntax standard, RSA Data Security, Inc., 2000.
[16] C. Adams, S. Farrell, T. Kause, T. Monen, Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP), Internet Engineering Task Force RFC 4210, 2005.
[17] J. Schaad, Internet X.509 Public Key Infrastructure Certificate Request Message Format, Internet Engineering Task Force RFC 4211, 2005.
[18] S. Micali. Simple and fast optimistic protocol for fair electronic exchange, in: Proceedings of 22th Annual ACM Symposium on Principles of Distributed Computing (PODC'03), ACM Press, 2003, pp. 12–19.
[19] S. Goldwasser, S. Micali, R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, SIAM Journal on Computing 17 (2) (1988) 281–308.
[20] C.-H. Wang, Y.-S. Kuo, An efficient contract signing protocol using the aggregate signature scheme to protect signers' privacy and promote reliability, ACM SIGOPS Operation System Review 39 (4) (2005) 66–79.