Efficient Fair Exchange with Verifiable Confirmation of Signatures

Liqun Chen

Hewlett-Packard Laboratories,
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK
liqun@hplb.hpl.hp.com

Abstract. We propose a new efficient protocol, which allows a pair of potentially mistrusting parties to exchange digital signatures over the Internet in a fair way, such that after the protocol is running, either each party obtains the other's signature, or neither of them does. The protocol relies on an off-line Trusted Third Party (TTP), which does not take part in the exchange unless any of the parties behaves improperly or other faults occur. Efficiency of the protocol is achieved by using a cryptographic primitive, called *confirmable signatures* (or *designated confirmer signatures* in its original proposal [9]). We recommend using a new efficient confirmable signature scheme in the proposed fair exchange protocol. This scheme combines the family of discrete logarithm (DL) based signature algorithms and a zero-knowledge (ZK) proof on the equality of two DLs. The protocol has a practical level of performance: only a moderate number of communication rounds and ordinary signatures are required. The security of the protocol can be established from that of the underlying signature algorithms and that of the ZK proof used.

1 Introduction

Since electronic commerce is playing a more and more important role in today's world, a related security issue - how to exchange electronic data, particularly digital signatures, between two parties over the Internet in a fair and efficient manner - is becoming of more and more importance. Imagine the following scenario that may happen in, for instance, signing electronic contracts and purchase of electronic goods. Two parties Alice and Bob need to exchange their digital signatures on agreed messages; but neither wants to send her/his signature before obtaining the other's because they do not trust each other. The basic requirement for Alice and Bob on the fairness of exchanging signatures is that either each of them gets the other's signature, or neither of them does.

1.1 The Related Previous Work

How to sort out the fair exchange problem has attracted much research attention. The original idea for the realisation of fair exchange is that two parties "simultaneously" disclose messages by many steps. Two mathematical models for realising simultaneous disclosure of messages have been proposed as follows.

The first is a computational model (e.g., [10,12,15,19,24,30]). In this approach, Alice and Bob exchange digital signatures (or agreed secret messages) piece by piece (e.g. bit by bit), where the correctness of each bit is verifiable. If both of them follow the approach correctly, they will receive the signatures at the end of a successful protocol run. If either of them aborts in the middle of the protocol running, this early stopper will at most obtain one more bit than the other party. This extra bit does not result in a significant advantage in finding the remaining secret bits unexchanged. Obviously, a virtue of this approach is that Alice and Bob can sort out the fair exchange problem without any intervention of a third party. The cost of this virtue is in two respects. (1) This approach is based on the assumption that Alice and Bob have equal computing power. However, this assumption may not be realistic and desirable for them. (2) This approach has a poor performance: many rounds (usually hundreds) of interactions between them are required.

The second type of model is a probabilistic model (e.g., [5,26]). For exchanging signatures on an agreed message, Alice and Bob sign and exchange many signatures on different events. Each event has a small probability binding with the agreed message. In order to increase the probabilities of their commitment to the message, they have to exchange a great number of signatures. This approach removes the requirement on equal computing powers of Alice and Bob. But it needs intervention of a third party in a weak form. In [26], an active third party defines the events by broadcasting a random number each day. In [5], a passive third party is invoked, only when a dispute between Alice and Bob occurs, to arbitrate the dispute according to a simple computation on events. Similarly to the first model, the major drawback of this approach is a poor performance.

In order to reduce the communicational and computational cost of simultaneous disclosure of messages, recent fair exchange research has proposed a variety of interventions of a Trusted Third Party (TTP), which can be on-line or off-line.

In an on-line TTP based approach (e.g. [11,13,17,31]), the TTP, who acts as a mediator between Alice and Bob, checks the validity of every transaction and then forwards correct data to both parties. The major disadvantage in this approach is that the TTP is always involved in the exchange even if both Alice and Bob are honest and no fault occurs, so that it results in another big cost of maintaining availability of the on-line TTP.

A number of off-line TTP based approaches have been proposed to reduce the requirement of TTP availability. In these approaches, the TTP does not take part in normal exchanges, it gets involved only where dishonest parties do not perform properly or other faults occur.

In [1,32], the TTP provide either of the following two services to guarantee the fairness. (i) The TTP is able to undo a transfer of an item, and/or produce a replacement for it. (ii) When a misbehaving party gets the other party's data and refuses to give his/her own one, the TTP will issue affidavits attesting to what happened. Obviously, neither of these TTP services meets the needs of many applications.

Bao, Deng and Mao in [4], which is based on the solution of [20], and Asokan, Shoup and Waidner in [2] separately proposed a novel off-line TTP based approach that uses verifiable public-key encryption to ensure fairness of signature exchanges. In [4], Alice first encrypts her ordinary signature under the TTP's public key and demonstrates the correctness of the encryption to Bob via an interactive ZK proof. Next Bob sends his ordinary signature to Alice, and Alice returns her ordinary one back. If Bob does not receive Alice's signature correctly, he will send Alice's encrypted signature and his own ordinary signature to the TTP. The TTP will do the corresponding decryption and check the validity of both signatures. If all the checks pass, the TTP will transfer these two signatures between Alice and Bob.

The approach of [2] is based on a primitive, called *a homomorphic inverse of a signature* (e.g., a DL for DSS [16] and Schnorr [28] signatures, and an RSA inverse for RSA [27] signatures). Alice and Bob first reduce a "promise" of a signature to the "promise" of a particular homomorphic inverse. Then, they encrypt their promised inverses under the TTP's public key and demonstrate the correctness of the encryption in a non-transferable way to each other. Once demonstrated of encryption, they disclose their promised inverses. If anyone of them (say Bob) does not receive a correct inverse of the other (Alice), he will send the encrypted homomorphic inverse of Alice and a promised inverse of his own to the TTP. The TTP will decrypt and check the validity of both signatures. After all the checks pass, the TTP will send Alice's inverse to Bob and then record Bob's one for Alice's possible requirement.

Although the idea of using verifiable encryption in an off-line TTP based fair exchange is clever, it is difficult to implement this idea in an efficient and generic manner because so far there has not been a generic and efficient construction of publicly verifiable encryption. A well-known solution of publicly verifiable encryption, [29], is based on inefficient "cut and choose" method. Bao recently in [3] proposed a more efficient scheme using Okamoto-Uchiyame trapdoor one-way function [25], which is not a generic construction. How to design an efficient and generic construction of publicly verifiable cryptographic systems is still an interesting and hard open problem.

In order to improve efficiency, [4] recommended the use of a modified Guillou-Quisquater signature algorithm [18] with the ElGamal encryption algorithm [14]. This protocol was recently attacked by Boyd and Foo [6] as the verifier is able to obtain the signer's signature without the help of TTP. For a more closed look at the properties of fair exchange, there is another problem in this protocol that the encrypted signature can not be simulated. Again to improve efficiency, [2] proposed a solution called off-line coupons where each party needs to retrieve the TTP's coupons before starting a fair exchange protocol. Clearly, it will increase the cost for maintaining availability and security of the off-line TTP service.

We finally state, in the author's view, that the previous work has not produced an efficient and widely acceptable approach for fair exchange of digital signatures over the Internet.

1.2 The New Contribution

In this paper, we propose a new approach for fair exchange of digital signatures which uses verifiable confirmation of signatures in place of verifiable encryption of signatures in [2,4]. Both verifiable encryption and verifiable confirmation of signatures can be used to provide off-line TTP based fair exchange. However, the existing constructions of verifiable confirmation are much more efficient and generic than that of verifiable encryption.

The contribution of the paper is organised as two parts. In the first part (the next section) we introduce a new off-line TTP modelled fair exchange protocol which is based on a cryptographic primitive, called *confirmable signatures* (or *designated confirmer signatures* in the original proposal [9]), to guarantee the fairness. In this protocol, the TTP acts as a designated confirmer. There is no restriction for the protocol as to which confirmable signature scheme will be used. In the second part (Section 3), we present a new realisation of confirmable signatures which is constructed by using the family of DL problem based digital signature algorithms. It is one of suitable confirmable signature schemes for the proposed fair exchange protocol.

2 Protocol for Fair Exchange

In this section, we present a fair exchange protocol, which allows a pair of parties to exchange digital signatures with an off-line TTP's intervention in a fair manner.

The protocol involves three players: two exchange parties, Alice (A) and Bob (B), plus one off-line TTP, Colin (C), who acts as a designated confirmer. Each of these players has a secret and public key pair denoted by S_X and P_X respectively (where $X \in \{A, B, C\}$), which is used for digital signature and verification. Suppose that there exists a secure binding between each player's identity and the corresponding public key. Such a binding may be in the form of a public key certificate that was issued by a certification authority. Suppose further that the communication channels between these three players are protected to guarantee integrity and confidentiality (if required).

2.1 Model, Notation and Explanation

We denote $Sig_X(m)$ $(X \in \{A, B, C\})$ as an ordinary signature on a message m signed using S_X, which can be universally verified using P_X. We denote $CSig_Y(m)$ $(Y \in \{A, B\})$ as a confirmable signature on m signed using S_Y. We denote $Sta\_of\_CSig_Y(m)$ as a validity statement of $CSig_Y(m)$, for instance, in the recommended confirmable signature scheme, as described in Section 3, $Sta\_of\_CSig_Y(m)$ is the equality of two DLs. It can be proved by using either S_Y or S_C.

A confirmable signature bound with its statement is universally verifiable and is as valid as an ordinary signature. Thus,

$$\{CSig_Y(m), Sta\_of\_CSig_Y(m)\} \equiv Sig_Y(m).$$

Without the statement, the binding between Y and $CSig_Y(m)$ cannot be claimed.

In order to prove $Sta\_of\_CSig_Y(m)$ from one party (as signer named Y) to the other (as verifier) in a non-transferable way, we make use of an interactive ZK proof between the two parties, named $Conf_Y$, which, on common inputs of m, P_Y, P_C, a string $Claim$ and on secret input of S_Y, outputs "true" or "false". That is,

$$Conf_Y(Sta\_of\_CSig_Y(m)|m, P_Y, P_C, Claim) = \text{ true } \mathit{or} \text{ false} .$$

If output is "true", it is proved that $Claim$ is $CSig_Y(m)$; if output is "false", it is proved that $Claim$ is not $CSig_Y(m)$.

In a confirmable signature scheme, the confirmer can make either a non-transferable confirmation or a transferable confirmation of $Sta\_of\_CSig_Y(m)$. For the purpose of the proposed fair exchange protocol, we only need the transferable one. In the protocol of the next subsection, an ordinary signature on $Sta\_of\_CSig_Y(m)$ signed using S_C will be used for the transferable confirmation of $CSig_Y(m)$. A confirmable signature suitable for the proposed fair exchange protocol has the following three properties.

– *Invisibility.* $CSig_Y(m)$ can be simulated by using a polynomial-time algorithm.
– *Unforgeability.* No polynomial-time algorithm can forge such a signature that can be confirmed to have a validity statement.
– *Undeniability.* Signer of $CSig_Y(m)$ cannot deny having issued this confirmable signature if $CSig_Y(m)$ is bound to $Sta\_of\_CSig_Y(m)$.

2.2 The Protocol

Suppose that Alice and Bob have agreed on a message (such as a contract) M. The protocol for fair exchange of signatures on M between Alice and Bob proceeds as follows. Without loss of generality, we assume that Alice is the protocol initiator.

Protocol FE

1. Alice computes her confirmable signature on M, $CSig_A(M)$, and sends it to Bob.
2. Alice and Bob run an interactive ZK protocol $Conf_A$, e.g. as described in Section 3.2, proving $Sta\_of\_CSig_A(M)$. If

$$Conf_A(Sta\_of\_CSig_A(M)|M, P_A, P_C, CSig_A(M)) = \text{false},$$

the proof is rejected and the protocol stops. If

$$Conf_A(Sta\_of\_CSig_A(M)|M, P_A, P_C, CSig_A(M)) = \text{true},$$

Bob computes and sends Alice his ordinary signature $Sig_B(M)$.

3. After receiving $Sig_B(M)$, Alice verifies whether it is a valid signature. If not, Alice halts; if it is valid, Alice accepts the signature, and then computes and sends Bob her ordinary signature $Sig_A(M)$.
4. Upon the receipt of $Sig_A(M)$, Bob verifies whether it is a valid one. If it is, Bob accepts the signature, and the protocol completes.
5. If Bob receives an invalid signature or nothing during a designed time period, Bob sends both $Sig_B(M)$ and $CSig_A(M)$ to Colin. Colin first checks whether $Sig_B(M)$ is Bob's valid signature on M, and secondly checks, by using his secret key S_C, whether $CSig_A(M)$ is Alice's valid confirmable signature on M. If either of these two checks does not pass, Colin does not provide a confirmation service. If both of the checks pass, Colin computes and sends Bob his signature on $Sta\_of\_CSig_A(M)$, and in the meantime, he forwards $Sig_B(M)$ to Alice.

2.3 Analysis of Protocol FE

We now consider the behavior of Alice and Bob. If both of them follow the protocol properly, it is easy to see that Alice and Bob will obtain each other's signatures without any involvement of Colin.

If Bob performs improperly, Bob may send Alice either an incorrect $Sig_B(M)$ or nothing in Item 2. In both of the cases, Alice does not send $Sig_A(M)$ to Bob in Item 3, and Bob has to ask Colin for confirmation of $CSig_A(M)$ if he wants Alice's signature. Based on Item 5, Colin makes such a confirmation for Bob only if Bob gives a valid $Sig_B(M)$, which will be forwarded to Alice.

If Alice does not follow the protocol properly, either of the following two situations may happen. (i) Alice sends Bob a non-confirmable signature in Item 1. In this case, she cannot demonstrate

$$Conf_A(Sta\_of\_CSig_A(M)|M, P_A, P_C, CSig_A(M)) = \text{true}$$

in Item 2 to Bob. (ii) She sends an invalid $Sig_A(M)$ or nothing to Bob in Item 3. In this case, Bob can obtain the confirmation of $CSig_A(M)$ from Colin.

As mentioned earlier, the fairness of exchanging signatures between two parties means that either each party gets the other's signature, or neither party does. In terms of the definition of fairness, we can conclude that neither Alice nor Bob can gain any benefit by performing improperly, so that Protocol FE can achieve fair exchange between Alice and Bob.

However, in Protocol FE, after accepting $CSig_A(M)$, Bob has the advantage of choosing stop or continuation. If it makes Alice feel unfairly treated, the protocol can be slightly changed. Following Alice having proved her confirmable signature to Bob, Bob proves his confirmable signature to Alice. Then Alice releases her ordinary signature and Bob releases his ordinary one. Both Alice and Bob can ask Colin for a confirmation service. As in Protocol FE, Colin always makes confirmation of a signature for one party and forwards an ordinary signature to another party. Before Colin provides the confirmation, Alice is able to ask Colin for invoking *abort* (i.e. by an *abort* sub-protocol as in [2]). Here Colin needs to maintain an extra record about "abort" and "confirmed".

A normal procedure of the protocol, where there is non-intervention of Colin, includes only five communication rounds: three rounds for non-transferable confirmation of $CSig_A(M)$ (Item 1 and 2 by using the recommended scheme of Section 3); and two rounds for exchange of $Sig_B(M)$ and $Sig_A(M)$ (Item 2 and 3).

Note that both parties' identifiers must be indicated in $CSig_A(M)$, which could be a part of the message M. Otherwise, Colin can know only that Alice is one of the exchange parties, and he cannot know who is another. In this case, an intruder (who may be Bob's colluder), given $CSig_A(M)$, can obtain the confirmation of $CSig_A(M)$ from Colin by providing his own signature on M. After the protocol is running, Alice will get an unexpected intruder's signature in place of Bob's one, which is not what she wants.

Protocol FE can be modified to meet the following different requirements of message styles.

Assume that Alice and Bob want to keep M confidential to Colin. They can use a one-way hash function, $h()$, and replace M with $h(M)$ in Protocol FE.

Assume that Alice and Bob want to sign two messages M_A and M_B, where both Alice's signature on M_A and Bob's signature on M_B can be universally verified. In this case Colin should be able to check if he is making a confirmation service for a real agreement between Alice and Bob. For this purpose, each file signed by one party must include an indicator of the file signed by the other. For example, as used in [4], Alice signs $M_A||h(M_B)$ and Bob signs $M_B||h(M_A)$, where $||$ denotes concatenation. Otherwise (e.g., Alice and Bob directly sign M_A and M_B respectively for such M_A and M_B that have no explicit relationship explanation), Bob may send Colin $CSig_A(M_A)$ with $Sig_B(M'_B)$ for the confirmation service. Finally Bob get a real confirmed signature of Alice, who will get only a signature on a meaningless message M'_B. Furthermore, if it is required that both M_A and M_B are confidential to Colin, Alice and Bob can have extra secret and public key pairs for encryption and decryption. In this case, M_A will be replaced by encrypted M_A under Bob's encryption public key and M_B will be replaced by encrypted M_B under Alice's encryption public key as well.

If it is required with certain applications, the protocol can be modified by including multiple confirmers instead of a single one.

3 A Confirmable Signature Scheme

The concept and the first realisation of confirmable signatures (or called designated confirmer signatures in [9]) was proposed by Chaum, [9], where he presented a realisation on the RSA signature algorithm. Following Chaum's idea, Okamoto proposed a more generic confirmable signature scheme [23]. However, that scheme was later attacked by Michels and Stadler [22] as the confirmer can forge signatures.

Michels and Stadler also proposed their own confirmable signature scheme based on a primitive called *the confirmer commitment scheme*. The scheme places a message in the position of a committal (i.e., commit to a message), and the confirmer is able to prove whether or not a given commitment contains a certain message. Using this scheme, two classes of ordinary digital signatures can be transformed into related confirmable signatures. The first class consists of the signatures that are based on proofs of some particular style of knowledge. Both the Schnorr signature and the Fiat-Shamir signature can be used in this way. The second class consists of the signatures that have the property of existential forgeability. For this kind of signature, an attacker can compute a universally verifiable message-signature pair without further constraint on the message. The RSA signature and the ElGamal signature are two good examples of this class.

This section presents a new confirmable signature scheme. In this scheme, a *confirmable signature* contains a *validity statement*, which is the equality of two DLs, and which can efficiently be proved either via running a ZK protocol, or via verifying an ordinary digital signature signed by the confirmer. Any DL based signature algorithm and any ZK protocol for proving the equality or the inequality of two DLs can be used in this scheme. The security of the scheme can be established from that of the underlying signature schemes and that of the ZK protocol used. In terms of efficiency the scheme is similar to the most efficient one of [22], which is based on the Schnorr signature scheme.

3.1 System Setup

Let p be a prime, and q be another prime which divides $p-1$. Let $G =< g >$ be a subgroup of $\mathbb{Z}_p^*$ of order q, in which computing DLs is infeasible. Let $h()$ denote a one-way hash function, and $a \in_R N$ denote to choose element a from the set N at random according to the uniform distribution.

A confirmable signature scheme involves three players: a Signer (say Alice), a Verifier (say Bob) and a designated Confirmer (say Colin). In the proposed fair exchange protocol described in Section 2.2, both the exchange parties, Alice and Bob, can be such a signer and verifier.

Alice, as a signer, has a secret and public key pair, denoted by $(S_A,\ P_A)$; and Colin has another secret and public key pair, denoted by $(S_C,\ P_C)$. These two key pairs can be generated as follows. Alice chooses $x \in_R \mathbb{Z}_q^*$ as S_A, and computes $P_A = (g, y)$ where $y = g^x \bmod p$. Colin chooses $w \in_R \mathbb{Z}_q^*$ as S_C, and computes $P_C = (g, z)$ where $z = g^w \bmod p$.

A confirmable signature scheme consists of the following two procedures: *signature issuance* and *signature confirmation.*

3.2 Signature Issuance

A signature issuance procedure runs between Alice and Bob. It consists of (i) Alice generating $CSig_A(m)$; and (ii) Alice demonstrating to Bob that $CSig_A(m)$ is a confirmable signature on a message m.

To generate $CSig_A(m)$, Alice chooses $u \in_R \mathbb{Z}_q^*$, computes $\tilde{y} = y^u \bmod p$ and $\hat{y} = z^{xu} \bmod p$. Next she generates a signature on a message m signed using u and ux as private keys. The basic idea of this signature is to make a transferable proof that: (i)someone knows how to express $\tilde{y}$ as a power of y and how to express $\hat{y}$ as a power of z; and (ii)this person has signed m using the DLs of both $\tilde{y}$ to the base y and $\hat{y}$ to the base z as private keys. Any existing secure signature algorithm, based on the DL problem, can be used to make this signature. The following is an example using the Schnorr signature [28],

$$k_1, k_2 \in_R \mathbb{Z}_q^*,\ r_1 = y^{k_1} \bmod p,\ r_2 = z^{k_2} \bmod p,$$
$$c = h(m, r_1, r_2),\ s_1 = k_1 - uc \bmod q,\ s_2 = k_2 - uxc \bmod q,$$
$$CSig_A(m) = (c, s_1, s_2).$$

The signature verification is to check if

$$c = h(m, y^{s_1}\tilde{y}^c, z^{s_2}\hat{y}^c)$$

holds. This signature is universally verifiable. However, because anyone can construct (c, s_1, s_2) by randomly choosing $\tilde{y}$ as a power of y and $\hat{y}$ as a power of z, without further proof, no one can see who is the issuer of the signature.

Proposition 1. *The above $CSig_A(m)$ is a confirmable signature with a validity statement $Sta\_of\_CSig_A(m)$, $\log_{\tilde{y}} \hat{y} = \log_g z \pmod{q}$.*

Proof. On the assumption that a random oracle model holds, the proposition is proved if the following three assertions can be proved: (i) given that $\log_{\tilde{y}} \hat{y} = \log_g z$, it can be proved that the issuer of $CSig_A(m)$ must be Alice; (ii) without the verification of $\log_{\tilde{y}} \hat{y} = \log_g z$, it cannot be claimed that the issuer of $CSig_A(m)$ is Alice; (iii) $\log_{\tilde{y}} \hat{y} = \log_g z$ can independently be verified by Alice and Colin.

By verifying the correctness of the digital signature, it can be proved that the issuer of (c, s_1, s_2) must know both $\log_y \tilde{y}$, denoted by u, and $\log_z \hat{y}$, denoted by v. The value $\log_g \hat{y}$, denoted by t, must be the product of three values: $\log_g y = x$, $\log_y \tilde{y} = u$, and $\log_{\tilde{y}} \hat{y}$. If $\log_{\tilde{y}} \hat{y}$ is $\log_g z = w$, then $t = xuw \bmod q$ and $v = xu \bmod q$. The person who knows v and u must know x. Since x is known only to Alice, the issuer of $CSig_A(m)$ must be Alice. The first assertion holds.

Without verifying $\log_{\tilde{y}} \hat{y} = \log_g z$, no one can claim that $CSig_A(m)$ was signed by Alice, since anyone knowing y and z is able to generate the signature (see the proof of Proposition 3 of Section 3.4). The second assertion holds.

Colin is able to prove $\log_{\tilde{y}} \hat{y} = \log_g z$, because $\log_g z$ is S_C. Alice can prove the knowledge of u and x, and hence she can demonstrate this statement (see the next subsection). The third assertion holds.

According to the definition of a confirmable signature and the above three assertions, it has been proved that $Sta\_of\_CSig_A(m)$ is $\log_{\tilde{y}} \hat{y} = \log_g z$ so that $CSig_A(m)$ is a confirmable signature. The proposition holds. □

The following interactive protocol, denoted by $Conf_A$, is used for Alice to demonstrate $Sta\_of\_CSig_A(m)$ to Bob.

Protocol $Conf_A$

Suppose that before the protocol starts, both Alice and Bob have $\tilde{y}$, $\hat{y}$ and $CSig_A(m)$.

1. Alice computes $\ddot{y} = z^x \bmod p$ and sends it to Bob.
2. Alice and Bob run an interactive ZK protocol proving
$$\log_g y = \log_z \ddot{y} \ (\text{mod} q).$$
3. Alice and Bob run an interactive ZK protocol proving
$$\log_y \tilde{y} = \log_{\ddot{y}} \hat{y} \ (\text{mod} q).$$
4. If both ZK proofs are accepted, Bob is convinced that $CSig_A(m)$ is a confirmable signature. Otherwise, the proof is rejected.

Several efficient ZK protocols for proving equality in DLs, e.g. [7,8], can be used for the proof.

Proposition 2. *Upon acceptance of $Conf_A$, Alice proves $\log_g z = \log_{\tilde{y}} \hat{y}$ (mod q) to Bob.*

Proof. Suppose $\tilde{y} = y^u \bmod p = g^{xu} \bmod p$ where $u \in \mathbb{Z}_q^*$. From the first ZK proof, Bob is convinced of $\ddot{y} = g^{xw} \bmod p$. From the second ZK proof, Bob is convinced of $\hat{y} = \ddot{y}^u = g^{xwu} (\text{mod} p)$. So the proposition follows, i.e. $\hat{y} = \tilde{y}^w \bmod p$. □

3.3 Signature Confirmation

In order to let Bob know whether or not a given statement is $Sta\_of\_CSig_A(m)$. Colin needs to demonstrate to him either
$$\log_g z = \log_{\tilde{y}} \hat{y} \ (\text{mod} q), \ \text{ or } \ \log_g z \neq \log_{\tilde{y}} \hat{y} \ (\text{mod} q).$$
A number of efficient protocols for a ZK proof on the equality or inequality of two DLs, e.g. [21], can be used for the proof. Colin can either run an interactive ZK protocol with Bob to make a non-transferable confirmation, or sign $Sta\_of\_CSig_A(m)$ for Bob to make a transferable confirmation. For the purpose of our fair exchange protocol, we need a transferable confirmation. A number of existing efficient interactive protocols for ZK proof of the equality or inequality of two DLs can be turned into non-interactive protocols, which can be used. The following is one example based on the Schnorr signature [28]. Colin signs $(g, z, \tilde{y}, \hat{y})$ using $S_C = w$ by two ordinary signatures.

The first signature makes a transferable proof in that there exist two values r_1 and r_2 satisfying $r_1 = g^k \bmod p$ and $r_2 = \tilde{y}^k \bmod p$ where $k \in \mathbb{Z}_q^*$. The signature is (c, s') generated as follows.
$$k' \in_R \mathbb{Z}_q^*, \ r_1' = g^{k'} \bmod p, \ r_2' = \tilde{y}^{k'} \bmod p,$$
$$c = h(r_1', r_2'), \ s' = k' - kc \bmod q.$$
The signature verification is to check if
$$c = h(g^{s'} r_1^c, \tilde{y}^{s'} r_2^c)$$

holds. If it does not hold, Bob can claim that Colin did not send a proper signature to him.

Based on acceptance of the first signature, the second signature provides a transferable proof on either $\log_g z = \log_{\tilde{y}} \hat{y}$ or $\log_g z \neq \log_{\tilde{y}} \hat{y}$. The resulting signature is (r_1, r_2, s), where

$$s = k + wh(r_1, r_2) \bmod q.$$

The signature verification is to check if

$$g^s = r_1 z^{h(r_1,r_2)} (\text{mod} p), \quad \tilde{y}^s = r_2 \hat{y}^{h(r_1,r_2)} (\text{mod} p)$$

holds. If the first equality does not hold, Bob can claim that Colin did not send a proper signature to him. Otherwise, Bob accepts the conviction of the signature. In this case, if the second equality holds, Bob accepts $\log_g z = \log_{\tilde{y}} \hat{y}$, and then further accepts that the related signature, $CSig_A(m)$, is a confirmable one. If the second equality does not hold, Bob accepts $\log_g z \neq \log_{\tilde{y}} \hat{y}$, and further accepts that the related signature is not a confirmable one.

Note that before generating the above two signatures, Colin may check if $\hat{y} = \tilde{y}^w$ mod$p$ holds firstly. If it does hold he can simply make a transferable proof on $\log_g z = \log_{\tilde{y}} \hat{y}$ by using the second signature only. With this signature, anybody is able to verify the correctness of $Sta\_of\_CSig_A(m)$. Hence $CSig_A(m)$ is universally verifiable.

3.4 Security of the Scheme

The confirmable signature scheme, specified above, allows the players of the scheme free to choose any DL based signature algorithms and to choose any efficient protocols for ZK proof on the equality or the inequality of two DLs. As long as the security property of those algorithms and protocols have been proved, i.e., (i) the verification of a digital signature is complete and sound; (ii) the error probability of an acceptance for a ZK protocol is negligible; (iii) they guarantee not to reveal useful information about x and w, the following three security properties hold under this scheme.

Proposition 3. *A confirmable signature, $CSig_A(m)$, can be simulated.*

Proof. A simulator, who knows g, y, z, q, p and m, is always able to generate a triple (c', s_1', s_2') in the following way. He/she simply chooses $u' \in_R \mathbb{Z}_q^*$ and $v' \in_R \mathbb{Z}_q^*$, computes $\tilde{y}' = y^{u'} \bmod p$ and $\hat{y}' = z^{v'} \bmod p$, and then signs m using u' and v' as private keys, by the same approach described in Section 3.2, to obtain (c', s_1', s_2'). For two fixed public keys y and z, and any message m, let $\mathcal{A}$ be any polynomial-time algorithm which, on input of a signature pair (m, σ), outputs whether or not (m, σ) is valid with respect to y and z. The value

$$Pr\{\mathcal{A}(m, (c', s_1', s_2')) = valid\} - Pr\{\mathcal{A}(m, (c, s_1, s_2)) = valid\}$$

is negligible.

Note that there is no binding between z^x and (c, s_1, s_2), hence z^x does not reveal any useful information for distinguishing a real $CSig_A(m)$ and a simulated one. Furthermore, the set $(g, g^x, z^x, \tilde{y}, \hat{y})$ is indistinguishable from the set $(g, g^x, z^x, \tilde{y}', \hat{y}')$, otherwise the Decision Diffie-Hellman assumption is not valid in the random oracle model. So the proposition holds. □

Proposition 4. *A confirmable signature, $CSig_A(m)$, is unforgeable, i.e., under the assumption on that it is computationally infeasible to compute DL in G, there is no polynomial-time algorithm which, on input of y, z, w, and any value $m' \in \{0,1\}^*$, outputs $CSig_A(m')$, with respect to $\tilde{y}'$ and $\hat{y}'$ satisfying $\log_g z = \log_{\tilde{y}'} \hat{y}' (\bmod q)$.*

Proof. It has been proved in Proposition 1 that, if $\log_g z = \log_{\tilde{y}'} \hat{y}'$, the value $\log_z \hat{y}'$ must be equal to $\log_g y * \log_y \tilde{y}'$. If there is a polynomial-time algorithm $\mathcal{A}$ which, on input of y, z, w and m', outputs $CSig_A(m')$ with respect to $\tilde{y}'$ and $\hat{y}'$ satisfying $\log_g z = \log_{\tilde{y}'} \hat{y}'$, $\mathcal{A}$ must be able to obtain $\log_g y$. This contradicts the assumption. Hence the proposition holds. □

This proposition proves that no one, including the confirmer Colin, is able to forge such a confirmable signature, $CSig_A(m)$.

Proposition 5. *A confirmable signature, $CSig_A(m)$, is undeniable.*

Proof. It is impossible for Alice to find any $\tilde{y}$, $\ddot{y}$ and $\hat{y} \in \mathbb{Z}_p^*$ satisfying $\log_g y = \log_z \ddot{y}$, $\log_y \tilde{y} = \log_{\ddot{y}} \hat{y}$ and $\log_g z \neq \log_{\tilde{y}} \hat{y}$. As has been proved in Proposition 1, given that $\log_g z = \log_{\tilde{y}} \hat{y}$, only the person knowing $\log_g y$ is able to make $CSig_A(m)$, so that Alice cannot deny having issued this confirmable signature. Therefore the proposition holds. □

4 Conclusions

Previous work on fair exchange of digital signatures did not produce an efficient approach that would be widely acceptable in electronic commerce. This paper has proposed a new efficient protocol for fair exchange of digital signatures between two potentially mistrusting parties. In the protocol, a TTP, acting as a designated confirmer, is needed only when one of the exchange parties does not follow the protocol properly or other fault occurs. This protocol has a practical level of performance: only a moderate number of communication rounds (e.g. 5 rounds for a normal procedure) and ordinary signatures (e.g. two Schnorr signatures for a confirmable signature and one Schnorr signature for a normal confirmation service) are required. It will be suitable for many electronic commerce applications over the Internet, such as contract signing and electronic purchase. The fairness property of the protocol is based on verifiable confirmation of digital signatures. The paper has presented an efficient and generic confirmable signature scheme recommended being used in the proposed fair exchange protocol.

Acknowledgements

I would like to thank Wenbo Mao, Markus Stadler, Markus Michels, Graeme Proudler and Siani Pearson for invaluable discussions and/or comments. I am also grateful to anonymous referees for helpful comments.

References

1. Asokan, A., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, Zurich, Switzerland, (1997) 6 - 17
2. Asokan, A., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In *Advances in Cryptology - EUROCRYPT '98, LNCS 1403*, Springer-Verlag, (1998) 591–606
3. Bao, F.: An efficient verifiable encryption scheme for encryption of discrete logarithms. To appear in CARDIS '98
4. Bao, F., Deng, R., Mao, W.: Efficient and practical fair exchange protocols with off-line TTP. In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, Oakland, California, IEEE Computer Press, (1998) 77-85
5. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.: A fair protocol for signing contracts. IEEE Transactions on Information Theory. 36(1) (1990) 40-46
6. Boyd, C., Foo, E.: Off-line fair payment protocols using convertible signatures. ASIACRYPT '98 (these proceedings)
7. Boyar, J., Chaum, D., Damgård, I., Pedersen, T.: Convertible undeniable signatures. In *Advances in Cryptology - CRYPTO '90, LNCS 537*, Springer-Verlag, (1991) 189–205
8. Chaum, D.: Zero-knowledge undeniable signatures. In *Advances in Cryptology - EUROCRYPT '90, LNCS 473*, Springer-Verlag, (1991) 458–464
9. Chaum, D.: Designated confirmer signatures. In *Advances in Cryptology - EUROCRYPT '94, LNCS 950*, Springer-Verlag, (1994) 86-91
10. Cleve, R.: Controlled gradual disclosure schemes for random bits and their applications. In *Advances in Cryptology - CRYPTO '89, LNCS 435*, pages. Springer-Verlag, (1990) 572-588
11. Cox, B., Tygar, J., Sirbu, M.: NetBill security and transaction protocol. In *Proceedings of First USENIX Workshop on Electronic Commerce*, (1995) 77-88
12. Damgård, I.: Practical and provably secure release of a secret and exchange of signatures. In *Advances in Cryptology - EUROCRYPT '93, LNCS 765*, Springer-Verlag, (1994) 201-207
13. Deng, R., Gong, L., Lazar, A., Wang, W.: Practical protocol for certified electronic mail. Journal of Network and Systems Management. **4**(3) (1996) 279-297
14. ElGamal, T.: A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory. **31**(4) (1985) 469-472
15. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. CACM. 28(6) (1985) 637-647
16. U.S. Department of Commerce/National Institute of Standards and Technology, *Digital Signature Standard*. Federal Information Processing Standard Publication (FIPS PUB) 186, May 1994.
17. Franklin, M., Reiter, M.: Fair exchange with a semi-trusted third party. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, Zurich, Switzerland, (1997) 1-5

18. Guillou, L., Quisquater, J.: A paradoxical identity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology -CRYPTO '88, LNCS 403*, Springer-Verlag, (1990) 216-231
19. Luby, M., Micali, S., Rackoff, C.: How to simultaneously exchange a secret bit by flipping symmetricall-based coin. In *Proceedings of the 24th IEEE Symposium on the Foundations of Computer Science (FOCS)*, (1983) 11-22
20. Mao, W.: Verifiable escrowed Signature. In *Proceedings of Second Australasian Conference on Information Security and Privacy, LNCS 1270*, Springer-Verlag, (1997) 240-248
21. Michels, M., Stadler, M.: Efficient convertible undeniable signature schemes. In the Proceedings of the 4th Annual Workshop on Selected Areas in Cryptography (SAC '97), (1997)
22. Michels, M., Stadler, M.: Generic constructions for secure and efficient confirmer signatures. In *Advances in Cryptology - EUROCRYPT '98, LNCS 1403*, Springer-Verlag, Berlin, (1998) 406–421
23. Okamoto, T.: Designated confirmer signatures and public-key encryption are equivalent. In *Advances in Cryptology - CRYPTO '94, LNCS 839*, Springer-Verlag, (1994) 61-74
24. Okamoto, T., Ohta, K.: How to simultaneously exchange secrets by general assumption. In *Proceedings of 2nd ACM Conference on Computer and Communications Security*, (1994) 184-192
25. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology - EUROCRYPT '98, LNCS 1403*, Springer-Verlag, Berlin, (1998) 308–318
26. Rabin, M., Transaction protection by beacons. Aiken Computation Lab. Harverd University Cambridge, MA, Tech. Rep. (1981) 29-81
27. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM. **21** (1978) 294–299
28. Schnorr, C.: Efficient identification and signatures for smart-cards. In *Advances in Cryptology - EUROCRYPT '89, LNCS 435*, Springer-Verlag, (1990) 239–252
29. Stadler, M.: Publicly verifiable secret sharing. In *Advances in Cryptology - EUROCRYPT '96, LNCS 1070*, Springer-Verlag, (1996) 190-199
30. Tedric, T.: Fair exchange of secrets. In *Advances in Cryptology - CRYPTO '84, LNCS 196*, Springer-Verlag, (1985) 434-438
31. Zhou, J., Gollmann, D.: A fair non-repudiation protocol. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, Oakland, California, IEEE Computer Press, (1996) 55-61
32. Zhou, J., Gollmann, D.: An efficient non-repudiation protocol. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, Rockport, Massachusetts, (1997) 126-132