Colluding Attacks to a Payment Protocol and Two Signature Exchange Schemes

Feng Bao

Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
Email: baofeng@i2r.a-star.edu.sg

Abstract. An untraceable fair network payment protocol is proposed by Wang in Asiacrypt'03, which employs the existent techniques of the offline untraceable cash and a new technique called restrictive confirmation signature scheme (RCSS). It is claimed that the fair payment protocol has both the fairness such that the buyer obtains the digital goods if and only if the merchant gains the digital cash and the untraceability and unlinkability such that no one can tell who is the original owner of the money. In this paper we show that the fairness is breached under a simple colluding attack, by which a dishonest merchant can obtain the digital money without the buyer obtaining the goods. We also apply the attack to some of the schemes of fair exchange of digital signatures proposed by Ateniese in ACM CCS'99. Our study shows that two of them are subjected to the attack. A countermeasure against the attack is proposed for the fair exchange of digital signatures. However, we are unable to fix the fair payment protocol if the untraceability and unlinkability are the required features.

1 Introduction

In Asiacrypt 2003, Wang proposed an untraceable fair network payment protocol, which is claimed to have untraceability, unlinkability and fairness [25]. The protocol is for online purchasing of digital goods with digital money. A buyer withdraws untraceable and unlinkable digital cash from a bank and buys some digital goods from an online merchant with the digital cash. The fairness is a feature that prevents either the buyer or the merchant from taking the advantage of the other. It guarantees that the buyer can obtain the goods if and only if the merchant gains the money. The protocol combines the techniques of the untraceable offline e-coin ([8], [9]) and a new primitive called restrictive confirmation signature scheme (RCSS). By RCSS, a signature confirmed by a designated confirmer can only convince some specified verifiers. In this paper we present a colluding attack where a dishonest merchant can breach the fairness such that he can obtain the money without the buyer obtaining the goods. The problem with the protocol is that the money is the untraceable and unlinkable e-coin, which has no link with the buyer's ID and hence can be separated from the RCSS-signed order agreement. That is the vulnerable point our attack exploits. The attack does not work if the digital money is internally linked to the buyer's ID. But the untraceability and unlinkability would be lost in that case.

We can also apply a similar colluding attack to the schemes of fair exchange of digital signature proposed by Ateniese in [4]. There are six schemes in [4] for fair exchange of 1) RSA signatures; 2) Gennaro-Halevi-Rabin signatures; 3) Cramer-Shoup signatures; 4) Guillou-Quisquater signatures; 5) Schnorr (or Poupard-Stern) signatures; and 6) ElGamal (or DSA) signatures, respectively. We show that 5) and 6) are subject to the attack, while 1), 2), 3) and 4) are not. The schemes 1), 2) and 3) have the same principle as Boyd-Foo scheme in [7], where TTP performs different converting functions for different users. The colluding attack does not apply to such schemes. The scheme 4) is an improved version of Bao-Deng-Mao scheme in [5]. A flaw of [5], which was first pointed out in [7], is removed in 4). The reason why 5) and 6) are subject to the attack is that Schnorr signatures and ElGamal signatures have a special feature, which Guillou-Quisquater signatures do not have. The feature does not affect the security requirements of digital signature. However, it is the key point in determining whether the attack works. The feature will be discussed later in this paper. The attack works only if the system allows new users to register at any time.

The rest of the paper is organized as follows. In Section 2, we describe the untraceable fair payment protocol proposed in Asiacrypt'03. In Section 3, we present a colluding attack breaching the fairness of the protocol and explain why the attack works. In Section 4, we describe the two schemes of fair exchange of digital signatures proposed in ACM CCS'99. In Section 5, we discuss the special feature of digital signatures that we exploit and present the colluding attack to the two schemes. We also give the countermeasure against the attack. Section 6 concludes the paper.

2 Untraceable Fair Network Payment Protocol

In this section we describe the untraceable fair network payment protocol proposed in [25]. For simplicity, we skip the details of the building-block RCSS (restrictive confirmation signature scheme) and put it in the Appendix A for interested readers. We also simplify the description of the protocol by assuming that the payment is in one e-coin instead of n e-coins as in [25].

Entities

$\mathcal{U}$ — the buyer, who buys soft goods from the merchant.
$\mathcal{M}$ — the merchant, who sells the soft goods to the buyer.
$\mathcal{B}$ — the bank, who issues e-coins to the buyers.
TTP — the trusted third party, who resolves dispute in payment protocol.

System Parameters and Cryptographic Keys

p, q, g — p, q large primes, $q|p-1$, g a generator of the subgroup G_q of order q of $\mathbb{Z}_p^*$.

$y_{\mathcal{B}}, x_{\mathcal{B}}$ — $\mathcal{B}$'s public and private keys, $y_{\mathcal{B}} = g^{x_{\mathcal{B}}} \mod p$.
y_{TTP}, x_{TTP} — TTP's public and private keys, $y_{TTP} = g^{x_{TTP}} \mod p$.
g_1, g_2 — two elements of G_q published by $\mathcal{B}$, for e-coin scheme.

Two Building-Block Techniques

RCSS — restrictive confirmation signature scheme. In RCSS, a signature signed by a signer S can be confirmed by a confirmer C, and C can convince only some specified verifiers $\mathcal{G}$ that the signature is valid and truly signed by S. RCSS is the main technique designed for the fair payment protocol in [25]. It is denoted by $Sign_{RCSS}(S, C, \mathcal{G}, m)$.

BP — interactive bi-proof of equality. In BP either $\log_\alpha Y = \log_\beta Z$ or $\log_\alpha Y \neq \log_\beta Z$ is proved. The proof system is denoted by $BP(\alpha, Y, \beta, Z)$ in [25], where no detailed description of BP is presented but the reader is referred to [16] and [19].

The untraceable fair network payment protocol consists of five processes, namely account opening, withdrawal, payment, dispute and deposit. The details are as follows.

Account Opening

The buyer $\mathcal{U}$ randomly selects $u_1 \in \mathbb{Z}_q$ and transmits $I = {g_1}^{u_1} \mod p$ to $\mathcal{B}$ if $Ig_2 \neq 1$. The identifier I used to uniquely identify $\mathcal{U}$ can be regarded as the account number of $\mathcal{U}$. Then $\mathcal{B}$ publishes ${g_1}^{x_{\mathcal{B}}}$ (we omit $\mod p$ here) and ${g_2}^{x_{\mathcal{B}}}$ so that $\mathcal{U}$ can compute $z = (Ig_2)^{x_{\mathcal{B}}} = ({g_1}^{x_{\mathcal{B}}})^{u_1} {g_2}^{x_{\mathcal{B}}}$ for himself.

Withdrawal

The buyer $\mathcal{U}$ performs the following protocol to withdraw an e-coin from the bank:

1. $\mathcal{B}$ randomly selects $w \in \mathbb{Z}_q^*$ and sends $e_1 = g^w$ and $e_2 = (Ig_2)^w$ to $\mathcal{U}$.
2. $\mathcal{U}$ randomly selects $s, x_1, x_2 \in \mathbb{Z}_q^*$ and computes $A = (Ig_2)^s$, $B = {g_1}^{x_1} {g_2}^{x_2}$ and $z' = z^s$. $\mathcal{U}$ also randomly selects $u, v, t_c \in \mathbb{Z}_q^*$ and computes $e_1' = {e_1}^u g^v, e_2' = {e_2}^{su} A^v$ and $(a_c, b_c) = (g^{t_c}, {y_{TTP}}^{t_c})$. Then $\mathcal{U}$ sends $c = c'/u \mod q$ to $\mathcal{B}$, where $c' = \mathcal{H}(A, B, z', e_1', e_2', b_c) + a_c \mod q$, where $\mathcal{H}$ is a collision-free hash function to $\mathbb{Z}_q^*$. Note that (a_c, b_c) is a pair of confirmation parameters.
3. $\mathcal{B}$ sends $r = cx_{\mathcal{B}} + w \mod q$ to $\mathcal{U}$.
4. $\mathcal{U}$ verifies whether $g^r = {y_{TTP}}^c e_1$ and $(Ig_2)^r = z^c e_2$. If the verification holds, $\mathcal{U}$ accepts and computes $r' = ru + v \mod q$. Note that $< A, B, (z', e_1', e_2', r', a_c, b_c) >$ represents a pseudo e-coin.

Payment

The Buyer $\mathcal{U}$ and the merchant $\mathcal{M}$ exchange the e-coin and the soft goods in this protocol. In the original protocol multiple e-coins are traded for the soft goods. We present a simplified version of one e-coin without loss of generality.

1. $\mathcal{U}$ selects goods and signs an order agreement $\theta = Sign_{RCSS}(\mathcal{U}, \mathcal{M}, TTP, OA)$, where $OA = \{ID_{\mathcal{U}}, ID_{\mathcal{M}},$ purchase data/information, goods description, $(A, B)\}$.

2. $\mathcal{U}$ sends the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c) >$ and θ to $\mathcal{M}$
3. $\mathcal{M}$ verifies the pseudo e-coin and θ. If all of them are valid and $A \neq 1$, then he sends $d = \mathcal{H}(A, B, ID_{\mathcal{M}},$ date/time$)$ to $\mathcal{U}$.
4. $\mathcal{U}$ sends $k_1 = du_1 s + x_1 \mod q$ and $k_2 = ds + x_2 \mod q$ to $\mathcal{U}$. In addition, $\mathcal{U}$ must run the interactive protocol of bi-proof $BP(g, a_c, y_{TTP}, b_c)$ with $\mathcal{M}$ to show $\log_g a_c = \log_{y_{TTP}} b_c$.
5. $\mathcal{M}$ accepts the pseudo e-coin and the payment transcripts $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ if the following verifications hold:
$$g^{r'} = y_{\mathcal{B}}{}^{\mathcal{H}(A,B,z',e'_1,e'_2,b_c)+a_c} e'_1$$
$$A^{r'} = z'^{\mathcal{H}(A,B,z',e'_1,e'_2,b_c)+a_c} e'_2$$
$$g_1{}^{k_1} g_2{}^{k_2} = A^d B$$
If the above verifications pass, $\mathcal{M}$ sends the soft goods to the buyer $\mathcal{U}$.
6. $\mathcal{U}$ checks the soft goods delivered by $\mathcal{M}$. If it matches the description in OA, $\mathcal{U}$ releases t_c to $\mathcal{M}$. Since each one can check $a_c = g^{t_c}$ and $b_c = y_{TTP}{}^{t_c}$ by himself, the coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c, t_c), (d, k_1, k_2) >$ (i.e., the pseudo e-coin plus t_c) denotes a true e-coin that can be directly cashed from the bank.

Disputes

If $\mathcal{U}$ refuses to send t_c to the merchant $\mathcal{M}$, $\mathcal{M}$ begins the dispute process in which TTP can convert the pseudo e-coin into the true e-coin.

1. $\mathcal{M}$ sends the order agreement OA, the RCSS signature θ, the soft goods and the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ to TTP.
2. The TTP checks the validity of the soft goods, pseudo e-coin and signature θ. If the pseudo e-coin is constructed properly, the soft goods from $\mathcal{M}$ is consistent with the description in OA, and θ is valid, TTP sends $\mathcal{M}$ a transformation certificate $TCer = (E_c, T_c)$, where $E_c = a_c{}^{\sigma}$ (σ is a random number selected by TTP) and $T_c = \sigma + x_{TTP} F(a_c, E_c) \mod q$ (F is a public collision-free hash function). The transformation certificate can be used to verify the relation of a_c and b_c by the following equation:
$$a_c{}^{T_c} = E_c b_c{}^{F(a_c,E_c)}$$
3. TTP sends the soft goods to the buyer $\mathcal{U}$.

Deposit

In a normal case, $\mathcal{M}$ forwards the payment transcript and the true e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c, t_c), (d, k_1, k_2) >$ to the bank for deposit. Nevertheless, if $\mathcal{U}$ maliciously aborts the payment process, $\mathcal{M}$ can start the dispute process to acquire the $TCer$ from TTP. In this situation, the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ plus $TCer = (E_c, T_c)$ can be the valid token for deposit. We can also regard $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2), (E_c, T_c) >$ as a true e-coin with different form.

3 Analysis of the Fair Payment Protocol

Before presenting our analysis, we copy the claimed security features of the fair payment protocol, which are expressed in the form of propositions and lemmas in [25].

Unforgeability. *No one except* $\mathcal{U}$ *can create his own pseudo e-coin* $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$.

Indistinguishability. *No one can distinguish between a valid pseudo e-coin and a simulated one without the help of the buyer or TTP.*

Convertibility. *If* $\mathcal{M}$ *accepts the pseudo e-coin, it is guaranteed that TTP can later convert the pseudo e-coins into the true e-coins which can be directly deposited in the bank.*

Fairness. *If the above unforgeability, indistinguishability and convertibility hold for the proposed payment protocol, it can be guaranteed that at the end of the transaction, the buyer* $\mathcal{U}$ *can obtain the soft goods if and only if the merchant* $\mathcal{M}$ *can gain the equivalent true e-coin.*

Untraceability. *No one except* $\mathcal{M}$ *and TTP can confirm the signature* θ*. That means only* $\mathcal{M}$ *and TTP can be convinced that the order agreement* OA *is valid.*

Unlinkability. *The bank or other parties cannot link a coin* $< A, B, (z', e'_1, e'_2, r', a_c, b_c) >$ *to the original owner.*

Idea of the Colluding Attack

In the fair payment protocol, the merchant $\mathcal{M}$ colludes with his conspirator $\mathcal{C}$. After $\mathcal{M}$ receives the pseudo e-coin from the buyer $\mathcal{U}$, $\mathcal{M}$ brings the pseudo e-coin to TTP but claims that the trade is between $\mathcal{C}$ and $\mathcal{M}$. Then the TTP will convert the e-coin to an equivalent true e-coin for $\mathcal{M}$ and send the soft goods to $\mathcal{C}$, while $\mathcal{U}$ will gain nothing. Next we present the attack in details and explain why there is no solution against the attack.

Attack Details and Explanation

1. The malicious merchant $\mathcal{M}$ honestly implements the Payment protocol till step 5. After the verifications pass, he halts the protocol. That is, he obtains the valid pseudo e-coin without giving the soft goods.
2. Then $\mathcal{M}$ asks his conspirator $\mathcal{C}$ to sign a forged order agreement between $\mathcal{M}$ and $\mathcal{C}$, $\theta' = Sign_{RCSS}(\mathcal{C}, \mathcal{M}, TTP, OA')$ where $OA' = \{ID_{\mathcal{C}}, ID_{\mathcal{M}},$ purchase data/information, goods description, $(A, B)\}$.
3. $\mathcal{M}$ starts the Dispute process by sending the order agreement OA', the RCSS signature θ' on OA', the soft goods and the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ to TTP. Note that TTP has no way to tell whether OA' and θ' are consistent with the pseudo e-coin or not because of the unlinkability and untraceability. Note that the d in the pseudo e-coin is $d = \mathcal{H}(A, B, ID_{\mathcal{M}},$ date/time) instead of $d = \mathcal{H}(A, B, ID_{\mathcal{M}}, ID_{\mathcal{U}},$ date/time). If d is replaced with $d = \mathcal{H}(A, B, ID_{\mathcal{M}}, ID_{\mathcal{U}},$ date/time), the attack does not work anymore but the unlinkability and untraceability would disappear.
4. TTP converts the pseudo e-coin into a true e-coin for $\mathcal{M}$ and forwards the soft goods to $\mathcal{C}$. The buyer $\mathcal{U}$ is left without obtaining anything.
5. The problem of the fair payment protocol is that the e-money is in the form of digital cash, which is generated with the bank's private key and has no link with the buyer $\mathcal{U}$. If the e-money is in the digital cheque form that is generated with $\mathcal{U}$'s private key, the attack would not work.

6. The protocol cannot be fixed by asking $\mathcal{U}$ to sign a pre-contract to indicate that the trade is between $\mathcal{U}$ and $\mathcal{M}$ or by any other means. The conspirator $\mathcal{C}$ can just simulate $\mathcal{U}$ by doing everything $\mathcal{U}$ does. No one can distinguish $\mathcal{C}$ from $\mathcal{U}$ since the money is unlinkable and untraceable.

4 Fair Exchange of Digital Signatures

Fair exchange protocols have been studied by many researchers in recent years in [1], [2], [3], [4], [5], [6], [7], [11] [12], [15], [20], [26] and many other papers. Among them, [2], [4] and [5] have the same principle in employing verifiable encryption schemes (VES) of digital signatures. In [4], six schemes are proposed in its sections 4.1, 4.2, 4.3, 4.4, 4.6 and 4.7, respectively. The first three schemes are actually not by VES but by the same method of [7]. The latter three schemes exploit the VES of Guillou-Quisquater signatures, VES of Schnorr signatures and VES of ElGamal signatures, respectively. We describe the latter two here in the same denotations as in [4]. Before our description, we introduce a technique that is the main building-block for the schemes.

Building-Block $EQ\_DLOG(m; {g_1}^x, {g_2}^x; g_1, g_2)$

$EQ\_DLOG(m; y_1, y_2; g_1, g_2)$ is a non-interactive proof system for proving $Dlog_{g_1} y_1 = Dlog_{g_2} y_2$ without disclosing the value $x = Dlog_{g_1} y_1 = Dlog_{g_2} y_2$. The proof is associated with a message m. Here $g_1, y_1 \in$ group G_1, $g_2, y_2 \in$ group G_2, and at least one of G_1 and G_2 has an unknown order with bit-length l. Let $\mathcal{H}$ be a hash function $\{0,1\}^* \to \{0,1\}^k$ and $\epsilon > 1$ be a security parameter. The proof $EQ\_DLOG(m, y_1, y_2; g_1, g_2)$ is implemented as follows.

Prover: randomly choose $t \in [-2^{\epsilon(l+k)}, 2^{\epsilon(l+k)}]$, compute $c = \mathcal{H}(m||y_1||y_2||g_1||g_2||{g_1}^t||{g_2}^t)$ and $s = t - cx$ (in integer $\mathbb{Z}$). (s, c) is the proof/signature of EQ_DLOG $(m; y_1, y_2; g_1, g_2)$.

Verifier: given (s, c) and (m, y_1, y_2, g_1, g_2), check if $c = \mathcal{H}(m||y_1||y_2||g_1||g_2||{g_1}^s {y_1}^c||{g_2}^s {y_2}^c)$ and $s \in [-2^{\epsilon(l+k)}, 2^{\epsilon(l+k)}]$. If both hold, the verification is passed.

4.1 Fair Exchange of Schnorr Signatures

Settings

• System parameters: The system parameters are p, q and α, where p, q are primes and $q|p-1$, α is an element of order q of $\mathbb{Z}_p^*$.

• TTP: TTP has a pair of public/private keys (n, g)/(factors of n) for either Naccache-Stern encryption scheme [18] (or Okamoto-Uchiyama encryption scheme [21]). The encryption of M under the public key (n, g) is $g^M \mod n$ (or $h^r g^M \mod n$). M can be computed if the factors of n are known. It is claimed in [4] that both schemes can be adopted but for the sake of simplicity Naccache-Stern encryption scheme is employed.

• Alice: Alice has a pair of public/private keys y/a for Schnorr signature scheme where $y = \alpha^a \mod p$.

• Bob: Bob also has a pair of public/private keys for signature. Bob's signature on message M is denoted by $\mathbf{S}_{Bob}(M)$.

• Message: m is a message, on which Alice and Bob are exchanging their signatures.

Fair Exchange by Verifiable Encryption

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, e)$, where $r = \alpha^k \mod p$, $e = \mathcal{H}(m||r)$ and $s = k + ea \mod q$, for a randomly chosen k from $\mathbb{Z}_q$. The verification of $\mathbf{S}_{Alice}(m)$ is to check if $e = \mathcal{H}(m||\alpha^s y^{-e})$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. The e is left in plaintext, from which no one else can compute s. Since Alice knows s, she can implement $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$. Then Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob. That is (C, e, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $e = \mathcal{H}(m||V y^{-e})$, if valid, sends $\mathbf{S}_{Bob}(m)$ to Alice, otherwise does nothing.
3. Alice verifies Bob's signature and, if valid, sends $\mathbf{S}_{Alice}(m)$ to Bob.
4. If Bob does not receive anything or if Alice's signature is invalid, then he sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ and $\mathbf{S}_{Bob}(m)$ to TTP. This provides a vehicle for TTP to understand whether the protocol was correctly carried out. If this is the case, TTP sends $\mathbf{S}_{Alice}(m)$ to Bob and $\mathbf{S}_{Bob}(m)$ to Alice.

4.2 Fair Exchange of ElGamal Signatures

The settings are exactly the same as in the fair exchange of Schnorr signatures in Section 4.1. The scheme of fair exchange of ElGamal signatures is as follows.

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, r)$, where $r = \alpha^k \mod p$ and $s = k\mathcal{H}(m) + ar \mod q$, for a randomly chosen k from $\mathbb{Z}_q$. The verification of $\mathbf{S}_{Alice}(m)$ is to check if $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. The r is left in plaintext, from which no one else can compute s. Since Alice knows s, she can implement $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$. Then Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob. That is (C, r, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $V = r^{\mathcal{H}(m)} y^r \mod p$, if valid, sends $\mathbf{S}_{Bob}(m)$ to Alice, otherwise does nothing.
3. Alice verifies Bob's signature and, if valid, sends $\mathbf{S}_{Alice}(m)$ to Bob.
4. If Bob does not receive anything or if Alice's signature is invalid, then he sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ and $\mathbf{S}_{Bob}(m)$ to TTP. This provides a vehicle for TTP to understand whether the protocol was correctly carried out. If this is the case, TTP sends $\mathbf{S}_{Alice}(m)$ to Bob and $\mathbf{S}_{Bob}(m)$ to Alice.

5 Colluding Attacks and Countermeasures

5.1 A Feature of Digital Signatures

The digital signatures we consider here are those that consist of two parts, such as the (s, e) of Schnorr scheme, the (s, r) of ElGamal scheme, the (d, D) of Guillou-Quisquater scheme and similar signatures of many other schemes. Let us denote a signature on message m with public/private keys PK/SK by (X, Y), and the verification formula of the signature by

$$\mathbf{Vef}(m, X, Y, PK) = 1 \tag{1}$$

The security requirement of digital signature demands that, for given PK, it is infeasible to compute m and (X, Y) such that (1) holds without knowing SK. (That is the unforgeability in passive attack model. In active attack model, the security requirment is that it is infeasible to forge a valid signature without knowing SK even given a signing oracle.) However, the following feature is not prohibited for the security of signatures, while it plays an important role in our colluding attack.

> **Feature.** Given $m, (X, Y), PK$ that satisfy (1), it is easy to find $Y' \neq Y$ and $PK' \neq PK$ such that $\mathbf{Vef}(m, X, Y', PK') = 1$ without knowing SK.

Schnorr Signature. Given a signature (s, e) on message m such that $e = \mathcal{H}(m||\alpha^s y^{-e})$, we can always find $e' \neq e$ and $y' \neq y$ such that $e' = \mathcal{H}(m||\alpha^s {y'}^{-e'})$. We just take $e' = \mathcal{H}(m||\alpha^s \alpha^t)$ for a randomly chosen $t \in \mathbb{Z}_q$, and then set $x' = -t/e' \mod q$ and $y' = \alpha^{x'} \mod p$. Hence Schnorr signatures have the feature.

ElGamal Signature. Given a signature (s, r) on message m such that $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$, we can find $r' \neq r$ and $y' \neq y$ such that $\alpha^s = {r'}^{\mathcal{H}(m)} {y'}^{r'} \mod p$. We take $r' = (\alpha^s/\alpha^t)^{(1/\mathcal{H}(m) \mod q)}$ for a randomly chosen $t \in \mathbb{Z}_q$, and then set $x' = t/r' \mod q$ and $y' = \alpha^{x'} \mod p$. Hence ElGamal signatures have the feature.

Guillou-Quisquater Signature. In Guillou-Quisquater scheme, $n = pq$ is generated by a trusted center, where p and q are safe primes. A large prime v is selected, and n and v are published as system parameters. The p, q are recommended to be destroyed after that. (It is also allowed that n is generated by each signer. In that case different signer has different n, v.) The public/private keys J/B have relation $B^v J = 1 \mod n$. A signature (d, D) can be generated with the private key B by setting $T = r^v$, $d = \mathcal{H}(m||T)$ and $D = rB^d$, where r is randomly chosen from $\mathbb{Z}_n$. The verification of (d, D) is $d = \mathcal{H}(m||D^v J^d)$. To generate $d' \neq d, J' \neq J$ such that $d' = \mathcal{H}(m||D^v {J'}^{d'})$ is not as simple as the problems for Schnorr and ElGamal signatures. We cannot solve the problem of computing the d'th-root $\mod n$ since the factorization of n is not known.

5.2 Attack to Fair Exchange of Schnorr and ElGamal Signatures

Attack to Fair Exchange of Schnorr Signature by Dishonest Bob

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, e)$, where $e = \mathcal{H}(m||r)$ and $s = k + ea \mod q$ for randomly chosen $k \in \mathbb{Z}_q$ and $r = \alpha^k \mod p$. The verification formula of $\mathbf{S}_{Alice}(m)$ is $e = \mathcal{H}(m||\alpha^s y^{-e})$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. Then she implements $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$ and $C = g^s \mod n$. After that Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob, i.e., (C, e, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $e = \mathcal{H}(m||Vy^{-e})$, if valid, halts the protocol. Then he computes e', y' such that $e' = \mathcal{H}(m||Vy'^{-e'})$, i.e., $e' = \mathcal{H}(m||V\alpha^t)$ for $t \in_R \mathbb{Z}_q$ and $y' = \alpha^{(-t/e' \mod q)} \mod p$, and asks his conspirator Cathy to register y' as her public key. Bob can ask Cathy to sign (C, e', V) and any other things that could be signed by Alice for any possible authentication.
3. Bob sends (C, e', V), the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $\mathbf{S}_{Bob}(m)$ to TTP and claims that the exchange is between Cathy and Bob.
4. TTP first verifies the proof of $EQ\_DLOG(m; V, C; \alpha, g)$, then decrypts s and verifies whether (s, e') is a valid signature of Cathy and whether $\mathbf{S}_{Bob}(m)$ is a valid signature of Bob. If all the verifications pass, TTP sends $\mathbf{S}_{Cathy}(m) = (s, e')$ to Bob and $\mathbf{S}_{Bob}(m)$ to Cathy. Hence Bob obtains Alice's signature (s, e) without Alice obtaining anything.

Attack to Fair Exchange of ElGamal Signature by Dishonest Bob

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, r)$, where $r = \alpha^k \mod p$ and $s = k\mathcal{H}(m) + ar \mod q$ for a randomly chosen k from $\mathbb{Z}_q$. The verification formula of $\mathbf{S}_{Alice}(m)$ is $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. Then she implements $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$. Finally Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob, which is (C, r, V) plus the proof of $EQ\_DLOG(m; V, C;\ \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$, if valid, halts the protocol. Then he computes r', y' such that $\alpha^s = r'^{\mathcal{H}(m)} y'^{r'} \mod p$, i.e., $r' = (V/\alpha^t)^{(1/\mathcal{H}(m) \mod q)}$ for $t \in_R \mathbb{Z}_q$ and $y' = \alpha^{(t/r' \mod q)} \mod p$, and asks his conspirator Cathy to register y' as her public key. Bob can ask Cathy to sign (C, r', V) and any other things that could be signed by Alice for authentication.
3. Bob sends (C, r', V), the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $\mathbf{S}_{Bob}(m)$ to TTP and claims that the exchange is between Cathy and Bob.
4. TTP first verifies the proof of $EQ\_DLOG(m; V, C; \alpha, g)$, then decrypts s and verifies if (s, r') is a valid signature of Cathy and if $\mathbf{S}_{Bob}(m)$ is a valid signature of Bob. If the verifications all pass, TTP sends $\mathbf{S}_{Cathy}(m) = (s, r')$ to Bob and $\mathbf{S}_{Bob}(m)$ to Cathy. Hence Bob obtains Alice's signature (s, r) without Alice obtaining anything.

For some applications it is possible that message m implies the two parties to be Alice and Bob instead of Cathy and Bob. However TTP is not supposed to semantically understand the content of m. TTP only confirms that the m in the EQ_DLOG proof is identical to the m in the signatures.

5.3 Countermeasures

Before presenting our countermeasure, we show an interesting fact that the attack does not apply to DSA signatures. In DSA scheme, a signature (s, r) under public key y satisfies $r^s = \alpha^{\mathcal{H}(m)} y^r \mod p$. Although it is also simple to compute r', y' such that $r'^s = \alpha^{\mathcal{H}(m)} y'^{r'} \mod p$, the attack does not work anymore because the commitment V is different from that of ElGamal signatures. In ElGamal scheme $V = \alpha^s$ while in DSA $V = r^s$. Therefore, in DSA the proof is $EQ\_DLOG(m; V, C; r, g)$. Recall that the proof of $EQ\_DLOG(m; V, C; r, g)$ is (c, σ) satisfying $c = \mathcal{H}(m||V||C||r||g||r^\sigma V^c||g^\sigma C^c)$. That is, r is included in the verification of (c, σ). An $r' \neq r$ would make (c, σ) fail to pass the verification. Forging a new proof of $EQ\_DLOG(m; V, C; r', g)$ is impossible since it is equivalent to knowing s.

Now it is easy to see that the countermeasure is quite simple: Alice includes her ID (or her public key) into the proof, i.e., $EQ\_DLOG(m||ID_{Alice}; V, C; \alpha, g)$. Even better, she includes more detailed information $\mathcal{I}$ about the exchange in the proof, i.e., $EQ\_DLOG(m||\mathcal{I}; V, C; \alpha, g)$. In such case, $\mathcal{I}$ is like a label that cannot be removed and replaced. While attaching $\mathbf{S}_{Alice}(\mathcal{I})$ is like a label stick from outside and can be replaced, and therefore is useless. The ASW fair exchange scheme in [2] is not subject to the attack since a similar label is adopted.

Such label technique would destroy the untraceability and unlinkability, therefore cannot be adopted to fix the fair payment protocol.

6 Conclusions

In this paper we present a colluding attack to breach the fairness of an untraceable fair payment protocol and two schemes of fair exchange of digital signatures. Their fairness actually has no problem in the situation where only the entities described in the protocols exist. The cryptographic techniques employed are also secure and efficient. However, the security flaws appear if we consider the real situation where more entities exist.

As many security experts have pointed out, security does not equal to cryptography and good cryptographic algorithms do not automatically guarantee the security of application systems. Every component is secure does not necessarily mean that the whole system is secure. For complex systems, security should be studied under various attacks from various angles very carefully. It takes long time and big effort before being able to make an assertion.

Another viewpoint reflected from the result of this paper is that the concrete implementation is very critical to security. We show that a tiny difference, such as whether to include r in $EQ\_DLOG(m; V, C; r, g)$, could make a big difference in security. Hence engineers who implement the security schemes should be very carefully in following every step of the schemes.

References

1. N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocol for fair exchange", Proceedings of the 4th ACM Conference on Computer and Communication Security, pp. 8-17, ACM Press, 1997.
2. N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures", Advances in Cryptology - Proceedings of Eurocrypt'98, LNCS 1403, pp. 591-606, Springer-Verlag, 1998.
3. N. Asokan, V. Shoup and M. Waidner, "Asynchronous protocols for optimistic fair exchange", Proceedings of the 1998 IEEE Symposium on Security and Privacy, IEEE Computer Press, Oakland, CA, 1998.
4. G. Ateniese, "Efficient verifiable encryption and fair exchange of digital signatures", Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), pp. 138-146, 1999.
5. F. Bao, R. H. Deng and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP", Proceedings of 1998 IEEE Symposium on Security and Privacy, pp. 77-85, IEEE Computer Press, 1998.
6. M. Ben-Or, O. Goldreich, S. Micali and R. Rivest, "A fair protocol for signing contracts", IEEE Transactions on Information Theory, IT-36(1):40–46, January 1990.
7. C. Boyd and E. Foo, "Off-line fair payment protocols using convertible signature", Proceedings of Asiacrypt'98, LNCS 1514, pp. 271-285, Springer-Verlag, 1998.
8. S. Brands, "Untraceable off-line cash in wallets with observers", Proceedings of Crypto'93, LNCS 773, pp. 302-318, Springer-Verlag, 1993.
9. D. Chaum, "Blind signature for untraceable payments", Advances in Cryptology – Proc. of Crypto'82, Plenum Press, pp. 199-03, 1983.
10. D. Chaum, "Designated confirmer signatures", Proceedings of Eurocrypt'94, LNCS 950, pp. 86-91, Springer-Verlag, 1994.
11. L. Chen, "Efficient fair exchange with verifiable confirmation of signatures", Proceedings of Asiacrypt'98, LNCS 1514, pp. 286-299, Springer-Verlag, 1998.
12. R. H. Deng, L. Gong, A. A. Lazar and W. Wang, "Practical protocol for certified electronic mail", Journal of Network and Systems Management, 4(3), pp. 279-297, 1996.
13. S. Even, O. Goldreich and A. Lempel, "A randomized protocol for signing contracts", CACM, Vol. 28, No. 6, pp.637-647, 1985.
14. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, IT-31(4):469-472, 1985.
15. M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party", Proceedings of the 4th ACM Conferences on Computer and Communications Security, pp. 1-5, April 1-4, 1997, Zurich, Switzerland.
16. A. Fujioka, T. Okamoto and K. Ohta, "Interactive bi-proof systems and undeniable signature schemes", Proceedings of Eurocrypt'91, LNCS 547, pp. 243-256, Springer-Verlag, 1992.
17. L. C. Guillou and J. J. Quisquater, "A paradoxical identity-based signature scheme resulting from zero-knowledge", Advances in Cryptology - Crypto'88, LNCS 403, Springer-Verlag, pp. 216-231.

18. D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues", Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS), pp. 59-66, 1998.
19. M. Michels and M. Stadler, "Generic constructions for secure and efficient confirmer signature schemes", Proceedings of Eurocrypt'98, LNCS 1403, pp. 406-421, Springer-Verlag, 1998.
20. T. Okamoto and K.Ohta, "How to simultaneously exchange secrets by general assumption", Proceedings of 2nd ACM Conference on Computer and Communications Security, pp. 184-192, 1994.
21. T. Okamoto and S. Uchiyama, "A new public key cryptosystem as secure as factoring", Proceedings of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 308-318, 1998.
22. C. Pomerance, "On the Role of Smooth Numbers in Number Theoretic Algorithms." In Proc. Internat. Congr. Math., Zrich, Switzerland, 1994, Vol. 1 (Ed. S. D. Chatterji). Basel: Birkh?user, pp. 411-422, 1995.
23. C. P. Schnorr, "Efficient signature generation for smart cards", Proceedings Crypto'89, LNCS, Springer-Verlag, pp.225-232, 1990.
24. M. Stadler, "Publicly verifiable secret sharing", Proceedings of Eurocrypt'96, LNCS 1070, Springer-Verlag, pp.190-199, 1996
25. C.-H. Wang, "Untraceable fair network payment protocols with off-line TTP", Proceedings of Asiacrypt 2003, LNCS 2894, pp. 173-187, Springer-Verlag, 2003.
26. J. Zhou and D. Gollmann, "A fair non-repudiation protocol", Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Press, pp. 55-61, Oakland, CA, 1996.

A Restrictive Confirmation Signature Scheme

In [25], RCSS is designed as follows.

- **System Setup.** The parameters p, q and g, where p, q are primes such that $q|p-1$ and g is an element of $\mathbb{Z}_p^*$ of order q. F_1, F_2 are two collision resistant functions. The private/public key pairs of the signer S, the confirmer C, the recipient R and the verifier V are $(x_S, y_S^{x_S} \mod p)$, $(x_C, y_C^{x_C} \mod p)$, $(x_R, y_R^{x_R} \mod p)$ and $(x_V, y_V^{x_V} \mod p)$, respectively.
- **Signing Protocol.** Assume the signer has signed an undeniable signature (a, b, δ) on message m related to the confirmer's public key, i.e., $a = g^t \mod p$, $b = {y_C}^t \mod p$ and $\delta = (F_1(m||a)+b)^{x_S} \mod p$, where t is randomly chosen by S. For delegating C the ability of confirming this signature, the signer randomly selects k, u, v_1, v_2 and constructs a proof of

$$(w, z, u, v_1, v_2) = Proof_{DVLogEQ}(c, g, y_S, F_1(m||a) + b, \delta, y_V),$$

where $c = (c_1||c_2)$, $c_1 = g^u {y_V}^{v_1} \mod p$, $c_2 = g^u {y_C}^{v_2} \mod p$, $w = F_2(c||g||y_S|| F_1(m||a) + b||\delta||g^k||(F_1(m||a) + b)^k)$ and $z = k - x_S(w + u) \mod q$. Thus, the RCSS on m denotes $Sign_{RCSS}(S, C, V, m) = (a, b, u, v_1, v_2, w, z, \delta)$.
- **Proof by the Signer.** The confirmer C also plays the role of the recipient R. That means C will be convinced that he is able to prove the validity

of the signature to V in this procedure. C checks the proof by computing $c = ((g^u {y_V}^{v_1} \mod p)||(g^u {y_C}^{v_2} \mod p))$ and verifying if

$$w = F_2(c||g||y_S||F_1(m||a) + b||\delta||g^z {y_S}^{(w+u)}||(F_1(m||a) + b)^z \delta^{(w+u)})$$

To prove the relation of a and b, the signer needs to run the interactive protocol of bi-proof $BP(g, a, y_C, b)$ to show $\log_g a = \log_{y_C} b$.

- **Confirmation Protocol.** The confirmer C can prove the validity of the signature to V by running the interactive protocol bi-proof $BP(g, y_C, a, b)$ with V to show $\log_g y_C = \log_a b$. The verifier V needs to check whether the signature $(a, b, u, v_1, v_2, w, z, \delta)$ is created properly, and he can be convinced that the signature is valid if he accepts the proof of $BP(g, y_C, a, b)$.
- **Conversion Protocol.** The confirmer can convert the designated confirmer signature to a general non-interactive undeniable signature. Since the signer has constructed the designated verifier proof in a non-interactive way, V can check the validity of the signature by himself. The verifier V no longer needs to ask C to help him verify the signautre. Here, C randomly selects $\sigma \in \mathbb{Z}_q^*$ and computes $E = a^\sigma \mod p$ and $T = \sigma + x_C F(a, E) \mod q$, where F is also a hash function. The confirmer sends (E, T) to the verifier V, thus, V can verify if $a^T = Eb^{F(a,E)}$ [10].