Off-line Fair Payment Protocols using Convertible Signatures

Colin Boyd and Ernest Foo*

Information Security Research Centre
School of Data Communications
Queensland University of Technology
Brisbane, Australia
{boyd,ernest}@fit.qut.edu.au

Abstract. An exchange or payment protocol is considered *fair* if neither of the two parties exchanging items or payment at any time during the protocol has a significant advantage over the other entity. Fairness is an important property for electronic commerce. This paper identifies a design framework based on existing fair protocols which use offline trusted third parties, but with convertible signatures as the underlying mechanism. We show that in principle any convertible signature scheme can be used to design a fair payment protocol. A specific protocol is detailed based on RSA undeniable signatures which is more efficient than other similar fair payment schemes. Furthermore, in this protocol the final signature obtained is always an ordinary RSA signature.

1 Introduction

As more and more electronic transactions are being conducted on insecure networks, it is becoming obvious that electronic transactions are governed by different forces from the ones which affect normal physical exchanges of currency and goods. The possibility that transactions can occur remotely is one of the greatest advantages of electronic transactions as well as one of its biggest challenges to protocol designers.

In a typical physical exchange two entities, for example a customer and a shopkeeper, are present at the same location. During the exchange the customer hands the shopkeeper some notes and coins. In return the shopkeeper hands the desired goods to the customer. Unfortunately, in electronic commerce the security of this scenario is suspect because of the remoteness of the shopkeeper and the customer. It is possible that, once the customer's coins have passed through cyberspace and have been received by the shopkeeper, the shopkeeper refuses to deliver the goods; or if the shopkeeper hands the goods to the customer first the customer may log off instead of paying the shopkeeper. These problems arise with electronic transactions because the customer and shopkeeper are separated by cyberspace. In a physical situation, if the customer attempts to take the goods without paying the shopkeeper has the option to detain him.

* Sponsored by Commonwealth Bank and the Australian Research Council

K. Ohta and D. Pei (Eds.): ASIACRYPT'98, LNCS 1514, pp. 271-285, 1998.

This problem is an obvious one to electronic transaction protocol designers. In the course of development of electronic commerce protocols, many schemes have been developed to solve the problem of electronic exchange. These protocols are referred to in the literature as *fair exchange* protocols. The main objective of all fair exchange protocols is to ensure that at no point during the execution of the protocol can either of the entities participating in the exchange gain any (significant) advantage over the other if the protocol is suddenly halted.

1.1 Previous Work

Until recently there have been two main approaches for achieving fair exchange. The first approach is to ensure that the exchange occurs simultaneously. One way of providing simultaneous exchange is to have the participants exchange information bit by bit in an interleaving manner [15].

The second approach is to ensure that the exchange will be completed even though one of the entities participating in the exchange refuses to continue. Fair exchange protocols which employ this approach often use a trusted third party to store the details of the transaction [8, 18]. These details are released if one of the entities refuse to complete the protocol.

The use of the trusted third party greatly reduces the efficiency of the protocol. For a once off transaction such as, say, exchange of an important contract, high efficiency need not be a priority. But for regular electronic transactions, such as remote purchase of electronic goods, efficiency is a critical issue. So most of the recent fair exchange protocols attempt to reduce the need for the trusted third party in the online execution of the transaction while ensuring that a trusted third party is always available to resolve disputes. Protocols which do not require a trusted third party during the online execution are referred to as being *offline*.

The basic method for fair exchange using an offline third party has been established in a few recent papers. This method seems first to have been presented by Mao [12] and was followed further by Asokan, Shoup and Waidner [1] and Bao, Deng and Mao [2]. The general idea in all these papers is for one party (sometimes both parties) to send a signature to the other in such a way that:

- the recipient is convinced that the signature is correct but cannot transfer the proof of correctness to other parties.
- the recipient is convinced that *if necessary* the offline third party will be able to make the signature available to any verifier.

The recipient of such a signature should then be willing to proceed with the transaction with the knowledge that in case of dispute the third party can make the signature universally verifiable. But normally the third party is not involved, thereby allowing great efficiency savings over protocols with an online trusted third party.

The way that the above properties have been achieved in previous work is that the signature is encrypted with the public key of the third party. A *verifiable encryption* protocol is then executed between the sender and recipient

of the signed message in order to achieve the second property above. Although these are all ingenious protocols they do suffer from some potential drawbacks. One is that the verifiable encryption protocols currently available are computationally expensive, perhaps too much so for practical use of these schemes in everyday transactions. Another drawback is that these protocols also require a large amount of storage. In order to reduce complexity of both computation and communications, non-interactive verifiable encryption has been proposed. However, this raises the question of whether a non-interactive proof that a signature is encrypted is really any different from a signature itself, since it alone is sufficient to prove to any third party that the signer has committed to the message. We believe that there is little difference in functionality and that convertible signatures with non-interactive proofs of correctness should be avoided in our fair payment protocols.

1.2 Our Approach

In this paper new fair exchange protocols using an offline trusted third party are proposed. The principal new idea is to make use of a well known cryptographic primitive known as a *convertible signature*. All the previously published offline fair exchange schemes use the same basic idea of allowing one party, say the merchant, to be able to verify that *if necessary* he can employ the third party to convert a restricted commitment, verifiable by the merchant, into a full signature providing non-repudiation. In other words, a signature verifiable only by the merchant is converted into a universal signature. Convertible undeniable signatures provide exactly this property.

Undeniable signatures were introduced by Chaum and van Antwerpen [5]. These are digital signatures which can only be verified with the assistance of the signer. The signer is able to confirm or deny the ownership of the signature. No entity other than the signer is able to verify ownership of the signature. The signer is unable to prove that a valid signature is invalid or similarly that an invalid signature is valid. Convertible undeniable signatures developed by Boyar, Chaum and Damgård [3] build on the properties of undeniable signatures. Like undeniable signatures, convertible undeniable signatures can only be verified with the assistance of the signer but in addition the signer is able to selectively convert a single undeniable signature into a normal digital signature or collectively convert all the signer's signatures into normal digital signatures which can be verified by anyone.

An extension of the idea of convertible signatures are *designated converter* signatures, defined by Chaum [4] in which conversion may be achieved by a designated third party separate from the original signer. (Actually Chaum called them *designated confirmer* signatures, but we have changed the name to emphasize the conversion property which we are interested in. In fact in all known examples either confirmation or conversion may be achieved according to whether an interactive or non-interactive protocol is used.) These appear even more suited to application in fair exchange than ordinary convertible signatures.

Surprisingly, the use of convertible signatures does not appear to have been proposed before in the context of fair exchange. Despite this each of the offline fair exchange protocols proposed in previous papers [1, 2, 12] can be seen as a new designated converter signature algorithm! This is because verifiable encryption of a signature with a designated third party's public key clearly allows that third party to convert the signature into a universally verifiable one simply by decryption. In this paper we will use existing convertible signature schemes and adapt them to work for fair exchange protocols.

We explore the use of convertible undeniable signatures in fair exchange protocols with an offline third party. We mainly concentrate on the convertible property of these signatures as the undeniable function is not necessary for fair exchange. We are able to propose a number of new protocols which are at least as efficient as any other known protocols of this type. We regard the following as the three main contributions of the current paper.

- A general framework for fair payment in which any convertible signature scheme may be used.
- A new fair payment protocol which is more efficient than similar fair exchange schemes.
- A new designated converter signature for which converted signatures are ordinary RSA signatures.

Asokan, Shoup and Waidner [1] and Mao [12] exchange signatures fairly between two parties. In this paper we focus on more specific goals in that we wish to conduct a payment transaction between a customer and a merchant. Bao, Deng and Mao [2] present two fair exchange protocols which may be used for payment. Their first protocol is inefficient since it relies on use of verifiable encryption protocols which require a high number of rounds for security. Their second protocol uses a more efficient verifiable encryption protocol, but unfortunately this protocol is faulty and allows anyone to verify the customer's signature.

In section 2, we present a general design model for fair payment. Section 3 discusses use of existing convertible signature schemes within the model. A new designated converter signature is presented in detail inside the framework in section 4. An attack on the second Bao, Deng and Mao protocol is presented in the appendix.

2 A Framework for Offline Fair Payment Protocol Design

2.1 Definitions and Notation

The following symbols will be used to represent common parameters for the entire paper. Other parameters which are only used by specific protocols will be defined in the protocol description.

C	The customer entity.
M	The merchant entity.
B	The bank, acquirer or notary entity.
TTP	The trusted third party. It is possible that the bank could play the role of the trusted third party also but for the purpose of this paper we assume trusted third party and the bank are separate entities.
m	Purchase Information. This is information regarding the goods' *product ID*, the *price* to be paid for the goods as well as the *merchant account number*. It is assumed that this information will uniquely identify the transaction and the merchant entity.
$Cert_X$	The certificate which verifies the public key of entity X with the appropriate certification authority. It also contains the customer's banking details which can only be decrypted by the bank entity and the customer's public key.
Goods	The goods which are described in m. These are assumed to be software goods which can be transmitted securely encrypted across open networks.

The following notation is used to denote cryptographic operations. X and Y always represent communicating parties and may be any of the four entities defined above.

$E_{XY}(Message)$	$Message$, encrypted with the key XY using symmetric key cryptography. It is assumed that the key is known only by X and Y and that only these entities may know the contents of $Message$.
$E_X(Message)$	$Message$, encrypted with a public key belonging to X using public key cryptography. It is assumed that the public key belonging to X is known to all entities but only the entity X knows the corresponding private key to decrypt the contents of $Message$.
$Sig_X(Message)$	$Message$, digitally signed by X using public key cryptography. This implies that X's public key is used to ensure that the message was transmitted by X. A message signed in this fashion can be verified by any entity.
$S_X(Message)$	$Message$, digitally signed by X using a convertible undeniable signature.
$H(Message)$	A cryptographic function which results in a digest and checksum of $Message$, using an algorithm such as the Secure Hash Algorithm (SHA) one-way hash function.

2.2 The Design Framework

The following design framework is the model which has been used to develop the offline fair payment protocols described in this paper. This model can be used with any convertible signature scheme to construct new offline fair payment protocols as long as there is a way to ensure that only the third party is able to convert signatures.

The basic protocol ensures fairness by having TTP force the completion of a transaction if a dispute occurs. If no dispute occurs only C and M need to participate in the transaction.

Registration A registration protocol between the customer and third party is required for our efficient protocol in section 4. It will be correctly argued that the need for registration is an overhead which somewhat reduces the efficiency of the new protocol. However, we would like to point out that in practice trusted third parties will not be offering their services free of charge, and registration is probably a necessary phase. It need only be carried out once to initialize the relationship between C and TTP. The purpose of the registration process is to ensure that C has been identified and approved by TTP. In section 4 it is specifically used to ensure that both the trusted third party and C share keys which are to be used in the case of a dispute.

Payment The payment phase of the protocol must be conducted for each transaction. It is during this phase that M and the customer exchange goods.

It is assumed that in this phase C has already gone through a bidding process with M and that the two entities have already settled on the items to be purchased and the price to be paid. This process may be as simple as C selecting fixed priced goods from M's web site. Thus C should already have all the information included in m defined above.

$$
\begin{aligned}
&P1.\ C \rightarrow M : S(m)\\
&P2.\ C \leftrightarrow M : M \text{ verifies interactively that } S(m) \text{ is valid}\\
&P3.\ M \rightarrow C : E_C(Goods)\\
&P4.\ C \rightarrow M : Sig_C(m)
\end{aligned}
$$

In step $P1$, C generates a partial signature of the transaction information m. The partial signature must be in such a form that only M can verify its correctness. In all our protocols M and C have to interact to verify this partial signature and this verification is done in step $P2$. Another property of the partial signature is that the trusted third party must be able to convert it into a normal signature which anyone could verify. This property is only used in case of a dispute.

Once M is satisfied that C's partial signature is valid he sends a signed copy of the requested goods to C along with the transaction information. This is done in step $P3$ of the protocol.

In step $P4$, C, on receipt of the goods, sends a normal signature to M. M can now show everyone, including the bank, that C has agreed to the transaction details in m. In practice M will follow this step with a deposit process, but we omit this from further discussion.

Disputes If M decides not to send C the goods requested in the purchase request, C does not send M her full signature approving the transaction. If C decides to cheat M by refusing to send her full signature in step $P4$, M can begin the dispute process in which the trusted third party forces the transaction to occur.

$$\begin{aligned} D1.\ M &\rightarrow TTP : Sig_M(S(m), E_{TTP}(Goods)) \\ &\quad\ TTP \text{ converts } S(m) \text{ to } Sig_C(m) \\ D2.\ TTP &\rightarrow M : Sig_C(m) \\ D3.\ TTP &\rightarrow C : E_C(Goods) \end{aligned}$$

In step $D1$, M sends to TTP the partial signature $S(m)$ and an encrypted copy of the goods $E_{TTP}(Goods)$. The trusted third party can now convert the partial signature, which can only be verified by M, into a normal signature which anyone can verify.

In step $D2$, TTP sends the normal signature to M. TTP also sends the goods to C in step $D3$, in case M is trying to falsely obtain C's converter string.

We assume here that since the goods in question are information ('soft') goods neither party will gain if the goods are in fact sent twice to C in a dispute resolution. In particular for the system to work it is essential that neither party should gain from falsely engaging in a dispute.

One problem that we have not addressed here is what should happen if the soft goods become old before they can be used by C, such as might happen with travel tickets or betting slips. This is an important issue in practical applications although somewhat out of scope of our concern here which is only to ensure that M and customer fairly exchange payment for goods. In practice use of validity windows and expiry times could solve this problem; for example, TTP would use the time of dispute in conjunction with the expiry time of the soft goods to resolve the issue. Note that previous fair exchange solutions have also left this issue unresolved. Another approach to this problem was taken by Asokan, Shoup and Waidner [1] in which users are allowed to send abort messages to TTP, which keeps a record of all aborted transactions.

Security and Efficiency The security of any protocol designed using this framework relies on the following properties.

Property 1. Only C can create the partially signed message $S(m)$.

If this does not hold a fraudulent customer can impersonate C in step $P1$ of the payment protocol by generating the partial signature and illegitimately purchase goods. Transactions would be forced despite the denial of C.

Property 2. Only M and TTP can confirm that the partial signature generated by C is valid or can convert the partial signature into a universally verifiable signature.

In most cases only M needs to verify that C has produced the partial signature. If other entities were able to verify C's partial signature at any time the fairness of the payment would not be present and M would have the advantage.

Property 3. If M accepts the validity of the transaction, then TTP can convert partial signature $S(m)$ into a normal signature $Sig_C(m)$.

This property of the protocol ensures that the transaction will be completed fairly and that C does not gain an advantage over M. If this property was not provided C could refuse to send her signature in step $P4$ and receive the goods without payment.

3 Solutions using Existing Signatures

The framework of section 2 is applicable for use with a number of existing convertible signature algorithms. Due to space limitations we give only a brief outline here to allow room for more detailed discussion of the new protocol in the next section. We highlight two distinct options for using the framework. The first is to use convertible signature schemes together with verifiable encryption while the second is to use designated converter signatures.

3.1 Convertible Signatures with Verifiable Encryption

In the first practical convertible undeniable signature scheme of Boyar, Chaum and Damgård [3], an undeniable signature consists of a triple (T, r, s) of elements in the integers modulo a large prime p[1]. The element required to convert such a signature into a universally verifiable signature is the discrete log t of T. A partial signature can thus be formed by adding a copy of t encrypted with the TTP's public key to the undeniable signature. If the merchant is convinced (i) that (T, r, s) is correct and (ii) that the ciphertext really is t encrypted with TTP's public key, then he can be sure that TTP can convert the undeniable signature into a universally verifiable one.

The biggest problem with such a solution is that known protocols for verifiable encryption of discrete logs are not very efficient. For example, the protocol of Stadler [17] requires around 40 rounds in its interactive version. Alternative, more general, protocols due to Asokan, Shoup and Waidner [1] have the same requirement. More recent convertible undeniable protocols, such as those of Damgård and Pedersen [6] could also be used in a similar fashion. The problem

[1] It should be noted that although this scheme was successfully attacked by Michels, Petersen and Horster [13], the attack only affects the situation where signatures are converted all at once and not converted individually as in our application.

with all these is to find an efficient verifiable encryption scheme which can be matched to the conversion information. This problem is the reason why previous fair exchange protocols have not been efficient.

Some schemes, such as those of Michels and Stadler [14] do not seem appropriate to use in this way since conversion of individual signatures works by converting an interactive proof to a non-interactive one. This means that verifiable encryption of a non-interactive proof would be required to use this method.

3.2 Designated Converter Signatures

Designated converter signatures can be used in a very direct way in the framework. Confirmation by the customer during payment is essentially identical to signature confirmation. TTP takes the role of the designated converter and so can complete the dispute procedure when presented with the signature.

The first designated converter signature protocol proposed by Chaum [4] is based on RSA signatures, but these signatures are never used as plain RSA signatures in the protocol. Instead the correctness of signatures is linked to knowledge of a certain discrete log. The definition also relies on the existence of a function which destroys the multiplicative property of RSA signatures while at the same time being easy to invert.

Converted signatures in Chaum's scheme are not ordinary RSA signatures but non-interactive proofs of knowledge of a discrete log. In fact it is impossible for the signature owner in Chaum's scheme to convert signatures in the same way as the designated converter. We believe it is important in our application that signatures converted either by the owner (merchant) or the designated converter (TTP) are indistinguishable. To achieve this in Chaum's scheme, a signer who converts must recalculate a brand new designated converter signature, using a designated converter public key for which it knows the corresponding private key, and provide a non-interactive proof of correctness.

Further designated converter signatures were provided by Okamoto [16]. His constructions rely on different assumptions from those of Chaum but share the same properties that converted signatures are non-interactive proofs and also that conversion by owner and designated converter are different.

In conclusion we may say that use of existing designated converter signatures may be used within our framework. These solutions are efficient in that they require only two rounds (4 moves) to achieve high security. Their major drawback is that converted signatures are not in the form of ordinary RSA or ElGamal-type signatures which are likely to be required in electronic commerce schemes.

4 Offline Fair Payment using RSA Based Designated Converter Signatures

The cryptographic tools used in this new protocol are entirely based on RSA public key encryption and signatures. C splits her secret key in such a way that TTP is able to complete a partial signature of the customer. TTP can force

the transaction to completion by ensuring that a complete signature can be generated.

The scheme is an adaptation of the recent RSA-based undeniable signature scheme of Gennaro, Krawczyk and Rabin (GKR) [10]. Although their scheme does allow for designated confirmer signatures this still leaves the same drawbacks identified in the previous section if used directly for fair exchange, in particular converted signatures would not be ordinary RSA signatures.

4.1 Registration

This is an efficient protocol requiring only one signature by each party. The registration stage of the protocol need only be conducted once (or at periodic intervals) and can be used to support any number of payments whether they are disputed or not. No state information need be stored by the third party once registration is complete.

C has an RSA key pair consisting of secret exponent d, public exponent e and modulus n. In order to use the results of GKR we assume that n is a *strong prime* so that $n = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$ for primes p, p', q, q'. C's public key is certified by some certification authority which, in general, has no connection with TTP, but which can be used by TTP or any merchant to verify the correctness of the key. We denote this certificate $Cert_C$. The certificate must assert that the modulus is correctly formed. A method for achieving this is given in the GKR paper [10].

$$R1.\ C \rightarrow TTP : Cert_C$$

When TTP receives this certificate he generates a random number d_1 which is less than n. We require that $(d_1, \phi(n)) = 1$ but since $\phi(n) = 4p'q'$ this can be practically ensured by demanding that d_1 is odd. This is to be part of the secret key which is shared between C and TTP. TTP must be able to reconstruct d_1 from the identity of C. A practical way to achieve this without demanding TTP to store data for each customer is to make $d_1 = 2H(K, C) + 1$ where K is a secret known only to TTP and H is a suitable hash function.

TTP then sends this key encrypted to C. This can be achieved using the public key in the certificate $Cert_C$.

$$R2a.\ TTP \rightarrow C : E_C(d_1)$$

C now calculates the second part of the secret key d_2 such that $d_1 d_2 e = 1 \bmod \phi(n)$. C must also create a reference message ω and calculate a reference signature $S(\omega) = \omega^{d_2}$. It is shown by GKR that we may safely choose $\omega = 2$.

The reference message and signature will be used by M to verify that TTP knows d_1 and can force a transaction to completion. The reference message and signature are sent to TTP.

$$R2b.\ C \rightarrow TTP : \omega, S(\omega)$$

On receipt of the reference message and signature the trusted third party checks that the reference signature is valid by verifying that the following equation holds:

$$S(\omega)^{d_1 e} \bmod n = \omega$$

If the equation holds then C must have generated the reference signature correctly. TTP then creates a ticket which consists of C's public keys and the reference message and signature. TTP now signs this ticket and sends it to C.

$$R2c.\ TTP \rightarrow C : Sig_{TTP}(Cert_C, \omega, S(\omega))$$

C can now use this ticket to purchase goods from merchants. TTP's signature is a guarantee to the merchant who receives the ticket that the transaction can be completed by TTP if C refuses, or is unable, to complete the transaction. This is achieved by proving that a partial signature signed with d_2 is signed with the same exponent as was used to sign $S(\omega)$. This is the basis of the GKR scheme.

4.2 Payment

C has to generate a partial signature of the purchase information $S(m) = \bar{m}^{d_2}$. (Here and below, $\bar{m}$ denotes m after preprocessing by any desired hashing and padding processes which we will not detail here.) C indicates that she wishes to conduct a payment by sending the purchase information, a partially signed version of the purchase information and the ticket received from TTP to M.

$$P1.\ C \rightarrow M : m, S(m), Sig_{TTP}(Cert_C, \omega, S(\omega))$$

M and C now complete a confirmation protocol which convinces M that TTP can complete the transaction. This is exactly the signature confirmation protocol of the GKR scheme. An efficient 4-move zero knowledge protocol shown below is given in their paper [10].

	Customer		Merchant
			Choose $i, j \in_R [1..n]$
			Set $Q = S(m)^{2i} S(\omega)^j \bmod n$
P2a.		$\xleftarrow{Q}$	
	Set $A = Q^{d_1 e}$		
P2b.		$\xrightarrow{\texttt{commit}(A)}$	
P2c.		$\xleftarrow{i,j}$	
P2d.		$\xrightarrow{A}$	
			Verify $A = \bar{m}^{2i} \omega^j \bmod n$

The function `commit` is a commitment function to ensure the zero knowledge property. It may be implemented as a hash or as RSA encryption with an unknown secret exponent if it is designed to avoid further security assumptions.

If M is satisfied the exchange can take place in step $P3$ and $P4$ of the protocol.

$$P3.\ M \rightarrow C\ : E_C(Goods)$$
$$P4.\ C \rightarrow M : Sig_C(m)$$

4.3 Disputes

If C does not complete the protocol by aborting after receiving the goods, M contacts TTP to resolve the dispute. M sends to TTP the ticket and signature received from C and the goods.

$$D1.\ M \rightarrow TTP : Cert_C, m, S(m)$$
$$E_{TTP}(Goods)$$

TTP first recovers $d_1 = 2h(K, C) + 1$ then calculates $S(m)^{d_1}$. TTP checks whether $S(m)^{d_1} = Sig_C(m)$ since $Sig_C(m) = m^{d_1 d_2}$. TTP may also check that the description of the goods in m corresponds with $Goods$ sent to TTP. If so, TTP accepts the claim of M and proceeds to send $Sig_C(m)$ to M and the goods to C.

$$D2.\ TTP \rightarrow M : Sig_C(m)$$
$$D3.\ TTP \rightarrow C\ : E_C(Goods)$$

4.4 Security and Efficiency

Let us again examine the three security properties for this protocol. It is intuitively reasonable that property 1 holds if RSA signatures are secure. In fact it has been shown that breaking a *multisignature* with two private keys d_1 and d_2 is as hard as breaking RSA [9]. The basic idea is that if the multisignature can be broken given known signatures and partial signatures then RSA may be broken by simulating partial signatures with random d_1 values and complete signatures with the public e value. This proof easily can be adapted to include the trusted party for whom d_2 is also known.

Property 2 can also be proven from the security of RSA multisignatures. Similar to the above case, an algorithm that can convert a partial signature into a complete one can be used to forge ordinary RSA signatures.

Finally, to prove property 3 we can use the properties of the GKR signature. It is proven [10, Theorem 1] that the prover (customer) in the payment protocol cannot convince the verifier (merchant) to accept an incorrect signature except with negligible probability. Thus the merchant will only accept if $S(m) = \bar{m}^{d_2}$.

It must be pointed out that the proofs in GKR only give confidence that signatures are true RSA signatures up to multiple by an element of order 2. To be precise, it is proven that $S(m) = \alpha \bar{m}^{d_2}$ where α is an element of order at

most 2. Thus a customer could give this slight variant instead of the true RSA signature. However, in this case, on conversion the third party will obtain $\alpha \bar{m}^{d_1 d_2}$ and can hence obtain α. But there are only two non-trivial elements of order 2 (since it is certified that n is the product of only two primes) and knowledge of one of these, say β, is sufficient to find a factor $(\beta - 1, n)$ of n. Hence, although C could attempt to cheat in this way, the result is that the third party can forge any signature of C.

To summarize, we can prove the following.

Lemma 1. *Properties 1, 2 and 3 all hold for the scheme. At the end of the transaction, C obtains the goods if and only if M gains a true RSA signature from the customer of the order.*

The use of RSA signatures in this protocol allows it to be more efficient than protocols using verifiable encryption. Asokan, Shoup and Waidner [1] present a general fair exchange protocol which can also be used with a range of signature and encryption schemes. This includes a scheme which also uses all RSA signatures and an encrypted signature verification step. But Asokan, Shoup and Waidner's protocol is less efficient in terms of messages sent by a factor of 10 when compared to the ones using designated converter signatures.

5 Acknowledgements

Thanks to Wenbo Mao for suggesting that we study the area of fair exchange protocols. Thanks also to the anonymous referees who found critical typographical errors in our manuscript.

References

1. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures. In *Advances in Cryptology - Proceedings of EUROCRYPT '98*, pages 591–606, Espoo Finland, May 1998. Springer-Verlag.
2. Feng Bao, Robert H. Deng, and Wenbo Mao. Efficient and Practical Fair Exchange Protocols with Off-line TTP. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, 1998.
3. Joan Boyar, David Chaum, and Ivan Dåmgard. Convertible Undeniable Signatures. In *Advances in Cryptology - Proceedings of CRYPTO '90*, pages 189–205. Springer-Verlag, 1991.
4. David Chaum. Designated Confirmer Signatures. In *Advances in Cryptology - Proceedings of EUROCRYPT '94*, pages 86–91, Perugia Italy, May 1994. Springer-Verlag.
5. David Chaum and Hans van Antwerpen. Undeniable Signatures. In *Advances in Cryptology - Proceedings of CRYPTO '89*, pages 212–216, 1989.
6. Ivan Damgård and Torben Pedersen. New Convertible Undeniable Signature Schemes. In *Advances in Cryptology - Proceedings of EUROCRYPT '96*, pages 372–386, Berlin Heidelberg, 1996. Springer-Verlag.

7. T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Transactions on Information Theory*, volume IT-31(4), pages 637–647, 1985.
8. Matthew K. Franklin and Michael K. Reiter. Fair Exchange with a Semi-Trusted Third Party. In *Proceedings of the 4th ACM Conference on COmputer and Communications Security*, April 1997.
9. Ravi Ganesan and Yacov Yacobi. A Secure Joint Signature and Key Exchange System. Technical report, Bellcore Technical Memorandum, 1994.
10. Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. RSA-Based Undeniable Signatures. In *Advances in Cryptology - Proceedings of CRYPTO '97*, pages 132–149. Springer-Verlag, 1997.
11. L. C. Guillou and J. J. Quisquater. A Paradoxical Identity-Based Signature Scheme Resulting from Zero Knowledge. In *Advances in Cryptology - CRYPTO '88*, pages 216–231. Springer-Verlag, 1988.
12. Wenbo Mao. Publicly Verifiable Partial Key Escrow. In *ACISP'97*, pages 240–248. Springer-Verlag, 1997.
13. Markus Michels, Holger Petersen, and Patrick Horster. Breaking and Repairing a Convertible Undeniable Signature Scheme. In *Proceedings of the 3rd ACM Conference on Computers and Communications Security*, pages 148–152, New Delhi, 1996. ACM Press.
14. Markus Michels and Markus Stadler. Efficient Convertible Undeniable Signature Schemes. In *SAC '97*, 1997.
15. T. Okamoto and K. Ohta. How to Simultaneously Exchange Secrets by General Assumption. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 184–192, 1994.
16. Tatsuaki Okamoto. Designated Confirmer Signatures and Public Key Encryption are Equivalent. In *Advances in Cryptology - Proceedings of CRYPTO '94*, pages 61–74, Santa Barbara California, August 1994. Springer-Verlag.
17. Markus Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology - Proceedings of EUROCRYPT '96*, pages 190–199, Berlin Heidelberg, 1996. Springer-Verlag.
18. Jianying Zhou and Dieter Gollman. A Fair Non-repudiation Protocol. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, 1996. IEEE Computer Press.

A Breaking the Bao, Deng and Mao Fair Exchange Protocol

The second protocol of Bao, Deng and Mao [2] uses verifiable ElGamal encryption [7] of a Guillou-Quisquater (GQ) signature [11] to provide fair exchange. The system wide public parameters are n, g, q, v where $n = PQ$ is the modulus used for GQ signatures and $P = 2p'q + 1$ and $Q = 2pq + 1$ where P, Q, p, p', q are all primes, g is an element of order q and v is the public exponent used for GQ signatures.

A GQ signature of a message M is a pair (d, D) for which $d = h(M, D^v J^d \bmod n)$ where h is a published one way hash function. In the protocol only D is encrypted using the public key PK_{TTP} of TTP. ElGamal encryption is used to form the ciphertext pair $(W, V_{TTP}) = (g^w \bmod n, D(PK_{TTP})^w \bmod n)$ for

a randomly chosen w. In order to bind D to the ciphertext, the value $V = D^v \bmod n$ is also calculated and used as part of the 'challenge' c generated using a hash function $\mathcal{H}$.

$$c = \mathcal{H}(g, W, (PK_{TTP})^v, V_{TTP}^v / V, a, A)$$

where $a = g^u \bmod n$ and $A = PK_{TTP}^{vu} \bmod n$ for a random value u. Finally the 'response' $r = u - cw \bmod q$ is calculated. The following parameters are then sent from the prover to the verifier.

$$M, (W, V_{TTP}), r, c, V, d$$

The verifier can use these parameters to ensure that the third party is able to decrypt the ElGamal ciphertext (W, V_{TTP}) to obtain the correct D value so that (d, D) is the GQ signature of M.

The attack consists of showing that the verifier is able to calculate $PK_{TTP}^w \bmod n$ and hence decrypt the ElGamal ciphertext to obtain D without the help of TTP. In other words the verifier (or any observer) can convert that signature without the help of TTP. The main observation is that since PK_{TTP} is in the orbit of g, to remove exponents of PK_{TTP} it is necessary only to invert them modulo q (which is known) and not modulo n (which is not known). Thus the verifier first calculates $A = (PK_{TTP}^v)/(V_{TTP}^v/V)^c \bmod n = PK_{TTP}^{uv} \bmod n$ (this is part of the intended verification process). Then the verifier calculates $A^{v^{-1} \bmod q} \bmod n = PK_{TTP}^u \bmod n$ and also $PK_{TTP}^r \bmod n$. This allows calculation of $PK_{TTP}^{cw} \bmod n$ since the following holds.

$$PK_{TTP}^r = PK_{TTP}^u / PK_{TTP}^{cw} \bmod n$$

Finally the verifier obtains $(PK_{TTP}^{cw})^{c^{-1} \bmod q} \bmod n = PK_{TTP}^w \bmod n$ as required.