

A Unified Approach to a Fair Document Exchange System

N. Zhang

Department of Computer Science
The University of Manchester
Oxford Road, Manchester, M13 9PL, UK

Q. Shi, and M. Merabti

School of Computing & Mathematical Sciences
Liverpool John Moores University
Byrom Street, Liverpool L3 3AF, UK

Abstract

This paper aims to utilise different approaches to fair document exchange to propose a unified approach for the development of a flexible, general and secure fair document exchange system able to guarantee strong fairness for a document exchange among any number of parties under various situations. At the heart of this unified approach are methods for a verifiable and recoverable encryption of a symmetric (or conventional) key for document encryption and decryption in a simple and efficient manner. The verifiability means that any party can verify the correctness of a key encrypted without actually viewing the key, and the recoverability implies that the party can be assured that a designated party or a group of designated parties can decrypt the encrypted key to recover the original key. These methods are essential for the unified approach to achieve strong fairness. The main contribution of the work presented in this paper is that the unified approach proposed represents the first effort on providing an efficient unified solution for achieving strong fairness in document exchange.

Keywords: telecommunications systems, software systems, e-commerce, fair document exchange.

1. Introduction

One of the principal activities in e-commerce is concerned with fair document exchange among a group of parties (e.g. companies, organisations or individuals). Such exchange is required when the documents to be exchanged contain valuable information. For example, a vendor exchanges valuable electronic goods for a payment from a customer, a group of companies or individuals exchange valuable electronic goods, and an individual or a company exchanges an important email or document for acknowledgements from a number of recipients. The valuable nature of the documents dictates that the exchange must be secure and fair to prevent a dishonest party from misbehaving and to avoid the situation where a party P_a can obtain an expected document from another party P_b , while P_b cannot get its expected document from P_a .

So far a number of approaches have been proposed to achieve fair exchange (Asokan et al., 1996 – Bao et al., 1999; Chen, 1998; Franklin and Reiter, 1997; Franklin and Tsudik, 1998; Khill et al., 2001; Okamoto and Ohta, 1994 – Ray and Ray, 2001; Vogt et al., 1999; Zhang et al., 1999). The main methods used by these approaches can be divided into three categories:

- (1) Online trusted neutral (or third) party (TNP) based approaches: An online TNP acts as an intermediary to assist other parties in a document exchange. It is online because it takes part in the exchange process to perform tasks such as collecting, verifying and forwarding data items related to keys for document decryption (Franklin and Reiter, 1997). The neutral party could also be semi-trusted in the sense that it may

misbehave but does not conspire with any party involved in the exchange (Franklin and Reiter, 1997). The use of a semi-trusted neutral party (STNP) makes its implementation easier.

- (2) Offline TNP based approaches: An offline TNP does not participate in an exchange in normal cases. In other words, the parties involved in the exchange can fairly complete the exchange without any involvement of the TNP if the exchange process is operated properly. The TNP is invoked to recover necessary information, e.g. a signature on a file or a message (Bao et al., 1998), so as to ensure a fair completion of the exchange only in abnormal cases where the exchange is not operated properly due to system faults or a party's misbehaviour. This type of approach becomes preferable as it offers the more cost-effective use of a TNP. As with the online TNP based approaches, the neutral party could be semi-trusted as well (Bao and Deng, 1999).
- (3) Approaches without use of any TNP: Most of them are based on gradual secret releasing, e.g. (Okamoto and Ohta, 1994), and probabilistic methods, e.g. (Rabin, 1983), which are not considered as cost-effective (Bao et al., 1998), e.g. they require many rounds of exchanges. There is another approach (Zhang et al., 1999) which utilises honest behaviours of over half of multiple parties involved in an exchange to achieve fairness.

Fairness can be divided into strong fairness and weak fairness (Asokan et al., 1997). Strong fairness means that at the end of an exchange among a group of parties, either each party can obtain every expected document from the others, or no party can obtain any expected document from any other party. Weak fairness implies that at the end of an exchange, either it achieves strong fairness, or any party P_a which cannot obtain an expected document from another party P_b can prove to an arbiter that P_b has received (or can receive) its document. In the latter case, an external dispute resolution system such as a court of law is needed to resolve the dispute, which may not guarantee that P_a will receive the expected document from P_b . Thus strong fairness is more desirable, which is the main issue to be addressed in the rest of this paper.

There are a number of fair exchange approaches such as those given in (Asokan et al., 2000 – Bao et al., 1999; Chen, 1998), which belong to category (2) above and can achieve strong fairness. These approaches are concerned mainly with exchanges involving signatures. The principal idea used by the approaches is based on a verifiable and recoverable encryption of a signature. The verifiability means that any party can verify the correctness of the encrypted signature without actually viewing the signature. The recoverability implies that only a designated offline TNP (or STNP) can decrypt the encrypted signature when the TNP is invoked for the legitimate recovery of the signature. This verifiable and recoverable signature encryption is essential for the approaches to achieve strong fairness based on an offline TNP.

However, existing approaches offering strong fairness suffer two main weaknesses. First, in an exchange of documents, the parties involved normally first exchange their documents encrypted with symmetric (or conventional) keys, and then exchange the keys for the decryption of the documents. The main issue for this exchange is about how to exchange the keys fairly. It would be desirable to apply the concept of verifiable and recoverable signature encryption to keys so as to achieve strong fairness for key exchange. As the verification of a key is different from that of a signature, appropriate methods for verifiable and recoverable key encryption are needed. To the best of our knowledge, no published approach to fair document exchange has applied the concept of such key encryption to achieve strong fairness based on an offline TNP/STNP.

Secondly, no existing approach has unified the above three categories of approaches to handle exchanges under various situations in a flexible, consistent and cost-effective manner, while guaranteeing strong fairness. Currently, most existing approaches fall in one of the three categories. This restricts their applicability as different situations may require the use of different categories of approaches, e.g. only an approach of category (3) is applicable if no TNP/STNP can be agreed by all the parties involved in an exchange. Consequently, a document exchange system has to incorporate the different categories of approaches, and little consistency between them hinders a cost-effective implementation of the system. Though the approach given in (Zhang et al., 1999) belongs to both of categories (1) and (3), it can only provide weak fairness. Additionally the approach presented in (Vogt et al., 1999) falls in categories (1) and (2), but it does not really offer strong fairness when it operates with an offline TNP.

The aim of this paper is to rectify the above weaknesses. Specifically, we will present a simple but efficient approach to verifiable and recoverable key encryption for each of the three categories (1) - (3) above. We will then combine these approaches to propose a flexible, general and unified approach for the development of a fair document exchange system, which can offer strong fairness. The unified approach can operate in any one of the following modes for a document exchange among any number of parties:

- (a) Use of an online STNP if half, or fewer than half, of the parties are honest, or
- (b) Use of an offline STNP if half, or fewer than half, of the parties are honest, or
- (c) Without use of any TNP/STNP if over half of the parties are honest.

These three modes are similar to the three categories (1) - (3) of approaches to fair document exchange described earlier, respectively, but the first two modes make use of STNPs instead of TNPs. Modes (a) and (b) have some advantages and weaknesses in comparison with each other. Mode (a) offers an easy implementation of a STNP as it can operate just like an ordinary party, but the STNP needs to participate in the exchange process online. Mode (b) provides a more cost-effective use of a STNP because it is invoked only under abnormal cases, but the STNP needs to perform different operations from ordinary parties so that special care must be given to its management and protection. These advantages and weaknesses should be taken into account in conjunction with application environments to determine which mode is more suitable.

The major novel contribution of this paper is two fold. First, it presents the first unified fair document exchange approach that applies the concept of verifiable and recoverable key encryption to achieve strong fairness under various situations. Secondly, the mode (b) of the unified approach is more efficient than other related work, and no existing fair document exchange approaches offer the mode (c) of the unified approach for document exchange with strong fairness.

The rest of the paper is organised as follows. Section 2 states the notation, assumptions and strong fairness requirement to be used in the subsequent sections. In Section 3, we present an approach to fair document exchange without use of any TNP/STNP, i.e. mode (c) above. Section 4 demonstrates how mode (a) can be handled as a special instance of the approach by employing an online STNP. In Section 5, we extend the approach to incorporate an offline STNP, i.e. mode (b). Section 6 proposes a unified approach to fair document exchange based on the approaches developed. In Section 7 we analyse the fairness of the unified approach with

regard to the strong fairness requirement stated. Section 8 compares the unified approach with related work. Finally our conclusions and future work are outlined in Section 9.

2. Preliminaries

2.1. Notation

The notation to be used throughout this paper is summarised as follows:

- $E_k(x)$ expresses the ciphertext of a data item x encrypted with a key k . $E_k(x)$ is computed using a public-key cryptosystem (e.g. RSA) if the corresponding decryption key is different from k , and using a conventional cryptosystem (e.g. triple DES) otherwise. Here we assume that the cryptosystems used are secure.
- pk_i and sk_i represent public and private keys of a party P_i , respectively.
- $o(x) = g^x \bmod p$ with $x < q$ is used as a one-way function (Schneier, 1996). Here, \bmod denotes the modulo operator, p is a prime number, q is a large prime factor of $p - 1$, and g is between 1 and p (i.e. $1 < g < p$). p , q and g are known by all the parties involved.
- $h(x)$ is a one-way hash function with the following properties: (a) for any x , it is easy to compute $h(x)$; (b) given x , it is hard to find x' ($\neq x$) such that $h(x) = h(x')$; and (c) given $h(x)$, it is hard to compute x . An example of such a one-way hash function is SHA-1 (Federal, 1995).
- $\lfloor x/y \rfloor$ denotes the quotient of an integer x divided by another integer y .
- x, y denotes the concatenation of data items x and y .
- $P_i \rightarrow P_j : M$ signifies that a party P_i sends a message M to another party P_j .

2.2. Assumptions

The case of a fair document exchange among n (> 1) parties can be described as follows:

- Every party P_i ($1 \leq i \leq n$) has a valuable document doc_i and a symmetric (or conventional) key k_i for the encryption and decryption of doc_i .
- Every party wishes to exchange its document for the document of each of the $n - 1$ other parties. All the parties have agreed a session number sn to distinguish the current session of exchange from others. Note that there are other classes of exchange (Franklin and Tsudik, 1998), e.g. a cyclic exchange means that each party P_i only wants doc_{i-1} (doc_n if $i = 1$) from P_{i-1} (P_n if $i = 1$), and offers doc_i to P_{i+1} (P_1 if $i = n$). Though we do not specifically address these classes of exchange in this paper, the approaches to be presented later can be applied to them by letting each party send certain messages only to designated parties.

The above document exchange will be carried out based on the following assumptions:

- P_{on} and P_{off} are online and offline STNPs, respectively, in the sense that either of them may misbehave by attempting to obtain a document doc_i , but each of them always correctly executes its specified operations

and does not conspire with any party P_i directly or indirectly. For easy presentation, we let *off* and *on* be equal to 0 and $n+1$, respectively, i.e. $P_{off} = P_0$ and $P_{on} = P_{n+1}$.

- For any two parties P_i and P_j , P_i knows P_j 's public key pk_j certified by a certification authority.
- Every party P_i has a certificate or confirmation associated with its document doc_i and key k_i , which is issued by an authority A_w . The certificate involves the following items:
 - $hed_i = h(E_{k_i}(doc_i))$ is the hash value of the ciphertext of doc_i encrypted with k_i .
 - $rk_i < q$ is a random number chosen by P_i , which is used to define $k_i = h(rk_i)$, $ok_i = o(k_i)$, and $ork_i = o(rk_i)$, where $o(x)$ and q have been defined in Section 2.1.
 - des_i is the content description of doc_i , e.g., if doc_i is a movie, then des_i contains its title and summary.
 - $sign_{w,i}$ is A_w 's signature on $h(hed_i, ok_i, ork_i, des_i)$ signed with its private key sk_w and a time stamp $ts_{w,i}$, i.e. $sign_{w,i} = E_{sk_w}(ts_{w,i}, h(hed_i, ok_i, ork_i, des_i))$.

The certificate can now be expressed as $CON_i = (hed_i, ok_i, ork_i, des_i, sign_{w,i})$.

$sign_{w,i}$ in CON_i represents authority A_w 's approval for the following cases:

- (1) If an encrypted document ed_i and a key k'_i meet the condition $h(ed_i) = hed_i$ and $o(k'_i) = ok_i$, then the decryption of ed_i with k'_i will recover a document, i.e. doc_i , with its contents matching description des_i .

This case will be used for document exchange in either mode (a) (i.e. use of P_{on}) or (c) (i.e. without use of any STNP) described in Section 1.

- (2) If an encrypted document ed_i and an item rk'_i meet the condition $h(ed_i) = hed_i$ and $o(rk'_i) = ork_i$, then the decryption of ed_i with key $k_i = h(rk'_i)$ will recover a document, i.e. doc_i , with its contents matching the description in des_i .

This case will be used for document exchange in mode (b) (i.e. use of P_{off}). The reason for $k_i = h(rk_i)$ will be explained in Section 5.

Note that authority A_w only needs to issue certificate CON_i once which can then be used by P_i to exchange doc_i for as many other documents as P_i can.

The rationale of this assumption is similar to that used and justified in (Franklin and Reiter, 1997). The assumption is essential for achieving strong fairness for the exchange, as it allows each party to verify the correctness of another party's encrypted document and its associated decryption key during the exchange. Without the assumption, a dishonest party could use a worthless document to exchange for the valuable document of another party, and this dishonesty cannot be detected until the exchange is completed. In other words, when the dishonesty is detected, the dishonest party has already unfairly gained other parties' valuable documents.

Here we suppose that all the other parties have received CON_i from P_i before the exchange starts.

To illustrate the application of the above certification, consider a simple example of on-line digital goods purchases in e-commerce. Let doc_1 represent a film produced and certified by a film producer, and P_1 an on-

line merchant contracted by the producer to sell the film on-line. The film only needs to be certified by the producer once. To issue a certificate for P_1 , the producer randomly chooses a number $rk_1 < q$ to compute $k_1 = h(rk_1)$, $ok_1 = o(k_1)$, and $ork_1 = o(rk_1)$, encrypts doc_1 with k_1 as a symmetric key, i.e. $E_{k_1}(doc_1)$, calculates $hed_1 = h(E_{k_1}(doc_1))$, assigns the title and summary of the film to des_1 , and uses its private key sk_p to produce a signature $sign_{p,1} = E_{sk_p}(ts_{p,1}, h(hed_1, ok_1, ork_1, des_1))$ with a time stamp $ts_{p,1}$. The producer then sends $CON_1 = (hed_1, ok_1, ork_1, des_1, sign_{p,1}, E_{k_1}(doc_1)$ and $E_{pk_1}(rk_1)$ encrypted with P_1 's public key pk_1 to P_1 . Note that for better security, the producer can further encrypt $E_{k_1}(doc_1)$ with a symmetric session key to prevent a STNP from gaining it, as will be discussed in Section 7. Upon receipt of the items from the producer, P_1 checks their correctness accordingly. For example, P_1 verifies the correctness of signature $sign_{p,1}$ in CON_1 by decrypting it with the producer's public key pk_p to check that the hash value recovered from the decryption is the same as $h(hed_1, ok_1, ork_1, des_1)$ and $ts_{p,1}$ is fresh. Once P_1 has proved the correctness of these items, P_1 can sell the film for as many times as P_1 can without any involvement of the producer.

A customer denoted as P_2 wants to purchase the film after seeing an advertisement about it on TV. To make the purchase, P_2 obtains an electronic payment (e.g. an electronic cheque) expressed as doc_2 , which is certified or issued by a bank or credit card company in a way similar to that used by the producer. P_1 and P_2 want to fairly exchange their documents doc_1 and doc_2 .

In this example, the film producer and bank (or credit card company) play the role of the authorities for the certification of documents doc_1 and doc_2 , respectively. This indicates that different authorities may certify different documents to be exchanged, and the document certification may be an inherent process rather than a separate process.

- In the case where offline STNP P_{off} is used, we assume that every party P_i has another certificate issued by P_{off} . To issue the certificate, P_{off} randomly chooses a number $rn_{i,off}$ ($0 < rn_{i,off} < q$) to compute the following items based on P_{off} 's private key sk_{off} and P_i 's public key pk_i :
 - $si_{i,off} = h(sk_{off}, pk_i, rn_{i,off}) \bmod q$ is a secret item to be shared (or known) only by P_i and P_{off} .
 - $pi_{i,off} = o(si_{i,off})$ is a public item associated with $si_{i,off}$, which can be known by any party.
 - $sign_{i,off} = E_{sk_{off}}(ts_{i,off}, h(pk_i, rn_{i,off}, pi_{i,off}))$ with time stamp $ts_{i,off}$ is P_{off} 's signature on $h(pk_i, rn_{i,off}, pi_{i,off})$, where the inclusion of P_i 's public key pk_i in $sign_{i,off}$ allows the certificate to be bound to P_i .

P_i 's certificate can now be expressed as $CER_{i,off} = (rn_{i,off}, pi_{i,off}, sign_{i,off})$.

P_{off} then transfers $E_{pk_i}(si_{i,off})$ and $CER_{i,off}$ to P_i . P_{off} does not store $si_{i,off}$ and $CER_{i,off}$, and given pk_i and $rn_{i,off}$, P_{off} can compute $si_{i,off}$ when needed. Note that $CER_{i,off}$ can be reused by P_i for as many document exchanges with the same or different parties as P_i wishes. This issue will be discussed further later.

$CER_{i,off}$ will be used for offline key recovery to be presented in Section 5. We assume that every party P_j ($1 \leq j \leq n$, and $j \neq i$) has received $CER_{i,off}$ and verified its correctness based on signature $sign_{i,off}$ (i.e. $h(pk_i, rn_{i,off}, pi_{i,off})$ calculated by P_j matches the hash value recovered by decrypting $sign_{i,off}$ with P_{off} 's public key pk_{off}) before the document exchange begins.

2.3. Strong Fairness Requirement

To exchange the documents fairly, each party P_i ($1 \leq i \leq n$) first sends its encrypted document $E_{k_i}(doc_i)$ to every party P_j ($1 \leq j \leq n$, and $i \neq j$), and these parties then engage in exchange of their document keys. This in fact turns the issue of fair document exchange into that of fair key exchange which needs to meet the following strong fairness requirement:

If a party P_a has obtained another party P_b 's document key k_b or can obtain k_b with assistance of some parties, then every party P_i has obtained any other party P_j 's key k_j or can obtain k_j with assistance of some parties.

This requirement ensures that under any circumstances including misbehaviour and collusion, either each party can get all the others' document keys, or none of them can get any other party's document key.

The following three sections will present approaches to key exchange for three modes (a) - (c) described in Section 1, respectively, which can satisfy the above strong fairness requirement.

3. Key Exchange without Any STNP/TNP

Suppose that at most m ($0 < m < \lfloor (n+1)/2 \rfloor$) out of the n parties might be dishonest or not trusted, e.g. they may intend to obtain other parties' document keys without allowing the others to get their keys. The remaining $n-m$ parties are honest. Note that when $m = 0$, i.e. all the parties are honest, there is no need to use any fair exchange approach for the document exchange. The upper bound of m ensures that over half of the parties wish to honestly exchange their documents. In this case, there is no need to employ any STNP for the exchange (i.e. mode (c) mentioned in Section 1). In the case of $m \geq \lfloor (n+1)/2 \rfloor$, solutions will be discussed in Sections 4 and 5.

The principal idea for key exchange in the case of $m < \lfloor (n+1)/2 \rfloor$ is an extension to the work described in (Zhang et al., 1999). The idea is to let each party P_i generate one secret item $sec_{i,j}$ based on its document key k_i for every party P_j , and a group VER_i of proof items for each party to check the correctness of the secret item received from P_i without knowing k_i . This group of secret items $sec_{i,j}$, denoted as SEC_i , should ensure that any $m+1$ of them can be used to recover k_i , but any fewer than $m+1$ of them cannot reveal any information about k_i . This allows any $m+1$ parties to jointly recover k_i after the secret items have been distributed to their corresponding parties, but any collusion among m (or fewer) dishonest parties is unable to compute k_i . SEC_i and VER_i represent a verifiable and recoverable encryption of key k_i . Once a party P_j has received a secret item $sec_{i,j}$ and a proof item group VER_i from every party P_i and confirmed the correctness of $sec_{i,j}$ based on VER_i , it is secure for P_j to release its key k_j to others. In case another party P_d 's key k_d fails to reach P_j after the release of k_j due to misbehaviour or unreliable communication, P_j can request other parties to jointly recover k_d by revealing their possessed secret items received from P_d .

The above idea can be implemented as the four-stage exchange process below.

Stage 1: Exchange of Verifiable and Recoverable Encrypted Keys

Each party P_i ($1 \leq i \leq n$) first defines an item $e = m$, and determines a function based on a set of $e+1$ coefficients $a_{i,l}$ ($0 \leq l \leq e$, and $a_{i,l} < q$):

$$y_i(x) = (a_{i,0} + \dots + a_{i,l} \times x^l + \dots + a_{i,e} \times x^e) \bmod q.$$

Here, $a_{i,0}$ is required to be equal to P_i 's key k_i , i.e. $a_{i,0} = k_i$, and the other coefficients $a_{i,l}$ ($1 \leq l$) are picked randomly by P_i . The reason for the use of item e is to make $y_i(x)$ applicable to the other two exchange modes to be described in Sections 4 and 5, which require different numbers of coefficients.

Based on function $y_i(x)$, P_i can generate n secret items $sec_{i,j}$ and e (or m) proof items $pro_{i,l}$:

$$sec_{i,j} = y_i(j) \text{ for every } j (1 \leq j \leq n), \text{ and } pro_{i,l} = o(a_{i,l}) \text{ for every } l (1 \leq l \leq e).$$

These items form the following groups:

$$SEC_i = (sec_{i,1}, \dots, sec_{i,n}) = (y_i(1), \dots, y_i(n)), \text{ and } VER_i = (pro_{i,1}, \dots, pro_{i,e}) = (o(a_{i,1}), \dots, o(a_{i,e})).$$

The verifiable and recoverable encryption of k_i can now be defined as (SEC_i, VER_i) . As each party P_j is to receive only one secret item $sec_{i,j}$ together with VER_i , the encrypted key will be held jointly by all the parties.

Additionally, P_i randomly picks a symmetric session key ssk_i for secure and easy transmission of its items to other parties. P_i then executes the following transaction:

$$P_i \rightarrow P_j : E_{pk_j}(sec_{i,j}, ssk_i, VER_i)$$

to send $E_{pk_j}(sec_{i,j}, ssk_i, VER_i)$ to each party P_j . The encryption of $sec_{i,j}$, ssk_i and VER_i with P_j 's public key pk_j is to prevent any other party from gaining $sec_{i,j}$ and outsiders from obtaining any one of these items. P_i can also add session number sn and a signature in the above transaction, which will be considered in Section 6.

Upon receipt of P_i 's message, P_j decrypts $E_{pk_j}(sec_{i,j}, ssk_i, VER_i)$ with its private key sk_j to recover $sec_{i,j}$, ssk_i and VER_i . P_j then defines items $pro_{i,0} = ok_i$ and $v_{i,j} = j$, and performs the following verification, where P_j knows $ok_i = o(k_i)$ as assumed in Section 2.2 and the purpose for the use of $pro_{i,0}$ and $v_{i,j}$ is the same as that for item e :

Verification 1: Check that $o(y_i(v_{i,j})) = (pro_{i,0} \times \dots \times pro_{i,l}^{v_{i,j}^l} \times \dots \times pro_{i,e}^{v_{i,j}^e}) \bmod p$.

This verification is to ensure that $sec_{i,j} = y_i(v_{i,j})$ is indeed a result calculated from the function associated with items $pro_{i,0}$ (or ok_i), ..., $pro_{i,e}$. The verification is based on the following fact:

$$\begin{aligned} o(y_i(v_{i,j})) &= g^{y_i(v_{i,j})} \bmod p \\ &= g^{a_{i,0} + \dots + a_{i,l} \times v_{i,j}^l + \dots + a_{i,e} \times v_{i,j}^e} \bmod p \\ &= (g^{k_i} \times \dots \times (g^{a_{i,l}})^{v_{i,j}^l} \times \dots \times (g^{a_{i,e}})^{v_{i,j}^e}) \bmod p \\ &= (o(k_i) \times \dots \times o(a_{i,l})^{v_{i,j}^l} \times \dots \times o(a_{i,e})^{v_{i,j}^e}) \bmod p \\ &= (pro_{i,0} \times \dots \times pro_{i,l}^{v_{i,j}^l} \times \dots \times pro_{i,e}^{v_{i,j}^e}) \bmod p. \end{aligned}$$

If $sec_{i,j}$ fails to pass verification 1, P_j requests P_i to retransmit its items. For repeated failures of verification 1, P_j can terminate the exchange, and informs other parties of the termination. This will also be applied to the other verifications to be introduced later.

Stage 2: Exchange of Acknowledgements

Once a party P_i has received $sec_{j,i}$ and VER_j from every party P_j and successfully verified $sec_{j,i}$ based on VER_j , P_i computes:

$$hsec_i = h(sec_{1,i}, \dots, sec_{n,i}), hpk_i = h(pk_1, \dots, pk_n), \text{ and } hver_i = h(VER_1, \dots, VER_n),$$

and then uses its private key sk_i to issue an acknowledgement ack_i of the successful verification with session number sn :

$$ack_i = E_{sk_i}(sn, hsec_i, hpk_i, hver_i).$$

In fact, ack_i is P_i 's signature on sn , $hsec_i$, hpk_i and $hver_i$, which ties all the parties (through their public keys, i.e. hpk_i), their proof item groups (i.e. $hver_i$), and the secret items possessed by P_i (i.e. $hsec_i$) to the current session sn . This signature indicates that P_i 's successful verification depends on the correctness of the n functions related to $hver_i$, which requires the further verification below (i.e. verification 2).

P_i executes the transaction below:

$$P_i \rightarrow P_j : ack_i$$

to send ack_i to each party P_j .

When a party P_j has received an acknowledgement from every party P_i , P_j carries out:

Verification 2: (a) Check that for any two parties P_a and P_b , sn , $hpka$ and $hver_a$ recovered from the decryption of acknowledgement ack_a with public key pk_a are the same as sn , $hpkb$ and $hver_b$ recovered from the decryption of ack_b with pk_b , respectively.

(b) Check that $hsec_a$ in ack_a is the same as $hsec_b$ in ack_b , only if offline STNP P_{off} is used.

Verification 2 (b) above will be used for the offline key recovery to be presented in detail in Section 5.

If verification 2 above is positive, P_j is assured that every party P_i has used the same function $y_i(x)$ associated with VER_i to generate its secret items, and that each of these items has been received, and its correctness verified, by the corresponding party. In other words, the verifiable and recoverable encryption of every key k_i , i.e. (SEC_i, VER_i) , has been produced and distributed to all the other parties correctly by P_i . These acknowledgements are used by P_j as the assurance that P_j can obtain the key k_i of any party P_i because any $m+1$ secret items of P_i , held by the other parties, can be used to compute k_i as to be detailed later.

Stage 3: Exchange of Document Keys

If verification 2 performed by a party P_i is positive, then it is secure for P_i to release its key k_i to all the other parties, i.e. P_i executes:

$$P_i \rightarrow P_j : E_{ssk_i}(k_i)$$

to transfer $E_{ssk_i}(k_i)$, encrypted with P_i 's session key ssk_i , to every party P_j .

Upon receipt of $E_{ssk_i}(k_i)$, P_j decrypts it with key ssk_i received at stage 1 to recover k_i . If $o(k_i) = ok_i$, P_j is assured that k_i is P_i 's correct document key.

When every party has received all the correct document keys needed, the exchange is completed successfully, so stage 4 below will not be performed.

Stage 4: Key Recovery

In case a party P_d misbehaves by refusing to send its key k_d to another party P_i , or the communication from P_d to P_i has broken down, after the release of k_i , P_i can recover k_d with assistance of other parties. To perform the key recovery, P_i groups all the parties' acknowledgements into:

$$ACK_i = (ack_1, \dots, ack_n).$$

P_i then sends to every party P_j (including P_d) a request for the recovery of k_d , which consists of P_d 's subscript d and ACK_i , ACK_i is used as evidence to justify the validity of the request. P_i sends out its request by the transaction below:

$$P_i \rightarrow P_j : d, ACK_i.$$

If P_j has not received all the parties' acknowledgements before P_i 's request arrives, P_j performs verification 2. When the verification is positive or P_j is already in the possession of all the correct acknowledgements, P_j sends $E_{pk_i}(sec_{d,j})$ (i.e. the encryption of secret item $sec_{d,j}$ with P_i 's public key pk_i) to P_i . Alternatively, P_j can send $E_{pk_i}(k_d)$ to P_i if P_j has obtained k_d .

As assumed earlier in this section, $n - m (> m)$ parties are honest. This ensures that P_i can receive responses from at least m honest parties possibly by repeating the request in case some communication channels break down temporarily. Here we assume that a communication channel between any two parties may break down temporarily, but any message transmitted (repeatedly if necessary) over the channel can eventually go through within the valid duration of the current document exchange.

After P_i has received w ($e \leq w < n$) secret items sec_{d,r_l} ($1 \leq l \leq w$). P_i can carry out verification 1 to check the correctness of each of these w items. If e out of the w items are correct, P_i can use them together with its possessed secret item $sec_{d,i} = y_d(v_{d,i})$ (see verification 1 for $v_{d,i}$) to solve the following equations for $k_d = a_{d,0}$ where for ease of presentation let the e correct secret items be $sec_{d,r_1} = y_d(v_{d,r_1}), \dots, sec_{d,r_e} = y_d(v_{d,r_e})$:

$$\begin{cases} y_d(v_{d,r_1}) = (k_d + \dots + a_{d,l} \times v_{d,r_1}^l + \dots + a_{d,e} \times v_{d,r_1}^e) \bmod q \\ \dots \\ y_d(v_{d,r_e}) = (k_d + \dots + a_{d,l} \times v_{d,r_e}^l + \dots + a_{d,e} \times v_{d,r_e}^e) \bmod q \\ y_d(v_{d,i}) = (k_d + \dots + a_{d,l} \times v_{d,i}^l + \dots + a_{d,e} \times v_{d,i}^e) \bmod q \end{cases}$$

In case a party P_b has not got all the acknowledgements, P_b can still request other parties for the recovery of a key. If a party P_j holds all the acknowledgements, P_j responds to the request just as described above, and also forwards all the acknowledgements to P_b . Otherwise, P_j informs P_b of unsuccessful key recovery.

Additionally a deadline can be added to allow the exchange to be completed before a specified time agreed by all the parties. This can prevent some parties from deliberately delaying the completion of the exchange so as to gain some advantage over others (Zhang et al., 1999).

Note that the efficiency of message exchanges among the parties can be improved using a ring model described in (Khill et al., 2001).

4. Key Exchange with an Online STNP

The approach for the mode (c) of exchange presented in Section 3 can also be applied to deal with the case of $m \geq \lfloor (n+1)/2 \rfloor$ (i.e. no less than half of the parties may be dishonest) by employing online STNP P_{on} ($= P_{n+1}$, as assumed in Section 2.2) agreed by all the parties, i.e. the mode (a) of exchange stated in Section 1. The role of P_{on} is to assist the parties in the fair exchange of their document keys. In the following, we will only show the differences of mode (a) from mode (c) in relation to the application of the approach:

- For stage 1 described in Section 3, by letting $e = 1$, the function of each party P_i ($1 \leq i \leq n$) becomes:

$$y_i(x) = (a_{i,0} + a_{i,1} \times x) \bmod q.$$

Here, $a_{i,0}$ and $a_{i,1}$ are the same as those defined in Section 3. P_i then uses $y_i(x)$ to generate its secret items:

$$sec_{i,j} = y_i(\lfloor j/(n+1) \rfloor + 1), \text{ for every } j (1 \leq j \leq n+1).$$

This means that P_i produces the same secret item $y_i(1)$ for every party P_j ($1 \leq j \leq n$), and $y_i(2)$ only for P_{on} (or P_{n+1}). This ensures that without any help from P_{on} , no parties will be able to compute P_i 's document key k_i even if they collude.

In this stage, P_{on} only receives secret items from the other parties and does not send any item to them as P_{on} has no document key to exchange.

When a party P_j receives $sec_{i,j}$ and VER_i from another party P_i , P_j defines $pro_{i,0} = ok_i$ and $v_{i,j} = \lfloor j/(n+1) \rfloor + 1$, and performs verification 1, as described in Section 3.

- In stage 2, each party P_i ($1 \leq i \leq n$) sends its acknowledgement ack_i to P_{on} in addition to all the other parties, and P_{on} also sends an acknowledgement ack_{on} to P_i , which is produced in the same way as for ack_i .
- In stage 3, each party P_i ($1 \leq i \leq n$) does not release its key k_i to P_{on} , and P_{on} sends out nothing.
- In stage 4, a party, which needs to recover another party's key, sends a key recovery request only to P_{on} .

The approach presented in Section 3 can also be easily applied to permit more than one online STNP to participate in key exchange so as to achieve better security and reliability (Zhang et al., 1999).

5. Key Exchange with an Offline STNP

In the case of $m \geq \lfloor (n+1)/2 \rfloor$, offline STNP P_{off} ($= P_0$, as assumed in Section 2.2) agreed by all the n parties can also be employed to assist them in the exchange of their document keys, i.e. the mode (b) of exchange stated in Section 1. The main difference from the mode (a) of exchange described in Section 4 is that P_{off} does not participate in the normal exchange process, i.e. stages 1 - 3 presented in Section 3, and it is invoked only when the recovery of a key is required, i.e. stage 4.

The approach for the mode (c) of exchange presented in Section 3 can also be applied to mode (b). However, as P_{off} does not take part in the normal exchange process, each party P_i ($1 \leq i \leq n$) must be assured without any online involvement of P_{off} that P_i will be able to obtain the document key of any party P_j ($1 \leq j \leq n$, and $i \neq j$), before P_i can release its key k_i . One way to produce this different type of assurance is to let P_j generate a

verifiable and recoverable encryption of its key k_j , i.e. (SEC_j, VER_j) , in such a way that P_i can have the whole encrypted key without being able to decrypt it to get k_j , and P_i can verify, without any other party's help, that P_{off} can recover k_j from (SEC_j, VER_j) for P_i . Once P_i has obtained such an encrypted key (SEC_j, VER_j) from every party P_j , it is secure for P_i to release its key k_i . Only when P_i has failed to receive another party P_d 's key k_d , P_i requests P_{off} for the recovery of k_d .

In the following, we will only present the differences of mode (b) from mode (c) based on the above discussion with regard to the application of the approach given in Section 3:

- In stage 1 described in Section 3, by letting $e = 0$, the function of each party P_i is simplified into:

$$y_i(x) = a_{i,0} \text{ mod } q.$$

Here, we define $a_{i,0}$ as follows, where items rk_i , $si_{i,off}$ and $m_{i,off}$ were specified in Section 2.2:

$$a_{i,0} = (rk_i + si_{i,off} \times h(m_{i,off})) \text{ mod } q.$$

$y_i(x)$ indicates that P_i produces the same secret item $sec_{i,j} = y_i(j) = a_{i,0}$ for every party P_j , and that VER_i is empty. Upon receipt of $sec_{i,j}$, P_j carries out verification 1 to check its correctness by setting items $pro_{i,0} = (ork_i \times pi_{i,off}^{h(m_{i,off})}) \text{ mod } p$ (i.e. $pro_{i,0} = (g^{rk_i} \times g^{si_{i,off} \times h(m_{i,off})}) \text{ mod } p = g^{a_{i,0}} \text{ mod } p = o(y_i(j))$) and $v_{i,j} = j$ (see Section 3), where $ork_i = o(rk_i)$ and $pi_{i,off} = o(si_{i,off})$ were defined in Section 2.2.

- In stage 2, verification 2 (b) specified in Section 3 ensures that every party has possessed the same group of secret items. This will convince P_{off} that any party can recover any document key with the assistance of P_{off} if necessary, and authorise P_{off} to perform key recovery only related to these secret items so as to prevent a dishonest party from requesting P_{off} to recover a key irrelevant to the current session of exchange.
- In stage 4, a party P_i , which wants to recover another party P_d 's key k_d , first groups all the parties' secret items, public keys, and acknowledgements into:

$$SEC_i = (sec_{1,i}, \dots, sec_{n,i}), PK_i = (pk_1, \dots, pk_n), \text{ and } ACK_i = (ack_1, \dots, ack_n).$$

P_i then only requests P_{off} for the key recovery by executing the transaction below:

$$P_i \rightarrow P_{off} : d, m_d, SEC_i, PK_i, ACK_i.$$

Having received P_i 's request, P_{off} conducts the following verification:

Verification 3: (a) Check the correctness of all the n acknowledgements in ACK_i by verification 2.

(b) Calculate $hsec_{off} = h(sec_{1,i}, \dots, sec_{n,i})$ and $hpk_{off} = h(pk_1, \dots, pk_n)$ using the n secret items in SEC_i and n public keys in PK_i , and confirm that $hsec_{off}$ and hpk_{off} are the same as $hsec_j$ and hpk_j in any acknowledgement ack_j in ACK_i , respectively.

(c) Confirm that both P_i 's public key pk_i and P_d 's public key pk_d are in PK_i .

Verification 3 (b) ensures that every party involved in the exchange has an acknowledgement in ACK_i and received the same set of secret items, so any party can request P_{off} for the recovery of a key. Verification 3 (c) makes sure that both P_i and P_d are valid parties participating in the exchange.

If verification 3 is positive, P_{off} computes:

$$si_{d,off} = h(sk_{off}, pk_d, rn_{d,off}) \bmod q, \text{ and } rk_d = (sec_{d,i} - si_{d,off} \times h(m_{d,off})) \bmod q.$$

P_{off} sends $E_{pk_i}(k_d)$ with $k_d = h(rk_d)$ to P_i , and saves session number sn and ACK_i to serve key recovery requests from other parties. P_i then decrypts $E_{pk_i}(k_d)$ received with private key sk_i to obtain k_d , and checks that $o(k_d) = ok_d$ as described in Section 3. Note that P_{off} does not send rk_d to P_i , so that P_i is unable to compute secret item $si_{d,off}$ from $sec_{d,i}$. This indicates that P_d can reuse $si_{d,off}$ for the exchange of its other document keys. This explains why rk_d was used to define $k_d = h(rk_d)$ in Section 2.2.

In case a party P_j has not got all the n acknowledgements, P_j can still request P_{off} for the recovery of a party P_d 's key k_d by sending only sn , d , m_d , SEC_j and PK_j to P_{off} . If P_{off} has ACK_i saved for session sn , P_{off} performs the key recovery as described above, and forwards ACK_i to P_j . Otherwise, P_{off} informs P_j of unsuccessful key recovery, and asks P_j to try later as a party with all the acknowledgements may send them to P_{off} for key recovery later.

6. A Unified Approach to Fair Document Exchange

We now combine the three approaches given in Sections 3, 4 and 5 to produce a unified approach to fair document exchange among the n parties. This unified approach will be presented using two protocols. The first protocol corresponds to the first three stages of the exchange process described in Section 3, namely normal cases of exchange without key recovery. The second protocol is for key recovery, i.e. the last stage of the exchange process, in case a normal exchange has failed.

The first protocol is presented in Table 1 together with the definitions of all the items used. Transaction E_1 in Table 1 represents the stage 1 of the exchange process, i.e. exchange of verifiable and recoverable encrypted keys. E_1 means that every party P_i sends its items to each party P_j . $items_{i,j}$ in E_1 is defined as the concatenation of the two items. $items_{i,j}$ implies that P_j receives $E_{pk_j}(sec_{i,j}, ssk_i, VER_i)$ and $E_{ssk_i}(E_{k_i}(doc_i))$ if P_j is not P_{on} ($= P_{n+1}$), and P_j gets $E_{pk_j}(sec_{i,j}, 0, VER_i)$ and 0 otherwise, i.e. P_{on} does not receive session key ssk_i and encrypted document $E_{ssk_i}(E_{k_i}(doc_i))$. This can prevent compromised P_{on} from gaining doc_i , as will be discussed in detail in Section 7. $sec_{i,j}$ and VER_i in $items_{i,j}$ are specified based on the three modes of exchange described in Sections 3, 4 and 5. The definition of $sec_{i,j}$ in Table 1 indicates that $sec_{i,j} = y_i(\lfloor j/hi \rfloor + 1)$ if P_{on} is used (i.e. $hi = n + 1$), and $sec_{i,j} = y_i(j)$ otherwise (i.e. $hi = n$). Similarly, the definition of $a_{i,0}$ implies that $a_{i,0} = (rk_i + si_{i,off} \times h(m_{i,off})) \bmod q$ if P_{off} ($= P_0$) is used (i.e. $lo = 0$), and $a_{i,0} = k_i \bmod q$ otherwise (i.e. $lo = 1$).

$sign_i$ in E_1 is P_i 's signature on sn and VER_i . The reason for excluding $sec_{i,j}$ and $E_{k_i}(doc_i)$ from $sign_i$ is that their correctness is verifiable based on VER_i and hed_i in CON_i (see Section 2.2), respectively. This exclusion allows P_i to use the same signature to send its messages to the other parties, which may include different secret items particularly when the exchange is operated in mode (c). This reduces the amount of P_i 's computation.

Having received P_i 's message, P_j performs verification 1 defined in Section 3. In addition, P_j checks that $sign_i$ is correct, i.e. $h(sn, VER_i)$ calculated by P_j is the same as that in $sign_i$ and its time stamp ts_i is fresh. If P_j is not P_{on} , P_j also decrypts $E_{ssk_i}(E_{k_i}(doc_i))$ with session key ssk_i received to recover $E_{k_i}(doc_i)$, and confirms that $h(E_{k_i}(doc_i))$ is equal to hed_i in CON_i .

Transactions E_2 and E_3 signify the stages 2 and 3 of the exchange process, i.e. the exchange of acknowledgements and exchange of keys, respectively. Their details have been described in the previous sections. Note that P_i sends its key to P_j by E_3 only if P_j is not P_{on} , as P_{on} is not allowed to view P_i 's document.

Protocol 1	
E_1 .	$P_i \rightarrow P_j : sn, items_{i,j}, sign_i$ if P_i is not P_{on}
E_2 .	$P_i \rightarrow P_j : sn, ack_i$
E_3 .	$P_i \rightarrow P_j : sn, E_{ssk_i}(k_i)$ if neither of P_i and P_j is P_{on}
Item	Definition
n, m	Numbers of parties involved in the exchange and possible dishonest parties, respectively
(lo, hi, e)	Indicator of the current exchange mode, which is defined as follows: For mode (a) (i.e. use of online STNP P_{on} or P_{n+1}), $lo = 1$, $hi = n + 1$, and $e = 1$ For mode (b) (i.e. use of offline STNP P_{off} or P_0), $lo = 0$, $hi = n$, and $e = 0$ For mode (c) (i.e. without use of any TNP/STNP), $lo = 1$, $hi = n$, and $e = m$
i, j	$1 \leq i, j \leq hi$, and $i \neq j$
sn, ssk_i	Session number, and P_i 's symmetric session key
doc_i, k_i, rk_i	P_i 's document, its document key, and random number with $k_i = h(rk_i)$, respectively
$pk_i, CER_{i,off}$	P_i 's public key, and its certificate associated with $rn_{i,off}$ and $si_{i,off}$ defined in Section 2.2
$a_{i,0}$	$= (k_i \times lo + (rk_i + si_{i,off} \times h(rn_{i,off})) \times (1 - lo)) \bmod q$; P_i 's coefficient
$a_{i,1}, \dots, a_{i,e}$	Different coefficients $a_{i,l} < q$ ($1 \leq l \leq e$) chosen randomly by P_i
$y_i(x)$	$= (a_{i,0} + \dots + a_{i,l} \times x^l + \dots + a_{i,e} \times x^e) \bmod q$; P_i 's function
$sec_{i,j}$	$= y_i(j + \lfloor hi/(n+1) \rfloor \times (\lfloor j/hi \rfloor + 1 - j))$; secret item computed by P_i for P_j
VER_i	$= (o(a_{i,1}), \dots, o(a_{i,e}))$; P_i 's proof item group
$items_{i,j}$	$= E_{pk_j}(sec_{i,j}, ssk_i \times (1 - \lfloor j/(n+1) \rfloor), VER_i), E_{ssk_i}(E_{k_i}(doc_i)) \times (1 - \lfloor j/(n+1) \rfloor)$; items sent by P_i to P_j
$sign_i$	$= E_{sk_i}(ts_i, h(sn, VER_i))$; P_i 's signature with time stamp ts_i
$hsec_i$	$= h(sec_{i,1}, \dots, sec_{i,n})$; hash value of the secret items possessed by P_i
hpk_i	$= h(pk_1, \dots, pk_n)$; P_i 's hash value of public keys
$hver_i$	$= h(VER_1, \dots, VER_n)$; P_i 's hash value of all the proof item groups
ack_i	$= E_{sk_i}(sn, hsec_i, hpk_i, hver_i)$; P_i 's acknowledgement

Table 1: Normal document exchange protocol

The second protocol is presented in Table 2 together with the definitions of all the additional items used. Transaction R_1 in Table 2 represents a key recovery request from a party P_i to every party P_r . The definition of the subscript r of P_r reflects the three modes of exchange: (a) $n + 1 \leq r \leq n + 1$ when P_{on} is used (i.e. $lo = 1$ and $hi = n + 1$), namely P_i only requests P_{on} for the key recovery; (b) $0 \leq r \leq 0$ when P_{off} is used (i.e. $lo = 0$ and $hi = n$); and (c) $1 \leq r \leq n$ otherwise (i.e. $lo = 1$ and $hi = n$). S_i in R_1 is a collection of the subscripts of all the keys which P_i wants P_r to recover. RN_i shows the two cases: (1) $RN_i = (rn_{s_1,off}, \dots, rn_{s_w,off})$ if P_{off} is used (i.e. $lo = 0$), as P_{off} or P_r needs the items in RN_i to perform offline key recovery; and (2) $RN_i = (0, \dots, 0)$ otherwise (i.e. $lo = 1$), because P_r does not use any item $rn_{sl,off}$ for online key recovery. Similar explanations are given to SEC_i and PK_i . ACK_i represents the group of all the n acknowledgements if P_i possesses all of them, and an empty group otherwise.

Protocol 2

$R_1. P_i \rightarrow P_r : sn, S_i, RN_i, SEC_i, PK_i, ACK_i, sign_{i,r}$

$R_2. P_r \rightarrow P_i : sn, res_r, sign_{r,i}$

Item	Definition
i, r	$1 \leq i \leq n$, and $(lo \times (1 + n \times \lfloor hi/(n+1) \rfloor)) \leq r \leq (lo \times (n + \lfloor hi/(n+1) \rfloor))$
S_i	$= (s_1, \dots, s_w)$; group of the subscripts for all the keys to be recovered, where for any s_l in S_i , $1 \leq s_l \leq n$
RN_i	$= (rn_{s_1,off} \times (1 - lo), \dots, rn_{s_w,off} \times (1 - lo))$; group of items in the certificates related to subscripts in S_i
SEC_i	$= (sec_{1,i} \times (1 - lo), \dots, sec_{n,i} \times (1 - lo))$; group of all the secret items
PK_i	$= (pk_1 \times (1 - lo), \dots, pk_n \times (1 - lo))$; group of all the public keys
ACK_i	$= (ack_1, \dots, ack_n)$ or $()$; group of all the acknowledgements defined in Table 1 or empty group
$sign_{i,r}$	$= E_{sk_i}(ts'_i, h(sn, S_i, RN_i, SEC_i, PK_i, ACK_i))$; P_i 's signature with time stamp ts'_i
res_r	$= E_{pk_i}(k_{s_1} \times (1 - lo) + sec_{s_1,r} \times lo, \dots, k_{s_w} \times (1 - lo) + sec_{s_w,r} \times lo)$ or 0; recovered keys, secret items, or zero
$sign_{r,i}$	$= E_{sk_r}(ts_r, h(sn, res_r))$; P_r 's signature with time stamp ts_r

Table 2: Key recovery protocol

Upon receipt of P_i 's request, P_r checks the correctness of signature $sign_{i,r}$, and responds to the request accordingly as defined in the previous sections. If P_r is P_{off} (i.e. $lo = 0$) and has recovered a key k_{s_l} for every subscript s_l in S_i , P_r encrypts all the recovered keys with P_i 's public key pk_i to produce $res_r = E_{pk_i}(k_{s_1}, \dots, k_{s_w})$. When P_r is not P_{off} (i.e. $lo = 1$) but would like to respond to P_i 's request with secret item $sec_{s_l,r}$ for every subscript s_l in S_i , P_r encrypts all the secret items with pk_i to generate $res_r = E_{pk_i}(sec_{s_1,r}, \dots, sec_{s_w,r})$. Otherwise, P_r cannot recover any key or secret item requested, so P_r defines $res_r = 0$.

Having received P_r 's response, P_i checks the correctness of signature $sign_{r,i}$. If $sign_{r,i}$ is correct, then P_i conducts the corresponding activities described in the previous sections.

7. Fairness Analysis

We now analyse the fairness of the unified document exchange approach consisting of the two protocols defined in Tables 1 and 2. The analysis is presented in terms of the three modes of the protocols, respectively.

- *Mode (c) of the protocols (i.e. no STNP is used)*: The fairness of this mode (i.e. the approach described in Section 3) is demonstrated by the following points:
 - (1) Once a party P_i has got all the n correct acknowledgements in its possession, P_i can obtain any other party P_j 's key k_j either directly from another party (e.g. P_j) or by solving the corresponding equations given in Section 3. P_j cannot dishonestly stop P_i obtaining k_j even if P_j colludes with other dishonest parties, as there are at least $m + 1$ honest parties able to jointly compute k_j based on their possessed secret items received from P_j .
 - (2) When a party P_i does not possess all the n correct acknowledgements, P_i cannot release its key k_i . As a result, if another party P_j holds all the n acknowledgements, P_j has to send to other parties a request for the recovery of k_i together with all the acknowledgements. Such a request is also repeated by other parties. This makes it highly likely for P_i to eventually receive all the n acknowledgements and thus be

able to obtain all the keys. Of course, P_i can request other parties for key recovery even if P_i does not possess all the acknowledgements, as described in Section 3.

The above measure can also ensure that even if a party P_j misbehaves by collecting the acknowledgements from all the other parties without sending its acknowledgement to them, P_j cannot benefit from its misbehaviour.

- (3) If no party has held all the n correct acknowledgements, no key or possessed secrets will be released by any party, so the exchange is unsuccessful.
- (4) Due to the one-way feature of $o(x)$, it is hard for a party P_i to derive k_j and $a_{j,l}$ directly from ok_j and $pro_{j,l}$ received from another party P_j . It is also hard for P_i to compute coefficients $a_{j,l}$ directly from $sec_{j,i}$ (Gong, 1994). Even when m dishonest parties collude in an attempt to illegitimately compute k_j or $a_{j,l}$, they can get a maximum of m secret items of P_j , which are insufficient for the computation of k_j or $a_{j,l}$. It is thus hard for P_i to obtain P_j 's key k_j illegitimately.
- (5) If a party P_i wants to cheat others at stage 1 (i.e. transaction E_1) by using an incorrect key k'_i to generate its secret items $sec'_{i,j}$, or by sending out some incorrect secret items $sec'_{i,j}$ or an incorrect proof item group VER'_i , to prevent other parties from recovering its key k_i , then P_i will not be successful. This is because verification 1 is based on $ok_i (= o(k_i))$ in CON_i certified by an authority as assumed in Section 2.2, and it is hard for P_i to find a different key $k'_i (\neq k_i)$ such that $o(k'_i) = ok_i$ (Schneier, 1996). Thus P_i 's misbehaviour will result in a failure of verification 1 or 2. This means that at least the honest parties will not release their keys or any possessed secret items, so P_i is unable to obtain a sufficient number of secret items for the computation of a key. Thus P_i gains no benefit from its misbehaviour.
- (6) Suppose that a party P_i sends an incorrect acknowledgement to others parties at stage 2 (i.e. transaction E_2). This will lead to a failure of verification 2, so no party will release its key and possessed secret items. Additionally, if P_i uses an incorrect number of acknowledgements or incorrect acknowledgements for key recovery at stage 4 (i.e. transaction R_1), and no party holds all the n correct acknowledgements, then no party will send any secret item or key requested to P_i .
- (7) Where a party P_i releases an incorrect key k'_i to another party P_j at stage 3 (i.e. transaction E_3), P_j can easily detect this by checking whether or not $o(k'_i)$ is equal to ok_i in CON_i . If necessary, P_j can request other parties for the recovery of P_i 's correct key k_i .
- (8) A party P_i can terminate the exchange if P_i has not sent out its acknowledgement ack_i . Otherwise, P_i 's termination leads to two cases. First, another party has obtained all the acknowledgements and thus wants to complete the exchange. In this case, P_i 's termination has no effect. Secondly, all the parties agree to terminate the exchange, or no party has got all the acknowledgements, so the exchange can be terminated successfully. It is evident that no party can benefit from the termination.
- (9) An outsider $P_o (\neq P_i \text{ for any } i (1 \leq i \leq n))$ cannot obtain any party P_i 's document doc_i . This is because doc_i is sent to other parties in the encrypted form of $E_{ssk_i}(E_{k_i}(doc_i))$, and keys ssk_i and k_i as well as the secret items associated with k_i are also transferred to other parties in encrypted forms. Without knowing

both ssk_i and k_i , P_o cannot decrypt $E_{ssk_i}(E_{k_i}(doc_i))$ to get doc_i even if P_o could intercept all the messages sent by the protocols. Here we assume that P_o does not know P_i 's private key sk_i and does not collude with any party P_j ($1 \leq j \leq n$).

The above points demonstrate that either each party can obtain all the other parties' document keys (and then their documents), or no party can obtain any other party's document key. Therefore the mode (c) can satisfy the strong fairness requirement stated in Section 2.3.

- *Mode (a) of the protocols (i.e. P_{on} is used)*: The fairness of this mode (i.e. the approach given in Section 4) is justified as follows:
 - (i) As mode (a) is offered as a special instance of mode (c) by employing online STNP P_{on} , it directly inherits the strong fairness of mode (c) which has been demonstrated above.
 - (ii) Additionally P_{on} only helps to achieve fair key exchange with no need to view any documents exchanged. In fact, as described in Section 6, each party P_i sends its document in the form of $E_{ssk_i}(E_{k_i}(doc_i))$ to every other party except P_{on} , and does not transfer session key ssk_i to P_{on} . Moreover, P_{on} only holds one of the two secret items generated from P_i 's key k_i , as described in Section 4. This means that even if P_{on} manages to intercept all the message transferred by the protocols, P_{on} can gain neither ssk_i nor k_i for the decryption of $E_{ssk_i}(E_{k_i}(doc_i))$, namely P_{on} cannot obtain doc_i , as long as P_{on} does not conspire with any party P_j . This shows that P_{on} can be a STNP.
- *Mode (b) of the protocols (i.e. P_{off} is used)*: The fairness of this mode (i.e. the approach presented in Section 5) is analysed by the following points:
 - (a) When a party P_i has got all the n correct acknowledgements in its possession, P_i can obtain any party P_j 's key k_j either directly from P_j or recovered by P_{off} . In the case of key recovery, the success of verification 2 in Section 3 performed by P_i implies that P_i has received a correct item $sec_{j,i}$ from every other party P_j . This ensures that any correct request of P_i for key recovery can pass verification 3 in Section 5 conducted by P_{off} , so P_{off} can recover any requested key for P_i .
 - (b) The analysis of modes (c) and (a) given above can also be applied to demonstrate the other fairness aspects of mode (b).

From the above analysis, we can conclude that the unified approach presented in Section 6 can guarantee the strong fairness defined in Section 2.3 under any one of its three modes.

8. Comparison with Related Work

In this section we compare the three modes of the unified approach described in Section 6 with related work, respectively.

- *Mode (a) of the approach (i.e. P_{on} is used)*: Existing fair exchange approaches relevant to the method presented in Section 4 (and 3) are those given in (Franklin and Reiter, 1997; Franklin and Tsudik, 1998). These approaches allow an online STNP to verify the correctness of document keys without actually seeing them, and to forward the verified keys to the corresponding parties, so as to achieve strong fairness. The

main difference from our method is that the STNP used by the approaches does not perform key recovery. As a result, the STNP has no need to see any document key, but it needs to execute different operations from the other parties involved in the exchange. In our method, online STNP P_{on} helps a party to recover a key upon its request, but P_{on} still cannot see any document key because it possesses only one of the two secret items for a key, which is insufficient for P_{on} to compute the key. Additionally, P_{on} can operate just like an ordinary party involved in the exchange, which makes the method simpler to implement and easier to incorporate more than one online STNP.

Other online TNP based approaches, e.g. (Vogt et al., 1999; Zhang et al., 1999), do not provide the verifiability of a document key, so they cannot really guarantee strong fairness.

- *Mode (b) of the approach (i.e. P_{off} is used):* A number of fair exchange approaches (Asokan et al., 2000 – Bao et al., 1999; Chen, 1998) have been proposed to achieve strong fairness based on an offline TNP or STNP. These approaches are mainly concerned with the exchange of signatures, and provide verifiable and recoverable signature encryption for offline signature recovery. This makes their design goal different from that of our approach which is concerned mainly with the fair exchange of document decryption keys. It is thus inappropriate to compare our approach with them.

Several approaches for the exchange of documents (e.g. digital products and payments) with the ability of offline key recovery (Ray and Ray, 2000; Ray and Ray, 2001) have also been developed to achieve strong fairness. However, these approaches have two main weaknesses. First, they need an online TNP to store an encrypted version of each exchanged document and the associated decryption key, and to be responsible for the distribution of the encrypted document. The TNP also helps to recover a decryption key for a party (e.g. sending the corresponding key stored to the party) in case a normal exchange has failed. Although the TNP is offline for the key recovery, it needs to be online for each exchange as a party involved in the exchange needs to acquire an encrypted document from the TNP. Thus, they are not really offline TNP based approaches, and also the TNP must be fully trusted and well protected as it can read all the documents stored. In our approach, P_{off} is offline and semi-trusted, so it is more cost-effective, secure and easier to implement.

Secondly, the main verification (i.e. checking the correctness of a document without being able to read it) used by these approaches is based on exchanged documents themselves rather than their decryption keys, which is very inefficient as a document is usually much larger than a key. The main verification (i.e. checking the correctness of a document key without seeing it) used by our approach is based on document keys. This makes our verification more efficient and independent of the size of a document.

Other fair exchange approaches based on an offline TNP, e.g. (Asokan et al., 1996; Asokan et al., 1997), cannot really guarantee strong fairness.

- *Mode (c) of the approach (i.e. no STNP is used):* The only relevant approach without use of any TNP/STNP is given in (Zhang et al., 1999). In fact, mode (c) is an extension to the work described in (Zhang et al., 1999). However, the approach presented in (Zhang et al., 1999) does not provide the verifiability of a document key, so it cannot assure strong fairness that is guaranteed by mode (c) presented in Section 3.

Other fair exchange approaches without use of any TNP/STNP are based on gradual secret releasing, e.g. (Okamoto and Ohta, 1994), and probabilistic methods, e.g. (Rabin, 1983), which are not cost-effective (Bao et al., 1998) and also unsuitable for an exchange involving multiple parties.

In summary, the mode (a) of our unified approach has some comparable existing work but offers an easier way of incorporating multiple online STNPs, the mode (b) is more efficient than other related approaches, and the mode (c) has no other similar approaches with strong fairness to compare. Additionally our unified approach is the first one to offer a unified solution to fair document exchange involving any number of parties, which can operate with an online or offline STNP or without any STNP while guaranteeing strong fairness.

9. Conclusions and Future Work

We have presented approaches to fair document exchange with the support of no STNP, an online STNP, and an offline STNP, respectively, in a unified manner. These approaches adopt the concept of verifiable and recoverable key encryption to achieve strong fairness. We have then combined the approaches to produce the first unified approach to fair document exchange involving any number of parties, which can operate under various situations while guaranteeing strong fairness. This unified approach will enable us to develop a flexible, general and secure system for fair document exchange. The fairness analysis of the unified approach has demonstrated that it can satisfy the strong fairness requirement stated in Section 2.3. The comparison with related work has confirmed the novel features of the approach. For the future work, we will apply the unified approach to develop a fair document exchange system, and demonstrate its efficacy and applicability.

Acknowledgements

This research is supported in part by the FIDES (Fair Integrated Data Exchange Services) project funded jointly by the UK Engineering and Physical Sciences Research Council (EPSRC) and Department of Trade and Industry (DTI). The project reference is LINK, GR/R55177.

We would like to thank the anonymous referees for their most constructive comments and suggestions.

References

- Asokan, N., Schunter, M., Waidner, M., 1996. Optimistic protocols for multi-party fair exchange. IBM Research Report (Zurich), RZ2892 (#90840).
- Asokan, N., Schunter, M., Waidner, M., 1997. Optimistic protocols for fair exchange. In: Proceedings ACM Conference on Computer and Communications Security, Zurich, Switzerland, pp. 6-17.
- Asokan, N., Shoup, V., Waidner, M., 1998. Asynchronous protocols for optimistic fair exchange. In: Proceedings IEEE Symposium on Security and Privacy, Oakland, CA, pp. 86-100.
- Asokan, N., Shoup, V., Waidner, M., 2000. Optimistic fair exchange of digital signatures. IEEE Journal on Selected Areas in Communications, 18, 593-610.
- Ateniese, G., 1999. Efficient verifiable encryption (and fair exchange) of digital signatures. In: Proceedings ACM Conference on Computer and Communications Security, Singapore, pp. 138-146.

- Bao, F., Deng, R., Mao, W., 1998. Efficient and practical fair exchange protocols with off-line TTP. In: Proceedings IEEE Symposium on Security and Privacy, Oakland, CA, pp. 77-85.
- Bao, F., Deng, R., 1999. An efficient fair exchange protocol with an off-line semi-trusted third party. In: Proceedings International Workshop on Cryptographic Techniques and E-Commerce, City University, Hong Kong, pp. 37-47.
- Bao, F., Deng, R., Nguyen, K.Q., Varadharajan, V., 1999. Multi-party fair exchange with an off-line trusted neutral party. In: Proceedings International Workshop on Database and Expert Systems Applications, Los Alamitos, CA, USA, pp. 858-862.
- Chen, L., 1998. Efficient fair exchange with verifiable confirmation of signatures. In: Proceedings Advances in Cryptology - ASIACRYPT '98, Springer-Verlag, Berlin, Germany, pp. 286-99.
- Federal Information Processing Standards Publication 180-1, 1995. Secure hash standard. National Institute of Standards and Technology.
- Franklin, M., Reiter, M., 1997. Fair exchange with a semi-trusted third party. In: Proceedings ACM Conference on Computer and Communications Security, Zurich, Switzerland, pp. 1-5.
- Franklin, M., Tsudik, G., 1998. Secure group barter: multi-party fair exchange with semi-trusted neutral parties. In: Proceedings Financial Cryptography (Lecture Notes in Computer Science 146, Springer-Verlag), Anguilla, British West Indies, pp. 90-102.
- Gong, L., 1994. New protocols for third-party-based authentication and secure broadcast. In: Proceedings ACM Conference on Computer and Communication Security, pp. 184-192.
- Khill, I., Kim, J., Han, I., Ryou, J., 2001. Multi-party fair exchange protocol using ring architecture model. *Computers & Security*, 20(5), 422-439.
- Okamoto, T., Ohta, K., 1994. How to simultaneously exchange secrets by general assumptions. In: Proceedings ACM Conference on Computer and Communication Security, New York, USA, pp. 184-192.
- Rabin, M.O., 1983. Transaction protection by beacons. *Journal of Computer and System Science*, 27, 256-267.
- Ray, I., Ray, I., Narasimhamurthi, N., 2000. A fair exchange e-commerce protocol with automated dispute resolution. In: Proceedings Annual IFIP WG 11.3 Working Conference on Database Security, Schoorl, Netherlands.
- Ray, I., Ray, I., 2000. An optimistic fair exchange e-commerce protocol with automated dispute resolution. In: Proceedings International Conference on Electronic Commerce and Web Technologies, EC-Web 2000, Lecture Notes in Computer Science Vol.1875, Springer-Verlag, Berlin, Germany, pp. 84-93.
- Ray, I., Ray, I., 2001. An anonymous fair exchange e-commerce protocol. In: Proceedings International Workshop on Internet Computing and E-Commerce, San Francisco, CA, pp. 1790-1797.
- Schneier, B., 1996. Applied Cryptography. John Wiley & Sons.
- Vogt, H., Pagnia, H., Gartner, F.C., 1999. Modular fair exchange protocols for electronic commerce. In: Proceedings Annual Computer Security Applications Conference, IEEE Computer Society, Los Alamitos, CA, USA, pp. 3-11.
- Zhang, N., Shi, Q., Merabti, M., 1999. A flexible approach to secure and fair document exchange. *The Computer Journal*, 42 (7), 569-581.

Ning Zhang is a Lecturer in Computer and Communication Networks in the Department of Computer Science at the University of Manchester in the UK. She received her PhD from the University of Kent at Canterbury in the UK in 1991. Between 1991 and 2000, She first worked for the University of York and then Manchester Metropolitan University in the UK before she joined the University of Manchester in 2000. Her main research interests include the design of protocols and architectures for secure communications, networks, and e-commerce. She is currently leading a research team working on a number of research projects on security, which are funded by various sources including the Engineering and Physical Sciences Research Council (EPSRC) that is the largest of the seven UK Research Councils.

Qi Shi is a Reader in the School of Computing & Mathematical Sciences at Liverpool John Moores University in the UK. He received his PhD in Computing from Dalian University of Technology, Dalian, People's Republic of China, in 1989. Between 1990 and 1994 he worked on a research project for the Department of Computer Science at the University of York in the UK. In 1994 he joined the School of Computing & Mathematical Sciences at Liverpool John Moores University. His current research interests include the development of formal security models, composable security theory, security protocols, mobile communication security, and network intrusion detection. He is currently supervising several research projects on network security, which are funded by various sources.

Madjid Merabti is a Professor, Director, and the Head of Research, School of Computing & Mathematical Sciences, Liverpool John Moores University, Liverpool, UK. He is a graduate of Lancaster University in the UK. He has over 10 years experience in conducting research and teaching in the areas of Distributed Multimedia Systems (Computer Networks, Operating Systems, and Computer Security). Prof. Merabti has over 50 publications in these areas, and he leads the Distributed Multimedia Systems Group which has a number of Government and Industry supported research projects in the areas of: Multimedia Networking, IP telephony, Differential Services Networks, Interactive TV, Multimedia Retrieval and Presentation, Mobile Networks Security, Intrusion Detection, and Security Architectures. He is collaborating with a number of international colleagues in the above areas.