20

How to Support Secure Electronic Contract Signing

N. Zhang
Distributed Systems Group
Dept. of Computing
Manchester Metropolitan University
Chester St., Manchester M1 5GD, UK
Email: N.Zhang@doc.mmu.ac.uk

Q. Shi
School of Computing and Mathematics Liverpool John Moores University
Byrom Building, Byrom Street
Liverpool L3 3AF, UK

Abstract

In an EDI environment, users may wish to do business over networks without physically being together. In such a situation, a contract signing protocol is required for business dealers to achieve simultaneous signing of a contract using communications. The aim of this paper is propose a protocol for contract signing. Our work is built upon the previous work done in the area by making use of their pros while eliminating their cons.

Key Words: *distributed EDI, contract signing, security protocols.*

1. INTRODUCTION

The increasing popularity of the Internet and its cost-effective accessibility to a wide range of users and information have been attracting more and more business communities to conduct their business activities electronically over networks. Such business activities are, in general terms, referred to as electronic commerce.

An important issue of electronic commerce is electronic contract exchange. Needlessly to say, adequate security services must be in place to support such exchange so as to protect contract partners from threats imposed by outsiders [6], or cheats committed by some insiders involved in the exchange.

For instance, consider that *Alice* and *Bob* negotiate over Email a contract *CONT* stating that *Alice* will sell her house to *Bob* at an agreed price. At the end of the negotiation, the two parties come to sign the contract. If *Alice* first signs *CONT* using a digital signature, and sends it to *Bob* through Email, then *Bob* may delay signing *CONT*, while looking around for a better deal (a better/cheaper house). During this period, *Alice* has already committed to the deal and is disadvantaged.

It is therefore essential to provide a protocol for secure contract signing services over networks. Ideally such a protocol should satisfy the following requirements:

Communications and Multimedia Security Vol. 3 S. Katsikas (Ed.)

(1) *Viability*: At the end of a proper execution of the protocol, all the parties involved should each obtain a copy of the singed contract with all the parties' valid signatures that are unforgeable, verifiable and non-repudiatable.

(2) *Fairness*: The protocol should be able to ensure simultaneous liability for all the parties. In other words, if all the parties execute the protocol properly, then they will all be committed to the contract at the end of the protocol execution. If, on the other hand, the protocol execution is terminated prematurely or finished inappropriately, then none of the parties will be bound to the contract.

(3) *Efficiency*: The protocol should reasonably balance the amount of communication and operation carried out by each party, so as to avoid any performance bottleneck.

(4) *Assurance*: The protocol should be free of the vulnerabilities that could be exploited by a dishonest insider to deceive others, or by an outsider to unlawfully gain the contract-related information or to maliciously jeopardise the validity of the contract signing process.

(5) *Tolerance*: The protocol should be capable of tolerating some failures caused by unreliable underlying networks. The implication of this is two-fold. First, it is impossible for a dishonest party to exploit any communication or system failure to impair the security of the protocol. Secondly, the protocol is still able to complete its execution despite of the occurrence of some failures.

(6) *Applicability*: The protocol should be effectively implementable in a heterogeneous and distributed network environment in which electronic contract signing is usually conducted. This means that the protocol should impose neither strong constraints on computational capacities of the parties nor tight requirements on their time synchronisation.

Previous work done in the area has proposed three approaches to solving the problem of contract signing. The first uses a trusted (third) party which is actively involved in the process of signing the contract [8, 11]. The trusted party's role is to hold the contract, to collect and verify the signatures from all the signing parties, and to finally distribute a copy of the signed contract to every party involved when all the signatures have been collected and proved to be valid. Should any dispute arise, the trusted party will act as a witness to resolve the dispute. Though the approach can achieve the requirement of applicability, and viability and fairness provided that the trusted party does not abuse the trust the parties have placed in it, it is weak in meeting the other requirements, namely, efficiency, assurance, and tolerance.

The second approach assumes the existence of a weak trusted party, e.g. a device or program, which either supplies a reliable source of randomness to assist the parties in a fair exchange of their signatures, or takes part in a retransmission of any message which has caused any dispute between the sender and recipient. The example protocols of this category are the beacon-based protocol [18], cancellation-centre-based protocol [9], and judge-based protocol [1, 3]. In order to achieve viability, fairness, and assurance, these protocols require that the weak trusted party is secure, and that the underlying communication system is reliable. Additionally, some protocols, such as the beacon-based protocol, are bandwidth consuming, and requires tight clock synchronisation among parties. However, this approach is more efficient in comparison with the first approach due to the fact that the role of the trusted party is reduced.

Though achieving absolutely simultaneous liability using serial communications without any assistance of a trusted party is very difficult if not impossible, the simultaneity can be

accomplished approximately by means of progressive exchanges of partial commitments to a contract.

The third approach is the one which uses this philosophy. It purely relies on some randomised algorithms or gradual secret releasing schemes to achieve a fair exchange of signatures between the parties. One example of the protocols using this approach is the Even's [7] which employs a large number of secret pairs of messages together with the *1-out-of-2 oblivious transfer* (OT½) primitive, in an exchange of signatures, to achieve fairness. Other examples are those based on gradual secret releasing [2, 4, 13, 16]. Rather than using the random nature of some randomised algorithms such as OT½ primitive to achieve fairness in a probabilistic way, the gradual secret releasing contract signing protocols employ secret gradual releasing technique to release some predefined secrets in a bit-by-bit or even a fraction of a bit manner. Such design does not require the participation of a trusted party, and therefore the likelihood of causing a performance bottleneck or a security vulnerable point is low or null. However, weaknesses do exist. Firstly, the number of secret pairs to be exchanged needs to be sufficiently large to achieve a high level of fairness, and this implies that the protocol execution is bandwidth consuming and tedious. Secondly, the protocol requires that the computational capabilities of all the parties be (approximately) the same, and we feel that this requirement is unrealistic in the context of an open, distributed and heterogeneous network environment in which global EDI or electronic commerce systems are built. Lastly, the security assurance level is low in that one party can still cheat the others by exploiting the protocols' weakness on simultaneous liability.

The aim of this paper is to propose a new protocol, which is able to comply with the requirements specified above, by making use of advantages of these existing protocols and eliminating their weaknesses. In detail, the next section gives the notations to be used throughout this paper. In Section 3, a novel protocol for signing contracts is proposed, based on the outcome of the survey, with respect to the requirements above. Finally, our conclusions are drawn in Section 4.

2. NOTATIONS

The notations to be used throughout this paper are summarised as follows:

- $X \| Y$ means the concatenation of data items X and Y.
- $\{PK, SK\}$ denotes a pair of keys used in a public-key cryptosystem such as RSA [19], where SK stands for the private (or secret) key and PK the public key.
- $\{M\}_K$ is the cipher-text of message M, encrypted by key K using a conventional (e.g. DES [14]) or public-key cryptosystem.
- $S_A(M) = \{M\}_{SK_A}$ is the digital signature of party A on message M, where SK_A is A's private key and $\{M\}_{SK_A}$ is public-key encryption. When M is a large message, its hash value uniquely associated with M could be used instead.
- $A \Rightarrow B$: M means party A sends party B message M.
- *CONT* denotes a contract to be signed which could be a hash value uniquely associated with all the contract details.

3. OUR WORK

Here, we propose a novel protocol which requires the presence of a trusted party, but any security misfortune happening to this trusted party, or any conspiracy with the trusted party, will not lead to any successful cheating among contract signers. The emphasis of the protocol design is placed on the satisfaction of the requirements set out in Section 1.

3.1. Assumptions

The protocol design is based on the following assumptions:

- There is a trusted party called *SS* (security server). Its responsibilities are to publicise received information such as joint signatures, and to maintain a database of all publicised information. This database is read-only and must be protected from unauthorised modification.

  Here, the publication of information means publicising the information in a way similar to a *public notice board.* The information publicised on the public notice board is unconditionally readable by any viewers or clients, regardless of their identities (of course, if the information is intended for a group of users, it can be encrypted using a key known only by these users). The practical implementation of such a public notice board may vary, e.g. it can be a known query-reply service, such as a news group or a Web site, provided by a server.

- All parties including *SS* have access to conventional and public-key cryptosystems as well as a Certification Authority [12] that generates and manages public-key certificates (*PC*s) for the parties. For example, PC_A is party *A*'s certificate issued by the *CA*, which specifies *A*'s distinguished name (e.g., name, organisation, email addresses, telephone number, etc.), his public key PK_A, and its issuing date.

3.2. The protocol

This protocol is based on a combined concept of conditional signatures and a public notice board. Its implementation is divided into two stages. First, signatures on a contract, encrypted using the public key of *SS*, are collected, in sequential order, from every party involved, and then sent to *SS* as a single message. These signatures are valid only under the conditions that:

(a) they will be publicised by *SS* before a specified time t_n;

(b) each of them is a correct signature in terms of its origin and the contract detail signed.

Secondly, when receiving the message at time t_{SS}, *SS* simply decrypts it to obtain these signatures, and then publicises them together with arrival time t_{SS}. After time t_n ($t_n \geq t_{SS}$), each party fetches the publicised signatures from *SS* and checks their validity in terms of the conditions above. If they are valid, all the parties are bound to the contract with effect from time t_n or t_{SS}.

To describe the protocol in detail, we assume that there are n parties who want to sign a contract *CONT*. Before the signing process starts, these parties first make an agreement on the following preliminary points:

- Sequential order $(P_1, t_1), \ldots, (P_n, t_n)$: in which signatures are collected and passed from one party to another. Here, every pair (P_i, t_i) ($1 \leq i < n$ and $t_i < t_{i+1}$) indicates that party P_i must ensure his signature, together with those received, to reach P_{i+1} by time t_i, and the last pair

(P_n, t_n) specifies that party P_n must let *SS* receive all the signatures by time t_n. This order should be decided by a mutual agreement among these parties, or by a random algorithm if they cannot reach such an agreement. As will be seen shortly, the order does not affect the fairness of the protocol.

- Session key K: which is only valid for this signing process and known only by these n parties involved in the deal.
- Session dependent random number rn: which is approved by all the parties and used to prevent a dishonest party from replaying a signature issued in a previous session.
- Session label *Index*: which is chosen jointly by all the parties and will be used as an index for them to fetch the associated information from *SS*.

Based on the agreement above, the protocol is formally presented as follows:

1. $P_1 \Rightarrow P_2$: $M_1(1)||M_1(2)$, before time t_1, where:

   $M_1(1) = \{Index||t_n\}_{PK_{SS}} ||\{\{PC_1||S_{P1}(CONT||decision_1||rn)\}_K\}_{PK_{SS}}$

   $M_1(2) = S_{P1}(M_1(1))$

   Here,

   - *Index* is included to tell *SS* that all the signatures should be publicised with this label *Index*.
   - t_n is included to notify *SS* that it publicises all the signatures received only if they arrive at *SS* no later than deadline t_n.
   - PC_1 is party P_1's public-key certificate. Its presence here is to inform the other parties involved of his identity and public key to facilitate them for the verification of the signature $S_{P1}(CONT||decision_1||rn)$.
   - $S_{P1}(CONT||decision_1||rn)$ is the signature of P_1 on the items: contract details *CONT*, P_1's decision on the contract $decision_1$, and session dependent number rn. The value of $decision_1$ equals *yes* if P_1 has agreed on the deal, and *no* otherwise.
   - $PC_1||S_{P1}(CONT||decision_1||rn)$ is encrypted using session key K, to protect the confidentiality of the parties' identities and the contract. It will be seen shortly that *SS* will make the signatures of all the parties readable by any viewer. This encryption thus ensures that only these n parties are able to interpret the signatures. If such confidentiality is not required, the encryption can be omitted.
   - The final encryption of $\{PC_1||S_{P1}(CONT||decision_1||rn)\}_K$ using *SS*'s public key PK_{SS} is to prohibit any other party from gaining P_1's signature on the contract prior to the publication of all the signatures by *SS*. In other words, the other parties do not know P_1's decision on the contract until all the parties' signatures are publicised by *SS* before time t_n.
   - $M_1(2)$ is P_1's signature on $M_1(1)$, which is used for the next party P_2 to verity the integrity and authenticity of $M_1(1)$. Obviously, $M_1(1)$ is the message to be forwarded to *SS* through P_2, while $M_1(2)$ is just for the verification purpose.

2. $P_i \Rightarrow P_{i+1}$: $M_i(1)||M_i(2)$, before time t_i, for every i $(1 < i < n)$, where:

   $M_i(1) = M_{i-1}(1)||\{\{PC_i||S_{Pi}(CONT||decision_i||rn)\}_K\}_{PK_{SS}}$

$M_i(2) = S_{Pi}(M_i(1))$

This message transfer is performed, only if (a) party P_i received message $M_{i-1}(1)||M_{i-1}(2)$ from P_{i-1} before time t_{i-1}; and (b) P_i has proved the correctness of $M_{i-1}(1)$ in terms of $M_{i-1}(2)$. The whole process is aborted, and each of the other parties is informed, otherwise. Message $M_{i-1}(1)||M_{i-1}(2)$ should be kept safely by P_i for purposes of resolving any future dispute.

3. $P_n \Rightarrow SS$: $M_n(1)||M_n(2)$, before time t_n, where:

   $M_n(1) = M_{n-1}(1)||\{PC_n||\{S_{Pn}(CONT||decision_n||rn)\}_K\}_{PK_{SS}}$

   $M_n(2) = S_{Pn}(M_n(1))$

   Having received message $M_n(1)||M_n(2)$ from P_n at time t_{SS}, SS confirms the integrity and originality of the message, decrypts $M_n(1)$ to obtain label *Index* and each signature, and publicises the following information:

   $Index||t_{SS}||M_{SS}(1)||M_{SS}(2)$, where,

   $M_{SS}(1) = \{PC_1||S_{P1}(CONT||decision_1||rn)\}_K||$

   $\{PC_2||S_{P2}(CONT||decision_2||rn)\}_K||$ ... $||$

   $\{PC_n||S_{Pn}(CONT||decision_n||rn)\}_K$, if $t_{SS} \leq t_n$

   or

   $M_{SS}(1) = Late$, if $t_{SS} > t_n$ (i.e. party P_n has missed deadline t_n)

   $M_{SS}(2) = S_{SS}(Index||t_{SS}||M_{SS}(1))$

   Finally SS saves messages $M_n(1)||M_n(2)$ and $M_{SS}(1)||M_{SS}(2)$ in its secure database for resolution of any future dispute.

4. After time t_n, each party fetches the publicised information from SS in terms of label *Index*, if the signing process was not aborted. All the parties are bound to the contract, provided that $M_{SS}(1)$ is not equal to *Late*, the signature of every party P_i contained in $M_{SS}(1)$ is proved to be correct, and item $decision_i$ in the signature is *yes*. Otherwise, the contract is invalid.

The process of the protocol execution above is depicted in Fig. 1.

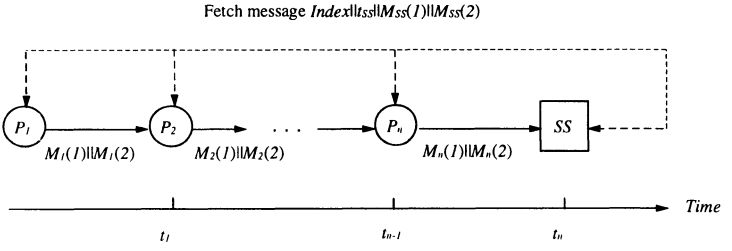

Figure 1

The presentation of the protocol above shows that its signature collection process is analogous to passing a black box from the originator P_1, through the other parties involved, to SS in sequential order. When the box reaches a party P_i prior to specified deadline t_{i-1}, he drops his

signature in the box and transfers it to the next party P_{i+1} by deadline t_i. Once the signature is dropped in the box, no party is able to remove it from the box without invalidating it. When the box finally arrives at *SS* before deadline t_n, *SS* opens the box to reveal all the parties' signatures. In other words, all the signatures are made public simultaneously, and no party is able to gain any advanced knowledge about the contents of the signatures of the other parties.

3.3. Analysis

In this section, we analyse the protocol, presented in Section 3.2, to see how it conforms to the requirements set out in Section 1.

Viability

The successful operation of the protocol ensures that all the signatures on the contract are valid due to the conditions (a) and (b) stated in Section 3.2. This can be illustrated in detail as follows:

- *Unforgeability*: The signature of each party comprises contract details *CONT* and session dependent number *rn*, and is encrypted using the party's private key known only by the party. Thus the signature on the contract is in theory unforgeable by any other party. However, there could exist a potential threat to the originality of the signature, as the party may cheat by intentionally compromising his private key and then claiming that the signature was not generated by him.

  To counter this threat, we assume that in addition to the tasks specified in Section 3.1, the *CA* is also responsible for recording all key compromise cases. When a party suspects his private key of being disclosed, he must notify the *CA* of revoking his keys, and request the *CA* to issue new keys. In this way, every party is held accountable for anything signed in his private key unless the revocation of the key has been lodged with the CA prior to the signing.

- *Verifiability*: During the process of signature collection, the integrity and originality of each message $M_i(1)$ received by party P_{i+1} are assessed in terms of $M_i(2)$. This indicates that each party P_i is held accountable for his message $M_i(1)||M_i(2)$ by the next party P_{i+1}. Once all the signatures are publicised, the integrity, authenticity and validity of each party's signature are further verified by using the party's public key to decrypt the signature to confirm that it contains the correct contract details *CONT* and session number *rn*.

- *Non-repudiation of receipt*: The protocol is able to prevent a party from false denial of reception of the signatures, since they are made public by *SS* and it is that party's responsibility for fetching them from *SS* [20].

Fairness

This is accomplished by means of trusted party *SS* that implements the combined concept of a public notice board and conditional signatures. Though the sequential process is adopted to collect signatures in order to reduce the involvement of *SS*, the protocol design still promises the following properties:

- During the process of signature collection, any party P_i can not alter the content of message $M_{i-1}(1)$ received without invalidating it and being accused of guilty (detailed discussion on this issue will be presented under the heading Assurance).

- Any party can not read the contents of the received signatures prior to specified time t_n under the assumption that the private key of *SS* is secure (an insecure case of the key will also be discussed under the heading Assurance).

These two properties enable simultaneous liability to be realised through the sequential process. In other words, all the parties are simultaneously bound to the contract at time t_n, provided each signature is correct and the decision in it is *yes*.

Efficiency

The involvement of trusted party *SS* in the protocol is kept minimal. This issue is illustrated as follows:

- The sequential process of signature collection and verification, rather than transmission of each party's signature directly to *SS*, is employed to avoid any communication between the parties and *SS* at this stage.
- The tasks performed by *SS* are limited to the decryption, publication and storage of the signatures.
- Though each party is permitted to fetch the publicised signatures directly from *SS* as depicted in Fig. 1, the amount of communication between the parties and *SS* can be reduced in such a way that only an elected party such as P_1 fetches the signatures from *SS* and then broadcasts them to the other parties. Particularly, such reduction will be significant when the number of the parties is large. However, this method should not deny any individual party's rights for fetching the signatures directly from *SS*, if the elected party fails to deliver them. This measure prevents the situation where a party rejects the validity of the contract simply because he has not received all the signatures from the elected party.

The facts above illustrate that *SS* is unlikely to become a performance bottleneck.

Assurance

The security assurance of the protocol is demonstrated in terms of the following cases:

- The combined concept of a public notice board and conditional signatures, adopted by this protocol, is able to minimise any security risks in case that trusted party *SS* encounters any security misfortune. This is shown in the instances below:
  - The confidentiality of *SS* is not important as all information is open to any viewer. However, only these parties involving the contract signing can really read the corresponding signatures since they are encrypted using the group session key known only by the parties. Obviously *SS* even can not interpret the signatures.
  - Consider that a party involved in the deal has compromised the security of *SS* and procured its private key without being detected. This bad party is certainly able to interpret signatures in the message received. Particularly, he can obtain the signatures of all the other parties before time t_n, if P_n is this bad party. Nevertheless, such disclosed signatures are invalid unless they are publicised by *SS* prior to t_n as stated in Section 3.2. Also time t_n should be chosen in such a way that the early disclosure of some signatures causes no damage to interests of their associated parties.
  - Suppose that the data of *SS* are tampered with by a malicious party. This gives rise to two possible scenarios. First, some signatures have been altered before they are publicised. This only leads to the detection of such alteration by those parties issuing

the signatures, and, as a result, the contract becomes invalid due to the condition (b) defined in Section 3.2. Secondly, some signatures are altered after they have been publicised. This does not affect the parties if some of them have got the signatures before the alteration, as the signatures can be circulated among them. It results in an invalid contract if these parties have all received the altered signatures.

- Assume that *SS* is corrupted and attempts to manipulate arrival time t_{SS} or *SS*'s clock. This leads to the following cases. First, all the signatures are publicised when t_{SS} later than t_n has been brought backwards to an earlier time than t_n. Depending on an agreement among the parties, the contract may be binding if no party could gain any benefit from such late publication, and not otherwise. Secondly, no signatures are publicised when t_{SS} no later than t_n has been brought forwards to a later time than t_n. This simply results in unsuccessful contract signing. Finally, there is no effect on the protocol when t_{SS} is altered in other ways. In fact, it is easy for the parties to detect the above alterations by closely monitoring *SS*.

A more effective method for dealing with the last two cases above will be described under the heading Tolerance.

- The process of signature collection is accountable. There could be the situation where a party P_i maliciously alters the content of message $M_{i-1}(1)$ received from party P_{i-1}, then includes the altered message in his message $M_i(1)$, and finally transfers $M_i(1)||M_i(2)$ to P_{i+1}. This guilty party is easily identified when the signatures are publicised by *SS*. This is because each party P_j must keep received message $M_{j-1}(1)||M_{j-1}(2)$ for resolution of any future dispute, and the altered message $M_{i-1}(1)$ in $M_i(1)$ held by P_{i+1} obviously differs from the original one acquired by P_i.

Tolerance

The protocol is resilient to the reality that underlying networks may be unreliable. This is illustrated in terms of the following scenarios:

- A communication channel between two parties P_i and P_{i+1} is unreliable. During the process of signature collection, this channel could transmit a corrupted (erroneous or lost) message from P_i to P_{i+1}, which is recognised by a mismatch in the integrity verification of the message. As a result, retransmission is required. If repeated retransmissions cause a timeout of deadline t_i, the signatures will not be publicised. The implication of this is that the contract is not signed. Since nobody is committed to the contract, such a unreliable channel offers no security loopholes for any dishonest party to deceive others. The same consideration is also applied to the case where a party's computer system malfunctions.
- *SS* itself, or a communication channel between *SS* and a party involved in the deal, is unreliable. This could give rise to the following incidents:
  - The message transferred from P_n to *SS* is corrupted or delayed. In this case, the transfer needs to be repeated until time t_n. Unsuccessful transfer before t_n implies an invalid contract.
  - *SS* publicises corrupted signatures. This simply results in an invalid contract.
  - *SS* fails to respond to requests on the signatures from the parties. An easy solution is to repeat the requests. In reality, a timeout should be defined, and the contract becomes invalid if *SS* gives no response within the timeout.

- Messages received from *SS* are corrupted. Similarly, they need to be requested repeatedly until the timeout is expired.

Additionally, several trusted parties could be employed to tackle the above problems in such a way that the parties' signatures are publicised simultaneously by these trusted parties. Obviously the signatures need to be encrypted based on the public keys of all the trusted parties, so they are decryptable by each of these parties. The benefit from this method is two-fold. First, the protocol is still able to successfully complete its operation despite of failures occurring in some trusted parties or their communication channels. This improves the reliability of the protocol. Secondly, cross-checking between the trusted parties enables the parties involved in the deal to discover and deter the compromise cases analysed earlier. This strengthens the security assurance of the protocol.

Applicability

The design of the protocol does not impose any restriction on the parties' computational capabilities. Though it requires the parties' clocks to be loosely synchronised due to the deadlines associated with each party, the time service offered by existing networks [15, 17] can well satisfy such a requirement. Hence, our protocol is more suited to distributed and heterogeneous network platforms.

4. CONCLUSIONS

We have briefly surveyed existing protocols for signing contracts electronically over networks, and proposed a new protocol, based on the outcome of the survey, which complies with the requirements set out in Section 1. The fairness and security assurance of this new protocol are simply accomplished by means of the combined concept of conditional signatures and a public notice board. In comparison with these existing protocols, the new protocol is more secure, reliable, efficient and applicable due to the following characteristics:

- Although trusted party *SS* is employed by the protocol, its role is very much weakened, i.e. it is only responsible for receiving, publicising and storing signatures. Any compromise of *SS*'s security causes no damage to interests of the parties involved.
- The possible implementation of the protocol on multiple trusted parties improves its security and reliability. This enables the protocol to operate appropriately even in a very hostile working environment involving unreliable network systems and malicious attacks on them. This feature has not been addressed in the existing protocols.
- The sequential process of signature collection, along with possible use of an elected party for fetching and broadcasting signatures, is adopted by the protocol to reduce the amount of direct communication to *SS*. This avoids performance bottleneck problems.
- The protocol demands no tight restrictions on time synchronisation and computational capabilities between different parties. It is more suitable for and implementable in a distributed and heterogeneous network platform on which electronic commerce is based.

REFERENCES

[1] Asokan, N., Schunter, M. and Waidner, M. (1996) Optimistic Protocols for Fair Exchange, *IBM Research Report (Zurich),* **RZ2858 (#90806).**

[2] Blum, M. (1983) How to Exchange (Secret) Keys, *ACM Transaction on Computer Systems*, **1(2)**, 175-193.

[3] Ben-or, M., Goldreich, O., Micali, S. and Rivest, R. (1990) A Fair Protocol for Signing Contracts, *IEEE Transactions on Information Theory*, **36(1)**, 40-46.

[4] Cleve, R. (1989) Controlled Gradual Disclosure Schemes for Random Bits and Their Applications, *Proceedings of CRYPTO'89,* 573-587.

[5] Diffe, W. and Hellman, M. (1976) New Directions in Cryptography, *IEEE Transactions on Information Theory,* **IT-22(6),** 644-654.

[6] Dosdale, T. (1994) Security in EDIFACT Systems, *Computer Communications,* **17(7)**, 532-537.

[7] Even, S. Goldreich, O. and Lempel, A. (1985) A Randomised Protocol for Signing Contracts, *Communications of the ACM,* **28**, 637-647.

[8] Ford, W. (1995) *Computer Communications Security: Principles, Standard Protocols, and Techniques*. Englewood Cliffs, NJ: Prentice Hall.

[9] Goldreich, O. (1984) A Simple Protocol for Signing Contracts, *Advances in Cryptology: Proceedings of Crypto'83,* (ed. D. Chaum), Plenum Press, New York, 133-136.

[10] Hastad, J. and Shamir, A. (1985) The Cryptographic Security of Truncated Linearly Related Variables, In *Proceedings of the 17th STOC.*

[11] ISO/IEC JTC 1/SC 27 N 1105, 2nd ISO/IEC CD 13888-1, Information technology - Security techniques - Nonrepudiation - Part 1: General Model.

[12] Kaufman, C., Perlman, R. and Spencer, M. (1995) *Network Security: Private Communication in a Public World*. Englewood Cliffs, NJ: Prentice Hall.

[13] Merkle, C. (1989) A Certified Digital Signature, *Advances in Cryptology: Proceedings of Crypto'89,* 218-238.

[14] National Bureau of Standards (1977), Data Encryption Standard, FIPS publications.

[15] Network Working Group Report (1989), Network Time Protocol Specification and Implementation, Request for Comment (RFC) 1119.

[16] Okamoto, T. and Ohta, K. (1994) How to Simultaneously Exchange Secrets By General Assumptions, *The 2nd ACM Conference on Computer and Communication Security,* Fairfax, Va., USA, 184-192.

[17] Postel, J. (1980) User Datagram Protocol, Request for Comment (RFC) 768.

[18] Rabin, O. (1983) Transaction Protection by Beacons, *Journal of Computer and System Science*, **27**, 256-267.

[19] Rivest, R., Shamir, A., and Adleman, L. (1978) A Method for Obtaining Digital Signatures and Public Key Cryptosystems, *Communications of the ACM*, **2**.

[20] Zhang, N. and Shi, Q. (1996) Security Issues in an EDI Environment, *Proceedings of 12th Annual Computer Security Applications Conference*, San Diego, California.

BIOGRAPHY

Dr. Zhang is currently a lecturer in the Dept. of Computing at Manchester Metropolitan University. Her major research work involves computer networks, distributed real-time systems, and design of secure communication protocols.

Dr. Shi is currently a lecturer in the School of Computing and Mathematical Sciences at Liverpool John Moores University. His research is concerned mainly with practical applications of formal methods to the design and evaluation of secure computer systems. Particularly, his work covers composable security theory, architectures of secure systems, and secure communication protocols.