Security and Privacy Assurance in Advancing Technologies: New Developments

Hamid R. Nemati
The University of North Carolina, USA

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE
Hershey · New York

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Security and privacy assurance in advancing technologies : new developments /
Hamid R. Nemati, editor.
p. cm.
Includes bibliographical references and index.
Summary: "This book provides a comprehensive collection of knowledge from experts within the field of information security and privacy and explores the changing roles of information technology and how this change will impact information security and privacy"--Provided by publisher.
ISBN 978-1-60960-200-0 (hardcover) -- ISBN 978-1-60960-202-4 (ebook) 1. Information technology--Security measures. 2. Management--Technological innovations. I. Nemati, Hamid R., 1958-
HD30.2.S437 2011
005.8--dc22
2010046637

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 17
A Resilient Fair Electronic Contract Signing Protocol

Harkeerat Bedi
University of Memphis, USA

Li Yang
University of Tennessee at Chattanooga, USA

ABSTRACT

Fair exchange between parties can be defined as an instance of exchange such that either all parties involved in the exchange obtain what they expected or neither one does. The authors examine a protocol by Micali that provides fair contract signing, where two parties exchange their commitments over a pre-negotiated contract in a fair manner. They show that Micali's protocol is not entirely fair and demonstrate the possibilities for one party cheating the other by obtaining the other party's commitment and not offering theirs. A revised version of this protocol by Bao which provides superior fairness by handling some of the weaknesses is also discussed. However, both these protocols fail to handle the possibilities of a replay attack. Their prior work improves upon these protocols by addressing the weakness that leads to a replay attack. This journal extends their prior work on fair electronic exchange by handling a type of attack which was not handled earlier and provides a brief survey of the recent work related to the field of fair electronic exchange. They also discuss the application of cryptography to our protocol which includes implementation of hybrid cryptography and digital signature algorithms based on elliptic curves to achieve features like confidentiality, data-integrity and non-repudiation.

INTRODUCTION

Commerce has come a long way since the beginning of our civilization. The ability to exchange goods and services for items of equivalent value has been widely exercised. Based on the kind of items exchanged between two parties, it can either be classified as a barter system where goods and services are exchanged for other goods and services, or the act of selling and buying where goods and services are sold or bought between parties in exchange for money.

The notion of *fair exchange* can be expressed as the ability to exchange goods or services for

DOI: 10.4018/978-1-60960-200-0.ch017

other goods or services in a fair manner where both the parties obtain what they expected. Being a fundamental concept, this can be implemented in various scenarios that may include exchanges based on barter system or buying and selling of goods.

With the advent of computers and the Internet, new means of performing commerce have been invented. E-commerce is one such solution where good and services are bought and sold between interested parties using computers over a network. With the rapid growth of the Internet, the magnitude of commerce performed online has also increased significantly. This increase is primarily because commerce conducted online is convenient and fast when compared to the traditional methods of trade. Even though commerce of this type offers benefits like speed and convenience, without properties like fairness and security, such services become less useful as they significantly increase the risk of failure. E-commerce cannot flourish or even sustain if it is not able to provide fairness and security. Therefore the concept of fair exchange plays a vital role in shaping such forms of commerce. When carried out online using computers and the Internet, such fair exchange is known as *fair electronic exchange*.

FAIR ELECTRONIC EXCHANGE

Fair electronic exchange can be demonstrated as e-commerce that takes place between two parties who are online and where exchange of goods and services is performed such that both either parties obtain what they expected or they obtain nothing at all. After an exchange is performed or aborted prematurely, none of the parties should have an unfair advantage over the other. If cheating takes place, where one party refuses to present their part of the exchange, other means for providing fairness should be available. These may include use of additional entities like a human judge or electronic ones that can comprehend the situation and act accordingly to provide fairness. Protocols that provide such facilities are termed as *fair exchange protocols*. Such protocols can be used for the following purposes:

a. *Certified E-Mail (CEM),* where a user named Alice sends a message to user a named Bob and gets a receipt from him in return. Providing the quality of fairness would include Alice getting the receipt only when Bob gets the message or Bob getting the message only when Alice gets the receipt. Associated protocols include (Zhou & Gollmann, 1996; Kremer & Markowitch, 2001; Ateniese & Nita-Rotaru, 2002; Imamoto & Sakurai, 2002)
b. *Electronic Contract Signing (ECS),* where both Alice and Bob wish to sign a contract that has already been negotiated. This would involve Alice sending her commitment (digital signature) on the contract to Bob and him sending his commitment (digital signature) on the same in return. Providing fairness would involve Alice receiving Bob's commitment only when her commitment is received by Bob and vice versa. This example demonstrates contract signing between two parties. Protocols that provide such functionality include (Ben-Or, Goldreich, Micali, & Rivest, 1990; Damgard, 1995; Bao, Deng, & Mao, 1998; Asokan, Shoup, & Waidner, 1998; Ateniese, 1999; Garay, Jakobsson, & MacKenzie, 1999; Micali, 2003; Bao, Wang, Zhou & Zhu, 2004). However, various multi-party contract signing protocols also exist and have also been proposed in (Baum-Waidner, 2001; Ferrer-Gomila, Payeras-Capella, Huguet-Rotger, 2001; Garay & MacKenzie, 1999;
c. *Online payment systems (OPS),* where Alice is the seller and Bob is the buyer and payment is given in return of the item of value (Cox, Tygar & Sirbu, 1995). Similar e-payment schemes in electronic commerce include

(Boyd & Foo, 1998; Park, Chong, & Siegel, 2003).

d. *Non-repudiation protocols,* where the parties involved in an exchange cannot later deny their participation or their actions performed. Protocols associated with such services include (Zhou & Gollmann, 1996; Kremer & Markowitch, 2000; Kremer, Markowitch & Zhou, 2002; Gurgens, Rudolph, & Vogt, 2003)

In the ideal case, where both Alice and Bob are guaranteed to be honest and the communication channel is secure and provides resilience, fair exchange can be achieved trivially without the aid of any external fairness provider. The above described scenarios can thus be carried out as follows:

Fair Certified E-Mail

- Alice sends her message to Bob.
- Bob sends his receipt for the message to Alice. The receipt may be the digital signature of Bob on the message. Being a digital signature, this step ensures non-repudiation.

Fair Electric Contract Signing

It is assumed that both parties have negotiated the contract before hand.

- Alice sends her digital signature on the contract to Bob.
- Bob sends his digital signature on the contract to Alice.

Fair Online Payment System

- Alice sells goods or services online by sending it to Bob.
- Bob buys these good or services by paying Alice online via e-check or e-money.

However in practice, honesty of the parties like Alice and Bob participating in an exchange can never be guaranteed. The availability of secure communication channels is not always possible and unsecured channels are easily prone to attacks.

Following are the types of outcomes that may take place rendering the above mentioned exchanges incomplete.

a. Cheating Bob can always refuse to send his signature on the messages to Alice after receiving her signature. In such a case, there is not much that Alice can do to obtain fairness.

b. An intruder can always stop the messages from reaching the other party. In this case the receipt or the messages signed by Bob may never reach Alice. Alice may think that she has been cheated where as Bob may be unaware of this taking place. This confusion may lead Alice to request for cancellation whereas Bob may not wish for the same.

c. An intruder who is listening to messages sent by Alice or Bob can replay those messages making the other party believe that they were sent from the original sender. In case of electronic contract signing, if the intruder replays Alice's messages, there is no way for Bob to learn that the messages are not from Alice but an intruder. This is because the signature on the intruder's message will always be verifiable using Alice's corresponding public key. Honest Bob may sign the message and think that he has signed a new contract with Alice, whereas Alice may never know of this taking place.

d. Bob can sign a fake contract is exchange of Alice's signature on the correct contract, thus cheating her by providing his commitment on a contract that does not solve her purpose.

Therefore to prevent such unwanted outcomes, external fairness providers are used to comprehend the situation if cheating is suspected and provide resolution. Such fairness providers are separate entities known as the *trusted third parties*. As the name suggests these entities are trusted by everyone. Considering the above scenarios, Alice can communicate with the external fairness provider and obtain fairness if Bob refuses to sign the contract after receiving Alice's signature. Alice can provide the provider with information relating to the contract along with the messages she sent to Bob. The provider, thus after verifying this information and being sure that Alice is not cheating can then provide fairness to her by regenerating Bob's part of exchange or issuing a certificate that can be used as a substitute for Bob's signature.

Based on their role and the method of providing fairness, either by preventing or handling such unwanted outcomes, these providers are classified into the following types:

One class of protocols depends on gathering evidence during the transaction that can later be used to provide fairness. Two parties during their transaction also send additional information along with their messages which can later be used for resolution if one believes that it has been cheated. The cheated party contacts a human judge which looks at the additional information exchanged (evidence) and provides fairness. Such protocols are classified as *weak fair-exchange protocols* due to their inability to provide fairness during the transaction.

This becomes a drawback since resolution is provided only after the transaction has been completed. In case of e-commerce where the location and availability of parties taking part in a transaction are not always fixed or known, such methods of dispute resolution may always not be possible.

To handle drawbacks like these, a second class of protocols has been defined by various researchers that provide means of avoiding disputes and obtaining resolution all within the transaction. Thus fairness can now be obtained during the transaction and such protocols are called as *strong-fair exchange protocols* (Ray & Ray, 2002). These protocols too use a trusted third party which can provide fairness in case of a dispute.

Trusted third parties that have to involve in every transaction occurring between two parties are known as *online trusted third parties*. Exchanges using such trusted third parties suffer from the disadvantage that these trusted third parties are required to involve in every transaction occurring between the interested individuals.

Trusted third parties that are not required to involve in every transaction but only required when a dispute occurs are known as *invisible trusted third parties* or *offline trusted third parties* and the protocols implementing them are known as o*ptimistic protocols*. Following are the advantages provided by protocols which use an invisible trusted third party:

The invisible third party intervenes only when cheating is suspected. In such a case the invisible third party solves the conflict by providing the complaining party with what it truly deserves. Either party can initiate this procedure if they feel they have been cheated.

An invisible third party generates no congestion or bottlenecks as it intervenes only when cheating occurs; which is usually very rare. Under normal execution, transactions between two parties are carried out directly, bypassing the third party altogether.

An invisible third party generates minimal expense and minimal liabilities as it is liable only for the few messages that is sends, which is only in case of a conflict. Even if a system adds a large number of clients which carry numerous transactions, the expense generated by such a system is minimal, since the third party intervenes only with cheating occurs and stays dormant remaining of the time.

Evolution of Fair Exchange Protocols

Earliest work on exchange protocols that provided fairness was based on a class of fair exchange protocols known as *gradual exchange protocols* (Tedrick, 1983, 1985). Such protocols provided fairness as a measure directly proportional to the number of rounds of messages exchanged between the two participating parties. Thus as the number of messages exchanged between parties increases, so does the probability of fairness. These protocols were complex and made use of advanced cryptographic techniques. They also required the assumption that both parties involved in the exchange are to have equivalent computing power. This assumption was removed by the introduction of an improved set of fair exchange protocols called *probabilistic protocols* (Ben-Or, Goldreich, Micali & Rivest, 1990; Markowitch & Roggeman, 1999). These protocols did not require both the parties to have equivalent computing power. However a good number of transmissions between the parties were still required to increase the probability of fairness to acceptable levels. This downside was addressed by the introduction of a new class of protocols that used trusted third parties for providing fairness. This reduced the number of transactions required to be exchanged to a smaller number. It begun with the use of *in-line trusted third parties* (Coffey & Saidha, 1996; Ben-Or, Goldreich, Micali, & Rivest, 1990; Deng, Gong, Lazar & Wang, 1996; Zhou & Gollmann, 1996), where the third party would required to involve in every transaction between the parties participating in the fair exchange protocol. Even though the number of transactions required was reduced, the involvement of third party during the protocol was very high. This made the third party as the bottleneck, since every transaction was required to be passed through the third party. Several improvements were made in this direction of implementation, beginning with the introduction of *online trusted third parties* (Zhang & Shi, 1996; Zhou & Gollmann, 1996), where the third party was required to involve only once during an entire instance of the protocol. This was major improvement since the involvement of third party was reduced substantially. Another major breakthrough was the introduction of *offline trusted third parties* (Asokan, Schunter & Waidner, 1997; Micali, 1997; Asokan, Shoup & Waidner, 1998; Asokan, Shoup, & Waidner, 2000; Ateniese 1999; Bao, Deng, & Mao, 1998; Park, Chong, & Siegel, 2003), where the third party was required to involve only when cheating was suspected or had occurred. This was again a key improvement since now the third party was required to interfere only in case of a conflict, and therefore did not become a bottleneck during the normal execution of the protocol and such protocols as stated previously were known as *optimistic protocols*.

This approach of implementing an offline trusted third party has also been used by researchers in execution of various non-repudiation protocols (Kremer & Markowitch, 2000; Zhou & Gollmann, 1997; Zhou, Deng & Bao, 1999). Non-repudiation with respect to exchange protocols can be defined as the concept of ensuring that a party having involved in an exchange or transmission cannot later deny their involvement. The advent of public key cryptography (Diffie & Hellman, 1976) and the introduction of digital signatures served as the foundation for non-reputation. A property like non-repudiation is required by fair exchange protocols to provide fairness efficiently. However, the first set of non-repudiation protocols proposed by the International Organization for Standards (1997a, 1997b, 1998) did not support fairness exclusively.

In this journal we analyze the several optimistic fair exchange protocols and demonstrate their weaknesses. We start by explaining Micali's protocol and analyzing its weaknesses. Micali's protocol is based on the use of an offline trusted third party where it is used to demonstrate an instance of exchange that implements fair contract signing.

We discuss the revisions made by Bao et al. (Bao, Wang, Zhou & Zhu, 2004) on Micali's protocol and how his protocol handles some of its weaknesses. We then explain our protocol and demonstrate how replay attacks which are not addressed by both these protocols are identified and can be prevented.

MICALI'S ELECTRONIC CONTRACT SIGNING PROTOCOL

In 2003, during the ACM Symposium on Principles of Distributed Computing (PODC) Silvio Micali presented a fair exchange protocol that could perform electronic contract signing (Micali, 2003). The protocol was also filed under US Patent No. 5666420 in 1997. In his protocol, contract signing was achieved in a fair way by using an invisible trusted third party. Fairness in this context means that either both parties are bound to the contract (i.e. obtain each others' signature on the contract) or neither one is.

However, the protocol is not as fair as claimed by the author, since either party is able to cheat the other under certain scenarios. The protocol also does not provide any means to handle or prevent replay attacks that may take place if an intruder replays messages sent earlier by one party to another.

Some of these scenarios are also illustrated in the paper by Bao et al. (2004). Following is the actual protocol for contract signing as proposed by Micali.

Protocol

Contract signing can be implemented between two parties say, Alice and Bob in three steps. Steps 4 and 5 are required only when a dispute occurs.

Pre-Requisites

It is assumed that both parties have negotiated the contract before hand. Alice begins by selecting the contract file "*C*" that she needs to sign with Bob. She also selects a random value *M* and creates a packet "*Z*".

Packet *Z* contains the following information:

1. Identity of sender: This is basically a string that represents the sender. Public key of the sender can also be used. In this case the sender is Alice "*A*".
2. Identity of the receiver: A string that represents the receiver. Public key of the receiver can also be used. In this case the receiver is Bob "*B*".
3. Random value *M*: This can also be a number. It is a value known only to Alice and will later be used for completion of the contract signing process.

This information is encrypted using the public key of the third party, which is known to everyone.

Thus *Z* can be represented as $Z = E_{TP}(A, B, M)$, where E_{TP} is performing encryption using the trusted third party's (*TP*) public key.

Due to the practical requirements for the encryption functions to be so secure, Micali also explains the implementation of the encryption function "E^{R}_{TP}", where *R* emphasizes the use of a unique random value for performing encryption while using the trusted third party's (*TP*) public key.

A, *B* are the identifiers for Alice and Bob respectively and *M* is the secret random value known only to Alice.

Steps

1. Once the packet *Z* is created, Alice initiates the protocol by sending her signature on the packet *Z* and the contract *C* to Bob.

A1: A → B: SIG_A (C, Z)

2. Upon receiving Alice's message, Bob sends his signature of *(C, Z)* and *Z* to Alice.

B1: B → A: SIG_B (C, Z) + SIG_B (Z)

3. After receiving Bob's message, Alice verifies his signatures on both *(C, Z)* and *Z* and if they are valid, she sends *M* to Bob.

A2: A → B: M

Dispute Resolution Phase

4. Bob receives the random *M* and uses it to reconstruct *Z*. If the newly created *Z* matches with the one he received in *Step 1*, he halts and the contract signing protocol is complete. Else, Bob sends his signature of *(C, Z)* and *Z* to the trusted third party.

If values of *Z* do not match:

B2: B → TP: SIG_B (C, Z) + SIG_B (Z)

5. Third Party verifies the signatures it received from Bob. If they are valid, it decrypts *Z* using its private key and sends *M* to Bob and SIG_B *(C, Z)* + SIG_B *(Z)* to Alice.

TP1: TP → A: SIG_B (C, Z) + SIG_B (Z)
TP2: TP → B: M

Micali defines the commitments of Alice and Bob on the contract *C* as the following:

Alice's commitment to contract *C*:

SIG_A (C, Z) and M

Bob's commitment to contract *C*:

SIG_B (C, Z) + SIG_B (Z)

To illustrate the fairness of the above mentioned contract signing protocol, Micali provided the following argument (Micali, 2003):

"Indeed, if Bob never performs Step B1, then he is not committed to C, but neither is Alice, because Bob only has received SIG_A(C, Z), and has no way of learning M. However, if Bob performs Step B1, then he is committed to C, but Alice too will be so committed: either because she will honestly send M to Bob, or because Bob will get M from the invisible TP. Again, if Bob tries to cheat bypassing Step B1 and accessing directly the invisible TP to learn M, then Alice will get SIG_B(C, Z) and SIG_B (Z) from the invisible TP, because the invisible TP will not help Bob at all unless it first receives both signatures, and because once it decrypts Z to find M, it will also discover that Alice is the first signatory, and thus that she is entitled to SIG_B(C, Z) and SIG_B (Z)."

In case of the use of the encryption function *"E^R_{TP}"* during the generation of *Z*, Alice would also be required to send *R* along with *M* in Step 3.

Analysis

This section discusses the vulnerabilities in Micali's contract signing protocol and how it is unfair in certain scenarios.

Insufficient requirements for dispute resolution: Third party only requires Bob's signatures *(SIG_B (C, Z) and SIG_B (Z))* during the dispute resolution phase and nothing from Alice's side is required. This can cause the following attack:

After receiving Alice's signature *(SIG_A (C, Z)),* dishonest Bob prepares a new contract *C^1*, creates the following signatures *SIG_B (C^1, Z)* and *SIG_B (Z)* and sends them both to the third party requesting for dispute resolution. Since these two signatures are valid and third party does not require any signatures from Alice, it forwards SIG_B *(C^1, Z)* and SIG_B *(Z)* to Alice and *M* to Bob.

This result in Bob having Alice's commitment on contract *C* and Alice only having Bob's commitment on a contract C^1 which is of no use to Alice.

Attack 1

```
A1:     A → B:     SIG_A (C, Z)
B1:     B → TP:    SIG_B (C^1, Z) + SIG_B
(Z)
```

Since *Z* does not contain any information about the contract *C* and the signature of Bob on C^1 and *Z* are valid, the third party provides Bob with the value of *M* contained in *Z* and Alice with Bob's signature over the fake contract C^1.

```
TP1:    TP → A:    SIG_B (C^1, Z) + SIG_B
(Z)
TP2:    TP → B:    M
```

Insufficient requirements for commitment of both parties: As per Micali's definition, for Alice to show Bob's commitment on the contract *C*, she only requires Bob's signatures on *(C, Z)* and *Z*, which are SIG_B *(C, Z)* and SIG_B *(Z)*. Alice is not required to provide the value *M* that creates *Z*. This flaw can be exploited such that Alice can always get Bob's commitment on a contract while Bob gets nothing. Following is the attack:

Dishonest Alice creates a random value of length *Z* and sends her signature of *(C, Z)* to Bob. Bob verifies Alice's signature and since it holds true, he sends his signatures of *(C, Z)* and *Z* to her. Alice now quits the protocol as she has received Bob's commitment. Bob on the other hand cannot get resolution form third party as *Z* is a random value and it cannot find *M* such that $Z = E_{TP}(A, B, M)$.

This leads to Alice obtaining an advantage since she is not required to present a value *M* that can recreate the packet *Z*.

Attack 2

Alice creates a random value of length *Z*.

```
A1:     A → B:     SIG_A (C, Z)
B1:     B → A:     SIG_B (C, Z) + SIG_B
(Z)
A2:     No response
```

Bob contacts the third party for resolution:

```
B2:     B → TP:    SIG_B (C, Z) + SIG_B
(Z)
```

The third party is not able to provide *M* to Bob as *Z* is a random value and it does not contain a value *M* such that $Z = E_{TP}(A, B, M)$.

```
TP:     Halts, as it is unable to
provide the value of M.
```

The above mentioned attacks are also explained in the paper by Bao et al. (2004). They handle these attacks by changing the requirements of the dispute resolution phase, the commitment parameters for both parties and the contents of *Z*.

Third party's dilemma: Micali's protocol does not clarify what the third party is required to do when it receives the packet *Z* such that decrypting it ($D_{TP}(Z)$) does not lead to the desired *(A, B, M)* even though the signatures SIG_A *(C, Z)*, SIG_B *(C, Z)* and SIG_B *(Z)* are valid. This dilemma occurs during the dispute resolution phase where $D_{TP}(Z) \neq (A, B, M)$ for any *M*. This indirectly means that the first two values of *Z* are not the identities of Alice and Bob.

In this case, the third party can either choose to ignore the request, since the identities do not match, or can proceed with dispute resolution phase and provide *M* to the requesting party. Bao's analysis on Micali's protocol show that either step taken by the third party leaves scope for possible attacks where one party can achieve an unfair advantage over the other.

Following is a brief overview of one of the possible attacks based on the above explained dispute resolution issue:

Attack 3

This attack takes place when the third party implements the policy of choosing to ignore the dispute resolution request if $D_{TP}(Z) \neq (A, B, M)$. To stage this attack, dishonest Alice colludes with A^1 and obtains the other party's (Bob) commitment without providing hers.

Steps

1. Alice chooses a random M and creates the packet $Z^1 = E_{TP}(A^1, B, M)$. She then initiates the protocol by creating her signature on the packet Z^1 and the contract C and sends it to Bob.

```
A1:     A → B:     SIG_A (C, Z^1)
```

2. On receiving Alice's message, Bob verifies her signature and since it is valid, sends his signature of $SIG_B(C, Z^1)$ and $SIG_B(Z^1)$ to Alice.

```
B1:     B → A:     SIG_B (C, Z^1), SIG_B
(Z^1)
```

3. After receiving Bob's message. Alice halts.

```
A2:     A → B:     Nothing
```

As Bob does not receive the value of M from Alice, he contacts the third party for dispute resolution. The third party verifies the signatures which are valid but observes that $D_{TP}(Z) \neq (A, B, M)$. As per the resolution policy defined above, the third party ignores this request since the identities of both parties do not match the signatures obtained. This leads to Alice (and her colluding partner A^1) obtaining a valid commitment from Bob over the contract C with A^1 and Bob obtains nothing. This is because Bob's signatures of $SIG_B(C, Z^1)$ and $SIG_B(Z^1)$ can be viewed as valid commitments of Bob to the contract C with the colluding party A^1 since $Z^1 = E_{TP}(A^1, B, M)$. It can be assumed that a contract usually contains the identities of both parties and that this attack can be identified since A^1 is not a part of the contract. However, such assumptions are not specified by Micali. It is not possible to identify the cheater in this type of attack since this attack can also be staged by Bob where he initiates the contract signing process and cheats in the first step by colluding with A^1.

Replay attacks: Even though the above mentioned attacks are addressed, both protocols (Micali and Bao) still leave one possible attack. These protocols provide no means of identifying a replay attack that may take place during a contract signing instance.

Since both the protocols do not handle this type of attack, one of them namely, Micali's original protocol is used for the replay attack demonstration.

Attack 4

Consider the normal execution of Micali's protocol:

```
A1:     A → B:     SIG_A (C, Z)
B1:     B → A:     SIG_B (C, Z) + SIG_B
(Z)
A2:     A → B:     M
```

Let us assume an intruder has access to the transmissions between Alice and Bob and is able to record all the messages sent by Alice. An intruder can thus replay a message that was previously sent by Alice to Bob.

Consider the following execution of the protocol by an intruder:

```
I1:     I → B:     SIG_A (C, Z)
B1:     B → A:     SIG_B (C, Z) + SIG_B
(Z)
I2:     I → B:     M
```

If the intruder replays the message *A1* as a new request, then there is no way for Bob to indentify the sender. Bob will assume the request as a genuine one since its signature will still be valid and respond back with his signature of the same. To this, the intruder can then send the message *A2*. This leads to commitment on a new contract between Alice and Bob. Bob assumes that he has signed a new contract with Alice and she on the other hand knows nothing about it.

Consider a scenario where Alice periodically purchases selected items from Bob using the above protocol. The contracts then signed by them would also be the same, which is also the case in real world transactions and if it is agreed that all contracts expire immediately upon fulfillment (i.e. Bob gets the order, he signs it, and then forgets about it), it would be hard to trace the intruder or even identify the attack. It should also be noted that the intruder does not need to involve the third party to stage this attack thus making it even harder to identify.

BAO'S ELECTRONIC CONTRACT SIGNING PROTOCOL

Micali's protocol suffers from several weaknesses due to the following design issues:

1. The packet *Z* does not contain any information about the contract the parties negotiated to sign.
2. No resolution of the third party's dilemma: where there trusted third party is not sure whether to resolve the dispute resolution request of ignore if $D_{TP}(Z) \neq (A, B, M)$.
3. The protocol provides no means or mechanisms for handling replay attacks that may take place if an intruder replays messages sent earlier from Alice to Bob.

In 2004, during the *Australasian Conference on Information Security and Privacy (ACISP),* Bao et al. proposed a contract signing protocol that improved upon Micali's work. Attacks made possible due to the first two design issues were identified and addressed by Bao. Our prior work (Bedi, Yang, & Kizza, 2009; Bedi & Yang, 2009) did not handle the second design issue but discussed the same in (Bedi & Yang, 2009) under the section titled "Scope for Further Improvement". Our present work now addresses this second design issue.

Attacks made possible due to the lack of contract information in packet *Z* structure were identified and addressed. Bao's protocol handled the first three types of attacks namely, Attack 1, 2 and 3, which were discussed in the previous section by changing the requirements of the dispute resolution phase, the commitment parameters for both the involved parties and the structure of the packet *Z*.

Protocol

In this protocol, contract signing between two parties, say Alice and Bob can be achieved through the following prerequisites and steps.

Prerequisites

It is assumed that both parties have negotiated the contract before hand. Alice begins by selecting the contract file "*C*" that she needs to sign with Bob and creates a hash of it *H(C)*. She also selects the random values *M* and *R* and creates a packet "*Z*".

Packet Z contains the following information:

1. Identity of sender: This is basically a string that represents the sender. Public key of the sender can also be used. In this case the sender is Alice "*A*".
2. Identity of the receiver: A string that represents the receiver. Public key of the receiver can also be used. In this case the receiver is Bob "*B*".

3. Random value *M*: This can also be a number. It is a value known only to Alice and will later be used for completion of the contract signing process.
4. Hash of the contract *C*.

This information is encrypted using the public key of the third party, which is known to everyone.

Thus packet *Z* can be represented as $Z = E^R_{TP}$ *(A, B, H(C), M), w*here E^R_{TP} is performing encryption using the trusted third party's (*TP*) public key with the randomness *R*.

A and *B* are the identifiers of Alice and Bob respectively. *M* is the random value known only to Alice and *H(C)* is the hash of the contract.

Steps

1. Once the packet *Z* is created, Alice then initiates the protocol by creating her signature on the packet *Z*, the contract *C*, the identities of her, Bob and the third party, and sending them to Bob.

A1: A → B: SIG_A (A, B, TP, C, Z)

2. On receiving Alice's message, Bob verifies her signature and if valid, sends his signature of *(A, B, TP, C, Z)* to Alice. Otherwise he halts.

B1: B → A: SIG_B (A, B, TP, C, Z)

3. After receiving Bob's message. Alice verifies his signatures on *(A, B, TP, C, Z)* and if valid, she sends *M* and *R* to Bob.

A2: A → B: M, R

Dispute Resolution Phase

4. Bob receives the values *M* and *R* and uses them to reconstruct *Z*. If the newly created *Z* matches with the one he received in Step 1, he halts and the contract signing protocol is complete. Else, Bob sends his and Alice's signature of *(A, B, TP, C, Z)* to the trusted third party.

If values of *Z* do not match:

B2: B → TP: SIG_B (A, B, TP, C, Z) + SIG_A (A, B, TP, C, Z)

5. Third Party verifies the signatures it received from Bob. If they are valid, it decrypts *Z* using its private key and sends *(SIG_B (A, B, TP, C, Z), M, R)* to Alice and *(SIG_A (A, B, TP, C, Z), M, R)* to Bob.

If contents of *Z* are legit and signatures are valid:

TP1: TP → A: SIG_B (A, B, TP, C, Z) + M + R
TP2: TP → B: SIG_A (A, B, TP, C, Z) + M + R

Else:

Halts or sends an error message

Following are the new commitments as revised by Bao:

Alice's commitment to the contract *C* can be defined as:

SIG_A (A, B, TP, C, Z), M, R
where $Z = E^R_{TP}$ (A, B, H(C), M)

Bob's commitment to the contract *C* can be defined as:

SIG_B (A, B, TP, C, Z), M, R
where $Z = E^R_{TP}$ (A, B, H(C), M)

The design weakness of the unavailability of a mechanism to identify and prevent replay attacks was not addressed by both Micali and Bao.

OUR FAIR CONTRACT SIGNING IMPLEMENTATION

We present a complete working implementation that provides fair electronic exchange. We implement an invisible third party that can be used to provide fairness if cheating is suspected.

Our work comprises of two parts. We begin with a revised protocol that is based on Micali's contract signing protocol. We improve upon Micali's protocol by handling certain types of weaknesses which can lead to attacks including where one party can obtain another's commitment to a contract without providing theirs. We also handle the possibilities of replay attack that can take place if an intruder replays the messages sent earlier from one party to the other.

This present work extends our prior work on fair electronic exchange (Bedi, Yang, & Kizza, 2009; Bedi & Yang, 2009) by handling a type of attack which was not handled earlier and also provides a brief survey of recent work related to the field of fair electronic exchange.

Our prior work (Bedi, Yang, & Kizza, 2009; Bedi & Yang, 2009) offered the unique contribution of handling replay attacks which were not addressed by both Micali and Bao. However one possible attack which is illustrated in the previous sections, titled "Third party's dilemma", was handled by Bao and not our protocol. Third party's dilemma was demonstrated in our prior publication (Bedi & Yang, 2009) under the section "Scope for Further Improvement" and now has been addressed in this present implementation.

To provide confidentiality, handle replay attacks and confirm the identities of both the parties, cryptography along with digital signatures is used. All messages communicated between parties involved in the contract signing process are encrypted using a hybrid cryptosystem. These two parts are explained in detail as follows.

Protocol

This section discusses our contract signing protocol and describes how it achieves fairness. Following is an adaptation of the protocol where the privacy of messages is not essential. It is assumed that both parties are not concerned about the privacy of their messages (or contracts), provided that fairness is guaranteed. This approach is taken to make the illustration of our protocol simple. Privacy of messages can be achieved using cryptography, which is explained in detail in the subsequent topics.

Following is the protocol under normal execution when both Alice and Bob are honest and there is no intruder. Contract signing between two parties, say Alice and Bob, can be achieved through following the below prerequisites and steps.

Prerequisites

It is assumed that both parties have negotiated the contract before hand. Alice selects the contract file "*C*" that she needs to sign with Bob. She also selects a random value *M* and creates a packet "*Z*".

Packet *Z* contains the following information:

1. Identity of sender: This is basically a string that represents the sender. Public key of the sender can also be used. In our case the sender is Alice "*A*".
2. Identity of the receiver: A string that represents the receiver. Public key of the receiver can also be used. In our case the receiver is Bob "*B*".
3. Random value *M*: This can also be a number. It is a value known only to Alice and will later be used for completion of the contract signing process.
4. Hash of the contract *C*, that is *H(C)*

This information is encrypted using the public key of the third party, which is known to everyone.

Thus Z can be represented as $Z = E^R_{TP}(A, B, H(C), M)$, where E^R_{TP} is performing encryption using the trusted third party's (*TP*) public key using the randomness *R*.

A and *B* are the identifiers of Alice and Bob respectively. *H(C)* is the hash of the contract file *C*. *M* is the random value known only to Alice.

Steps

1. Alice sends a nonce NA_1 to Bob. Nonce is a random number used only once for the prevention of replay attacks.

A1: $A \rightarrow B$: NA_1

2. On receiving Alice's nonce, Bob signs it using his private key and sends it back to her along with his nonce. This step ensures Alice that it was indeed Bob who signed the message as Bob's private key is a secret known only to Bob.

B1: $B \rightarrow A$: $SIG_B (NA_1) + NB_1$

3. After receiving the above package Alice can verify the digital signature of Bob on NA_1 using his public key. If it matches, Alice is sure that it is indeed Bob and there is no replay attack. Alice now signs Bob's nonce using her private key so that he can be sure of the same. Alice also signs *Z* along with *H(C)* and sends all of it to Bob.

This step makes Alice partially committed to the contract as she has signed the hash of contract using her private key.

A2: $A \rightarrow B$: $SIG_A (NB_1) + SIG_A (H(C), Z) + Z + C$

4. On receiving Alice's signatures, Bob can now verify them using her public key. If they match, Bob is sure that it is indeed Alice and there is no replay attack. It is now Bob's turn to send his commitment to Alice. Bob does this by signing the Alice's signature he received and sending it back to Alice.

B2: $B \rightarrow A$: $SIG_B (SIG_A (H(C), Z))$

5. After receiving the message, Alice can verify Bob's signature and if it holds true, she sends him the values *M* and *R* signed by her.

A3: $A \rightarrow B$: $SIG_A (M, R) + M + R$

Bob receives this package and learns the values *M* and *R* which is then used to reconstruct *Z* along with public values *A* and *B*. If the newly created *Z* matches with the one he received in Step 3, the transaction is complete and both the parties have successfully signed the contract together.

Following is the protocol under abnormal execution where Alice does not provide Bob with the correct secret values of *M* and *R* in the final step.

Dispute Resolution Phase

Upon receiving Alice's signature of *Z* and *H(C),* Bob sends his signature of the same to Alice. Alice is then required to send the values of *M* and *R* to Bob to provide her complete commitment on the contract. For the purpose of discussion let us assume that Alice cheats Bob by not providing him with the correct secret values of *M* and *R* after she receives his signature on the contract.

Thus Bob is left with a partial commitment from Alice, where as Alice has Bob's complete commitment. Bob can thus execute the following steps to obtain resolution.

1. Bob sends the packet *Z* that he initially received from Alice, the contract *C* that was to be signed, Alice's signature of *Z* and *H(C)* and his own signature on Alice's signature to the third party.

B1: B → TP: $Z + C + SIG_A (H(C), Z) + SIG_B (SIG_A (H(C), Z))$

2. Third party, upon receiving Bob's request performs the following steps:
   a. Computes the value of *H(C)* from *C*.
   b. Decrypts the packet *Z* since it knows its own private key and extracts the secret *M*.
   c. Verifies the contents of *Z* to contain the identities *A* for Alice and *B* for Bob and *H(C)* for the hash of the contract.
   d. Verifies the signatures of Alice and Bob. Third party can do this since it knows the public keys of both Alice and Bob. If the signatures are valid, and the contents of the packet *Z* are verifiable with their identities and the hash of the contract, third party provides Bob with *M* it recovered from *Z,* the value *R* and Alice with the signature of Bob over $SIG_A (H(C), Z)$.

If contents of *Z* are legit and the signatures are valid:

TP1: TP → B: $M + R$
TP2: TP → A: $SIG_B (SIG_A (H(C), Z))$

Else:

No action or report error message.

If the secrecy of the contract is necessary, the cheating party can send the hash of the contract instead of the original contract to the third party during the dispute resolution phase. The third party then can skip the step where it is required to compute the hash of the contract, and can proceed directly with the remaining steps. This completes the resolution phase as the third party provides Bob with the correct values of *M* and *R* which gives him complete commitment from Alice on the contract.

Protocol Description

To sign a contract in theory would mean exchange of signatures from both the parties on the same contract. That is if Alice and Bob were to sign a contract with each other, Alice would need to have Bob's signature (commitment) on the contract in the form of $SIG_B(C)$ and Bob would need to have Alice's signature $SIG_A(C)$.

For this to be fair it would require that each party gets the other party's signature only when their signature is received by the other party. Implementation of this exchange may be straightforward under conventional transactions where both the parties are physically available or geographically locatable. However under e-commerce where the transactions take place over the Internet, it becomes reasonably complicated. This is because these transactions transcend geographical boundaries and it is not always possible to contact the other party for resolution or queries once the transaction is complete.

A party can always refuse to provide their signature once they receive the same from the other party. And since locating someone over the Internet or to know when the other party will be available online again is not always possible, fairness cannot be guaranteed.

Therefore to provide fairness, additional information (e.g. packet *Z*) is exchanged along with the contract (or messages) during these fair exchange transactions. This additional information is then examined and used for resolution if a party does not respond appropriately.

Thus the commitments for both parties are modified as follows:

For Bob to have Alice's commitment on the contract, he would need:

SIG_A (H(C), Z), M and R;
where $Z = E^R_{TP}$ (A, B, H(C), M)

For Alice to have Bob's commitment on the contract, she would need:

SIG_B (SIG_A (H(C), Z)), M and R;
where $Z = E^R_{TP}$ (A, B, H(C), M)

To prove the commitments, parties will be required to present not only the signatures but also the contents *M, R, A, B* and *H(C)* that altogether satisfy $Z = E^R_{TP}(A, B, H(C), M)$. Failing to do so shall render the commitments as well as the signed contract as invalid.

The additional information used in our protocol is $Z = E^R_{TP}(A, B, H(C), M)$ and values *R* and *M*.

The packet *Z* is created using the following information,

1. Identities of both the parties: Those are *A* for Alice and *B* for Bob. We use text strings for this purpose. The public keys of parties or any other type of identifiable information can also be used.
2. Hash of the Contract: A Message digest created by using a cryptographic hash function. It is usually of fixed length and is unique to the data hashed. A change in hash generated would represent a change in the data hashed.
3. Random number *M*: Alice creates this random number that is used as a part of the contract signing process and for contract verification by Bob. Initially, Alice signs the packet *Z* which includes this random number *M* along with contract *C* and sends it to Bob. After receiving her signature, Bob still does not have her complete commitment over the contract as he does not know the value of *M* that was used to create the packet *Z*. Bob cannot find this value on his own since the packet *Z* is encrypted using the third party's public key and the third party's corresponding private key is a secret known only to it. Bob also cannot get it from Alice unless he provides his commitment of the same to Alice. Thus the only option available to Bob is to send his commitment to Alice. Alice then verifies his signature and if valid, sends him the values *M* and *R*. Upon receiving *M* and *R* from Alice, Bob can reconstruct *Z* using *A*, *B* and *H(C)*. Bob does so by encrypting these values using the third party's public key along with the random *R*. If the newly created *Z* matches with the one he received earlier, he has Alice's complete commitment on the contract.
4. The value *R***:** It is this randomness that is used by the public-key cryptosystem during encryption to produce the same cipher text for a given data using the same public key. This is usually not the case since most public-key cryptosystems produce different cipher texts under the above scenario. In our protocol, during the final step, comparison of cipher texts is performed for contract verification purposes. Thus the production of same cipher text over the same data and same public key becomes a requirement and can only be achieved if the randomness used for encryption is stored and reused.

Our Contribution

Our contribution in the revised protocol encompasses the following design changes:

Inclusion of hash of contract in *Z:* Information about the hash of the contract is added in the packet *Z*. A Hash is basically a fixed length value returned by a cryptographic hash function that takes the contract file data as the input. These hash values are usually unique for a given message (contract) and changes if the message is altered. If one party modifies the contract during the contract signing protocol, the hash generated on the modified contract will not match with the one generated on the original contract. If the hashes on this contract do not match between the parties, it can be concluded that they have different contracts and cheating can be identified. It can be concluded so as it is extremely unlikely to find the same hash on two different contracts. We specifically assume that SIG_A *()* and SIG_B *()* are secure signing algorithms that exhibit the following four properties:

1. It is easy to compute hash for any given data.
2. It is extremely difficult to recreate the data from its hash.
3. It is extremely unlikely to find the same hash for two different data.
4. It is extremely difficult to modify a given data without changing its hash.

Exchange of random nonce prior to exchange of commitments: Random nonces are exchanged between both parties prior to exchange of the contract commitments. This step ensures that both parties are certain about the other's identity. It also helps to identify a replay attack that may occur if an intruder tries to replay messages previously sent by Alice.

This can be achieved through the following three steps. Let the two parties be Alice and Bob,

1. Alice sends a random nonce to Bob.
2. Bob signs the nonce it received from Alice using his private key and sends his random nonce to her.
3. Alice verifies the signed value using Bob's public key. If valid, she signs his nonce using her private key and sends it to Bob. Bob can verify the same using her public key.

Inclusion of hash of contract in signature: Hash of the contract instead of the actual contract is used during the digital signature generation process. This step ensures privacy of the contract between both the parties. On the contrary, if the contract is part of the digital signature, during dispute resolution the cheated party would also have to provide the contract along with the other information to the third party. This is necessary since all information that is part of the signature is required in order to verify it which also includes the contract. If the secrecy of the contract is essential to the cheated party, executing this step would make them lose the same. Protocols by Micali and Bao both use the actual contract instead of its hash during the digital signature generation process.

Handling of Attacks

This section discusses how the attacks discovered in Micali's protocol are being handled by our protocol,

Insufficient Requirements for Dispute Resolution by the Third Party: In Micali's protocol, the third party only requires Bob's signatures *(*$SIG_B(C, Z)$ *and* $SIG_B(Z)$*)* during the dispute resolution phase. Nothing from Alice's side is required. This can lead to the following attack:

After receiving Alice's signature *(*$SIG_A(C, Z)$*)* dishonest Bob can prepare a new contract C^1, create the following signatures $SIG_B(C^1, Z)$ and $SIG_B(Z)$ and send them to the third party requesting for resolution. Since these two signatures are valid and third party does not require any signatures from Alice or even the contract, it forwards $SIG_B(C^1, Z)$ and $SIG_B(Z)$ to Alice and M to Bob. This result in Bob having Alice's commitment on contract C and Alice only having Bob's commitment on another contract C^1.

This attack is handled by our protocol since now the third party requires signatures of both parties participating in the contract signing process. Also, since hash of the contract file is now included in the packet Z, if Bob signs a fake contract, the hashes will change and the signature will not be verifiable.

Insufficient requirements for commitment of both parties: As per Micali's definition, for

Bob to be committed to the contract *C*, Alice only requires Bob's signatures on the contract *C* and packet *Z* that are $SIG_B(C, Z)$ and $SIG_B\ (Z)$. Alice is not required to provide the value *M* that creates *Z*. This flaw can be exploited such that Alice can always get Bob's commitment on a contract while Bob gets nothing. Following is the attack:

Dishonest Alice creates a random value of length *Z* and sends her signature of *(C, Z)* to Bob. Bob verifies Alice's signature and since it holds true, he sends his signatures of *(C, Z)* and *Z* to her. Alice now quits the protocol as she has received Bob's commitment. Bob on the other hand cannot get resolution form third party as *Z* is a random value and it cannot find *M* such that $Z = E^R_{TP}\ (A, B, M)$. This leads to Alice obtaining an advantage since she is not required to present a value *M* that can recreate the packet *Z*.

This attack is handled by our protocol by changing the requirements of contract commitment such that Alice is also required to provide a value of *M* and *R* such that $Z=E^R_{TP}\ (A, B, M)$.

Third party's dilemma: During the dispute resolution phase in Micali's protocol, it is not stated how the third party should react to a scenario where decrypting *Z* does not yield to the valid *A*, *B* and *M* values, that is, $D_{TP}\ (Z) \neq (A, B, M)$ for any *M*, even though the signatures $SIG_A\ (C, Z)$, $SIG_B\ (C, Z)$ and $SIG_B\ (Z)$ are valid. This indirectly means that the first two values of *Z* are not the identities of Alice and Bob. Bao's analysis on Micali's protocol show that either step taken by the third party leaves scope for possible attacks where one party can achieve an unfair advantage over the other.

Such attacks are handled by our protocol by changing the requirements for commitment of both the parties over the contract and by changing the contents of the packet *Z* to also include the hash of the negotiated contract *H(C)*. Bob is now is required to sign over Alice's signature which he received and not just the *H(C)* and *Z*. Therefore, by signing over Alice's signature and not just the *H(C)* and packet *Z*, Bob confirms his intention to commit over the contract with only Alice and not anyone else. Therefore now Alice cannot collude with A^1 and obtain Bob's commitment over the contract as Bob signs over Alice's signature and the packet *Z* contains A^1 as one of the parties and not Alice. This mismatch illustrates cheating and is not allowed by our protocol.

Replay Attacks: Even though the above mentioned attacks are addressed, both protocols (Micali and Bao) still leave one possible attack. These protocols provide no means of identifying a replay attack that may take place during a contract signing instance.

Let us assume an intruder has access to the transmissions between Alice and Bob and is able to record all the messages sent by Alice. An intruder can thus always replay a message that was previously sent by Alice. Bob will assume the request as genuine as its signature will still be valid and respond back with his signature of the same. This leads to signature of a new contract between Alice and Bob. Bob assumes that he has signed a new contract with Alice and she on the other hand knows nothing about it.

This attack is handled by our protocol since both parties are required to exchange random nonce between each other before they exchange their commitments. Therefore if an intruder replays a contract signing request sent previously by Alice, Bob would respond with a nonce which has to be signed using Alice's private key. Since Alice's private key is a secret known only to Alice, the intruder will not be able to provide the signature and the contract signing process would halt, preventing the replay attack.

To recognize a replay attack Bob can also recompute Z by using previously obtained values of *M* from Alice. If a match occurs Bob can conclude that it is a replay attack. However, only limited values of *M* (or contracts) can be accessible to a party in practice due to limits on database sizes. Our use of nonce removes this limitation and does not require storage of previous values of *M* or even the contract since the attack can be easily identified if nonce verification fails.

Role of Cryptography

In our protocol, cryptography is being used for two main purposes which include creation and verification of digital signatures and implementation of hybrid cryptography. Following is a brief discussion on both these purposes and their implementation in our system.

Digital Signatures

Digital signatures are derived from asymmetric cryptography where messages signed by a party using its private key can later by verified by anybody using the party's corresponding public key. This provides the property of non-repudiation where the signer cannot refuse to have signed the message since the private key used to sign the message is a secret known only to the signer. Digital signatures can be considered equivalent to traditional handwritten signatures in many aspects and when implemented properly are extremely effective. In a contract signing instance, the initiating party can sign messages using any of the various digital signature algorithms. This way during the transaction it can always be proved that the messages signed and sent by the initiating party indeed belong to it.

We implement a digital signature algorithm based elliptic curves known as Elliptic Curve Digital Signature Algorithm or ECDSA (ANSI X9.63, 1999) and our implementation produces a digital signature of 448 bits (Piotrowski, Langendoerfer & Peter, 2006) on the provided data.

Hybrid Cryptography

Our system uses hybrid cryptography for secure communication since it offers better efficiency and properties like data integrity and non-repudiation when compared individually to techniques like asymmetric and symmetric cryptography. Following is a brief discussion on both these forms of cryptography (asymmetric and symmetric) for a clear understanding of our implementation and its benefits.

Symmetric Cryptography: This process of cryptography uses the same key for encryption and decryption. The secrecy of the information is dependent on how well this secret key can be kept private. Compared to its counterpart *asymmetric cryptography* which is explained later, symmetric cryptography is fast and efficient. For this reason it is used widely for encryption and decryption of large files. Even though this process is highly efficient, it suffers from the following disadvantages:

- Key Sharing: Since an initial exchange of the secret key is required between the parties before they can begin encrypting and decryption data, safe transmission or sharing of this key becomes a problem.
- Key Management: A key is required to be shared between every two parties who are willing to exchange information securely. Therefore in a large network of users who want to exchange information with other users, a unique key is required for every user pair. This storage and management of keys become difficult for each user who wants to participate is such transactions.
- Integrity: Since the receiver cannot verify that the message has not been altered before receipt, the integrity of data can be compromised.
- Repudiation: Since the same secret key has to be shared between users, the sender can always repudiate the messages because there are no mechanisms for the receiver to make sure that the message has been sent by the claimed sender.

Asymmetric Cryptography

This process of cryptography also known as *public key cryptography* uses different keys for encrypting and decrypting information. These keys together form a key-pair and are known as public and private keys. Public keys are shared with everyone in the world. Private keys on the other are kept secret and known only to the individual to whom it belongs. For a party to send information securely to another, they need to encrypt the information using the recipient's public key. The recipient can then decrypt this data and recover the message by using their corresponding private key. Since this private key is a secret known only to the recipient, the information can be communicated securely without the requirement of initial key exchange thus handling the key sharing problem. Key management also becomes convenient since there are no unique keys that are required for each user pair willing to communicate. The user can simply use the recipient's public key and start secure communication. Non-repudiation and data integrity can be achieved by making the sender also encrypt or sign the message using their private key which can then be verified by anybody using the signer's corresponding public key.

Even though asymmetric cryptography offers features like non-repudiation and data integrity, its execution is still far slower than symmetric cryptography making it less favorable for encrypting and decrypting large files.

Hybrid Cryptography

Hybrid cryptography handles the above mentioned disadvantages in symmetric and asymmetric cryptography. It does so by using both these cryptosystems together which provides the convenience of asymmetric cryptography along with the efficiency of symmetric cryptography.

Hybrid cryptography consists of the following two stages:

1. **Data encapsulation:** The process in which data to be communicated securely is encrypted using symmetric cryptography schemes which are highly efficient.
2. **Key encapsulation:** The symmetric secret key used to encrypt the data is then encrypted using asymmetric cryptography schemes.

To encrypt a message to Bob, Alice performs the following steps:

1. Creates a random symmetric key and encrypts the message using the data encapsulation scheme.
2. Encrypts the symmetric key using Bob's public key under the key encapsulation scheme.
3. Sends both the encrypted message and the encrypted symmetric key to Bob.

To recover the message sent by Alice, Bob performs the following steps:

1. Use his private key to decrypt the encrypted symmetric key.
2. Use the recovered symmetric key to decrypt and recover the original message.

Since the major part of the transmission that includes the actual message is encrypted using symmetric cryptosystem, the efficiency of the system is improved. By encrypting only the symmetric key using asymmetric cryptography properties like key management, data integrity and non-repudiation can be addressed.

We use Elliptic Curve Integrated Encryption Scheme, also known as ECIES for performing asymmetric cryptography operations. We use Advanced Encryption Standard (AES) for symmetric cryptography. It is the current cryptography standard for symmetric cryptosystems and was announced by the National Institute of Standards and Technology (NIST) under the Federal Information Processing Standards (FIPS, 2001) 197 on November 26, 2001. It is also one of the most popular algorithms used for symmetric cryptography at present.

Elliptic Curve Cryptography Overview

Both Elliptic Curve Integrated Encryption Scheme (ECIES) and Elliptic Curve Digital Signature Algorithm (ECDSA) are part of cryptography that is based on elliptic curves. Also known as Elliptic Curve Cryptography or ECC, it is an approach to public key cryptography that is primarily based on elliptic curves which are defined over a finite field. A field is basically a mathematical group that offers operations for addition, subtraction, multiplication, and division that always construct results within the field. A finite field can be defined as a field that contains only finitely many elements. It is this property of being finite that makes it possible to perform cryptography with these elliptic curves that exists over the fields. The use of elliptic curves for cryptography was proposed independently by Neal Koblitz (1987) at the University of Washington and Victor Miller (1986) at IBM in 1985. Being grown into a mature public key cryptosystem, it is endorsed by the United States government (NIST, 1999).

The security of any cryptographic system is based on a hard mathematical problem that is computationally infeasible to solve. For example, RSA gets its security from the difficulty of factoring large numbers. The public and private keys used in RSA cryptography is a function of a pair of large prime numbers, and recovering the plain text from the cipher text that was created using the public key is believed to be computationally equivalent to finding the factors of the primes used to create the pair of public and private keys. Elliptic Curve Cryptography along with many other cryptographic systems achieves their security from the difficulty of solving the discrete logarithmic problem (DLP). ECC to be specific is based upon the difficulty of solving Elliptic Curve Discrete Logarithm Problem (ECDLP) which offers a better implementation when compared to previous generation techniques as used by RSA. ECDLP can be demonstrated with the help of the equation $Ax = B$. For very large values of x, it gets computationally infeasible to derive its value as no efficient algorithm is available for solving it. The primitive approach of solving this would be to keep adding and/or multiplying the value of A to itself until the result matches B. This approach is used on elliptic curve groups where, a point on the group is selected and multiplied by a scalar value. When the scalar value is very large, it becomes computationally infeasible to solve the problem. The primitive approach then becomes using the addition and doubling operations together until the matching value is observed. For example, $7P$ can be expressed as $2* ((2 * P) + P)) + P$. This calculation of a point nP is referred to as Scalar Multiplication of a point. ECDLP is based upon the intractability of scalar multiplication products.

Not all curves can provide strong security and that ECDLP for some curves can be resolved efficiently. Therefore NIST offers a set of recommended curves (NIST, 1999) whose security properties are well understood and can be safely used for cryptography. Standardization of elliptic curves also makes it convenient for interoperability and use by external third party cryptographic providers to provide cryptographic solutions that comply with the security standards.

Performance Improvements Using Elliptic Curve Cryptography

Majority of security systems still use first generation public key cryptographic algorithms like RSA and Diffie-Hellman (DH) which were developed in mid-1970. For these systems the current NIST recommended public key parameter size is 1024 bits. NIST states that these systems can be used securely till 2010 after which it is recommended to shift to other systems which provide better security. One alternative can be to keep increasing the bit size to higher values so that these systems can be used for some more time. Another option can be to shift to next generation cryptographic systems like elliptic curve cryptography which provide equivalent security for smaller key sizes and are also more efficient.

Following is Table 1 that compares ECC with schemes like RSA and Diffie-Hellman in terms of key sizes required for securing symmetric keys for varying length by NSA (2009).

We can see that NIST recommends 1024 bit key sizes for securing 80 bits symmetric keys. The same security can be provided by ECC by using 160 bit key size which makes ECC a better solution. Securing a 256 bit symmetric key would require a RSA key with the bit parameters of size 15,360 which is fifteen times the current size used in Internet today. Comparing to ECC, one would only require keys of size 521 bits which is far smaller.

ECC is also more computationally efficient when compared to RSA and Diffie-Hellman. Even though ECC has more complex arithmetic than RSA and DH, the security added per bit increase in key size does make for the extra time used to handle such complexity. Following is Table 2 that compares the computation required by ECC and schemes like Diffie-Hellman for varying key sizes by NSA (2009).

We can see that as the security level based on key sizes is increased, the difference in the computation required increases at a higher rate which makes ECC much more efficient than the first generation cryptographic algorithms.

Table 1. NIST recommended key sizes

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Due to the above mentioned reasons which include smaller key sizes, better computational efficiency and greater security, Elliptic Curve Cryptography is considered as a better solution when compared to first generation techniques like RSA and DH. National Security Agency has also decided to move to Elliptic curve cryptography for protecting both classified and unclassified national security information (NSA, 2009).

CONCLUSION

In this journal we examine the notion of fair exchange and its applications. We analyze one exchange protocol by Micali that provides fair contract signing based on the concept of fair exchange. The protocol claims fairness by providing resolution if a party refuses to provide their commitment. We show that Micali's protocol is not completely fair and demonstrate scenarios where one party can cheat the other without being identified. We discuss Bao's protocol which is a revised version of Micali's protocol that provides superior fairness by handling certain types of weaknesses. Both these protocols fail to handle the possibilities of a replay attack. Our prior work improves upon these protocols by addressing the weakness that leads to a replay attack. This present work extends our prior work on fair electronic exchange by handling a type of attack which was not handled earlier and also provides a brief survey of the recent work related to the field of

Table 2. Relative computation costs of Diffie-Hellman and Elliptic Curves

Security Level (bits)	Ratio of DH Cost: EC Cost
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

fair electronic exchange. In addition to our protocol, our implementation of hybrid cryptography and digital signature algorithms which provide features like confidentiality, data-integrity and non-repudiation are also discussed.

REFERENCES

ANSI X9. 63. (1999). *Elliptic Curve Key Agreement and Key Transport Protocols*. Washington, DC: American Bankers Association.

Asokan, N., Schunter, M., & Waidner, M. (1997). Optimistic protocols for fair exchange. In T. Matsumoto (Ed.), *4th ACM Conference on Computer and Communications Security*. (pp. 6, 8–17). Zurich, Switzerland: ACM Press.

Asokan, N., Shoup, V., & Waidner, M. (1998). Optimistic fair exchange of digital signatures. *Advances in Cryptology-EUROCRYP*, *T98*, 591–606.

Asokan, N., Shoup, V., & Waidner, M. (2000). Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, *18*(4), 591–606. doi:10.1109/49.839935

Ateniese, G. (1999). Efficient verifiable encryption (and fair exchange) of digital signatures. *Proceedings of the 6th ACM conference on Computer and communications security* (pp. 138-146). ACM.

Ateniese, G., & Nita-Rotaru, C. (2002). Stateless-recipient certified e-mail system based on verifiable encryption. *Topics in Cryptology-CT-RSA*, *2002*, 182–199.

Bao, F., Deng, R. H., & Mao, W. (1998). Efficient and Practical Fair Exchange Protocols with Off-line TTP. *Proceedings of IEEE Symposium on Security and Privacy*, (pp. 77-85).

Bao, F., Wang, G., Zhou, J., & Zhu, H. (2004). Analysis and Improvement of Micali's Fair Contract Signing Protocol. *Information Security and Privacy*, *3108*, 176–187. doi:10.1007/978-3-540-27800-9_16

Baum-Waidner, B. (2001). *Optimistic asynchronous multi-party contract signing with reduced number of rounds. ICALP'01, LNCS 2076* (pp. 898–911). Berlin: Springer.

Bedi, H., & Yang, L. (2009). Fair Electronic Exchange Based on Fingerprint Biometrics. [IJISP]. *International Journal of Information Security and Privacy*, *III*(3), 76–106.

Bedi, H., Yang, L., & Kizza, J. (2009). Extended Abstract: Fair Electronic Exchange using Biometrics. *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies* (pp. 1-4). Oak Ridge: ACM.

Ben-Or, M., Goldreich, O., Micali, S., & Rivest, R. (1990). A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, *36*(1), 40–46. doi:10.1109/18.50372

Boyd, C., & Foo, E. (1998). *Off-line fair payment protocols using convertible signatures. Advances in Cryptology—ASIACRYPT'98* (pp. 271–285). New York: Springer.

Coffey, T., & Saidha, P. (1996, Jan.). Non-repudiation with mandatory proof of receipt. ACM SIGCOMM. *Computer Communication Review*, *26*(1), 6–17. doi:10.1145/232335.232338

Cox, B., Tygar, J. D., & Sirbu, M. (1995). NetBill security and transaction protocol. In *Proc. 1st USENIX Workshop on Electronic Commerce*, 77–88.

Damgard, I. B. (1995). Practical and provably secure release of a secret and exchange of signatures. *Journal of Cryptology*, *8*, 201–222. doi:10.1007/BF00191356

Deng, R., Gong, L., Lazar, A., & Wang, W. (1996). Practical protocol for certified electronic mail. *Journal of Network and Systems Management*, *4*(3), 279–297. doi:10.1007/BF02139147

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. doi:10.1109/TIT.1976.1055638

Ferrer-Gomila, J. L., Payeras-Capella, M., & Huguet-Rotger, L. (2001). Efficient optimistic n-party contract signing protocol. *Information Security Conference, LNCS 2200*,pp. 394-407, Berlin: Springer.

FIPS. (2001, November). *Federal Information Processing Standards.* Retrieved from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Garay, J., Jakobsson, M., & MacKenzie, P. (1999). Abuse-free optimistic contract signing. *Advances in Cryptology—CRYPTO'99*, (pp. 449--466).

Garay, J., & MacKenzie, P. (1999). Abuse-free multi-party contract signing. *1999 International Symposium on Distributed Computing, LNCS 1693*, pp. 151-165, Berlin: Springer.

Gurgens, S., Rudolph, C., & Vogt, H. (2003). On the security of fair non-repudiation protocols. *Information Security Conference (ISC)*, LNCS 2851, pp. 193-207. Springer-Verlag, 2003.

Imamoto, K., & Sakurai, K. (2002). A certified e-mail system with receiver's selective usage of delivery authority. *Progress in Cryptology-INDOCRYPT*, *2002*, 326–338.

ISO/IEC 13888-1 (1997a). *Information technology - Security techniques - Non-repudiation - Part 1: General*

ISO/IEC 13888-2 (1998). Information technology - Security techniques - Non-repudiation - Part 2: Mechanisms using symmetric techniques

ISO/IEC 13888-3 (1997b). Information technology - Security techniques - Non-repudiation - Part 3: Mechanisms using asymmetric techniques

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, *48*, 203–209. doi:10.1090/S0025-5718-1987-0866109-5

Kremer, S., & Markowitch, O. (2000). Optimistic non-repudiable information exchange. In J. Biemond (Ed.), *21st Symp. on Information Theory in the Benelux, Werkgemeenschap Informatie- en Communicatietheorie*. Enschede (NL), Wassenaar (NL), 2000, (pp. 139–146).

Kremer, S., & Markowitch, O. (2001). Selective receipt in certified e-mail. *Progress in Cryptology-INDOCRYPT*, *2001*, 136–148.

Kremer, S., Markowitch, O., & Zhou, J. (2002). An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25 (17), 1606-1621. Elsevier, Nov. 2002.

Markowitch, O., & Roggeman, Y. (1999). Probabilistic non-repudiation without trusted third party. *In Second Conference on Security in Communication Networks'99*, Amalfi, Italy.

Micali, S. (1997). *Certified E-mail with invisible post offices*. Presented at the RSA '97 conference (1997).

Micali, S. (2003). Simple and Fast Optimistic Protocols for Fair Electronic Exchange. In *Proceedings of the ACM Symposium on Principles of Distributed Computing,*pp. 12-19.

Miller, V. (1986). *Uses of elliptic curves in cryptography", Lecture Notes in Computer Science 218: Advances in Cryptology - CRYPTO '85* (pp. 417–426). Berlin: Springer-Verlag.

NIST. (1999, July). *Recommended Elliptic Curves for Federal Government Use*. Retrieved from http://csrc.nist.gov/csrc/fedstandards.html

NSA. (2009). *The Case for Elliptic Curve Cryptography*. Retrieved from http://www.nsa.gov/business/programs/elliptic_curve.shtml

Park, J. M., Chong, E. K., & Siegel, H. J. (2003). Constructing fair exchange protocols for e-commerce via distributed computation of RSA signatures. *Proceedings of 22nd Annual ACM Symposium on Principles Distributed Computing (PODC)*, pp. 172-181. New York: ACM press.

Piotrowski, K., Langendoerfer, P., & Peter, S. (2006). How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks.*

Ray, I., & Ray, I. (2002, May). Fair Exchange in E-commerce. *ACM SIGecom Exchange*, *3*(2), 9–17. doi:10.1145/844340.844345

Tedrick, T. (1983). How to exchange half a bit. In Chaum, D. (Ed.), *Advances in Cryptology: Proceedings of Crypto 83* (pp. 147–151). New York, London: Plenum Press.

Tedrick, T. (1985). Fair exchange of secrets. In G. R. Blakley, D. C. Chaum (Eds.), *Advances in Cryptology: Proceedings of Crypto 84, Vol. 196 of Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, (pp. 434–438).

Zhang, N., & Shi, Q. (1996). Achieving non-repudiation of receipt. *The Computer Journal*, *39*(10), 844–853. doi:10.1093/comjnl/39.10.844

Zhou, J., Deng, R., & Bao, F. (1999). Evolution of fair non-repudiation with TTP. *ACISP: Information Security and Privacy: Australasian Conference*. (Vol. 1587 of Lecture Notes in Computer Science)Springer-Verlag, 1999, (pp. 258–269).

Zhou, J., & Gollmann, D. (1996). A fair non-repudiation protocol. In *IEEE Symposium on Security and Privacy, Research in Security and Privacy*, IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Security Press, Oakland, CA, 1996, (pp. 55–61).

Zhou, J., & Gollmann, D. (1997). An efficient non-repudiation protocol. In *Proceedings of The 10th Computer Security Foundations Workshop,* IEEE Computer Society Press, 1997, (pp. 126–132).