ABSTRACT
Fair exchange between two parties can be defined as where: either both parties acquire what they expected or neither does. Protocols that facilitate such transactions are known as “fair exchange protocols”. We analyze one such protocol for contract signing that was presented by Micali. In this paper we show that Micali’s protocol is not completely fair and demonstrate the possibilities for one party to cheat by obtaining the other party’s commitment and not offer his. A revised version of this protocol by Bao et al. provides improved fairness by handling the above mentioned attack but does not handle replay attacks. Our proposed protocol improves upon the revised protocol by addressing replay attacks and removing redundant information from signatures thus making the overall transmission smaller. Our protocol also provides client authentication using a fingerprint based authentication technique and our use of hybrid encryption using Elliptic Curves instead of first generation techniques further improves efficiency.

Categories and Subject Descriptors

General Terms
Algorithms, Performance, Design, Reliability, Security, Theory, Legal Aspects, Verification

Keywords

1. INTRODUCTION
1.1 Fair Electronic Exchange
Properties like fairness and confidentiality are very important in protocols that facilitate electronic transactions. Fairness in this context means two mistrusting parties involved in a transaction should either get what they deserve or nothing at all. None of the parties should get an unfair advantage over the other after the transaction is complete or even if it is prematurely aborted. Also if a party cheats, there should be means for providing fairness to the cheated party. Protocols that provide such mechanisms and features are termed as “Fair Exchange Protocols” and can be used for various purposes like,

1. Certified E-Mail (CEM) where Alice sends a message to Bob and gets a receipt from him in return.
2. Electronic Contract Signing (ECS) where both Alice and Bob wish to sign a contract. This would involve Alice sending her signature of the contract to Bob and him sending his signature of the same in return. This contract signing protocol is possible between two parties. However, various multi-party contract signing protocols have also been proposed in the following papers [4, 5, 6].
3. 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 [7].

Fair Exchange Protocols with External Fairness Providers:
Fair exchange becomes difficult when honesty of both parties and security and resilience of the communication channel cannot be guaranteed. Fair exchange in such cases can be achieved by use of external fairness providers like a trusted third party or a human judge that can process the transaction and provide fairness. Fair exchange protocols based on their procedure of dispute resolution are classified into the following categories:

The first category of protocols depends on gathering evidence during the transaction that can later be used to provide fairness. Such protocols are classified as weak fair-exchange protocols. The drawback of such protocols is that they provide resolution only after the transaction has been completed. In case of e-commerce where the location and availability of parties taking part in a transaction is not always fixed or known, such methods of dispute resolution may not always be possible.
The second category of protocols handles the issue above faced by providing means of obtaining resolution during the same transaction. Such protocols are known as "strong-fair exchange" protocols [8] and use an entity that is known as the trusted third party (TP) to provide fairness.

The third category of protocols provides a measure of fairness which is directly proportional to the number of rounds of messages passed between the two transacting parties and is known as gradual exchange protocols [e.g. 9]

Trusted third parties based on their mode of operation are classified into the following types:

Trusted parties that are required to involve in every transaction occurring between any two parties are known as online trusted third parties. This creates a bottleneck in the network when there are a large number of users making many transactions.

Trusted parties that are not required to involve in every transaction but only when a dispute occurs are known as invisible trusted third parties; and the protocols implementing them are known as Optimistic Protocols. This type of protocols were first conceived by Ben-Or, Goldreich, Micali and Rivest [10]. Invisible TP are advantageous in that they do not create any bottleneck since they are involved only when cheating occurs. Due to this reason they also generate minimal expense and liabilities.

2. RELATED WORK

Our fair exchange protocol is based on the one proposed by Micali. In 2003, Micali presented a protocol for contract signing during the ACM Symposium on PODC [1, 2]. In his protocol, contract signing was provided in a fair way by using an invisible trusted third party. However, the protocol fails to achieve the claimed fairness when executed under certain scenarios. Following is the actual protocol for contract signing as proposed by Micali.

2.1 Micali’s Electronic Contract Signing Protocol (ECS1):

Pre-requisites: Alice selects the file which is the contract C that she needs to sign with Bob. Alice also selects a random value M and uses it to create \( Z = E_{TP}(A, B, M) \)

\( E_{TP} \) is performing encryption using the trusted third party’s (TP) public key and the randomness source R. A, B are the identifiers of Alice and Bob respectively. M is a random value that will be used later for verification purpose.

Protocol:

1. Alice sends her signature of Z and C to Bob.
   \( A1: A \rightarrow B: \ SIG_A(C, Z) \)

2. Upon receiving Alice’s message, Bob sends his signature of (C, Z) and Z to Alice.
   \( B1: B \rightarrow A: \ SIG_B(C, Z) + SIG_B(Z) \)

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.
   \( A2: A \rightarrow B: \ M \)

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 contact signing protocol is complete. Else, Bob sends his signature of (C, Z) and Z to the trusted third party.
   \( B2: B \rightarrow TP: \ SIG_B(C, Z) + SIG_B(Z) \)

5. Third Party verifies both the signatures it received from Bob. If they are valid TP decrypts Z using its private key and sends \( SIG_B(C, Z) + SIG_B(Z) \) to Alice and M to Bob.
   \( TP1: TP \rightarrow A: \ SIG_B(C, Z) + SIG_B(Z) \)
   \( TP2: TP \rightarrow B: \ M \)

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

Alice’s commitment to contract C:
\( SIG_A(C, Z) \) and M

Bob’s commitment to contract C:
\( SIG_B(C, Z) + SIG_B(Z) \)

2.2 Analysis

This section talks about the vulnerabilities in Micali’s contract signing protocol and how it is unfair in certain scenarios.

2.2.1 Insufficient Requirements for Dispute Resolution:

TP only requires Bob’s signatures \( (SIG_B(C, Z) \) and \( SIG_B(Z) \) during the dispute resolution phase and verifies nothing from Alice’s side. This can cause the following attack:

After receiving Alice’s signature \( (SIG_A(C, Z)) \), dishonest Bob can prepare a new fake contract C\(^1\) and create the following signatures \( SIG_B(C, Z) \) and \( SIG_B(Z) \) and send them both to TP. Since these two signatures are indeed valid (signed by Bob) and TP does not require any signatures from Alice, it forwards \( SIG_B(C, Z) \) and \( SIG_B(Z) \) to Alice and M to Bob.

This result in Bob having Alice’s commitment on contract C and Alice having Bob’s commitment on a fake contract C\(^1\).

This attack can be handled by changing the requirements of TP for dispute resolution to also include Alice’s signatures in Bob’s requests.

2.2.2 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 (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, 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 TP as Z is just a random value and it cannot find a value M such that \( Z = E_{TP}^R(A, B, M) \).
This attack can be handled by changing the requirements of commitments such that Alice is also required to provide a value M such that \( Z = E_{TP}^a (A, B, M) \).

The mentioned attacks are addressed in the paper by F. Bao et al [3]. This paper also handles these attacks by changing the requirement of dispute resolution phase, commitment parameters for both parties and contents of Z.

The ability to recognize a replay attack is not available in both protocols (Micali and F. Bao et al. [3]). If an old message is replayed by an intruder; the other party has no means of recognizing whether it’s a genuine contract signing request or a replay attack.

3. OUR SYSTEM

3.1 Fair Contract Signing Protocol – Without Privacy (ECS)

Our protocol is a fair contract signing protocol that uses an invisible trusted third party. Following is an adaptation of our protocol where privacy of messages is not essential. This approach is taken for simple understanding of the protocol.

Privacy of messages can be easily achieved by encrypting them with other party’s public key before transmission. A combination of asymmetric key encryption and symmetric key encryption is used for better performance. Our system uses Advanced Encryption Standard (AES) for symmetric cryptography and Elliptic Curve Integrated Encryption Scheme (ECIES) for asymmetric cryptography. Elliptic Curve Digital Signature Algorithm (ECDSA) is used for creation and verification of digital signatures. Messages are thus encrypted using symmetric key encryption and only the symmetric key is encrypted using asymmetric encryption providing better efficiency. This type of encryption is known as Hybrid Encryption.

Prerequisites: Alice selects a file which is the contract C that she needs to sign with Bob and creates a hash of it H(C). Alice also selects two random values M and R and uses them to create Z = \( E_{TP}^b (A, B, H(C), M) \)

Protocol:

1. Alice sends a nonce \( NA_1 \) to Bob. Nonce is a random number used only once for prevention of replay attacks.
   
   \( A_1: A \rightarrow B: NA_1 \)

2. Upon receiving Alice’s nonce, Bob signs it using his private key and sends it back to Alice 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.
   
   \( B_1: B \rightarrow A: SIG_B (NA_1) + NB_1 \)

3. After receiving Bob’s message, Alice verifies the signature of Bob and if it matches, she signs her nonce so that he can be sure of the same. Alice also signs Z along with H(C) and sends it all to Bob.
   
   \( A_2: A \rightarrow B: SIG_A (NB_1) + SIG_A (H(C), Z) + Z + C \)

4. After receiving Alice’s messages, Bob verifies them and if they match he is sure that it is not a replay attack. Bob now sends his commitment to Alice by signing H(C) with Z.
   
   \( B_2: B \rightarrow A: SIG_B (H(C), Z) \)

5. After receiving, Alice verifies Bob’s signature and if it holds true, sends him the values M and R signed by her.
   
   \( A_3: A \rightarrow B: SIG_A (M, R) + M + R \)

Dispute Resolution Phase:

6. Bob receives the message and re-creates Z using M and R. If this newly created Z matches with the one he received in Step 3, he halts and the contract signing is complete. Else he sends his and Alice’s signatures along with C and Z to TP.

   \( B_1: B \rightarrow TP: Z + C + SIG_A (H(C), Z) + SIG_B (H(C), Z) \)

7. Third party, upon receiving Bob’s request computes H(C) from C. Extracts M from Z. Verifies the contents of Z to include A, B and H(C). And verifies signatures of both parties. If valid, it sends M to Bob and Bob’s commitment to Alice. Else it takes no action.

   If contents of Z are legit and signatures are valid:

   \( TP_1: TP \rightarrow B: M \)

   \( TP_2: TP \rightarrow A: SIG_B (H(C), Z) \)

   The new commitments for both parties are defined as follows:

   Alice’s commitment to contract C:

   \( SIG_A (H(C), Z), M \) and R

   Bob’s commitment to contract C:

   \( SIG_B (H(C), Z), M \) and R

   Where \( Z = E_{TP}^a (A, B, H(C), M) \)

3.2 Improvements by our protocol

Replay Attacks:

Both protocols (Micali and Bao) provide no means of identifying a replay attack that may occur during a contract signing instance. Therefore one of them – Micali - is used for illustration.

Consider the normal execution:

\[ A_1: A \rightarrow B: SIG_A (C, Z) \]

\[ B_1: B \rightarrow A: SIG_B (C, Z) + SIG_B (Z) \]

\[ A_2: A \rightarrow B: M \]

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

Consider the following execution of this protocol by an intruder:

\[ I_1: I \rightarrow B: SIG_A (C, Z) \]

\[ B_1: B \rightarrow A: SIG_B (C, Z) + SIG_B (Z) \]

\[ I_2: I \rightarrow B: M \]

If the intruder replays the message A1 as a new request, there is no way for Bob to identify the sender. Bob will assume the request as genuine as its signature of Alice will still be valid and will therefore respond back with its signature of the same. To this, the intruder can then send the message A2. This leads to 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.

If the contract contains a time stamp, expiration date that determines its freshness, such an attack could be identified. It could be argued that such information is present in any written contract. However, this should be specifically stated as part of the protocol and not left for the user to assume. Both protocols do not state this as part of their protocol specification.
Consider a scenario where Alice periodically purchases selected items from Bob using the above protocol. Then the contracts signed by them would also be the same, which is also the case in real world transactions. 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) then it would be hard to trace the intruder or identify the attack. It should also be noted that the intruder does not need to involve the TP to stage this attack thus making it further hard to identify.

Our protocol handles this attack by making parties exchange nonce before exchanging commitments. Making exchange of nonce as separate steps ensures that commitments are exchanged only when a party is sure of the other’s identity. Thus the protocol can be aborted in the beginning if nonce verification fails and commitment need not be generated or exchanged thus preventing replay attacks and making the system proactive in nature.

To recognize a replay attack Bob can also re-compute 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 limitations in database sizes. Our use of nonce removes this limitation and does not require storage of previous values of M since the attack can be easily identified if nonce verification fails.

Reduction in signature sizes and parameters:
Both protocols require the transmission of contract C twice between the parties. Our protocol does so only once followed by use of its hash for further communications. This leads to savings in network bandwidth and time for transmitting large contracts.

The new commitments of both parties in Bao’s revised protocol require signatures to include the following parameters: A, B, TP, C, and Z. The parameters A and B in these signatures are redundant and provide no apparent benefits in terms of security or fairness since Z already includes them. Our protocol does not include them, thus reducing redundancy and transmission size.

3.3 Use of Biometrics and Cryptography:
Properties like user authentication increase the robustness of a system that implements such fair exchange protocols and creates a complete system that can be used for making end-to-end transactions.

Our system uses a minutiae based fingerprint authentication system. Minutiae are distinct features found on a fingerprint such as ridge endings or bifurcations [12]. Compared to its competitive graph-based technique, the amount of resources required for storage and processing are less [11]. Fingerprints have a property of being unique and immutable, allowing for a rather small False Acceptance Rate (1%) and False Rejection Rate (0.1%). The extracted minutiae cannot be used to recreate the original fingerprint making it a one way procedure that increases security.

Our system uses ECIES for asymmetric cryptography. The elliptic curve used by our system is “prime192v1” which is a standard and is specified in X9.62 and FIPS PUB-186-2. ECDSA-224 that produces a 448 bit signature is used for creation and verification of digital signatures. NIST recommends RSA 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 [13].

4. CONCLUSION
Our contract signing protocol addresses a type of attack where an intruder can replay messages sent earlier by Alice and trick Bob in committing to the replayed contract without Alice’s or TP’s knowledge. It removes redundant parameters from the signatures of both parties and requires the original contract to be only transmitted once making the overall transmission packet small. The improvements provided by Bao’s revised protocol are also covered in our protocol. Our system also includes Minutiae based fingerprint authentication scheme where users are first authenticated before attempting to begin the contract signing process. Our implementation of a hybrid cryptosystem which includes combination of ECIES and AES provides better performance when compared to first generation schemes like RSA and DES. Use of ECDSA for creation and verification for digital signatures further improves performance.

5. REFERENCES