International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 8, October - 2012

Rsa Based Solution For Fair Contract Signing

Rajasree R.S.

.

ABSTRACT

*A fair contract signing protocol allows two potentially mistrusted parties to exchange their commitments to an agreed contract over the Internet in a fair way so that either each of them obtains the others signature or neither party does. Based on the RSA scheme a new digital contract signing is proposed. The proposed protocol satisfies new property abuse freeness. That is if the protocol is executed unsuccessfully, none of the two parties can show the validity of intermediate results to others. Here the first abuse free fair contract signing protocol based on the RSA signature is presented and it is showed that it is both secure and efficient.***

Introduction

Contract signing plays a very important role in any business transaction, in particular in situations where the involved parties do not trust each other to some extent already. Contract signing is truly simple due to the existence of "simultaneity." That is, both parties generally sign two hard copies of the same contract at the same place and at the same time.

After that, each party keeps one copy as a legal document that shows both of them have committed to the contract. If one party does not abide by the contract, the other party could provide the signed contract to a judge in court. As electronic commerce is becoming more and more important and popular in the world, it is desirable to have a mechanism that allows two parties to sign a digital contract via the Internet.However; the problem of contract signing becomes difficult in this setting, since there is no simultaneity any more in the scenario of computer networks. In other words, the simultaneity has to be mimicked in order to design a digital contract-signing protocol. Information is exchanged in computer networks nonsimultaneously, so at least an unfair state must be passed through. From the view point of technique, the problem of digital contract signing belongs to a wider topic: fair exchange. Actually, fair exchange includes the following different but related issues: contract-signing protocols, certified e-mail systems nonrepudiation protocols and e-payment schemes in electronic commerce .In this paper, the problem of digital contract signing between two parties is focused. Since a party's commitment to a digital contract is usually defined as his/her digital signature on the contract, digital contract signing is essentially implied by fair exchange of digital signatures between two potentially mistrusted parties. There is a rich history of contract signing (i.e., fair exchangeof digital signatures) because this is a fundamental problem in electronic transactions.

2 Existing System

According to the involvement degree of a trusted third party (TTP), contract-signing protocols can be divided into three types: 1) gradual exchanges without any TTP; 2) protocols with an on-line TTP; and 3) protocols with an off-line TTP. Early efforts mainly focused on the first type of protocols to meet computational fairness: Both parties exchange their commitments/secrets "bit-by-bit."If one party stops prematurely, both parties have about the same fraction of the peer's secret, which means that they can complete the contract off-line by investing about the same amount of computing work, e.g., exclusively searching the remaining bits of the secrets.

The major advantage of this approach is that no TTP is involved. However, this approach is unrealistic for most real-world applications due to the following reasons. First of all, it is assumed that the two parties have equivalent or related computation resources. Otherwise, such a protocol is favorable to the party with stronger computing power, who may conditionally force the other party to commit the contract by its own interest. At the same time, such protocols are inefficient because the costs of computation and communication are extensive. In addition, this approach has the unsatisfactory property of uncertain termination. In the second type of fair exchange protocols an on-line TTP is always involved in every exchange. In this scenario TTP is essentially a mediator: a) Each party first sends his/her item to the TTP; b) then, the TTP checks the validity of those items; c) if all expected items are correctly received, the TTP finally forwards each item to the party who needs it. Contract-signing protocols with an on-line TTP could be designed more easily since the TTP facilitates the execution of each exchange, but may be still expensive and inefficient because the TTP needs to be paid and must be part of every execution.

3 PROPOSED SYSTEM

This paper shows the importance of abuse-freeness and security in contract signing by proposing a new contract-signing protocol for two mutually distrusted parties. The protocol is based on an RSA multisignature, which is formally proved to be secure .This protocol is fair and optimistic. Furthermore, different from the above existing schemes,theprotocol is abuse-free.

3.1 Fairness

Our protocol guarantees the two parities involved to obtain or not obtain the other's signature simultaneously.This property implies that even a dishonest party who tries to cheat cannot get an advantage over the other.

3.2 Optimism

The TTP is involved only in the situation where one party is cheating or the communication channel is interrupted.

3.3 Abuse-Freeness

If the whole protocol is not finished successfully,any of the two parties cannot show the validity of the intermediate results generated by the other to an outsider,either during or after the procedure where those intermediate results are produced.

3.4 Provable Security

Under the standard assumption that the RSA problem is intractable, the protocol is provably secure in the random hash function model, where a hash function is treated as if it were a "black box "containing a random function.

Timely Termination:

The execution of a protocol instance will be terminated in a predetermined time. This property is implemented by adding a reasonable deadline in a contract. If one party does not send his/her signature to the other party after the deadline , both of them are free of liability to their partial commitments to the contract and do not need to wait any more.

6) **Compatibility**:

In our protocol, each party's commitment to a contract is a standard digital signature. This means that to use the protocol in existing systems, there is no need to modify the signature scheme or message forma at all. Thus, it will be very convenient to integrate the contract-signing protocol into existing software for electronic transactions.

7) **TTP's Statelessness**:

To settle potential disputes between users, the TTP is not required to maintain a database to searching or remembering the state information for each protocol instance, so the overhead on the side of the TTP is reduced greatly.

8) **High Performance**:

In a typical implementation, the protocol execution in a normal case requires only interaction of several rounds between two parties, transmission of about one thousand bytes of data, and computation of a few modular exponentiations by each party.

Existing System

Alice sets an RSA modulus n=pq whwre p and q are two safe k bit primes and sets her public key $e \in_R Z^*_{\Phi(n)}$, and calculates her private key

$$d=e^{-1} \bmod \Phi(n) \qquad (1)$$

where mod$\Phi(n)$ is Euler's totient function. Then, she registers her public key with a certification authority (CA) to get her certificate .After that, Alice randomly splits d into

d_1 and d_2 so that $d=d_1+d_2$, where $e \in_R Z^*_{\Phi(n)}$. To get a voucher V_A from a TTP, Alice is required to send $(CA,e_1,d_2)$to the TTP, where

$$e_1=d_1^{-1} \bmod \Phi(n) \quad (2)$$

The voucher is the TTP's signature that implicitly shows two facts:

1)e_1 can be used to verify a partial signature generated by using secret key d_1, and

2) the TTP knows a secret d_2 matches with RSA key pairs $(d_1,e_1)$and (d,e) .When Alice and Bob want to exchange their signatures on a message m, Alice first computes

$$\mu 1=h(m)d1 \bmod n \quad (3)$$

andsends (CA,VA,μ_1) to Bob, where h(.)is a secure hash function.Upon receiving , Bob checks the validity of CAand VA, and whether $h(m)=\mu_1^{e1} \bmod n$. If all those verificationsgo through, Bob returns his signature μ_B to Alice, since he is convinced that the expected

$$\mu_2=h(m)^{d2} \bmod n \quad (4)$$

can be revealed by Bob or the TTP. After receiving valid μ_B Alice reveals $\mu_2=h(m)^{d2} \bmod n$ to Bob. Finally, Bob obtains Alice'signature μ_A for message by setting , $\mu_A= \mu_1\mu_2$since we have

$$h(m)\Xi\mu_A^e=h(m)^{(d1+d2)e}=h(m)^{de} \bmod n. \quad (5)$$

The security problem in Park 'sscheme is that an honest-but-curious TTP can easily derive Alice's private key d.The reason is that with the knowledge of (n,e,e_1,d_2), the TTP knows that the integer $e-(1-ed_2)e_1$ is a nonzero multiple of$\Phi(n)$. It is well known that knowing such a multiple of $\Phi(n)$,Alice's RSA modulus n can be easily factored. Consequently, the TTP can get Alice's private key by the extended Euclidean algorithm.

4 TRAPDOORCOMMITMENT SCHEMES

A cryptographic primitive, called trapdoor commitment schemes has been used in order to achieve abuse-freeness.

Strong RSA-Based Trapdoor Commitment Scheme

The following RSA-based efficient trapdoor commitment scheme

1) TCgen: The receiver Bob first generates two large primes P_b and q_B ,, sets an RSA modulus $n_{B=}$ p_b q_B,, selects a random number s, picks a 160-bit prime number u,such that $GCD(u,\Phi(n_B))=1$, and selects a collision-resistant hash function h_2.Then, TCgen outputs the commitment public key p_k and trapdoor t_d

2)TCcom : To commit to a string r with arbitrary length, the sender sends the receiver com $\bar{r}=s^{h2(r)}t^u \bmod n_B$

3)TCver : To decommit$\bar{r}$ the sender reveals (r,t), so that the receiver can check if $\bar{r}=s^{h2(r)}t^u \bmod n_B$

4) TCSim: Given an answer (r,t)to a commitment com=$\bar{r}$,by using the trapdoor $\mu_B$receiver Bob can decommit $\bar{r}$ w.r.t. any string r1 .

5 REGISTRATION PROTOCOL

1) Alice first sets an RSA modulus n=pq, where p and q are two -bit safe primes, i.e., there exist two primes p' and q' such that $p=2p'+1$and $q=2q'+1$. Then, Alice selects her random public key e, and calculatesher private key

$$d=e^{-1} \bmod \Phi(n),$$

where $\Phi(n)=(p-1)(q-1)$.Finally, Alice registers her public key with a CA to get her certificate CA, which binds her identity and the corresponding pubic key (n,e)together.

2) Alice randomly splits d into d1and d2 such that $d= d1+ d2 \bmod \Phi(n)$ and computes $e_1 = d_1^{-1} \bmod \Phi(n)$.Then, Alice sends $(CA,w,\mu_w,d_2)$to the TTP but keeps (d,d_1,d_2,e) secret.

3) The TTP first checks that Alice's certificate CA is valid.After that, the TTP checks that the triple (w,μ_w,d_2) is prepared correctly. If everything is in order, the TTP stores d_2 securely, and creates a voucher VAby computing

$$VA =SignTTP(CA,w,\mu_w) \quad (6)$$

5.1 Signature Exchnage Protocol

1) First, the initiator Alice computes her partial signature $\mu_1=h(m)^{d1}$and then sends the triple (CA,VA,µ1) to the responder Bob. Here,h(.) is a cryptographically secure hash function.

2) Upon receiving $(C_A,V_A,\mu_{1)}$ Bob first verifies thatC is Alice's certificate issued by a CA, and that V_A is Alice's voucher created by the TTP. Then, Bob checks if the identitiesof Alice, Bob, and the TTP are correctly specified as part of the contract m. If all those validations hold, Bob initiates the following interactive zero-knowledge protocol with Alice to check whether µ1is indeed Alice's valid partial signature on contact m.

a) Bob picks two numbers i,j at random,and sends a challenge c to Alice by computing $c=\mu_1{}^{2i}\mu_w{}^j$ mod n.

b) After getting the challenge c, Alice calculates the respondence $r=c^{e1}$ mod n, and then returns her commitment $\bar{r}$=TCcom(r,t) to Bob by selecting a random number t, where TCcom is the commitment algorithm of a secure trapdoor commitment scheme which depends on Bob's public key.

c) When the commitment $\bar{r}$ is received, Bob sends Alice the pair(i,j) to show that he prepared the challenge c properly.

d) Alice checks whether the challenge c is indeed prepared correctly, i.e., $c=\mu_1{}^{2i}\mu_w{}^j$ mod n.. If the answer is positive, Alice decommits the commitment $\bar{r}$ by revealing the respondence (r,t) to Bob.With the knowledge of (r,t), Bob accepts μ_1 as valid if and only if

$$r=h(m)^{2i}\ w^j \bmod n \text{ and } \bar{r}=TC_{com}(r,t). \qquad (7)$$

3) Only if μ_1 is Alice's valid partial signature and the deadline t specified in contract is sufficient for applying dispute resolution from the TTP, Bob sends his signature μ_B on contract m to Alice, since he is convinced that another partial signature µB can be released by the TTP, in case Alice refuses to do so.

4) Upon receiving μ_B, Alice checks whether it is Bob's valid signature on message m. If this is correct, she sends Bob the partial signature μ_B, by computing $\mu_2=h(m)^{d2}$ mod n. When Bob gets μ_2 , he sets $\mu_A= \mu_1\ \mu_2$ mod n , and accepts as μ_2 valid .In this case, Bob can recover Alice's standard RSA signature μ_2 on message m from µA. If Bob does not receive the value of $\mu\ _2$or only receives an invalid μ_2 from Alice timely, he applies help from the TTP via the dispute resolution protocol before the deadline t expires.

5.2 Dispute Resolution Protocol

1) The TTP first verifies whether CA , VA , and μ_B are Alice's valid certificate, voucher, and Bob's signature on contract m respectively. After that, the TTP checks whether the deadline t embedded in m expires, and whether Alice, Bob,and itself are the correct parties specified in m. If any validation fails, the TTP sends an error message to Bob. Otherwise,continue.

2) Then, the TTP computes $\mu_2=h(m)^{d2}$mod n and checks whether $h(m)^2=(\mu_1\ \mu_2)^{2e}$. If this equality holds,the TTP sends $(m,\mu_2\ )$to Bob and forwards $(m\ ,\mu_B\ )$ to Alice.

6 Security Discussion

6.1 Case 1:Alice is honest, but Bob is cheating.

If Bob cheats in any possible way, he cannot learn other information except $\mu\ _1$is valid Upon receiving the valid value of $\mu\ _1$, Bob has to make a choice whether he should send his signature $\mu\ _B$ on contract m to Alice. If Bob does, honest nitiator Alice returns back her second partial signature $\mu_2=h(m)^{d2}$ as Bob expects. In such a situation, Bob gets Alice's signature on contract m by setting $\mu_A= \mu_1\ \mu_2$ mod n while Alice also obtains Bob's signature μ_B simultaneously. If Bob does not send µB or only sends an incorrect μ_B to Alice, he cannot get the value of μ_2 from ice. Furthermore, in this setting, Bob also cannot get the value of μ_2 from the TTP so that Alice does not obtain his signature μ_B. Once those values are submitted, Bob indeed gets μ_2 from the TTP but Alice receives (m, μ_B) from the TTP, too. Therefore, once again, Bob and Alice get the other's signature on contract m at the same time

6.2 Case 2: Bob is honest, but Alice is cheating.

In our signature exchange protocol, Alice may cheat in any or some of the following steps: step (1), step (2), and step (4). First of all, according to the specification of our signature exchange protocol, to get the signature on contract from the honest responder Bob, the initiator Alice has to convince Bob accepting as a valid partial signature in step (2). Step (2) is confirmation protocol for RSA undeniable signatures,and that their protocol satisfies the property of soundness. The soundness means that the possible cheating Alice (prover), even computationally unbounded, cannot convince

Bob (verifier) to accept an invalid as valid with nonnegligible probability. Therefore, we conclude that to get from Bob, Alice has to send valid μ1 (with valid CAandVA) instep (1) and perform honestly in step (2). Alice is not so silly by preparing and sending μ_1 to Bob. Bob can drive her private key (and then compute signature μ_B. Therefore, to get signature μB from Bob, Alice has to compute $\mu_1=h(m)^{d1}$ and send it to Bob. In this situation, Bob receives valid $\mu_1=h(m)^{d1}$ from Alice before Alice gets valid μ_B from Bob. After that, step (4) is the only one possible cheating chance for Alice, i.e., she may refuse to reveal μ_2 or just send an incorrect μ_2 to Bob. However, this cheating behavior does not harm Bob essentially, since he can get the value of μ_2 from the TTP via our dispute resolution protocol. The reason is that Bob has received valid μ1before he sends μB to Alice. After getting the value of from the TTP, Bob can recover Alice's signature according to the recovery algorithm specified in . Therefore, in case (b) where Bob is honest but Alice is dishonest, Alice cannot get Bob's signature such that Bob does not obtain her signature. Based on the above analysis, we conclude that the proposed protocol is not advantageous to any dishonest party. In other words, our contract-signing protocol satisfies the property of fairness.

7 Conclusion

This protocol can be adapted to fair payments in e-commerce .In this setting, one customer purchases digital goods from a merchant via the Internet by paying with a digital check or cash. The extended scheme could implement such an electronic transaction between two parties fairly. That is, it is guaranteed that the customer gets the digital goods from the merchant if and only if the merchant gets the money from the customer.
Finally, using the technique of threshold RSA signature the proposed protocol could be extended for the scenarios where the trust on a single TTP needs to be distributed into multiple TTPs, or a contract is required to be signed only by a given quota of members cooperatively

. References

[[1] Abadi, N. Glew, B. Horne, and B. Pinkas, "Certified e-mail with a light on-line trusted third party: Design and implementation," in *Proc. 2002 Int.World Wide Web Conf. (WWW'02)*, 2002, pp. 387–395, ACM Press.
[2] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Proc. EUROCRYPT'98*, 1998, vol. 1403, LNCS, pp. 591–606, Springer-Verlag.
[3] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 4, pp. 591–606, Apr. 2000.
[4] G. Ateniese, "Efficient verifiable encryption (and fair exchange) of digital signature," in *Proc. ACMConf. Computer and Communications Security*