DIGITAL CONTRACT SIGNATURE SCHEME BASED ON MULTIPLE CRYPTOSYSTEM

Wang Lianhai, Manu Malek
Computer Science Department, Stevens Institute of Technology, Hoboken, NJ07030

Keywords: Digital contract signature, digital signature, multisignature, concurrent signature, public key authentication.

Abstract: This paper presents a new type of signature, contract digital signature, based on Discrete Logarithm(DL) and Elliptic Curve(EC) cryptosystems. Contract signature is similar to a real-life contract. No less than two signers take part in a contract signature. After introducing the concept and definition of contract signature, a scheme based on Discrete Logarithm (DL) and Elliptic Curve (EC) cryptosystems is presented. This scheme allows signers, whose ordinary signature schemes use many different cryptographic systems, to generate a single signature. The scheme requires neither a trusted arbitrator nor a high degree of interaction between signers. We then prove that this scheme is secure under the discrete logarithm assumption.

1 INTRODUCTION

Digital signature(Rivest et al., 1978) is one of the major means for authentication, and also one of the major research subjects of modern cryptography. Digital signature is the digital equivalent of hand signature, whose major functionality is to achieve user's authentication of a digital message. It is playing an increasingly important role in building e-commerce and office automation. Digital signature has also gained extensive interest due to its being utilized in governmental agencies and internal operations of enterprises.

A variety of digital signatures have been proposed, such as multisignature(Okamoto, 1988)(Ohta and Okamoto, 1991), proxy signature(Mambo et al., 1996), proxy multisignature(Xiaoming et al., 2002)(Lian-hai and Qiu-liang, 2004), and threshold digital signature(Park and Kurosawa, 1996). However, to date there has been no digital signature which simulates a real-life contract. In real-life, two or more signers, who may be from different companies, participate in a contract. Such contracts play a very important role in various economic activities, so it is necessary to propose a kind of signature that is similar to a real-life contract, with the following characteristics:

(1) In its generation, this kind of signature uses the signer's public key and private key of ordinary signature, making it more practical than other signature types.

(2) Signers may use different parameters or cryptosystems. This is different from the multisignature (Okamoto, 1988)(Ohta and Okamoto, 1991) proposed by Okamoto, which uses the same parameters in same cryptosystem.

(3) Fairness in the signing process: In the signing process, if one signer can't obtain the signature, the others also can't obtain the signature

This kind of signature is the digital equivalent of contracts and can be named "Contract Signature". It can be easily distinguished from multisignature, which use parameters in the same cryptosystem and seldom considers fairness in the signing process. It can also be distinguished from the "private contract signature" used in a contract signing protocol or concurrent signature. This latter type of signature resolves the problem of fair exchange of signatures, but contract signature does that with a single signature in a collaborative and simultaneous manner.

Furthermore, contract signature does not have the drawback of data expansion with the number of signers, which is more similar to a real-life contract. Moreover, in some applications, co-signers may be associated with different roles/ positions and, therefore, have different management liabilities and authorization capabilities. Thus, contract signatures

Lianhai W. and Malek M. (2006).
DIGITAL CONTRACT SIGNATURE SCHEME BASED ON MULTIPLE CRYPTOSYSTEM.
In *Proceedings of the International Conference on Security and Cryptography*, pages 267-274
DOI: 10.5220/0002097802670274

generated by co-signers with different signing orders often imply different meanings (In this case, it is not necessary to consider fairness of signature generation). Contract signature also suits well to joint announcement between different governments, ministries, and enterprises. So it can be used more widely than signing protocols and concurrent signatures.

1.1 Related Work

There are many real-life situation where multiple signers need to sign the same message. A simple solution is that every signer sign the message using an ordinary signature scheme. But this has the drawback that the data increases with the number of signers. A multisignature (Okamoto, 1988)(Ohta and Okamoto, 1991) scheme , whose goal is to design a signature scheme without data expansion with the number of signers, is a digital signature scheme that allows multiple signers to generate a single signature in a collaborative and simultaneous manner. Moreover, in some applications, co-signers in a signing group may be associated with different roles/ positions and, therefore, have different management liabilities and authorization capabilities. Thus, multisignatures generated by the same group of co-signers with different signing orders often imply different meanings.

At the same time, as more business is conducted over the Internet, the fair exchange problem assumes an increasing importance. Early work on solving this problem was based on the idea of timed release or timed fair exchange of signatures [8, 9, 10, 11]. Here, the two parties sign their respective messages and exchange their signatures "little-by-little" using a protocol. Typically, such protocols are highly interactive with many message flows. An alternative approach to solving the problem of fair exchange of signatures involves the use of a (semi-trusted) third party or arbitrator T who can be called upon to handle disputes between signers. The idea is that A registers her public key with T in a one-time registration, and thereafter may perform many fair exchanges with other entities. To take part in a fair exchange with B, A creates a partial signature which she sends to B. Entity B can be convinced that the partial signature is valid (perhaps via a protocol interaction with A) and that T can extract a full, binding signature from the partial signature. However, the partial signature on its own is not binding for A. B then fulfils his commitment by sending A his signature, and if valid, A releases the full version of her signature to B.

This protocol is fair since if B does not sign, then A's partial signature is worthless to B, and if B does sign but A refuses to release her full signature, then B can obtain it from T. The third party is only required in case of dispute; for this reason, protocols of this type are commonly referred to as optimistic fair exchange protocols.

The main problem with such an approach is the requirement for a dispute-resolving third party with functions beyond those required of a normal Certification Authority. In general, appropriate third parties may not be available.

Concurrent signatures(Chen et al., 2004) and concurrent signature protocols are also proposed to solve this problem. In a concurrent signature protocol, two parties A and B can interact without the help of a third party to sign (possibly identical) messages M_A and M_B in such a way that both A and B become publicly committed to their respective messages at the same moment in time (i.e., concurrently). This moment is determined by one of the parties through the release of an extra piece of information k which is called a keystone. Before the keystone's release, neither party is publicly committed through their signatures, while after this point, both are. In fact, from a third party's point of view, before the keystone is released, both parties could have produced both signatures, so the signatures are completely ambiguous. In a concurrent signature scheme, A first generates a keystone k and sends k to B. A and B each generate ambiguous signatures σ_A and σ_B, which are similar to ring signatures (Rivest et al., 2001), in which an anonymous signer A wants to have the option of later proving his authorship of a ring signature. The solution was to choose bits h_B pseudo-randomly and later to reveal the seed used to generate h_B. In a concurrent signature scheme, before k is released, it can't be assured that σ_A is signed by A, and it can't be assured that σ_B is signed by B. These assurances become possible when k is released. But these concurrent signatures have the drawback that the data expansion increases with the number of signers, and also do not adapt to the cases of joint announcement between different governments, ministries, and enterprises. Moreover, A may hide k and want to publicize it when the contract signature is of benefit to him. In this sense, the protocol is not fair.

1.2 Our Contributions

We introduce the notion of *contract signature*. A contract signature scheme allows multiple signers, who may be from different companies and may use different parameters or cryptosystems, to generate a single signature in a collaborative and simultaneous manner. Signers They use the same public key and private key as in their ordinary signature. This makes the management of public key or digital ID easier, and make contract signatures more practical than multisignatures.

In the following sections, the paper presents a contract signature scheme based on Discrete Logarithm (DL) and Elliptic Curve (EC) cryptosystems. Our contributions can be summarized as follows:

(1) This scheme resolves the problem of signers, whose ordinary signature scheme use different cryptographic systems from each other, generating a single signature. This is a major question that should be resolved in each contract signature. In the scheme described here, there are two signers, one's ordinary signature scheme is based on Discrete Logarithm(DL), and the other's scheme is based on Elliptic Curve (EC) cryptosystem. They can generate a secure contract signature.

(2) In this signature generating process, if one side can't obtain the signature, the others can't obtain the signature either. We use a keystone, similar to concurrent signatures, and set up a time limit, to resolve the fairness property of generation, without requiring a trusted third party.

(3) The scheme is secure: we prove that it is secure under the discrete logarithm assumption and ECCDH(Elliptic Curve Computational Diffie-Hellman) assumption.

In Section 2 we provide the formal definition of a contract signature and its properties. In Section 3 we present a contract signature scheme based on DL and EC cryptosystems, and prove the correctness of the scheme. In Section 4 we analyze the scheme's security.

2 DEFINITION OF CONTRACT SIGNATURE

2.1 Description

Assume that n signers take part in the signing process, and Signer i $(i = 1,...,n)$ uses $(P_i, A_i, K_i, S_i, V_i)$ to generate a digital signature, in which P_i is a finite set of plaintexts, A_i is a finite set of cipher texts of signature, K_i is a finite set of keys, $k_i \in K_i$ is the key to generate a signature, $sig_{k_i} \in S_i$ is the signing algorithm and $ver_{k_i} \in V_i$ is the validating algorithm of signature.

Definition 1. A *Contract Signature scheme* is a digital signature scheme which is constituted by (P, A, K, S, V) , and satisfies the following proprieties:

(1) P is a finite set of messages

(2) A is a finite set of signatures

(3) $K = (K_1, K_2, ..., K_n)$ is a key Vector

(4) For each $k_i \in K_i (i = 1, 2, ..., n)$, there exists a signing algorithm $sig_{(k_1,k_2,...,k_n)} \in S$ and a validating algorithm of signature $ver_{(k_1,k_2,...,k_n)} \in V$. For each message x, if $y = sig_{(k_1,k_2,...,k_n)}(x)$, then $ver_{(k_1,k_2,...,k_n)}(x, y)$ would be true; otherwise it would be false.

(5) For each $k_i \in K_i (i = 1, 2, ..., n)$, $sig_{(k_1,k_2,...,k_n)} \in S$ and $ver_{(k_1,k_2,...,k_n)} \in V$ can be calculated in polynomial time. $ver_{(k_1,k_2,...,k_n)} \in V$ should be published; Everyone can use it to validate signatures.

2.2 Basic Characteristics

Following are the characteristics of a contract signature

Basic Security: In the process of signature generation, any information broadcast by signer i is useless for others to compute the private key of signer i. That is, any broadcast information does not affect the security of private key or ordinary signature of signer i.

Unforgeability: If signer i does not take part in the process of signature generation, no one can counterfeit his contract signature. In the generation process of contract signature, the scheme should be secure even though other *i*-1 signers conspire to forge the contract signature of signer i.

Undeniability: Once having generated a contract signature with proxy, none of the signers

can deny it.

Fairness: In the signing process, if one can't obtain the signature, the others can't obtain the signature either.

Key-dependence: The signature key for a contract signature generator depends on each corresponding signer's private key which was used by the signer to generate an ordinary signature.

3 CONTRACT SIGNATURE SCHEME BASED ON DL AND EC CRYPTOSYSTEM

A contract signature generation may involve different parameters and different cryptosystems. As a result, a contract signature scheme is comparatively more complicated than other types of signature. Here, we study a simple form described below. In later research, we will discuss other complex forms.

In the case studier here, only two signers participate in a digital contract generation, where their digital signature schemes are based on Discrete Logarithm (DL) or Elliptic Curve (EC) cryptosystems.

3.1 Parameters

Let the two signers be A and B.

3.1.1 A's Parameters

A's ordinary signature parameters are the following: p_1 is a large prime, an elliptic curve E_1 over $GF(q_1)$, $G_1 \in E_1(GF(q_1))$ is a finite point of prime order in $GF(q_1)$. It can be presented as follows: $D = (q_1, FR, a_1, b_1, G_1, n_1), (d_1, Q_1)$, where

q_1 is field size, where $q_1 = p_1$;

An indication FR of the representation used for the elements of $GF(q_1)$;

Define the equation of the elliptic curve E_1 over $GF(q_1)$:

$y^2 = x^3 + a_1 x + b_1$ where $a_1, b_1 \in GF(q_1)$;

$G_1 = (x_{C1}, y_{C1})$ is a point of $E(GF(q_1))$ and $ord(G_1) = n_1$;

n_1 is a large prime Integer, $n_1 > 2^{160}$ and $n_1 > 4\sqrt{q_1}$;

d_1 is A's private Key, Q_1 is A's public key.

3.1.2 B's Parameters

B's ordinary signature parameters are as follows: large prime p_2, g is generator of the unique cyclic group of order p_2 in $Z_{P_2}^*$; h is a hash function; B's private key is $x_B \in [1, p_2 - 1]$ and B's public key is $y_B = g^{x_B} \bmod p_2$.

3.1.3 Common Parameters

h is a hash functions;

Let the message for signature be m_1

3.2 Generation

STEP1: A does the following:

- Selects a random integer $k_1 \in [1, n_1 - 1]$;
- Computes $R_1 = k_1 G_1 = (x_1, y_1)$, $r_1 = x_1 \bmod n_1$. If $r_1 = 0$, then go to STEP1;
- Selects a random integer $k_3 \in [1, p_2 - 1]$, and a random integer $k \in [1, p_2 - 1]$;
- Computes $r_3 = g^{k_3} \bmod p_2$ and $f_1 = g^{h(g^k \bmod p_2)} \bmod (p_2 - 1)$;
- Sends r_1, r_3 and f to B.

STEP2: B does the following :

- Selects random integers $k_2 \in [1, p_2 - 1]$, $k_4 \in [1, n_1 - 1]$;
- Computes $r_2 = g^{k_2} \bmod p_2$. If $r_2 = 0$, then go to STEP2;
- Computes $R_4 = k_4 G_1 = (x_4, y_4), r_4 = x_4 \bmod n_1$;
- Selects a time variable t_1, where t_1 is the time limit before which A must publicize k or respond to B, and B can announce that the contract signature has been blanked out within a day if he has not received k;
- Sends t_1, r_2 and r_4 to A;

STEP3: A and B compute $n = n_1(p_2 - 1)$

From n_1 being a prime, we know that $(p_2 - 1)^{-1} \bmod n_1$ and $n_1^{-1} \bmod (p_2 - 1)$ exist. Let

$l_3 = (p_2 - 1)^{-1} \bmod n_1$

$l_4 = n_1^{-1} \bmod (p_2 - 1)$

$r = r_2^{r_2} r_3^{r_3} \bmod p_2$

$R = R_1 r_1 + R_4 r_4$

$f = r f_1 \bmod (p_2 - 1)$

$m = m_1 \| t_1$, Here "||" denotes concatenation.

If $f + h(m) \bmod p_2 - 1 = 0$ or $f + h(m) \bmod n_1 = 0$, then go to STEP1; Otherwise

A computes $s_1 = ((f + h(m))d_1 - r_1 k_1) l_3 (p_2 - 1) - r_3 k_3 l_4 n_1 \bmod n$, and sends s_1 to B;

Step4: B uses the following equations to validate s_1:

$$R_1 r_1 + G_1 s_1 = Q_1 (f + h(m)) \quad (3.1)$$

$$r_3^{r_3} g^{s_1} = 1 \bmod p_2 \quad (3.2)$$

If equations don't hold, B terminates the signing process. Otherwise, B computes $s_2 = ((f + h(m))x_B - r_2 k_2) l_4 n_1 - r_4 k_4 l_3 (p_2 - 1) \bmod n$ and sends s_2 to A;

STEP5: A uses the following equation to validate s_2:

$$r_2^{r_2} g^{s_2} = y_2^{h(m)+f} \bmod p_2 \quad (3.3)$$

$$R_4 r_4 = -G_1 s_2 \quad (3.4)$$

A Computes $s = s_1 + s_2$, (R, r, s, f) is the contract signature obtained by A. A sends k to B;

STEP6: if B receives k, then he uses the following equation to validate k:

$$f = r g^{h(g^k \bmod p_2)} \bmod (p_2 - 1)$$

If the equation holds, then B computes $s = s_1 + s_2$, (R, r, s, f) is the contract signature obtained by B.

If B does not receive k before t_1, or if the above equation does not hold, then B will announce that (R, r, s, f) will be blanked out within a day, and sends his ordinary signature to others signers.

The interactions between A and B are shown in Figure 1

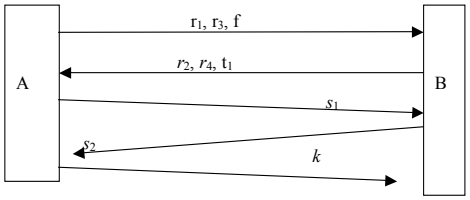

Figure 1: Interactions between A and B.

3.3 Verification

When m and (R, r, s, f) are received by us, we first compute t_1 from m, and check whether B has announced that this signature has been blanked out before $t_1 + 1$ day. If yes, the verification fails. Then we check whether R is a point on E_1. If not, the verification fails.

We use the following equations to validate whether (R, r, s, f) is a contract signature of m:

$$R + G_1 s = Q_1 (h(m) + f) \quad (3.5)$$

$$r g^s = y_B^{h(m)+f} \bmod p_2 \quad (3.6)$$

$$f = r g^{h(g^k \bmod p_2)} \bmod p_2 - 1 \quad (3.7)$$

If the above equations hold, then accept the contract signature.

3.4 Correctness

In this subsection we prove the correctness of the procedure.

3.4.1 Proof that Equations (3.1), (3.2), (3.3) and (3.4) Work

Note that $n = n_1 (p_2 - 1)$, therefore, $n_1 \mid n$. So if c is an integer, then $c \bmod n \bmod n_1 = c \bmod n_1$. Thus

$$\begin{aligned} & s_1 \bmod n_1 \\ &= s_1 \bmod n \bmod n_1 \\ &= ((f + h(m))d_1 - r_1 k_1) l_3 (p_2 - 1) - r_3 k_3 l_4 n_1 \bmod n \bmod n_1 \\ &= (f + h(m))d_1 - r_1 k_1 \bmod n_1 \end{aligned}$$

$$\begin{aligned} & s_1 \bmod p_2 - 1 \\ &= s_1 \bmod n \bmod p_2 - 1 \\ &= ((f + h(m))d_1 - r_1 k_1) l_3 (p_2 - 1) - r_3 k_3 l_4 n_1 \bmod n \bmod p_2 - 1 \\ &= ((f + h(m))d_1 - r_1 k_1) l_3 (p_2 - 1) - r_3 k_3 l_4 n_1 \bmod p_2 - 1 \\ &= -r_3 k_3 \bmod p_2 - 1 \end{aligned}$$

$$
\begin{aligned}
&s_2 \bmod (p_2-1)\\
&= s_2 \bmod n \bmod (p_2-1)\\
&= ((f+h(m))x_B - r_2k_2)l_4n_1 - r_4k_4l_3(p_2-1) \bmod n \bmod (p_2-1)\\
&= (f+h(m))x_B - r_2k_2 \bmod (p_2-1)
\end{aligned}
$$

$$
\begin{aligned}
&s_2 \bmod n_1\\
&= s_2 \bmod n \bmod n_1\\
&= ((f+h(m))x_B - r_2k_2)l_4n_1 - r_4k_4l_3(p_2-1) \bmod n \bmod n_1\\
&= -r_4k_4 \bmod n_1
\end{aligned}
$$

$$
\begin{aligned}
r_2^{r_2} g_2^{s_2} &= r_2^{r_2} g^{(f+h(m))x_2 - r_2k_2} \bmod p_2\\
&= r_2^{r_2} g^{(f+h(m))x_2} g^{-r_2k_2} \bmod p_2\\
&= r_2^{r_2} g^{(f+h(m))x_2} r_2^{-r_2} \bmod p_2\\
&= y_B^{h(m)+f} \bmod p_2
\end{aligned}
$$

So equation (3.3) is verified.

$$
\begin{aligned}
R_1r_1 + G_1s_1 &= R_1r_1 + G_1((f+h(m))d_1 - r_1k_1)\\
&= R_1r_1 + G_1(f+h(m))d_1 - G_1r_1k_1\\
&= R_1r_1 + G_1(f+h(m))d_1 - R_1r_1\\
&= Q_1(f+h(m))
\end{aligned}
$$

So equation (3.1) is verified.

$$
\begin{aligned}
r_3^{r_3} g_2^{s_1} &= r_3^{r_3} g^{-r_3k_3} \bmod p_2\\
&= r_3^{r_3} g^{-r_3k_3} \bmod p_2\\
&= r_3^{r_3} r_3^{-r_3} \bmod p_2\\
&= 1 \bmod p_2
\end{aligned}
$$

So equation (3.2) is verified.

$$
\begin{aligned}
-G_1s_1 &= -G_1(-r_4k_4)\\
&= r_4R_4
\end{aligned}
$$

So equation (3.4) is verified. □

3.4.2 Proof that Verification Equation Works

To prove that verification works, we must prove that equation (3.5) and (3.6) work.

First we prove that equation (3.5) works.

STEP1: Compute $s \bmod n_1$

$$
\begin{aligned}
&s \bmod n_1 = s_1 + s_2 \bmod n \bmod n_1\\
&= ((f+h(m))d_1 - r_1k_1)l_3(p_2-1) - r_3k_4l_4n_1 +\\
&\quad ((f+h(m))x_B - r_2k_2)l_4n_1 - k_4r_4l_3(p_2-1) \bmod n \bmod n_1\\
&= ((f+h(m))d_1 - r_1k_1 - k_4r_4)(p_2-1)(p_2-1)^{-1} \bmod n_1\\
&= (f+h(m))d_1 - r_1k_1 - k_4r_4 \bmod n_1
\end{aligned}
$$

$$
\begin{aligned}
&R + G_1s\\
&= R_1r_1 + R_4r_4 + G_1((f+h(m))d_1 - r_1k_1 - r_4k_4)\\
&= R_1r_1 + R_4r_4 + G_1(f+h(m))d_1 - G_1r_1k_1 - G_1r_4k_4\\
&= R_1r_1 + R_4r_4 + G_1(f+h(m))d_1 - R_1r_1 - R_4r_4\\
&= Q_1(f+h(m))
\end{aligned}
$$

So equation (3.5) is verified.

Next, we prove that equation (3.6) works.

$$
\begin{aligned}
&s \bmod (p_2-1) = s_1 + s_2 \bmod n \bmod (p_2-1)\\
&= ((f+h(m))d_1 - r_1k_1)l_3(p_2-1) - r_3k_3l_4n_1 +\\
&\quad ((f+h(m))x_B - r_2k_2)l_4n_1 - k_4r_4l_3(p_2-1) \bmod n \bmod (p_2-1)\\
&= ((f+h(m))x_B - r_2k_2 - r_3k_3)n_1^{-1}n_1 \bmod (p_2-1)\\
&= (f+h(m))x_B - r_2k_2 - r_3k_3 \bmod (p_2-1)
\end{aligned}
$$

$$
\begin{aligned}
rg^s &= r_2^{r_2} r_3^{r_3} g^{(f+h(m))x_B - r_2k_2 - r_3k_3} \bmod p_2\\
&= r_2^{r_2} r_3^{r_3} g^{(f+h(m))x_B} g^{-r_2k_2} g^{-r_3k_3} \bmod p_2\\
&= r_2^{r_2} r_3^{r_3} y_B^{(f+h(m))x_B} r_2^{-r_2} r_3^{-r_3} \bmod p_2\\
&= y_B^{f+h(m)} \bmod p_2
\end{aligned}
$$

So equation (3.6) works. □

4 SECURITY ANALYSIS

We will discuss the security of our scheme under Discrete Logarithm Assumption and ECCDH Assumption.

4.1 Security Characteristics of the Scheme

We will prove that this scheme is secure under the Discrete Logarithm Assumption and ECCDH Assumption.

4.1.1 Basic Security

We have proved that in this scheme, signer A and signer B cannot compute d_1 or x_B from s_1

or s_2 under the computational discrete logarithm assumption and ECCDH assumption. The proof is rather long, and omitted due to space limitation.

4.1.2 Nonfakability

We have proved that the scheme is secure under outsider impersonation attacks, insider impersonation attacks, and Man-in-the-middle attack. The proof is rather long, and omitted due to space limitation..

4.1.3 Fairness

In a contract signature, fairness is very important. In our signature scheme, if one can't obtain the signature, the others also can't obtain the signature.

To show this, note that in Step4, B obtains the contract signature (R,r,s,f), but if B does not send s_2 to A, then A can't obtain the contract signature (R,r,s,f). But we should note if A has not received s_2, he will not send k to B; So the contract signature (R,r,s,f) obtained by B does not satisfy the equation (3.7). Before B can compute k, he should first resolve the discrete logarithm problem to compute $h(g^k \bmod p_2)$, and them compute $g^k \bmod p_2$ from $h(g^k \bmod p_2)$, and then compute k from $g^k \bmod p_2$. From our discrete logarithm assumption, it can't be computed within polynomial time. If k cannot be obtained, the contract signature (R,r,s,f) obtained by B is not a valid signature. So if A has not obtained the contract signature, B also has not obtained the valid contract signature.

In Step5, A has obtained the valid contract signature, but if she wants to that his signature is valid, he should publicize k, then B also obtains k, and obtains the valid contract signature. So, the process is fair. Moreover, A may hide k and want to publicize it when the contract signature is of benefit to him. But if B can not receive k before time t_1, then B will publicize his ordinary signature which announce that (R,r,s,f) has been blanked out within a day and sends his ordinary signature to others signers. If A didn't send k to B before time t_1, then (R,r,s,f) will be blanked out.

If a few days later B regrets having signed the contract, he want to public his ordinary signature which announces that (R,r,s,f) has been blanked out. But because A has publicized k, or one day has passed, his ordinary signature which announce that (R,r,s,f) have been blanked out is valid. So our scheme is fair.

4.1.4 Nonrepudiation

From 4.1.1 and 4.1.2, we know that only A can compute s_1 and only B can compute s_2, No one can impersonate A and B to sign a contract signature. So A and B can't deny that they have signed a contract signature.

5 CONCLUSIONS AND FUTURE WORK

This paper has presented a new type of signature: contract digital signature scheme based DL and EC Cryptosystems. The scheme resolved the problem of how signers whose ordinary signature schemes use different cryptographic systems, generate a single signature. The scheme requires neither a trusted arbitrator nor a high degree of interaction between signers. We believe that contract signatures can be used more widely than signing protocols and concurrent signatures.

The scheme presented here is only a simple form of contract signature. There are many more related problems for further study, such as the multi-party contract signature scheme, contract signature scheme based on RSA and DL Cryptosystems, and contract signature scheme based on RSA, DL, and EC Cryptosystems.

REFERENCES

R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications for the ACM, 1978,21:120-126.

T. Okamoto. Digital multisignature scheme using bijective public-key cryptosystems. ACM Transaction on Computer systems, 1988,6(4): 432-441.

K. Ohta and T. Okamoto. A digital multisignature scheme base on the Fiat-Shamir scheme. In Advances in Cryptology ASIACRYPT '91, LNCS Vol. 739, H. Imai, R. Rivest and T. Matsumoto ed., Springer-Verlag, 1991. 139-148.

M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures: Delegation of the power to sign messages. In IEICE Trans. Fundamentals, 1996, Vol. E79-A, No. 9, Sep., pages 1338--1353.

Wang Xiaoming, Fu Fangwei, Security analysis of a sort of proxy multisignature scheme [J] Journal of China Institute of Communications 36(4), 2002: Page 98-102

Wang Lian-hai ,Xu Qiu-liang. Proxy signature Based on the elliptic curve cryptosystem.[J] Application Research Of Computers, 4, 2004, Page 122-126.

C.Park, and K. Kurosawa. New ElGamal Type Threshold Digital signature Scheme. IEICE Trans. Fundamentals, 1996, E79-A(1):86-93.

D. Boneh, and M. Naor, Timed commitments (extended abstract). In Advances in Cryptology - CRYPTO 2000, LNCS vol. 1880, pp. 236-254. Springer-Verlag, 2000.

S. Even, O. Goldreich, and A. Lempel, A randomized protocol for signing contracts. In Commun. ACM, vol. 28(6), pp. 637-647, June 1985.

O. Goldreich, A simple protocol for signing contracts. In Advances in Cryptology - CRYPTO 1983, pp. 133-136, Springer-Verlag, 1983.

J. Garay, and C. Pomerance, Timed fair exchange of standard signatures, In Proc. Financial Cryptography 2003, LNCS vol. 2742, pp. 190-207, Springer-Verlag, 2003.

Liqun Chen, Caroline Kudla, and Kenneth G. Paterson. Concurrent Signatures. In: *EUROCRYPT 2004*, LNCS 3027, pp. 287-305. Springer-Verlag, 2004.

R. Rivest, A. Shamir and Y. Tauman, How to leak a secret. In Advances in Cryptology - ASIACRYPT 2001, LNCS 2248, pp. 552-565, Springer-Verlag, 2001

A. Menbezes, Elliptic Curve Public Key Cryptosystem, Kluwer Academic Publishing, Boston, 1993.