

A Secure and Fair DSA-based Signature Exchange Protocol

Aleksandra Nenadić, Ning Zhang, Stephen Barton
Computer Science Department, University of Manchester
Oxford Road, Manchester M13 9PL, UK
{anenadic, nzhang, s.k.barton}@cs.man.ac.uk

Abstract

This paper presents a novel protocol for fair exchange of DSA-based signatures to support an e-commerce activity of digital contract signing. The protocol employs an off-line semi-trusted third party (STTP), which participates in the protocol execution only in exceptional circumstances and which involvement in the protocol is transparent. The protocol can tolerate the STTP's own misbehaviour (i.e. if it does not collude with other participants of the protocol) and it prevents the STTP from gaining the exchanged signatures or the signed document. The protocol is based on the principle of verifiable and recoverable signature encryption, and in comparison with related protocols is more efficient.

1. Introduction

As we are rapidly heading into an era of e-commerce, secure systems are pressingly needed to facilitate execution of e-commerce transactions. One category of such transactions is electronic contract signing. Contract signing is a major part of every business and legally binding contracts are used to facilitate valuable agreements carried out in commercial activities. Automating the process of contract signing would cut business journeys, save time and reduce costs to businesses. However, the fact that e-commerce transactions are conducted over untrustworthy communication channels has introduced a whole range of security concerns that must be addressed. One of the most important security concerns in contract signing is that of *unfair exchange*, which occurs when one party unfairly obtains the contract signature of another without giving out his own, thereby leaving the other party legally bound to the contract while he is not.

The notion of fairness is well established in a traditional contract signing process, in which all the parties involved are present at the time of signing and sign the contract at the same time, i.e. simultaneously. However, in e-commerce environments, deals are signed over a serial communication network that does not provide simultaneous exchange of messages, and, therefore,

adequate security protocols are needed to facilitate contract signing and guarantee fairness to all the parties involved.

This paper presents a novel contract signing DSA-CS protocol, in which the exchanged signatures are based on DSA. DSA-CS protocol provides *strong fairness* for the exchange of signatures, and enables automated dispute resolution with the help of an off-line STTP. The participation of the STTP in the protocol is *transparent*, i.e. even in cases when the STTP has to be invoked, signatures it recovers are identical to those generated by the original signers. The *confidentiality* of the contract document and the exchanged signatures is maintained throughout the protocol execution - neither a party external to the contract signing process nor the STTP can obtain any information related to these items. The protocol is designed in such a way that only the initiator of the protocol needs to actively take part in (possible) dispute resolution, while the responder only has a passive role. This property enables to minimize the communication overhead on the responder and to safeguard him against possible denial-of-service attacks by malicious senders. In order to improve the protocol's efficiency, measures are taken to reduce the number of computationally expensive cryptographic operations, and, in comparison with the related protocols, our protocol is more efficient.

The paper is structured as follows. Section 2 gives an overview of the related work. Section 3 states the requirements DSA-CS protocol is aimed at satisfying, and the notation and the assumptions used in the protocol design. Section 4 presents the design of the protocol's main building block - the VRES cryptographic primitive, and Section 5 presents the formal description of DSA-CS protocol. The security of the protocol is analysed in Section 6, and the protocol is evaluated and compared to related work in Section 7. Finally, we draw conclusions in Section 8.

2. Related work

A number of protocols have been proposed up to date to achieve fair contract signing. Historically, the protocol

design has evolved from *two-party* approach (i.e. without the involvement of any trusted third party (TTP)), to the *TTP-based* approach (where fairness is guaranteed by the involvement of a TTP).

The two-party protocols go as early as 1980s. Protocols [6, 11] and, more recently, [10, 15] are based on gradual exchange of small parts of the items to ensure that the exchange of the items occurs pseudo-simultaneously. This is achieved by having the parties release their secret items bit-by-bit in an interleaving manner - one party releases a bit of his item (together with the proof of correctness of the bit), and in return receives a bit of his counterpart's item (and its proof of correctness), and this process continues until both of them have received and released all the bits. Obviously, to achieve fairness, both items must be of the same length. The gradual secret release has been applied in the design of protocols for contracts signing in the following way: each party randomly generates a secret, and declares that he is committed to the previously agreed contract if his counterpart knows his secret. Then the parties engage in the gradual secret release protocol to fairly exchange their secrets.

A major disadvantage associated with two-party approach is that a large number of exchange rounds is needed to ensure an acceptable level of fairness, thus the level of traffic generated into the underlying network is high. Additionally, gradual secret release protocols require that participating parties have approximately equal computational power in order to guarantee fairness. Otherwise, the party with stronger computational capabilities can launch a brute-force attack after receiving the first several bits, and work out the remaining bits of his counterpart's secret. Although reasonably convincing in theory, this approach is too impractical for real life applications. In addition, protocols of this kind provide no guarantees of the quality of secrets exchanged, i.e. parties can fairly exchange bits of their secrets, but only at the end of the protocol to discover that the received bits are gibberish.

In order to overcome these weaknesses, a TTP is introduced in the protocols to assist the participants with the exchange and achieving fairness. In early TTP-based protocols, the TTP is *on-line*, i.e. it mediates every exchange process, even when participating parties are not attempting to cheat [7, 12, 14, 19]. Basically, both parties send their items to the TTP, which verifies the correctness of the items and forwards them to the rightful recipients. The TTP is also responsible for generating and storing security sensitive transactional records, making it a focal point of security attacks. In on-line TTP-based approach, a protocol cannot be performed without the TTP, making it a potential communicational and performance bottleneck and susceptible to denial-of-service attacks. In addition, as all the exchanged data is exposed to the on-line TTP, it has to

be fully and unconditionally trusted. Clearly, it is desirable to design protocols where the involvement of and security/storage requirements placed on the TTP are reduced.

In *off-line* TTP-based protocols [1-5, 9, 13, 16-18], the TTP is not involved in the protocol run under normal circumstances, i.e. when the participants do not attempt to cheat or are willing to resolve possible disputes themselves. Only when the exchange process fails to complete due to a network failure or a party's misbehaviour, the TTP is invoked to assist the exchange finally come to a fair completion. However, in off-line TTP-based contract signing protocols [1], in cases when the TTP intervenes it generates a replacement signature that has the same legal value but is structurally different from the signature produced by the original signer. This makes the TTP's participation in the protocol *non-transparent* and signatures protocol-dependant.

Recently, a new category of off-line TTP-based contract signing protocols, capable of making the role of the TTP *transparent*, have been proposed based on a cryptographic primitive that we call *Verifiable and Recoverable Encrypted Signature* (VRES) [2-5, 9, 13, 17]. In these protocols, merely by looking at the exchanged signatures, it is not possible to decide whether or not the TTP has intervened in the protocol execution. The VRES represents an encrypted signature such that a VRES receiver is assured that it indeed contains the correct signature, without obtaining any information about the signature itself (*verifiability property*). At the same time, the VRES receiver is assured that a designated TTP can decrypt the VRES, if a dispute arises (*recoverability property*). In this way, the VRES receiver is convinced that it is secure for him to release his signature first during an exchange process, as the agreed third party can recover the original signature from the VRES, should the VRES sender refuse to send his original signature later on.

This paper proposes a novel and efficient method for the VRES. Unlike protocols [2-5, 9], which are using interactive zero-knowledge (ZK) proofs for the VRES verification, in our VRES scheme no on-line interactions between the participants are required. A more detailed comparison with the related VRES-based contract signing protocols is presented in Section 7.

3. Preliminaries

3.1. Requirements

The design of the DSA-CS protocol is aimed at satisfying the following requirements.

Strong fairness [1] (S2): At the end of an exchange, if one party has obtained the other party's signature or can obtain it with the assistance of a third party, then the other

party has obtained this party's signature or can obtain it with the assistance of the third party.

Confidentiality (S2): A contract to be signed and the corresponding signatures should not be disclosed to other parties, including the third party.

Limited role of the third party (S3): The protocol should employ a STTP, rather than a TTP, and reduce security and storage requirements placed on the STTP. As defined in [12], STTP differs from TTP as it may misbehave on its own, but will not collude with any party involved in the exchange, whereas the TTP is expected not to misbehave at all and the level of trust placed in the TTP is unconditional.

Transparency of the STTP (S4): Signatures recovered by the STTP in case of dispute should be identical to those sent by the original signers.

3.2. Notation

- $Sign_i(x)$ expresses the signature of a party P_i on a data item x using DSA signature scheme.
- (x_i, y_i) denotes P_i 's DSA private/public key pair.
- $h(x)$ is a strong-collision-resistant one-way hash function, such as SHA-1.
- x, y denotes the concatenation of data items x and y .

3.3 Assumptions

(A1) Parties P_a and P_b share document C and wish to fairly exchange their respective DSA-based signatures on document C , i.e. (r_a, s_a) and (r_b, s_b) .

(A2) P_a and P_b have agreed to employ an off-line STTP P_t to help with the exchange if they cannot reach a fair completion themselves. P_t may misbehave by attempting to access document C or signatures, but P_t does not conspire with either of P_a and P_b .

(A3) Each party P_i ($i \in \{a, b, t\}$) has a pair of private and public DSA keys, denoted as (x_i, y_i) . Public key y_i is certified by a Certification Authority and known by all the other parties.

(A4) For the purpose of the VRES generation and possible recovery, P_b has obtained a certificate C_{bt} from P_t , certifying his additional public/private key pair, denoted as (z_{bt}, w_{bt}) , to be detailed below.

(A5) Communication channels between all the parties are authenticated and confidential, e.g. through the use of Secure Socket Layer (SSL) protocol.

(A6) Party P_a initiates the contract signing protocol.

4. DSA-based VRES scheme

The VRES scheme comprises the following four stages: (1) initialisation stage, (2) VRES generation, (3) VRES verification, and (4) VRES recovery. We assume that party P_b generates the VRES for his ordinary DSA signature $(r_b,$

$s_b)$, party P_a performs the VRES verification, and party P_t the VRES recovery. We recall that to verify P_b 's DSA signature (r_b, s_b) , one should check that the following equation holds:

$$(g^{h(C) \times s_b^{-1}} \times y_b^{r_b \times s_b^{-1}}) \bmod p \bmod q = r_b. \quad (1)$$

Initialisation stage. During the initialisation stage, parties P_b and P_t agree on a one-time shared secret key w_{bt} , which should be used for one protocol execution only (the reason for this will be explained later). The related public key z_{bt} is defined as $z_{bt} = g^{w_{bt}} \bmod p$. P_t issues a certificate C_{bt} to party P_b for the public key z_{bt} , defined as $C_{bt} = (P_b, y_b, z_{bt}, h_{bt}, S_{bt})$. Here, P_b represents P_b 's identity, y_b is P_b 's ordinary DSA public key, z_{bt} is P_b 's newly generated public key, and h_{bt} is defined as $h_{bt} = (h(x_t, y_b)^{-1} \times w_{bt}) \bmod p$, where x_t is P_t 's private key. By including h_{bt} in the C_{bt} , P_t has no need to safe-keep the secret key w_{bt} , as P_t can compute it from h_{bt} as $w_{bt} = (h(x_t, y_b) \times h_{bt}) \bmod p$. S_{bt} in C_{bt} is P_t 's signature on all the items from the certificate, i.e. $S_{bt} = Sign_t(P_b, y_b, z_{bt}, h_{bt})$. The purpose of this certificate is to establish a secret key w_{bt} between P_b and P_t , which is to be used by P_b to generate the VRES, and by P_t to recover P_b 's signature from the VRES in case of any dispute.

VRES generation. To generate verifiable and recoverable encryption of his DSA signature (r_b, s_b) , denoted as (v_1, v_2, v_3) , party P_b computes:

$$\begin{aligned} v_1 &= (r_b \times g^{(z_{bt} \times s_b)} \bmod p) \bmod q, \\ v_2 &= (z_{bt} \times r_b \times s_b^{-1}) \bmod q, \\ v_3 &= (s_b^{-1} + z_{bt} \times s_b \times h(C)^{-1} \bmod p) \bmod q. \end{aligned}$$

VRES verification. To verify P_b 's VRES (v_1, v_2, v_3) , party P_a performs the following verification:

- Check the correctness of P_b 's certificate C_{bt} by verifying P_t 's signature S_{bt} in $C_{bt} = (P_b, y_b, z_{bt}, h_{bt}, S_{bt})$.
- Confirm that the following equation holds:

$$(g^{v_3 \times h(C)} \times y_b^{v_2 \times z_{bt}^{-1}}) \bmod p \bmod q = v_1. \quad (2)$$

Here, $v_1 = (r_b \times g^{(z_{bt} \times s_b)} \bmod p) \bmod q = (r_b \times g^{((g^{w_{bt}})^{s_b})} \bmod p) \bmod q = (r_b \times g^{(y_b^{w_{bt}})} \bmod p) \bmod q$, so v_1 can be "decrypted" to recover r_b either using P_b 's private key x_b or using the secret key w_{bt} , shared between P_b and P_t .

The purpose of verification (a) is to verify that P_t has indeed issued the certificate C_{bt} for P_b 's public key z_{bt} , and that it knows the corresponding secret key w_{bt} . The purpose of verification (b) is to convince P_a that P_b 's VRES (v_1, v_2, v_3) indeed contains the correct signature (r_b, s_b) , and that number $z_{bt} \times s_b = y_b^{w_{bt}} \bmod p$ was used in the computation of (v_1, v_2, v_3) . Hash value $h(C)$ is also included in the verification to ensure that the signature is computed on the correct document C . The equivalence of equation (2), used

for the VRES verification, and equation (1), used for the DSA signature verification, implies that P_b 's correct signature (r_b, s_b) was used. This can be proved as follows. By replacing the expressions v_1, v_2 and v_3 into equation (2), one gets:

$$(g^{(s_b^{-1} + z_{bt}^{x_b} \times h(C)^{-1}) \times h(C)} \times y_b^{(z_{bt} \times r_b \times s_b^{-1}) \times z_{bt}^{-1}} \bmod p) \bmod q = r_b \times g^{(z_{bt}^{x_b})}.$$

Dividing both sides of the above equation with $g^{(z_{bt}^{x_b})}$, which we can since $\gcd(g^{(z_{bt}^{x_b})}, p)=1$ and $\gcd(g^{(z_{bt}^{x_b})}, q)=1$, one obtains equation (1) (i.e. (2) \Rightarrow (1)). The other direction (i.e. (1) \Rightarrow (2)) is trivial - by multiplying both sides of equation (1) with $g^{(z_{bt}^{x_b})}$, one obtains equation (2).

VRES recovery. In order for P_t to recover P_b 's signature (r_b, s_b) from VRES (v_1, v_2, v_3) , P_t first has to derive the shared private key w_{bt} from P_b 's certificate C_{bt} using its private key x_t , i.e. $w_{bt} = (h(x_t, y_b) \times h_{bt}) \bmod q$, and then uses w_{bt} to recover r_b from v_1 as $r_b = (v_1 \times g^{-(y_b w_{bt})} \bmod p) \bmod q$. The other part of P_b 's signature can then be computed as $s_b = z_{bt} \times r_b \times v_2^{-1} \bmod q$.

VRES security. The security of the VRES scheme can be discussed in relation to two issues – firstly, is it possible to convert the VRES (v_1, v_2, v_3) into P_b 's normal signature (r_b, s_b) without knowing the private key x_b or w_{bt} , and, secondly, it is possible for P_b to forge the VRES and trick P_a into accepting it?

In order to convert the VRES into the ordinary signature, one should either know P_b 's private key x_b or private key w_{bt} shared between P_b and P_t and use it to compute the number $z_{bt}^{x_b} = y_b^{w_{bt}} \bmod p$ and then use this number to recover r_b from v_1 or s_b from v_3 . However, computing private keys x_b and w_{bt} from the corresponding public keys y_b and z_{bt} is equivalent to solving the discrete logarithm problem.

Should it be possible to forge the VRES, P_b could generate another signature $(r_b', s_b') \neq (r_b, s_b)$ on a different document C' , and use it to create VRES' and cheat P_a into accepting it as a verifiable and recoverable encryption of signature (r_b, s_b) on document C . This would, however, mean that the equation (2) holds, and, since equations (2) and (1) are equivalent, that the following holds as well:

$$(g^{h(C) \times s_b'^{-1}} \times y_b^{r_b' \times s_b'^{-1}} \bmod p) \bmod q = r_b'$$

The latter can, however, be true if and only if (r_b', s_b') is P_b 's correct signature on C , owing to the security of the DSA signature scheme.

Note that it is crucial for the security of the scheme that the secret key w_{bt} is not reused. Should, for instance, party P_b reuse the key pair (w_{bt}, z_{bt}) for two exchange sessions

with party P_a , it would enable P_a to compute the number $z_{bt}^{x_b}$ after the first session, and then use it to “break” P_b 's VRES and compute his signature from the second session. Party P_b should obtain multiple certificates C_{bt} from P_t , and discard the used certificate at the end of each session.

5. DSA-CS protocol

This section presents the formal description of the DSA-CS protocol, which consists of two protocols - the signature exchange and the signature recovery protocol. The signature exchange protocol handles the case of a normal exchange between parties P_a and P_b without P_t 's involvement. The recovery protocol deals with cases when the exchange protocol has failed to complete successfully, during which P_t is invoked to recover the receipt and restore fairness.

5.1. Signature exchange protocol

The exchange protocol is performed according to the following steps:

- (E1): $P_a \rightarrow P_b: r_a, p_1, p_2$
- (E2): $P_b \rightarrow P_a: v_1, v_2, v_3, at_b, C_{bt}$
- (E3): $P_a \rightarrow P_b: s_a$
- (E4): $P_b \rightarrow P_a: r_b$

Step (E1): P_a produces partial encryption of his DSA signature (r_a, s_a) , denoted as (p_1, p_2) , as follows:

$$p_1 = g^{s_a^{-1}} \bmod p, \text{ and } p_2 = y_a^{s_a^{-1}} \bmod p,$$

and transfers the items p_1, p_2 , and r_a to P_b .

Step (E2): P_b verifies the correctness of P_a 's partial encrypted signature (p_1, p_2) by confirming that the following equation holds:

$$(p_1^{h(C)} \times p_2^{r_a} \bmod p) \bmod q = r_a.$$

If this verification is negative, P_b may ask P_a to re-send message (E1) or may terminate the protocol execution. Otherwise, P_b is ensured that P_a 's partial signature encryption (p_1, p_2) contains the correct s_a , and P_b produces VRES (v_1, v_2, v_3) together with his authorisation token at_b . at_b is defined as P_b 's signature on items C_{bt}, v_1, p_1 and P_a , i.e. $at_b = \text{Sign}_b(C_{bt}, v_1, p_1, P_a)$, and at_b represents P_b 's conditional authorisation stating that P_t can recover r_b from v_1 , which will enable P_a to derive s_b from v_2 , if and only if P_a provides P_t with an item s_a , such that $g^{s_a^{-1}} \bmod p = p_1$. P_b transfers his VRES (v_1, v_2, v_3) , authorisation at_b , and C_{bt} to P_a .

Step (E3): P_a performs the VRES and checks the correctness of authorisation by verifying P_b 's signature at_b . If either of these verifications is negative, P_a may ask P_b to re-send message (E3) or terminate the protocol execution. Otherwise, P_a sends the other part of his signature s_a to P_b .

Step (E4): Upon receipt of s_a from P_a , P_b verifies the correctness of P_a 's signature (r_a, s_a). If this verification fails, P_b may ask P_a to re-send message (E4) or terminate the protocol execution. Otherwise, P_b sends r_b to P_a . Upon receipt of r_b , P_a uses it to derive s_b . P_a then verifies the correctness of P_b 's signature (r_b, s_b). If this verification is positive, the exchange process is completed successfully - parties P_a and P_b have obtained each other's signature on document C . If this verification is negative, or P_a fails to receive r_b from P_b , P_a may initiate the signature recovery protocol.

5.2. The Signature Recovery Protocol

The signature recovery protocol can be invoked by party P_a only, and is executed as follows:

(R1): $P_a \rightarrow P_t: C_{bt}, v_1, p_1, s_a, at_b$

(R2): $P_t \rightarrow P_a: r_b$

(R3): $P_t \rightarrow P_b: s_a$

Step (R1): P_a transfers the items C_{bt}, v_1, p_1, s_a and at_b to P_t to request the recovery of r_b from VRES. To check the correctness of P_a 's request, P_t verifies P_b 's signature in authorization at_b and confirms that $g^{s_a^{-1}} \bmod p = p_1$. If any of these verifications fail, P_t rejects P_a 's request. Otherwise, P_t recovers r_b from v_1 .

Step (R2): P_t sends r_b to P_a .

Step (R3): P_t forwards s_a to P_b .

6. Protocol analysis

The security of DSA-CEGD protocol is dependant on that of its main cryptographic building block – the VRES, which was discussed in Section 4. The compliance with the security requirements specified in Section 3.1 can be discussed as follows.

Strong fairness (S1): Suppose that P_b has obtained P_a 's signature (r_b, s_b), i.e. P_b has received the correct r_a in step (E1) and s_a in step (E3) or from P_t in step (R3). In this case, P_a must have received the correct VRES (v_1, v_2, v_3) in step (E2). Consequently, P_a will obtain r_b from P_b in step (E4), or from P_t in step (R2). P_a can then use it to derive s_b from v_2 to obtain P_b 's complete signature (s_b, r_b). Similarly, suppose that P_a has obtained P_b 's complete signature (s_b, r_b), i.e. P_a has received the correct items in step (E2) and r_b in step (E4) or from P_t in step (R2). This implies that P_b has received the correct r_a in step (E1) and s_a in step (E3) or from P_t in step (R3). Therefore, P_b can obtain P_a 's complete signature (s_a, r_a).

Data confidentiality (S2): Since communication between parties P_a and P_b is carried out through a confidential channel, the exchanged data is not exposed to any outsiders. In addition, document C is not used in the recovery process and P_t only deals with P_a 's and P_b 's partial signatures, i.e. s_a and r_b , respectively. Disclosing s_a

and r_b to P_t does not affect the secrecy of r_a and s_b , respectively. This implies that P_t does not have the full access to the exchanged signatures nor the contract document.

Limited role of the STTP (S3): The role of P_t is limited to off-line signature recovery and issuing additional certificates to participating parties. P_t need not store any information related to the shared secret key w_{bt} , and P_t does not have the access to the exchanged signatures and the contract document. Therefore, the security and storage requirements placed on P_t are reduced.

Transparency of the STTP (S4): It is clear from the protocol description that the signatures recovered by P_t are identical to those produced by the original signer.

7. Comparison with related work

In this section we evaluate the overheads introduced by the DSA-CS protocol and compare it with related VRES-based contract signing protocols suited for DSA signatures [3-5, 9, 13, 17]. The evaluation and comparison are performed in terms of the number of messages specified in the protocols and computationally expensive operations (i.e. modular exponentiations) used in formation of all the messages. The results are shown in Table 1.

Protocols [3-5, 9] differ from ours as they employ interactive ZK protocols for the VRES verification, which can be resource consuming. Therefore, our protocol is much more efficient and the amount of data transmitted is considerably smaller. Protocols [4, 5] are flawed as they allow party P_a to successfully launch the following attack: P_a refuses to execute step (E3) and instead gets P_t to recover P_b 's signature for him immediately after step (E2). In other words, they provide no condition under which P_t will recover P_b 's signature for P_a . Protocol [3] does not give the concrete instances of encryption algorithms used in the protocol, and therefore it is difficult to calculate the number of modular exponentiations and to evaluate it. However, the authors report that, only for the VRES verification, their protocol requires around 80 modular exponentiations and transmits around 4KB of data (when 40 interactive rounds are used in ZK proof), which greatly exceeds the number of exponentiations and amount of data transmitted during the execution of our entire exchange protocol (around 320 bytes). Protocols [3, 4, 9, 13, 17] do not provide contract and signatures' confidentiality, and in protocols [3, 13] both participants can be actively involved in dispute resolution. Authors of [17] have not presented the detailed protocol design, but rather have just explained the design of the VRES scheme the protocol is based on, which makes it difficult to evaluate the protocol itself. However, their VRES scheme has been recently broken and therefore the protocol is excluded from comparison.

Table 1. Evaluation and comparison with related protocols

| Protocol | DSA-CS protocol | Ateniese's protocol [4] | Bao et al.'s protocol [5] | Chen's protocol [9] | Garay et al.'s protocol [13] |
|---|-----------------|-------------------------|---------------------------|---------------------|------------------------------|
| Process | | | | | |
| # exp. in VRES generation | 3 | 3 | 3 | 4 | ≥ 3 |
| # exp. in VRES verification | 4 | 14 | $3+9i^*$ | $\geq 17^{**}$ | 64 |
| # exp. in VRES recovery | 2 | ≥ 16 | 1 | 4 | N/A |
| # exp. in exchange protocol | 19 | 22 | $8 + (3+9i)^*$ | $\geq 26^{**}$ | N/A |
| # exp. in recovery protocol | 5 | ≥ 20 | 5 | 11 | N/A*** |
| Contract & sign. confidentiality | Yes | No | No | No | No |

* i is number of rounds in ZK proof.

** The concrete ZK proof of equality of discrete logarithms used in the protocol is not specified, e.g. using an efficient ZK proof from [8] the number of exponentiations for the VRES verification in [9] is 17 and for the exchange protocol is 26.

*** There are two different recovery protocols for P_a and P_b .

8. Conclusion

The growing importance of e-commerce and the increasing number of applications in this area has led research into studying methods of how to perform contract signing over the Internet securely and reliably. This paper presented a novel, efficient and fair DSA-CS protocol to facilitate this e-commerce activity, in which the exchanged signatures are DSA-based. Automated dispute resolution in the protocol is performed with the help of an off-line and transparent STTP. The protocol analysis has demonstrated that it satisfies the security requirements specified, and that, in comparison with related protocols, it is more efficient. In our future work we will formally verify the protocol's security properties and prototype the protocol as part of the Fair Integrated Data Exchange System (FIDES).

9. References

[1] Asokan N., Schunter M., Waidner M., Asynchronous Protocols for Optimistic Fair Exchange, Proceedings of the IEEE Symposium on Security and Privacy, pp. 86-100, 1998.

[2] Asokan N., Schunter M., Waidner M., Optimistic Fair Exchange of Digital Signatures, Extended Abstract, Proceedings of Advances in Cryptology - EUROCRYPT '98, LNCS, Springer-Verlag, Berlin, Germany, vol. 1403, pp. 591-606, 1998.

[3] Asokan N., Schunter M., Waidner M., Optimistic Fair Exchange of Digital Signatures, IEEE Journal on Selected Areas in Communications 18, pp. 593-610, 2000.

[4] Ateniese G., Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures, Proceedings of ACM Conference on Computer and Communications Security, pp. 138-146, 1999.

[5] Bao F., Deng R., Mao W., Efficient and Practical Fair Exchange Protocols with Off-line TTP, Proceedings of IEEE Symposium on Security and Privacy, pp. 77-85, 1998.

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

[7] Bürk H., Pfizmann A., Value Exchange Systems Enabling Security and Unobservability, Computers & Security 9, pp. 715-721, 1990.

[8] Chaum D., Pedersen T. P., Wallet Databases with Observer, Proceedings of Advances in Cryptology - CRYPTO '92, LNCS, Springer-Verlag, Berlin, pp. 89-105.

[9] Chen L., Efficient Fair Exchange with Verifiable Confirmation of Signatures, Proceedings of Advances in Cryptology - ASIACRYPT '98, LNCS, Springer-Verlag, Berlin, Germany, vol. 1514, pp. 286-299, 1998.

[10] Damgard I.B., Practical and Provably Secure Release of a Secret and Exchange of Signatures, Advances in Cryptology - EUROCRYPT '93, LNCS, Springer-Verlag, Berlin, Germany, vol. 765, pp. 200-217, 1994.

[11] Even S., Goldreich O., Lempel A., A Randomized Protocol for Signing Contracts, Communications of the ACM 28 (6), pp. 637-647, 1985.

[12] Franklin M. K., Reiter M., Fair Exchange with a Semi-Trusted Third Party, Proceedings of ACM Conference on Computer and Communications Security, pp. 1-5, 1997.

[13] Garay J. A., Jakobsson M., MacKenzie P., Abuse-free Optimistic Contract Signing, Proceedings of Advances in Cryptology - CRYPTO '99, LNCS, Springer-Verlag, Berlin, Germany, vol. 1666, pp. 449-466, 1999.

[14] Ketchpel S., Transaction Protection for Information Buyers and Sellers, Proceedings of the Dartmouth Institute for Advanced Graduate Studies '95: Electronic Publishing and the Information Superhighway, Boston, USA, 1995.

[15] Okamoto T., Ohta K., How to Simultaneously Exchange Secrets by General Assumptions, Proceedings of ACM Conference on Computer and Communication Security, pp.184-192, 1994.

[16] Ray I., Ray I., An Optimistic Fair Exchange E-commerce Protocol with Automated Dispute Resolution, Proceedings of 1st International Conference on Electronic Commerce and Web Technologies EC-Web 2000, LNCS, Springer-Verlag, Berlin, vol.1875, pp. 84-93, 2000.

[17] Wu C., Varadharajan V., Fair Exchange of Digital Signatures with Offline Trusted Third Party, International Conference on Information and Communication Security, pp. 466-470, 2001.

[18] Zhang N., Shi Q., An Efficient Protocol for Anonymous and Fair Document Exchange, Computer Networks Journal 41, Elsevier Science Publisher, pp. 19-28, 2003.

[19] Zhou J., Gollmann D., Observations on Non-repudiation, Proceedings of Advances in Cryptology - ASIACRYPT '96, LNCS, Springer, vol. 1163, pp. 133-144, 1996.