Asian Journal of Information Technology 6 (6): 726-741, 2007

A Model of a Secure Multi-Party Transaction Protocol

[1]L. Josephine Mary and [2]S.P. Rajagopalan
[1]Department of Information Technology, Dr. M. G. R. University, Chennai-95, India
[2]School of Computer Science and Engineering,
Dr. M. G. R. University, Chennai-95, India

Abstract: Multi-party signing and encapsulation of messages (transactions/documents) that support multiple asynchronous signers and multiple recipients. There must be some mechanisms and protocols to create and distribute protected transactions/documents between participants with flexible verification of such transactions/documents.The main objectives of our new model of a Multi-party transaction protocol to support for secure creation, distribution, verification and different legal assurance levels of signatures of multi-party (sequential and parallel) processing of transactions between multiple signers and verifiers. Therefore, it may be used for protection of various electronic transactions or documents exchanged between multiple participants.

Key words: Multi-party transaction protocol, sequential signature, parallel signature, digital signature

INTRODUCTION

Parties involved in multi-party transactions: In general, 4 types of participants may be involved in a multi-party transaction. They are: An initiator of the transaction, multiple signers of the transaction, a sender of the transaction and multiple verifiers of the transaction. An initiator is the participant who initiates creation or verification of a multiparty transaction. Signers are participants who sign a transaction. A sender is the participant who envelops and distributes the transaction to multiple verifiers. Verifiers are participants who receive and verify multi-party transaction.

These participants may have more than one of the above roles. For example, an initiator and a sender of a multi-party transaction may also be a signer of the transaction. Therefore, only signers and verifiers are the two distinctive groups of participants in multi-party transactions. An initiator and a sender of a multi-party transaction may also be a signer of the transaction. Therefore, only signers and verifiers are the two distinctive groups of participants in multi-party transactions. Depending on the time of their involvement in a multi-party transaction, signers and verifiers are categorized as follows:

- Static signers.
- Static verifiers.
- Dynamic signers.
- Dynamic verifiers.

PKCS7 (1993), S/MIME (Ramsdell, 1999), XML signature [XML Signature] and XML encryption (XML Encryption, 2001) are 4 different document formatting standards which could be used as a basic encapsulation format for multi-party transactions. Since these standards support multiple signatories and multiple recipients, they could be used to handle all combinations of multiple signers and multiple verifiers mentioned in this study.

Security requirements for multi-party transactions: A model of a secure multi-party transaction/document handling protocol must meet the following requirements:

- Possibility to support creation and verification of multiple signatures,
- Support for secure distribution of transactions to multiple recipients,
- Possibility for verification of multiple signatures to be performed after a long period of time relative to their creation, 4.possibility of managing signatures in accordance with person's role(s) inside an organization,
- Support for verification of the signing time with a reasonable and well defined approximation,
- Support for different legal assurance levels of signatures depending on different context of applications.

Corresponding Author: L. Josephine Mary, Department of Information Technology, Dr. M. G. R University, Chennai-95, India

DIFFERENT LEGAL ASSURANCE LEVELS OF SIGNATURE

The formal requirements for legal transactions, including the need for signatures, vary in different legal systems and also vary with the passage of time.There is also variance in the legal consequences of failure to cast the transaction in a required form. The statute of frauds of the common law tradition. During this century, most legal systems have reduced formal requirements, or atleast have minimized the consequences of failure to satisfy formal requirements.

In current practice, formalization usually involves documenting the transaction on paper and signing or authenticating the paper. Traditional methods, however, are undergoing fundamental change. Documents continue to be written on paper, but sometimes merely to satisfy the need for a legally recognized form. In many instances, the information exchanged to effect a transaction never takes paper form. Computer-based information can also be utilized differently than its paper counterpart.

The basic nature of transactions has not changed, the law has only begun to adapt to advances in technology. The legal and business communities must develop rules and practices which use new technology to achieve and surpass the effects historically expected from paper forms. To achieve the basic purposes of signatures outlined above, a signature must have the following attributes:

Signer authentication: A signature should indicate who signed a document, message or record and should be difficult for another person to produce without authorization.

Document authentication: A signature should identify what is signed, making it impracticable to falsify or alter either the signed matter or the signature without detection.

Signer authentication and document authentication are tools used to exclude impersonators and forgers and are essential ingredients of what is often called a "nonrepudiation service" in the terminology of the information security profession. A nonrepudiation service provides assurance of the origin or delivery of data in order to protect the sender against false denial by the recipient that the data has been received, or to protect the recipient against false denial by the sender that the data has been sent. Thus, a nonrepudiation service provides evidence to prevent a person from unilaterally modifying or terminating legal obligations arising out of a transaction effected by computer-based means.

Affirmative act: The affixing of the signature should be an affirmative act which serves the ceremonial and approval functions of a signature and establishes the sense of having legally consummated a transaction.

Efficiency: Optimally, a signature and its creation and verification processes should provide the greatest possible assurance of both signer authenticity and document authenticy, with the least possible expenditure of resources.

Digital signature technology generally surpasses paper technology in all these attributes.

Digital signature: Applications such as banking, stock trading and the sale and purchase of merchandise are increasingly emphasizing electronic transactions to minimize operational costs and provide enhanced services. This has led to phenomenal increases in the amounts of electronic documents that are generated, processed and stored in computers and transmitted over networks. This electronic information handled in these applications is valuable and sensitive and must be protected against tampering by malicious third parties (who are neither the senders nor the recipients of the information). Sometimes, there is a need to prevent the information or items related to it (such as date/time it was created, sent and received) from being tampered with by the sender (originator) and/or the recipient. Traditionally, paper documents are validated and certified by written signatures, which work fairly well as a means of providing authenticity. For electronic documents, a similar mechanism is necessary. Digital signatures, which are nothing but a string of ones and zeroes generated by using a digital signature algorithm, serve the purpose of validation and authentication of electronic documents. Validation refers to the process of certifying the contents of the document, while authentication refers to the process of certifying the sender of the document. In this study, the terms document and message are used interchangeably.

CONVENTIONAL AND DIGITAL SIGNATURE CHARACTERISTICS

A conventional signature has the following salient characteristics:

- Relative ease of establishing that the signature is authentic.
- The difficulty of forging a signature.
- The nontransferability of the signature.

- The difficulty of altering the signature.
- The nonrepudiation of signature to ensure that the signer cannot later deny signing.

A digital signature should have all the above features of a conventional signature plus a few more as digital signatures are being used in practical, but sensitive, applications such as secure e-mail and credit card transactions over the Internet. Since a digital signature is just a sequence of zeroes and ones, it is desirable for it to have the following properties: The signature must be a bit pattern that depends on the message being signed (thus, for the same originator, the digital signature is different for different documents); the signature must use some information that is unique to the sender to prevent both forgery and denial; it must be relatively easy to produce; it must be relatively easy to recognize and verify the authenticity of digital signature; it must be computationally infeasible to forge a digital signature either by constructing a new message for an existing digital signature or constructing a fraudulent digital signature for a given message and it must be practical to retcopies of the digital signatures in storage for arbitrating possible disputes later.

To verify that the received document is indeed from the claimed sender and that the contents have not been altered, several procedures, called authentication techniques, have been developed. However, message authentication techniques cannot be directly used as digital signatures due to inadequacies of authentication techniques. For example, although message authentication protects the two parties exchanging messages from a third party, it does not protect the 2 parties against each other. In addition, elementary authentication schemes produce signatures that are as long as the message themselves.

Basic notions and terminology: Digital signatures are computed based on the documents (message/ information) that need to be signed and on some private information held only by the sender. In practice, instead of using the whole message, a hash function is applied to the message to obtain the message digest. A hash function, in this context, takes an arbitrary- sized message as input and produces a fixed-size message digest as output. Among the commonly used hash functions in practice are MD-5 (message digest 5) and SHA (secure hash algorithm). These algorithms are fairly sophisticated and ensure that it is highly improbable for two different messages to be mapped to the same hash value. There are two broad techniques used in digital signature computation-symmetric key cryptosystem and public-key cryptosystem (cryptosystem broadly refers to an encryption technique). In the symmetric key system, a secret key known only to the sender and the legitimate receiver is used. However, there must be a unique key between any two pairs of users. Thus, as the number of user pairs increases, it becomes extremely difficult to generate, distribute and keep track of the secret keys. A public key cryptosystem, on the other hand, uses a pair of keys: A private key, known only to its owner and a public key, known to everyone who wishes to communicate with the owner. For confidentiality of the message to be sent to the owner, it would be encrypted with the owner's public key, which now could only be decrypted by the owner, the person with the corresponding private key. For purposes of authentication, a message would be encrypted with the private key of the originator or sender, who we will refer to as A. This message could be decrypted by anyone using the public key of A. If this yields the proper message, then it is evident that the message was indeed encrypted by the private key of A and thus only A could have sent it.

Hence, our new model of a Multi-party transaction protocol to support for secure creation, distribution, verification and different legal assurance levels of signatures (Digital signature) of multi-party (sequential and parallel) processing of transactions between multiple signers and verifiers.

A MODEL OF A SECURE MULTI-PARTY TRANSACTION PROTOCOL

Multi-party transaction formats: PKCS#7, S/MIME and XML signatures/encryption are suitable security encapsulation standards, which could be used as our multiparty transaction formats. According to these standards, a multi-party transaction could be prepared in different formats by following two different scenarios. PKCS#7 standard is used to describe these possible multi-party transaction formats.

Signature formats of multi-party transactions: Sequential signatures and parallel signatures are 2 types of signature formats which could be applied to a multi-party transaction. An organization, which for instance requires approval of three parties to process a transaction, could be considered as an example of sequential signatures. In an organization, financial officer must sign the transaction first and then forward it to a manager. After the manager signs it, director must also sign in order to complete the transaction. Therefore, these three parties sign the transaction sequentially by including the

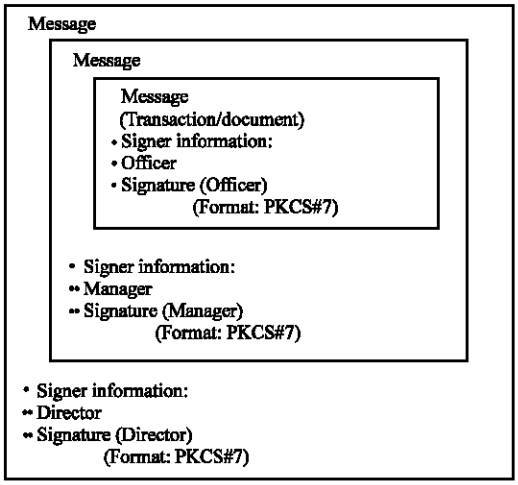

Fig. 1: The structure of a message with sequential signatures

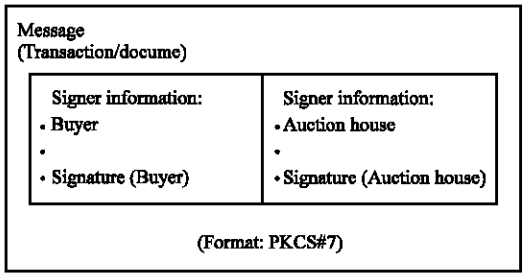

Fig. 2: The structure of a message with parallel signatures

signature of the previous party in the signing data, as shown in Fig. 1. This signature format is called sequential signatures.

In parallel signatures format, each participant creates his/her own signature of the message (transaction/document) without signatures of other participants like buyer and auction house sign the transaction independently, as shown in Fig. 2.

When creating a multi-party transaction, an initiator of the transaction should format the transaction in a specific way to indicate to other participants the required signing format. Figure 3 shows this initial structure of the transaction/document in case of a hypothetical organization.

In order to create parallel signatures, a transaction/ document is formatted with multiple Signer Information fields, which are addressed to each signer. Figure 4 shows this initial structure of the transaction/document in case of auction house. As shown in Fig. 4, Encrypted Digest (signature) field of Signer Information of auction house

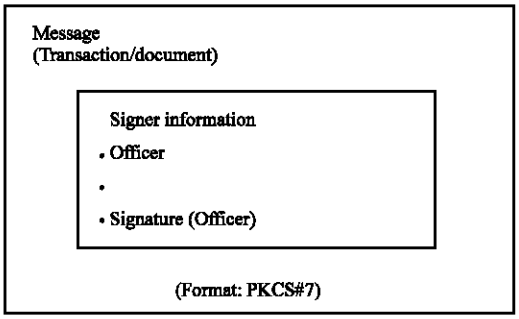

Fig. 3: Initial structure of a multi-party transaction

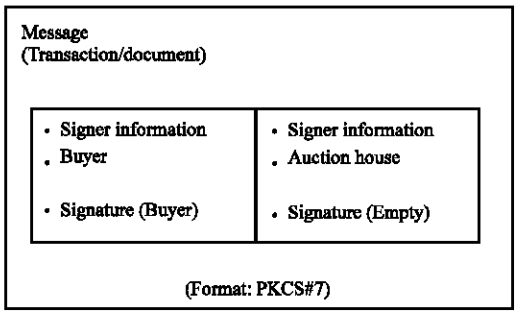

Fig. 4: Initial structure of a multi-party transaction for creation of parallel signatures

is empty. Auction house must complete this initial structure of the transaction/document after it is received. In general, signers must create sequential signatures if there is no Signer Information field, which is addressed to them. If there is a Signer Information field addressed to them, signers must complete the corresponding Signer Information field by inserting his/her signature (parallel signatures).

Procedures and a protocol for handling these signing formats are described later in this study.

Enveloping formats of multi-party transactions: Two different formats could be used to envelop a transaction/ document for multiple recipients (verifiers). With the first type of format, a transaction/document is enveloped sequentially for multiple verifiers, one after another. Therefore, this format is called sequential envelopes. For example, let's assume a hypothetical organization, which would like to arrange their transaction flow through a director, then a manger and finally to a marketing officer. In that organization all incoming transactions are first received by the director. He/she opens transactions and forwards them to his/her manager. He/she opens transactions next and forwards them to the marketing officer. Finally, marketing officer opens transactions.

Alternatively, this organization might prefer to receive transactions which are first enveloped for the marketing officer, then enveloped for the manager and finally enveloped for the director, as shown in Fig. 5.

In case of sequential envelopes, all participants must open their envelopes sequentially in order to get the content of the transaction. With the second type of enveloping format, a transaction/document could be enveloped independently for each recipient (verifier). This format is therefore, called parallel envelopes. Each verifier of the transaction/document can read the content of parallel envelopes without assistance of other verifiers. Online auction house could be used again as an example of parallel envelopes. If multiple sellers are involved in a transaction, auction house can independently envelop the transaction to each seller, as shown in Fig. 6. Both enveloping formats can use a transaction/document in clear data format or in any of the signature format above as an input data. However, a transaction/document is always in one of the above signature formats during our multi-party transaction protocol.

Multi-party transaction tokens: This section describes our new concept of Multi-Party Transaction (MPT) tokens. Multi-party transaction protocol which is described in uses these tokens to perform secure transactions between multiple participants.

Overall structure of MPT tokens: Two types of multi-party transaction tokens are used in our multi-party transaction protocol. They are called MPT request token and MPT response token. Both tokens have two components called MPT token header and MPT token body. MPT token header is used to route the token between multiple participants and MPT token body carries a message (transaction/document). The structure of MPT request and response token is defined in the study. Both MPT request and response token bodies contain a transaction or a document in multi-party transaction formats. However, in some situation MPT response token may contain empty token body. The format of MPT tokens (requests/response) is shown in Fig. 7.

The structure of the MPT request token: This study describes the structure of the MPT request token. As mentioned earlier, MPT request token has two parts, i.e., MPT request token header and MPT request token body.

MPT request token header contains the following information:

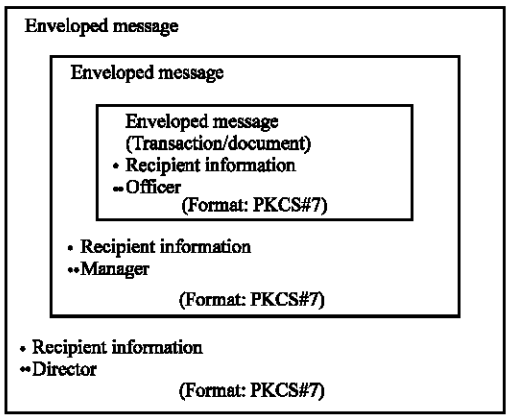

Fig. 5: The structure of a message with sequential envelopes

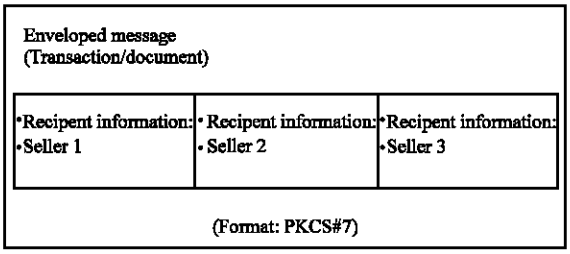

Fig. 6: The structure of a message with parallel envelopes

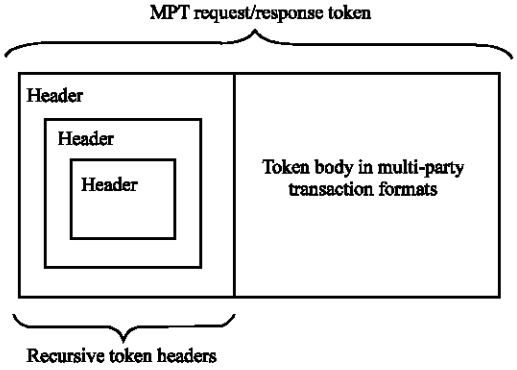

Fig. 7: The structure of the MPT request/response token

- Version
- Recipient name
- Recipient's roles
- Recipient's URI (optional)
- Signer's name
- Timestamps of message signatures (optional)
- Previous MPT token header (optional)
- Signature algorithm identifier
- Signature of the above fields (signature of the header)
- Signer's certificate (optional)

Version: This field is the version number of the MPT request token header. Recipient name: A very general definition of a name, taken from X.509 (v3),which enables recipient name identifiers other than distinguished names to be used. This will be important when the recipient does not have a distinguished name.

Recipient's roles: This field describes one or more recipient's roles. Recipient's roles are: Signer, Sender and Verifier are described as follows:

Signer: If the recipient's role is Signer, the recipient must sign the attached transaction/document.

Sender: If the recipient's role is Sender, recipient must envelop the transaction/document and distribute it to multiple recipients.

Verifier: If the recipient's role is Verifier, the recipient must open the envelope of the transaction/document and verify its signatures or perform both actions according to the transaction/document format.

Recipient's universal resource identifier: This is a valid IP address or domain name of the recipient. Signer's name: The distinguished name of the signer of the token. As a part of the verification process, the entity verifying the token shall ensure that this name is consistent with the subject name of the certificate.

Timestamps of message signatures: This field is used to carry the timestamps of transaction/document signatures. This field is used to bind token body with corresponding token header. It also allows to prove the time when the token is created.

Previous MPT token header: This field contains a multiple recursive MPT request token headers created by a single or multiple participants. The initial MPT request token header is generated by one participant, which is then encapsulated in another MPT request token header generated by the same or by another participant. This enables a signed trust chain to be built.

Signature algorithm identifier: This field contains signature algorithm used to sign the token header. This identifier generally implies both an asymmetric algorithm and a hash algorithm.

Signature of the above fields: This field contains the signature of the above fields, i.e., the signature of the complete MPT token header.

Signer's certificate: This field enables exchange of certificates, which may help participants to verify the credentials. If this field is populated, it must be consistent with the signer's name in the MPT request token header. This information is provided by the entity generating the token to facilitate verification. In most cases, this field is empty, since during authentication process all participants receive each others certificates.

MPT request token body contains:

- Message (document/transaction)

Message (document/transaction): Message field contains a transaction or a document in a multi-party transactions format.

The structure of the MPT response token: This study describes the structure of the MPT response token. As mentioned earlier, MPT response token has also two parts, i.e., MPT response token header and MPT response token body.

MPT response token header contains the following information:

- Version
- Status
- Error code (optional)
- Signer's name
- Timestamps of message signatures (optional)
- Previous MPT token header (optional)
- Signature algorithm identifier
- Signature of the above fields (signature of the header)
- Signer's certificate (optional)

Version: This field is the version number of the MPT token header.

Status: This is the verification status of the MPT request token. Three different statuses are: Processed, rejected and error.

Error code: This field contains the error code in case of problems. Signer's name: This is the distinguished name of the signer of the token. As a part of the verification process, the entity verifying the token shall ensure that this name is consistent with the subject name of the certificate.

Timestamps of message signatures: This field is used to carry the timestamps of transaction/document signatures. The format of the timestamp depends on the

time-stamping protocol. This field is used to bind token body with corresponding token header. It also allows to prove the time when the token is created.

Previous MPT token header: This gives a mechanism to include multiple recursive MPT response token headers created by single or multiple participants. The initial MPT response token header is generated by one participant, which is then encapsulated in another MPT response token header generated by the same or another participant. This enables a signed trust chain to be built. Signature algorithm identifier: This is the signature algorithm used to sign the token header. The identifier generally implies both an asymmetric algorithm and a hash algorithm.

Signature of the above fields: This field contains the signature of the above fields which is the signature of the MPT token header.

Signer's certificate: This provides a mechanism for exchange of certificates which may help the recipient to verify the credentials. If this field is populated, it must be consistent with the signer's name in the MPT token. This information is provided by the entity generating the token to facilitate verification. In most cases this field is empty, since during our authentication process all participants receive each others certificates.

MPT response token status may have the following values

Processed: If the MPT request token was processed successfully, then the corresponding MPT response token contains processed status.

Rejected: If the MPT request token verification failed, then the corresponding MPT response token contains rejected status. This may happen if signature verification fails or if some other verification fails to meet participants' security policies.

Error: If any other problem occurs during the MPT request/response token processing, then the corresponding MPT response token contains error status.

MPT response token body contains:

- Message (document/transaction)

Message (document/transaction): Message field may be empty or may contain a transaction/document in a multi-party transaction format.

MULTI-PARTY TRANSACTIONS SIGNING PROCEDURES

In this study, various signing procedures for multi-party transactions are described. MPT request and response tokens, defined in the section above, are used in these procedures. Before starting a signing procedure, an initiator performs MPA protocol and establishes a secure communication session with multiple participants (signers). During that procedure, the initiator receives certificate of each signer. In turn, each signer receives identity information, a certificate of the initiator and a session key. Since the initiator starts a signing procedure, he/she decides which signature format is used (sequential signature or parallel signatures). He/she also decides who will send the signed transaction/document to multiple recipients (verifiers), i.e. who is the sender of the transaction/document. For some type of transactions the initiator can distribute the signed transaction/document without asking any other participant to distribute it.

"Sequential" versus "Parallel" signing scenarios: Signing of multi-party transactions may be performed according to the sequential or parallel scenario, as shown in Fig. 8. In a sequential scenario each signer signs a transaction sequentially, one after the other. In a parallel scenario all signers sign a transaction simultaneously.Sequential scenario is suitable for creation of signatures in both sequential and parallel signature formats. However, parallel scenario is suitable only for creation of parallel signatures.

Sequential signing procedure: A procedure to create multiple signatures based on a sequential scenario is shown in Fig. 9. In this procedure it is assumed that the initiator is also one of the signers of the transaction/ document. Any number of dynamic signers may be added in the middle of a signing procedure. As mentioned earlier, this sequential scenario could be used to create sequential signatures, as well as parallel signatures.

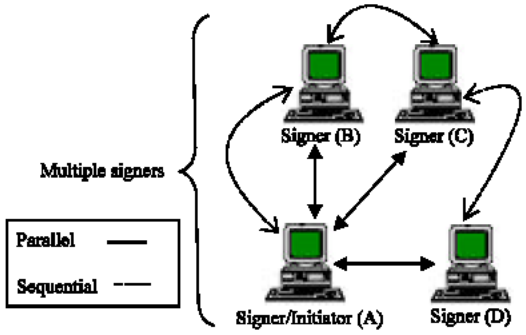

Fig. 8: Scenarios for creation of multiple signatures

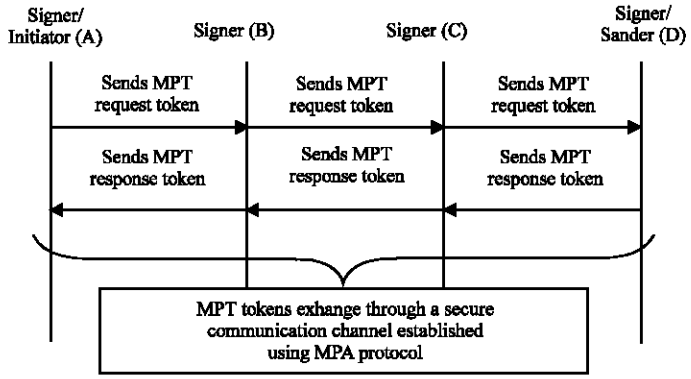

Fig. 9: Sequential creation of multiple signatures

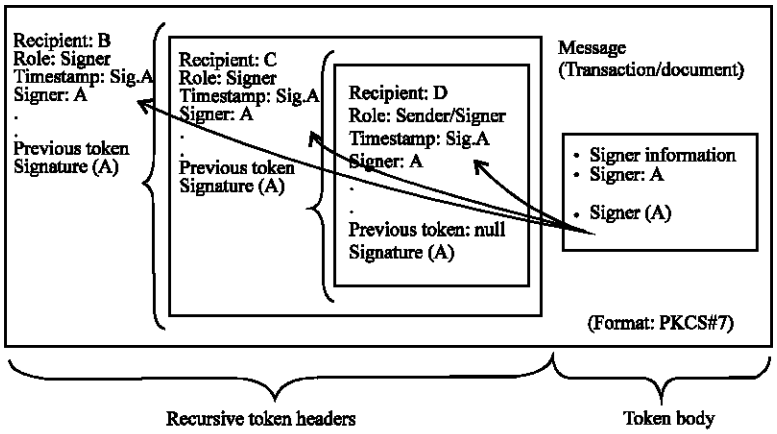

Fig. 10: MPT request token for sequential signing procedure

Figure 10 shows the complete structure of the MPT request token which can be used to create sequential signatures using a sequential scenario. As shown in the Fig. 10, MPT request token header consists of several recursive MPT request token headers and the role of each MPT token header is Signer. These individual headers are addressed to each signer who are known initially when creating the transaction. Since the initiator is also a signer, he/she signs the message transaction/document) before sending it. Timestamp of this signature should be included in each MPT request header. This is necessary in order to bind the attached transaction/document with the MPT request headers. All of these MPT request token headers are signed by the initiator. Since all signers have already established secure communication session using our MPA protocol, nobody other than signers could access MPT tokens. However, any accidental or unauthorized alterations of the MPT request token headers or body could be detected, because all MPT token headers and body are signed by the initiator.

After creation of the MPT request token, the initiator sends it to the first signer indicated by the recipient's name of the outer MPT request token header through secure communication channel, as shown in Fig. 9. The first signer (recipient B) removes the outer token header which is addressed to him/her and performs necessary actions according to his/her role. As shown in Fig. 10, the role of B is Signer and therefore, he/she signs the message after verification of signatures of the MPT token headers and the message body. Before signing the transaction/ document, the signer should also verify the timestamp value against the initiator's signature of the transaction/ document. As mentioned earlier, signing method (sequential/parallel) is decided based on the structure of the message body. If the message body contains PKCS#7 Signer Information which is addressed to the signer, then he/she should create parallel signatures. Otherwise, he/she should create sequential signatures. Finally, the intermediate signer creates new MPT request token by attaching the signed transaction/document to the rest of the MPT token header. This MPT request token is forwarded to the next signer, addressed in the outer MPT request token header, through the secure communication channel.

Each signer performs the same procedure, until the MPT request token reaches the last signer. At that stage, the last signer receives the transaction/document signed by all previous signers. The last signer sends back to his/her previous signer a MPT response token with signing status through the secure communication channel, as shown in Fig. 9. The body of that MPT response token contains the transaction/document signed by all signers. The timestamp of the last signer's signature is included in the header of that MPT response token. When each intermediate signer receives this MPT response token, he/she adds his/her own header with the timestamp of its signature and forwards MPT response token to the previous signer through a secure communication channel. Finally, the initiator receives MPT response token which contains signed transaction/ document and all required timestamps. In other words, the initiator receives all required proofs which guarantee that everybody signed the transaction/document. The initiator may archive timestamps and the signed transaction/ document for future verification. This completes the sequential signing procedure. If the initiator is not a sender, the initiator forwards the signed transaction/ document to a sender for distribution.

In order to do that, the initiator first establishes a secure communication session using MPA protocol.Then he/she creates a new MPT request token with a single MPT token header addressed to the sender. The field Recipient's role in this MPT request token header is sender. The body of this MPT request token contains signed transaction/document, which must be distributed. All corresponding timestamps should be included in the MPT token header. After receiving this MPT request token, the sender distributes signed transaction/ document to multiple verifiers. If message level protection is required, the sender should envelop the signed transaction/document before distributing it. Procedures for enveloping a multiparty transaction will be described later in this chapter. Otherwise, the sender should establish a secure multi-party communication session using MPA protocol and distribute signed transaction/ document to multiple verifiers. After distribution process is completed, the sender sends back a MPT response token with distribution status to the initiator. The body of this MPT response token is empty. As mentioned earlier, the initiator can also distribute signed transaction/ document to multiple verifiers by treating himself/herself as a sender.

Parallel signing procedure: A signing procedure which is based on the parallel signing scenario is shown in Fig. 11. In this procedure MPT request tokens are

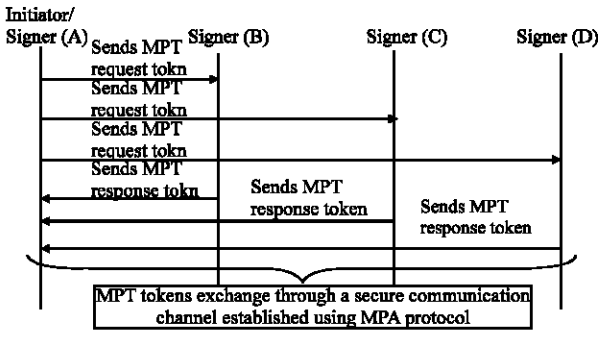

Fig. 11: Parallel creation of multiple signatures

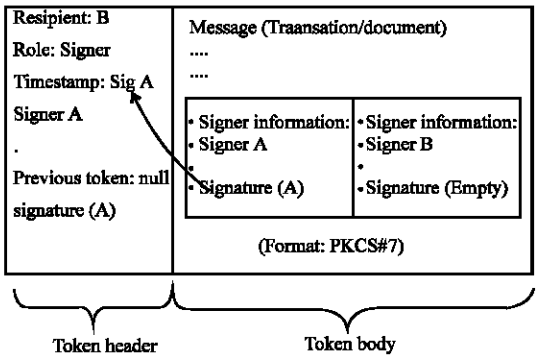

Fig. 12: MPT request token for parallel signing procedure

simultaneously sent to each signer through secure channel established using MPA protocol. Therefore, separate MPT request token headers must be created for each signer, all known at the time when a transaction/ document is created. MPT request token body contains transactions/documents to be signed in multi-party transaction formats. Since this procedure creates parallel signatures, the initiator must also add signer information into the PKCS#7 Signer Information field, as shown in Fig. 12. Finally, MPT request token bodies are attached to the corresponding MPT request token headers and sent to the corresponding signers through a secure channel. The initiator must sign MPT request token headers and bodies before sending them to each signer. In addition, the initiator must include timestamp of his/her signature of the transaction/document in the MPT request token header before signing it. Since MPT request token headers and message bodies are signed by the initiator, any accidental or unauthorized alternations could be detected.

After receiving the MPT request token, each signer first checks his/her role, indicated in the MPT request token header. Since the role of each token header is Signer, as shown in Fig. 12, each signer signs the message

after verification of signatures of the MPT token header and message body. The signing format (sequential/parallel) is decided based on the structure of the request token body. As mentioned earlier, if the token body consists of the Signer Information field, which is addressed to the signer, then he/she must create parallel signatures. Finally, each signer sends back a MPT response token which contains the signed transaction/ document with parallel signatures to the initiator through the secure communication channel. Timestamps of all signatures are included in the header of the MPT response token. After the initiator receives MPT response tokens from all signers, he/she assembles all Signer Information fields into a single signed transaction/ document in parallel signatures format. The initiator could archive this signed transaction/document together with the timestamps for future verification. This completes the parallel signing procedure.

Finally, as in the case of sequential signing procedure, the initiator forwards the assembled signed transaction/document to a sender for distribution to multiple verifiers. In some cases, an initiator may be also be a sender of the transaction/document. In that case the initiator distributes the signed transaction/document to multiple verifiers. If message level protection is required, a sender must envelop the signed transaction/document before distribution. Multi-party transaction enveloping procedure is described in the next study.

MULTI-PARTY TRANSACTION ENVELOPING AND DISTRIBUTION PROCEDURES

In order to achieve message level protection a signed transaction/document must be enveloped for multiple verifiers before being distributed through an open network. Therefore, enveloping of a transaction/document is the next important step for secure multi-party transactions.

As mentioned earlier, a sender is the participant who envelops a transaction/document on behalf of others. The initiator of signing procedure must forward the signed transaction/document to a sender for enveloping and distribution. In order to do that the initiator first establishes a secure communication session with the sender using MPA protocol. Then he/she creates a new MPT request token with a single MPT token header, which is addressed to the sender. The role in this MPT request token header is sender. The body of this MPT request token contains the transaction/document, which is required to be distributed in sequential or parallel signature formats. All corresponding time stamp(s) should be included in the MPT token header. After receiving this MPT request token, the sender distributes the signed transaction/document to multiple verifiers with or without enveloping. If message level protection is required, the sender must envelop the signed transaction/document before distributing it. In order to envelop the transaction/ document, the sender first performs our MPA protocol and establishes a secure communication session between multiple verifiers. After this authentication, the sender receives all verifiers' certificates. If the sender already has all certificates of verifiers, he/she can perform enveloping of the transaction/document without prior authentication. The sender finally envelops the signed transaction/ document in sequential envelopes or parallel envelopes format. We assume that the sender knows in advance the preferred enveloping format of verifiers. If message level protection is not required, the sender could establish a secure multi-party communication session using MPA protocol and distribute signed transaction/document to the multiple verifiers without enveloping.

In both cases, the sender must create MPT request token(s) in order to distribute signed transaction/ document. The format of the MPT request token depends on the format of the transaction/document. In case of sequential envelopes, the sender creates MPT request token containing recursive MPT request token header addressed to each verifier. The recipient's role field in these MPT request token headers is verifier. All timestamps of the attached signed transaction/document are also included in the MPT token headers. The sender must sign all MPT token headers sequentially. Finally, single MPT request token is created attaching the signed transaction/document in sequential envelopes format to the recursive MPT token header. In case of parallel envelopes, the sender creates separate MPT request token headers for each verifier. The recipient's role field in each MPT request token header is verifier. All timestamps of the attached signed transaction/document are included in each MPT token header. The sender must sign each MPT token header after creating it. Finally, separate MPT request tokens are created for all verifiers by attaching the signed transaction/document in the format of parallel envelopes to each MPT request token header. However, recursive MPT request token headers, as in the case of sequential envelopes, can also be used in order to distribute a signed transactions/documents in parallel envelopes format.Which format of MPT request token header should be used depends on preferred multi-party transaction verification procedures. Possible verification procedures are described in the next section. In case of sequential signatures or parallel signatures, both formats of the MPT request token can be used to distribute the signed transaction/document. Again, which

format of MPT request token header should be used, depends on the preferred multi-party transaction verification procedures.

After creation of MPT request token(s), the sender should distribute them to multiple verifiers. In case of multiple MPT request tokens, the sender should simultaneously distribute the corresponding MPT request tokens to each verifier. However, in case of a single MPT request token with recursive MPT request token headers, the sender must send the MPT request token to the verifier addressed in the outer MPT request token header. After distribution process is finished, the sender sends back a MPT response token with distribution status to the initiator. The body of this MPT response token is empty. After the initiator receives that MPT response token, distribution procedure is completed. As mentioned earlier, the initiator can also distribute the signed transaction/ document to multiple verifiers by treating himself/herself as a sender without passing it to other participants.

Multi-party transaction verification procedures: In this study several procedures for verification of multi-party transactions are described. These procedures may be used to verify signed or signed and enveloped transactions/documents.

"Sequential" versus "Parallel" verification scenarios: Verification of secure multi-party transactions could be handled according to the sequential or parallel scenarios, as shown in Fig. 13. In the sequential verification procedure participants verify multi-party transactions one after the other. In the parallel verification procedure all participants verify multi-party transactions simultaneously. Sequential verification scenarios could handle transactions/documents in parallel envelopes and sequential envelopes formats.

However, parallel verification scenarios are suitable only for transactions/documents in a parallel envelopes format. Both sequential and parallel scenarios can be used to verify multi-party transactions in sequential signatures or parallel signatures formats.

An initiator of a multi-party transaction verification procedure should not always be its verifier. For example, in some situations a sender of a multi-party transaction may also be the initiator of verification procedures. Whichever participant that distributes multi-party transaction to multiple verifiers is the initiator of the verification procedure. The initiator should know the information about all other verifiers. In both procedures the initiator performs our MPA protocol between himself/herself and multiple verifiers and establishes a secure communication session.

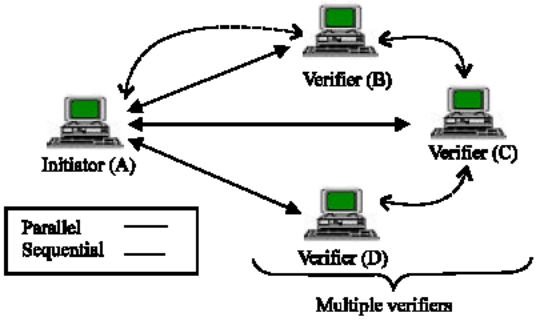

Fig. 13: Verification scenarios for multi-party transactions

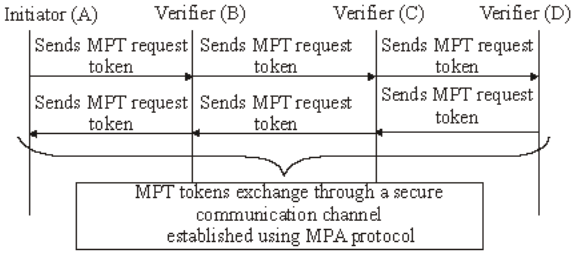

Fig. 14: Sequential verification procedure

Sequential verification procedure: Figure 14 shows the procedure which is based on sequential verification scenario. As mentioned earlier, this procedure could be used to verify transactions/documents in both sequential envelopes and parallel envelopes formats. It could also verify transactions/documents in both sequential signatures and parallel signatures formats.

In order to perform sequential verification procedure an initiator must createMPT request token with recursive MPT request token headers addressed to each verifier. The Recipient's role field of each MPT token should be Verifier. In case of sequential envelopes, recursive token header should be created according to the opening sequence of sequential envelopes. In case when an initiator of the signing procedure is also an initiator of the verification procedure (sender), he/she knows the opening sequence of sequential envelopes.A transaction/ document to be verified is attached as the body of the MPT request token. Timestamps of the attached transaction/document should be included in each MPT token header. The initiator forwards this MPT request token to the first verifier addressed by the outer MPT request token header through secure communication channel. The first verifier removes his/her MPT request token header and retrieves the attached transaction/ document. In case when the attached transaction/ document is in sequential signatures or parallel signatures formats, all signatures must be verified.

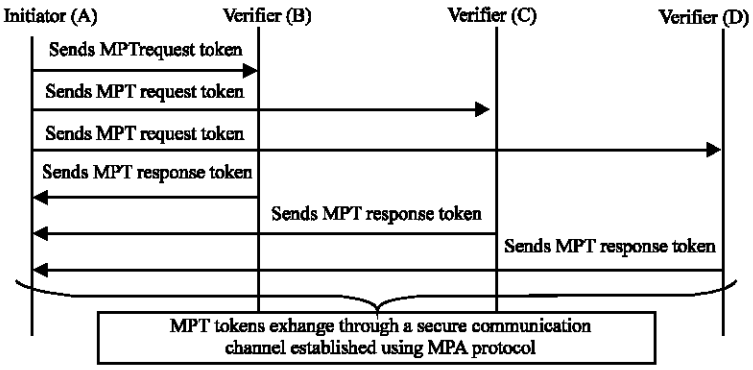

Fig. 15: Parallel verification procedure

Timestamps should also be verified against the corresponding signatures. In case when the attached transaction/document is in parallel envelopes or sequential envelopes formats, the envelope must be opened. Then the verifier must verify multiple signatures if the opened transaction/document is in sequential signatures or parallel signatures formats. However, the verifier may again get back the transaction/document in parallel envelopes or sequential envelopes format in case when the attached transaction/document is in sequential envelopes format. In that case, the verifier creates a new MPT request token attaching the opened transaction/ document to the rest of the MPT request token headers and forwards it to the next verifier through a secure communication channel. Otherwise, the verifier creates a new MPT request token attaching the received transaction/document to the rest of the MPT request token headers and forwards it to the next verifier through the secure communication channel. Finally, the verifier sends the new MPT request token to the next verifier, which is addressed by the outer MPT request token header through the secure communication channel.

All verifiers perform the same procedure until the MPT request token reaches the last verifier. The last verifier verifies all signatures in case the transaction/ document is in sequential signatures or parallel signatures format. Timestamps should also be verified against the corresponding signatures. The last verifier first opens the enveloped transaction/document and then verifies all signatures in case when the attached transaction/document is in parallel envelopes or sequential envelopes format. In both cases, he/she creates a MPT response token addressed to the initiator with the signature verification status.

The last verifier signs the transaction/document and includes the timestamp of this signature in the MPT token header. It proves that he/she received the transaction/ document and verified it at the approximate time indicated in the timestamp. This timestamp also binds the MPT response token with the transaction/document. The body of this MPT response token is empty. Finally, the last verifier sends back this MPT response token to the previous verifier through a secure communication channel.

All intermediate verifiers create a new MPT response token including their verification status and the previous MPT response token header(s). Verification status of intermediate verifiers may depend on the verification status of previous verifiers. For example, in case of sequential envelopes intermediate verifiers don't have access to the signed transaction/document and therefore verification status of the last verifier should be taken into consideration. In the study we discuss these situations with more details. Intermediate verifiers also sign the transaction/document and include the timestamp of the signature in the MPT response token header. This proves that he/she received the transaction/document and verified it at the approximate time indicated in the timestamp. In case of sequential envelopes intermediate verifiers don't have access to the transaction/document in clear form and therefore they sign the enveloped transaction/document. The body of this new MPT response is also empty. Each verifier sends this MPT response token to his/her previous verifier/initiator through a secure communication channel. All intermediate verifiers perform the same procedure until the MPT response token reaches the initiator. The final recursive MPT response token contains MPT response token headers, which are signed sequentially by each verifier. In other words, the initiator receives signed response, which indicates the verification status of the transaction/ document. The initiator may archive this token together with the transaction/document for future verifications.

As mentioned earlier, sequential verification procedure is suitable for verification of transactions/ documents in both sequential and parallel envelope

formats. Our second verification procedure, based on parallel verification scenario, is suitable only for parallel envelopes. However, it could be used to verify transactions/documents in both sequential and parallel signatures formats.

Parallel verification procedure: In the parallel verification procedure, an initiator creates separate MPT request tokens for each verifier known initially, as described in the study. The sender simultaneously sends the corresponding MPT request tokens to each verifier, as shown in Fig. 15.

Each verifier verifies all signatures in case the attached transaction/document is in sequential signatures or parallel signatures format. Timestamps should also be verified against the corresponding signatures. Each verifier opens the enveloped transaction/ document and verifies signatures of the transaction/ document in case the attached transaction/document is in parallel signatures or sequential signatures formats. Any verifier can invite any number of dynamic verifiers to the verification procedure, if the transaction/document is in sequential signatures or parallel signatures format. In that case, the verifier performs sequential verification procedure between dynamic verifiers treating himself/ herself as an initiator. Then each verifier sends an MPT response token with the verification status to the initiator. In addition, the verifier must sign the transaction/ document and include the timestamp of the signature in the MPT token header. This proves that the verifier received the transaction/document and verified it at the approximate time indicated in the timestamp. Finally, the verifier must sign the MPT token header and create MPT response token with empty token body. This MPT response token is sent back to the initiator. Parallel verification procedure is completed when MPT response tokens from all verifiers are received. The initiator may archive all MPT response tokens together with the transaction/document for future verifications. The complete multi-party transaction protocol is described.

MULTI-PARTY TRANSACTION (MPT) PROTOCOL

Multi-Party Transaction (MPT) protocol. MPT protocol is based on MPT tokens and signing, enveloping and verification procedures. Upon receipt of a MPT request token, a recipient of the MPT protocol determines if: _ the MPT request token (header and message) is well formed; if not, recipient creates MPT response token with Error status and the corresponding error code;

- Recipient name in the outer header is addressed to himself/herself; if not, recipient creates MPT response token with error status and the corresponding error code; _ recipient roles are Verifier, Signer or Sender; (Recipients could have more than one role).
- The token contains the information needed by the recipient; if any information is missing, the recipient creates MPT response token with error status and the corresponding error code; _ signature of the MPT request token header is valid. If signature verification fails, the recipient creates MPT response token with rejected status and the corresponding error code. UCVP protocol may be used in this process to verify certificates.

After performing all the tests above, the recipients must take the following actions, according to the roles assigned in the MPT token header. In the case of multiple roles, the priority must be assigned hierarchically, first to the role of the verifier, then the role of the signer and finally the role of the sender.

If the role is verifier: The verifier is one of the recipients who should process a transaction/document in the body of the MPT request token. MPT request token body may contain enveloped transaction/document or signed and enveloped transaction/document. Therefore, verifier may only open the envelope or open the envelope and verify signature(s) according to the transaction format. During the signature(s) verification, the recipient must also verify the attached timestamp(s) against the corresponding signatures. If any of these verifications fail, MPT response token with error status and the corresponding error code should be generated and sent back to the previous recipient. (For some recipients previous recipient may be the initiator or the sender of the transaction/document.)

In the next step, the recipient should determine whether he/she should generate a MPT response token or a new MPT request token. This new MPT request token contains recursive MPT request token header. This recursive MPT request token header includes valid MPT token header, which is in the Previous MPT token header field (if this field is not empty) Finally, recipient creates new MPT request token by attaching the verified transaction/document to the recursive MPT token header. This new MPT request token is sent to the next recipient, who is addressed by the outer MPT token header. If the Previous MPT token header field is empty, the recipient should generate a MPT response token with the Processed MPT token header status and an empty MPT

token body. Recipient name of this MPT response token header contains signer's name of the MPT request token. Recipient signs the transaction/document and includes the timestamp of this signature in the MPT response token header. Finally, MPT token header should be signed by the recipient. This MPT response token should be sent to his/her previous recipient. If the Previous MPT token header field contains valid MPT request token header, the recipient creates a new MPT request token. Valid MPT request token header, which is in the Previous MPT token header field, is the new MPT request token header. If the opened transaction/document is in sequential envelopes or parallel envelopes format, the opened transaction/document will be new MPT request token body. Otherwise, the received MPT request token body is the new MPT request token body. This new MPT request token is sent to the next recipient, addressed by the outer MPT token header. If the received transaction/ document or opened transaction/document is in unknown PKCS#7 data format, MPT response token with Error status and the corresponding error code should be created. This MPT response token should be sent back to the previous recipient.

If the role is Signer: Signer is one of the recipients who signs a transaction/document attached to the body of the MPT request token. The recipient who has the role of a Signer should first verify the signature(s) of the attached transaction/document and corresponding timestamp(s). If this verification fails, the participant must create MPT response token with Rejected MPT response token status and the corresponding error code. This response token should be sent back to the previous signer/initiator (For some recipients the previous signer is the initiator).

After successful signature verification of MPT token header(s) and body, the recipient should determine the signing method. In order to do that, he/she retrieves Signer Information fields from the received MPT token body. If one of these Signer Information fields contain his/her information, then he/she should create parallel signatures. In other words, the signer should complete the corresponding Signer Information field by creating his/her signature. If Signer Information field does not contain his/her information, the signer should create sequential signatures. In other words, he/she should create a new message body in the sequential signatures format by including the received signed transaction/ document. If this signature creation process fails, MPT response token with Error status and the corresponding error code should be generated.

This MPT response token should be sent back to his/her previous recipient/initiator. When creation of multiple signatures is completed, the recipient should determine where to send the signed transaction/ document. This may be based on the rest of the MPT request token header which can be retrieved from the Previous MPT token header field of the MPT request token header. In case that Previous MPT token header is empty, the signer should create MPT response token header with Processed status. Recipient's name of this MPT response token contains the initiator name. The corresponding timestamp(s) should be included in this MPT response token header. Signed transaction/ document should be attached to the body of that MPT response token header. Finally, recipient signs the MPT token header and sends it back to his/her previous recipient/initiator.

In case that Previous MPT token header field contains valid MPT request token header, the signer creates new MPT request token attaching the signed transaction/document to the valid MPT request token header. This new MPT request token is then sent to the recipient indicated in the outer MPT request token header.

If the role is Sender: Sender is one of the recipients who should distribute a transaction/document to multiple verifiers. In other words, sender is the entity who knows the information about verifiers and the verification procedure they will use to verify the transaction/ document.

The recipient who has a role of a Sender should first prepare a single MPT request token with recursive MPT tokens header or multiple MPT request tokens. If any failure occurs during this process, MPT response token with Error status and the corresponding error code should be generated. This MPT response should be sent back to the initiator.

In case of multiple MPT request tokens, the sender should simultaneously distribute the corresponding MPT request tokens to the corresponding verifiers.

However, in case of a single MPT request token with recursive MPT request token header, the sender must send the MPT request token directly to the recipient addressed by the outer MPT request token header.

Upon receipt of a MPT response token, a recipient determines if:

- The MPT response token (header and message) is well formed; if not, the recipient creates MPT response token with Error status and the corresponding error code and sends it back to the recipient who originated the MPT response token.

- The token contains the information needed by the recipient; if any required information is missing, the recipient creates MPT response token with Error status and the corresponding error code and sends it back to the recipient who originated the MPT response token.
- Signature of MPT response token header is valid. If signature verification fails, the recipient creates MPT response token with Rejected status and the corresponding error code and sends it back to the recipient who originated the MPT response token. UCVP protocol may be used in this signature verification process to verify certificates.

After performing the above tests, the recipient must take the following actions according to the recipient's name of the outer MPT response token header.

- If the outer MPT response token is addressed to himself/herself.

All MPT tokens with Processed status may be archived for future verifications.An appropriate action must be taken for all other MPT response tokens with Error and Rejected status.

- If the outer MPT response token is addressed to other recipients

In this case the recipient is an intermediate recipient in the MPT protocol. If the received MPT response token has Processed status, the recipient must have the MPT request token previously received from the same target of the received MPT response token. In addition, that MPT request token should have been processed successfully. Therefore, the recipient creates a new MPT response token with the Processed status. This new MPT response token header should be addressed to the same target recipient. If the recipient's role is Signer, timestamps of his/her signature should be included in the MPT token header. If the recipient's role is Verifier, he/she must sign the verified transaction/document and include the corresponding timestamp in the MPT response token header. The received MPT token header(s) should be included in the Previous MPT token header field. The received MPT token body should be attached as the body of this new MPT response token. If the received MPT response token body is empty, new MPT response token body should also be empty. Finally, new MPT response token is sent to the initiator/sender/previous recipient.

All other MPT response tokens with Error and Rejected status should be returned to the initiator/ sender/previous recipient of the MPT protocol. After returning that MPT response token, the recipient should close MPT session. The features of our new model of the multi-party transaction protocol may be summarized as follows. MPT protocol can perform transactions between static participants as well as dynamic participants. Since it always distributes signed transactions/documents with strong protection of private signing keys (multi-application smart cards) and with strict verification of certificates, it provides non-repudiation. MPT protocol attaches timestamps of all signatures of the transaction/ document, so signing time may be verified. The MPT response token with Processed status created during the multi-party transaction verification can be used to verify the transaction again in the future. MPT protocol always exchanges transactions/documents over secure communication channel or it envelops transactions/ documents before exchanges. Therefore, MPT protocol provides confidentiality between multiple participants at the communication level, as well as at the document level. Using MPT protocol, a transaction/document can be enveloped or signed sequentially. This means that MPT protocol allows participants to perform protection/ verification of transactions/documents according to their organizational roles.

CONCLUSION

Our model of a multi-party transaction protocol adequately addresses all security requirements for multi-party transactions. It provides functionality to create transactions with multiple signatures in different signature formats. The concept of signature formats supports creation of secure transactions based on the participant's role in an organization. Since our multi-party transaction protocol also creates timestamps of signatures, signature creation time can also be verified in the future. In general, multiple signatures together with timestamps provide authenticity, integrity and non-repudiation security services for multi-party transactions. Since our multi-party transaction protocol distributes enveloped transactions through a secure communication channel, it provides confidentiality security service during transmission, as

well as in storage. The concept of multiple transaction enveloping formats allows enveloping of transactions for multiple recipients based on their organizational role. The concept of multi-party transactions verification allows verification of transactions by multiple verifiers. Since these verifiers create signed tokens with verification results, validity of transactions can be verified and proved long after their creation and different legal assurance levels of signatures of multi-party (sequential and parallel) processing of transactions between multiple signers and verifiers. Therefore, it may be used for protecton of various electronic transactions or documents exchanged between multiple participants.

REFERENCES

PKCS7, 1993. "PKCS#7 V1.5: Cryptographic Message Syntax Standard", Public-Key Cryptography Standards, RSA Laboratories, [Online] Available at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/index.html.

Ramsdell, 1999. Ramsdell, B., "S/MIME Version 3 Message Specification", RFC2633, The Internet Engineering Task Force, [Online] Available at ftp://ftp.ietf.org/rfc/rfc2633.txt.

XML Encryption, 2001. "XML Encryption Working Group", World Wide Web Consortium (W3C), [Online] Available at http://www.w3.org/Encryption/.