Asian Journal of Information Technology 6 (8): 873-883, 2007
ISSN: 1682-3915

Multi-Party Security System

[1]L.J. Mary and [2]S.P. Rajagopalan
[1]Department of Information Technology, Dr. M. G. R. University,
Periyar E.V.R. High Road, Maduravoyal,Chennai-600 095,Tamil Nadu, India
[2]School of Computer Science and Engineering, Dr. M.G.R. University,
Chennai-600 095,Tamil Nadu, India

Abstract: Transaction protocols and network applications are rapidly changing from a simple client-server model towards multi-party, multi–purpose, multi– device models. The main objective of a model of Multi-Party Security System (Multi-PaSS) provides the security services like Security infrastructure, supporting secure transactions between several parties in an open global environment with registration, certification, distributed remote authentication and trusted security administration, Secure multi-party protocols for protection of transactions in transfer, storage, authenticity delegation, authorization forwarding, etc. and Security extensions of various network applications suitable for multi-party transactions providing user authentication, cryptography, certification function, message protection, etc.. The structure of the Multi-PaSS may be shown in the form of three layers. The bottom layer, called security platform, contains cryptographic modules and security tokens, such as ATM cards. The middle layer comprises a collection of security proxies (Multi-PaSS proxies), which can securely communicate with each other performing security operations in a multi-party environment. The top layer, security infrastructure, comprises multiple cross-certified Public Key Infrastructure (Multi-PKI domains) and other supporting servers. All participants of a transaction must communicate through Multi-PaSS proxy servers and they handle authentication and all transactions between them. This authentication is performed using sequential and parallel concept. Since, MPT protocol messages contain corresponding timestamps, verifiers can verify creation time and verification time. Multiple signatures together with timestamps provide authenticity, integrity and non-repudiation security services for multi-party transactions.

Key words: Multi-Party Security System (Mult-PaSS), Time Stamp Server (TSS), Time-Stamping Authority (TSA), Hardware Security Module (HSM), Public Key Infrastructure (PKI)

INTRODUCTION

Present computer networks are a complex assembly of databases, web and other application servers, web browsers and various network devices. In such an environment transactions are usually not simple client-server arrangements, but complex actions involving multiple participants. When more than two participants are involved in a transaction, the situation with security becomes considerably more complex (Piccinelli, 2001). With the distributed nature of computer networks and a variety of transactions, it is very difficult to specify a model of security architecture for multi-party transactions. Simple extension of client-server security protocols to multi-party security protocols is not possible (Kempster, 1999). Time stamping is also, necessary for multi-party transactions in order to archive reliable non-repudiation. Most of the Time stamping systems use a trusted third party, so called Time-Stamping Authority (TSA) (Adams, 2001). The Timestamp is a digital attestation of the TSA that the specific electronic document, together with its digital signature, has been presented to TSA at a certain time. Time stamping techniques enable verification whether an electronic document was created or signed at a certain time (Zhou, 1996). Multi-PaSS uses the concept of time stamping based on TSAs to provide long-term dispute resolution of transactions.

Security requirements: Security requirements for a model of a security architecture supporting multiparty transactions are:

Corresponding Author: L. Josephine Mary, Department of Information Technology, Dr. M. G. R. University,
Periyar E.V.R. High Road, Maduravoyal,Chennai-600 095,Tamil Nadu, India

- Unique registration of participants (users, servers etc.),
- Mobility of users and dynamic participation,
- Scaling to hundreds or thousands of participants under different security polices,
- Supporting strong authentication between participants,
- Creation of transactions with multiple signatures,
- Protection of transactions for multiple verifiers (multiple envelopes),
- Strong verification of transactions (in real-time or delayed),
- Applicability to various electronic transactions.

Timestamp: A timestamp is the current time of an event that is recorded by a computer. Through mechanisms such as the Network Time Protocol (NTP), a computer maintains accurate current time, calibrated to minute fractions of a second. Such precision makes it possible for networked computers and applications to communicate effectively. The timestamp mechanism is used for a wide variety of synchronization purposes, such as assigning a sequence order for a multi-event transaction so that if a failure occurs the transaction can be voided. Another way that a timestamp is used is to record time in relation to a particular starting point. In IP telephony, for example, the Real-time Transport Protocol (RTP) assigns sequential timestamps to voice packets so that they can be buffered by the receiver, reassembled and delivered without error. When writing a program, the programmer is usually provided an application program interface for a timestamp that the operating system can provide during program execution.

Network time protocol: Network Time Protocol (NTP) is a protocol that is used to synchronize computer clock times in a network of computers. Developed by David Mills at the University of Delaware, NTP is now an Internet standard. In common with similar protocols, NTP uses Coordinated Universal Time (UTC) to synchronize computer clock times to a millisecond and sometimes to a fraction of a millisecond.

Accurate time across a network is important for many reasons; even small fractions of a second can cause problems. For example, distributed procedures depend on coordinated times to ensure that proper sequences are followed. Security mechanisms depend on coordinated times across the network. File system updates carried out by a number of computers also depend on synchronized clock times. Air traffic control systems provide a graphic illustration of the need for coordinated times, since flight paths require very precise timing (imagine the situation if air traffic controller computer clock times were not synchronized).

UTC time is obtained using several different methods, including radio and satellite systems. Specialized receivers are available for high-level services such as the Global Positioning System (GPS) and the governments of some nations. However, it is not practical or cost-effective to equip every computer with one of these receivers. Instead, computers designated as primary time servers are outfitted with the receivers and they use protocols such as NTP to synchronize the clock times of networked computers. Degrees of separation from the UTC source are defined as strata. A radio clock (which receives true time from a dedicated transmitter or satellite navigation system) is stratum-0; a computer that is directly linked to the radio clock is stratum-1; a computer that receives its time from a stratum-1 computer is stratum-2 and so on.

The term NTP applies to both the protocol and the client/server programs that run on computers. The programs are compiled by the user as an NTP client, NTP server, or both. In basic terms, the NTP client initiates a time request exchange with the time server. As a result of this exchange, the client is able to calculate the link delay, its local offset and adjust its local clock to match the clock at the server's computer. As a rule, six exchanges over a period of about 5-10 min are required to initially set the clock. Once synchronized, the client updates the clock about once every 10 min, usually requiring only a single message exchange. Redundant servers and varied network paths are used to ensure reliability and accuracy. In addition to client/server synchronization, NTP also supports broadcast synchronization of peer computer clocks. NTP is designed to be highly fault-tolerant and scalable.

Types of timestamps

Subtracting timestamps: The result of subtracting one Timestamp (TS2) from another (TS1) is a timestamp duration that specifies the number of years, months, days, hours, minutes, seconds and microseconds between the two timestamps.

Incrementing and decrementing timestamps: The result of adding a duration to a timestamp or subtracting a duration from a timestamp is itself a timestamp.

Secure timestamping and confidential auditing: Another interesting protocol is cryptographic timestamping. The purpose is to prove that a particular piece of content (i.e. some array of bits) existed at a particular period of time.

The basic idea goes back to the anagram publication technique that Robert Hooke, Galileo and some other early scientists used to prove that they discovered certain things long before they published them.

The modern protocol uses cryptographic hash functions instead of anagrams. Any set of bits (digital content, a network event, whatever) is passed through the hash function, turning into into a unique random-looking string of bits. Those bits are then published to multiple timestamp servers on the Internet. The timestamp servers create a chain of hashes. Using the chain of published hashes, it is easy to later prove that hash was published before one event and after another event, thus proving the time of publication and the hash uniquely corresponds to a particular content.

Secure email timestamping: A server includes a dedicated hardware card that is responsible for digesting an incoming email, appending a date and time to the digest to create a time stamp and signing the result with a private digital signature. This provides a secure time stamp for an email that is resistant to falsification and tampering by the sender of an email and which can be verified by a recipient of the email.

Previous work on timestamps: Lamport proposed the assignment of integer timestamps to events of an execution involving send and receive statements (Lamport, 1978). Integer timestamps can be used to produce totally ordered sequences of events of an execution involving send and receive statements such that these sequences do not violate the "happened before" relations among events of the execution (i.e. if event e "happened before" event f, then e appears before f in any of these totally ordered sequences). However, integer timestamps cannot be used to determine the "happened before" relation between two events of the same execution. To solve this problem requires the use of vector timestamps, each consisting of n values, where n is the number of processes involved in an execution. How to assign vector timestamps for events of an execution involving asynchronous communication (i.e. non-blocking send and blocking receive) is shown in (Mattern, 1989). The assignment of vector timestamps for events of an execution involving asynchronous and/or synchronous communication is described in (Fidge, 1991). A technique for improving the implementation of vector timestamps for message passing programs was proposed in (Singhal, 1992). Netzer considered optimal tracing and replay of parallel programs that contain accesses to shared variables, but do not contain messages (Netzer, 1993). He presented an algorithm that uses vector timestamps for read and write events on shared variables in a parallel program.

During an execution of a parallel program, the number of parallel threads is usually not a constant. A number of timestamp techniques for a parallel program avoid the use of vector timestamps with their size being the total number of parallel threads in the program. Dinning and Schonberg considered parallel programs that use doall-endall statements for parallelism and some coordination statements for synchronizing accesses to shared variables (Dinning, 1990). A block of a parallel program is defined as an instruction sequence, executed by a single thread, that does not include doall, endall, or any coordination statements. A technique, called task recycling, assigns a vector timestamp to a block, where the size of this vector timestamp is the maximum number of parallel threads in the outermost doall-endall statement that contains the block. Audenaert considered parallel programs that use fork and join statements for parallelism and send and receive statements for synchronizing accesses to shared variables (Audenaert, 1997). He described a technique that assigns a clock tree, which is a tree of vector timestamps, to a fork, join, send or receive event. The average size of a clock tree is much smaller than the size of a vector timestamp based on task recycling.

Shen (Netzer, 1993) improve Yang and Shieh's timestamp-based authentication scheme to withstand Chan and Cheng's attack.

Time stamp server: Network appliance that allows you to integrate secure digital signatures and auditable time stamping functionality into security applications. nCipher's Time Stamp Server (formerly the DSE 200) is an easily deployed and cost effective time stamping solution, comprising a networked appliance and a developer's toolkit. It allows you to integrate secure digital signatures and auditable time stamping functionality into your applications. Once calibrated, via an authenticated secure network connection, the Time Stamp Server (TSS) is ready to provide time stamps to any PKIX-compliant time stamp request; avoiding reliance on the system clock of host servers which are unreliable and vulnerable to tampering.

Time certain technology: Time certain's technology is a service which provides cryptographically secure timestamps of customer data. These timestamps are provided by a TimeCertain-owned proprietary server appliance which resides on the customer's local network. The time certain chronologics server appliance is a self-contained, secure, dedicated device, consisting of a host

machine and an internal FIPS 140-1 Level 3 validated Hardware Security Module (HSM). The appliance is rack mountable and has a 2U height. The appliance listens for timestamp requests on a specified port. The appliance is only configurable through the directly connected keyboard and monitor. The service (and appliance) is non-invasive with respect to a firewall -- the customer need not open any 'holes' through a firewall.

A Timestamp request includes a SHA-1 cryptographic digest of the original data. The TimeCertain Chronologics Server Appliance passes this digest into its internal HSM(Hierarchical Storage Management). This HSM contains a secure internally generated RSA private key, which exists solely for the purpose of timestamping; an internal secure clock, from which it obtains the time data; and a secure monotonically increasing counter, from which it obtains a serial number. The HSM constructs a timestamp which includes the time data and serial number. The HSM signs the timestamp using the secure private key. The timestamp also, includes either the certificate required to verify the signature or some identifier of that certificate. The appliance returns the timestamp to the requestor. The timestamp and integrity of the original data can later be verified. The HSM internal secure clock cannot be synchronized without the agreement of both the customer and time certain. Time certain also provides a Client Toolbox, which facilitates generating timestamp requests, sending requests to a time certain chronologics server appliance, retrieving responses from the server and verifying both the timestamp and integrity of the original data. The toolbox is available for Java development and will soon be available for C development. The time certain client toolbox also includes sample code to facilitate integration into specific infrastructures and processes. The TimeCertain Client Toolbox comes with no warranty and is provided as a courtesy to developers.

Timestamps can be requested from the server by any software capable of constructing an ASN.1 data stream, generating a SHA-1 hash and performing TCP/IP communication. Verification can be performed by any software capable of parsing an ASN.1 data stream, generating a SHA-1 hash and performing RSA-SHA1 signature verification.

Time stamping ensures the security and integrity of enterprise data exchange: In the information age, enterprises have come to rely heavily on the electronic exchange of information. Time stamping, consequently, has become a necessity for companies needing to verify the exact time a document was created and/or modified. This technology has become one of the most important aspects of Public Key Infrastructure (PKI) technology. in the absence of accurate time stamping significant negative consequences materialize when some industries, such as financial services, require accuracy within fractions of a second. nCipher delivers a timestamp solution that is highly accurate, secure and verifiable, allowing the organization to integrate secure digital signatures and auditable time stamping into their critical applications. click here to read more about time stamping solutions.

nCipher delivers secure auditable time stamping: nCipher helps organizations protect critical data and offers solutions in identity management, data protection, enterprise key management and cryptographic hardware. nCipher's time stamping solutions include:

The Time Source Master Clock (TMC) 200 is a network appliance that securely distributes precise time throughout the enterprise. A Rubidium atomic clock delivers remarkable accuracy, mitigating the need to regularly re-calibrate the device. Using DS/NTP, a secure transit protocol incorporating mutual authentication, the TMC200 establishes a secure link to a Secure Root Clock or Time Stamp Server. The TMC200 produces a signed certificate verifying the traceability and calibration of the time stamp at the end of the transaction. A FIPS 140-2 Level 3 Hardware Security Module helps secure cryptographic keys to prevent time values from being altered during transit.

Time Stamp Server (TSS) is nCipher's cost effective solution for integrating secure digital signature and auditable time stamping into applications. The TSS appliance is a networked server that cryptographically

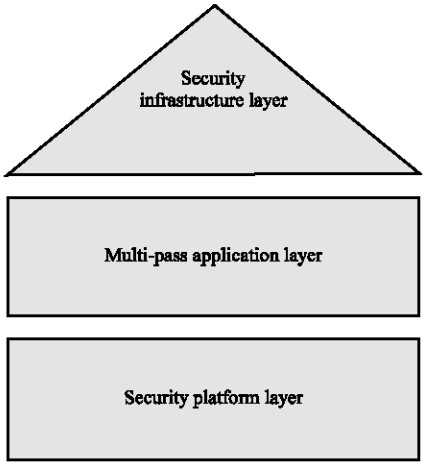

Fig. 1: The layered architecture of Multi-PaSS

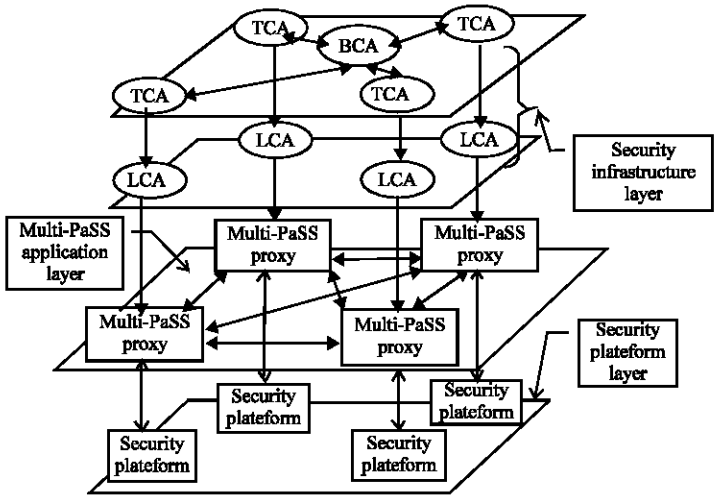

Fig. 2: The detail structure of Multi-PaSS

signs, seals and timestamps documents, delivering a digital signature or stamp for any PKIX-compliant request. This allows organizations to avoid system clocks on host servers which can be highly unreliable and easily compromised. Designed to operate with independently provided calibration and audit services, the TSS delivers secure and traceable links to official Coordinated Universal Time (UTC) time sources. nCipher's TCC contains a developers tool kit that includes sample code for quick and simple integration of document signing functionality into critical applications.

Beyond time stamping: additional solutions for data protection: nCipher delivers a number of solutions that allow enterprises to identify who can access data, to protect data during transit and at rest and to comply with privacy-driven regulations. nCipher's user management and provisioning system allows enterprises to easily and efficiently manage user identities. nCipher's data protection solution comprises access controls and cryptographic hardware. An enterprise key management solution automates key delivery to distributed applications. And cryptographic hardware helps accelerate Secure Socket Layer (SSL) operations, protect sensitive application code and secure encryption and signing keys. nCipher's timestamping products allow organizations to integrate secure digital signatures and auditable time stamping functionality into their critical applications. Timestamping has emerged as one of the key components of Public Key Infrastructure (PKI) technology, delivering non-repudiation and ensuring that the integrity of data is verifiable at a future point in time.

Time Source Master Clock (TMC) 200: The TMC200 provides secure distribution of accurate time to multiple Time Stamp Servers. nCipher's TimeSource Master Clock (TMC200) is a network appliance incorporating a Rubidium atomic clock that can securely distribute accurate time throughout an organization. Deploying an authenticated and encrypted version of the industry-standard NTP protocol ensures the secure delivery of auditable time to multiple Time Stamp Servers from a single source.

Hence our new model of a Multi-Party Security System (Multi-PaSS) provides the security services like Security infrastructure, Secure multi-party protocols and Security extensions of various network applications suitable for multi-party transactions. All participants of a transaction must communicate through Multi-PaSS proxy servers and they handle authentication and all transactions between them. This authentication is performed using sequential and parallel concept. Since MPT protocol messages contain corresponding timestamps, verifiers can verify creation time and verification time. Multiple signatures together with timestamps provide authenticity, integrity and non-repudiation security services for multi-party transactions. Time Stamp Server provides timestamp services to participants of multi-party transaction.

MODEL OF MULTI-PARTY SECURITY SYSTEM

Internal structure of multi-PaSS: The concept of a Multi-PaSS is based on several concepts and protocols. In this study, the internal structure of Multi-PaSS is described.

Overall layered Architecture: All components and protocols of Multi-PaSS may be categorized in three layers, as follows.

- The bottom, supporting layer is the so–called security platform, comprising cryptographic modules, certification modules, multi-application ATM card modules, encapsulation modules, all local security administration modules, etc.
- The middle layer is collection of Multi-PaSS servers (security proxies) interconnected through an open network into a homogeneous and transparent security infrastructure providing communication security services, protection of transactions, secure processing of transaction, etc using Multi-party Authentication and Transaction protocols.
- The top layer, security infrastructure layer, consists of certification authority servers in the form of the hierarchy or mesh of certification infrastructures, each performing registration, certification, authorization and ATM cards administration services. Certificate infrastructures in this layer may be either mutually cross-certified or cross-certified through a Bridge Certification Authority (BCA). In addition, various other types of security supporting servers, such as time-stamping servers and X.500 directory servers, are also located in this layer.

Figure 1 shows this layered architecture of Multi-PaSS. Each layer is a complex assembly of components and protocols.

Figure 2 shows the detailed structure of Multi-PaSS. As shown in the figure, security infrastructure layer consists of different types of CA servers, such as Local CA (LCA), Top Level CA (TCA) and Bridge CA (BCA). In addition, we assume that all TCAs are cross-certified through a BCA. Middle layer is a collection of Multi-PaSS proxy servers interconnected through an open network. In order to perform multi-party transactions, all users and servers (participants) must communicate through these Multi-PaSS proxy servers. The bottom layer is security platform layer, which consists of cryptographic modules, ATM card modules and other supporting security modules. Multi-PaSS proxy servers use these modules in order to perform multi-party transactions. This study describes important components and features of each of these layers in detail.

Security infrastructure layer: Security infrastructure layer is used to establish the overall trust model between participants of multi-party transactions. All participants performing multi-party transactions must be registered by some certification authority located in a security infrastructure layer. Certification Authority, which performs end user/server registration, is usually called Local Certification Authority (LCA). LCA provides user/server registration and certification. LCA should be "close" to the end users/servers and therefore located in an organization of a particular community of users/servers. In practice, individual LCA servers are usually located in individual security domains. When registering with a LCA, a particular user/server automatically accepts LCA's certification policy. Therefore, each participant must check whether LCA's certification policy is acceptable for his/her requirements before registering with the LCA. The LCA allows to store registration information and certificates in a X.500 directory server or in a personal security token. Secret information, such as private keys, must be stored in a personal security token, such as ATM card. In addition to regular CA (Certification Authority) functions, LCA in Multi-PaSS may provide some other functions such as certificate path building and verification. We assign these new roles to the LCA because:

- LCA is the entity trusted by all its users/servers and
- LCA knows the latest status of its certificates.

This extended LCA functionality provides architecture for efficient verification of certificate, which is suitable for multi-party transactions. In general, each local certification authority in a Multi-PaSS performs the following functions:

- Registration of users/applications (participants);
- Certification of participants;
- Fetching/verifying certificates from high level certificate authorities.

According to its certification policy;

- Issuing and administration of personal security tokens, such as multiapplication ATM cards;
- Creating and maintaining certificate revocation information;
- Certificate verification and path building. Top-level Certificate Authority (TCA), which has self-signed certificate, is the other important component in a security infrastructure layer. In order to perform multi-party transactions with the participants belonging to various certificate infrastructures, TCAs in these certificate hierarchies must be cross certified either directly or through a Bridge Certificate Authority (BCA). Crosscertification concept and procedure using a Bridge Certificate.In addition to CA servers, timestamp servers are another important component of the security infrastructure layer. Timestamp servers provide timestamp services to participants of multi-party transactions. LCA may also function as a timestamp server. However, in practice, a timestamp server is usually a separate sever, certified itself by some LCA.

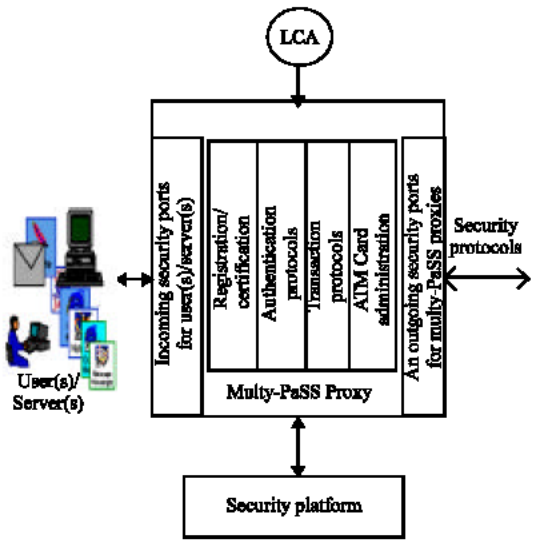

Fig. 3: Internal Architecture of a Multi-PaSS Proxy Server

In addition to these security related servers, some supporting servers, such as X.500 directory servers, are also located in a security infrastructure layer. X.500 directory servers are used to store information, such as user certificates. LCA may operate X.500 directory server for its users/servers. In case when timestamp servers or directory servers are separate servers, they should be also certified by a LCA.

Multi-PaSS application layer: Multi-PaSS application layer is the core layer of Multi-PaSS. It consists of a set of proxy servers called Multi-PaSS proxies. In other words, Multi-PaSS separates security services from applications and assigns these security services to some kind of an application level security server called Multi-PaSS proxy server. Figure 3 shows the internal architecture of a Multi-PaSS proxy server. Multi-PaSS proxy servers listen to several network ports for security services for other Multi-PaSS proxies and security services for its local participants. These ports are called outgoing security ports and incoming security ports, respectively. All participants (users/servers) must communicate with their local Multi-PaSS proxy server through its incoming security ports. For example, a web browser that requires security services from a Multi-PaSS proxy server could access it as an HTTP proxy server through an incoming security port.

However, configuration procedure for applications to communicate with a Multi-PaSS proxy server is application dependent. Somehow, all local application clients must be configured to communicate via a Multi-PaSS proxy's incoming security port. An outgoing security ports of a Multi-PaSS proxy server allows only connections from other Multi-PaSS proxies.

As shown in the Fig. 3, Multi-PaSS proxy server performs registration and certification, some functions of the ATM cards administration, authentication of local and remote users and protection of transactions on behalf of its local users/servers. For example, all local participants request certification services from the LCA server through Multi-PaSS proxy server. In other words, Multi-PaSS proxy server sends certificate request to the LCA on behalf of the participant, obtains a certificate and stores it back in the participant's personal security token. In addition to that, Multi-PaSS proxy server cooperates with LCAs when performing certificate verification.

Multi-PaSS proxy server uses MPA (Multi-Party Authentication) protocol and MPT (Multi-Party Transaction) protocol to perform multi-party transactions. These protocols are performed between Multi-PaSS proxy servers through multiple outgoing security ports on behalf of users/servers.The implementations of these security services are in the security platform layer, in the form of security services modules. Multi-PaSS proxy server uses necessary security modules in order to perform all security protocols and services required for multi-party transactions.

Security platform layer: All cryptographic modules and implementation of security protocols are in the security platform layer, as service modules. In other words, security platform layer contains all necessary service modules, which are required for functioning of a Multi-PaSS proxy server.

Protocols and service modules from different vendors may also be installed in the security platform layer. In that case, Multi-PaSS proxy server should select the vender and necessary service module before executing the corresponding security service. In addition, Multi-PaSS proxy server can download dynamically unavailable services through an open network. Figure 4 shows the internal architecture of the security platform layer. The ATM card services module provides functionality of a multi-application ATM card. Certification services module provides all registration and certification functions. Multi-party authentication protocol is implemented and interfaced via authentication services. Multi-party transactions module provides necessary functions for the MPT protocol. Not only Multi-PaSS proxy servers themselves, but also service modules in the security platform layer use services from each other. For example, certification services module in the security platform layer uses services from other service modules, such as ATM card service modules. Therefore, security platform layer

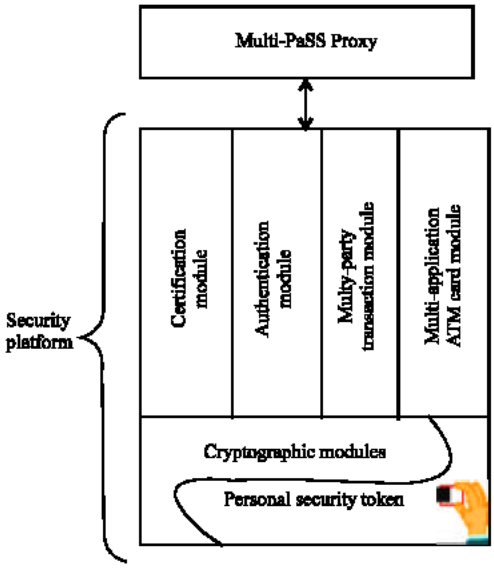

Fig. 4: Internal architecture of a security platform layer

provides services to the Multi-PaSS proxy servers as well as to other service modules in the security platform layer.

In addition to the described modules, security platform layer also, contains several other supporting service modules, such as modules necessary to access X.500 directories. In general, security platform layer consists of all service modules required for multi-party transactions. Since, Multi-PaSS proxy servers can dynamically download necessary services from remote servers into multiapplication ATM cards, service modules available in the security platform layer may be dynamically updated.

Multi-party transactions processing: This study first describes how participants organize themselves in groups in order to process multi-party transactions. Then it describes the arrangement of the model of security architecture (Multi-PaSS) in an organization performing multi-party transactions. Finally, it explains the creation and verification process for multi-party transactions.

Groups of signers and verifiers: Multiple participants in different cross-certified PKI domains can perform signing and verification of multi-party transactions. These participants could be called signing group and verification group. Figure 5 shows these signing and verification groups belonging to different cross-certified PKI domains. Doted circles show these cross-certified PKI domains. Small circles are the participants of multi-party transactions, registered in these individual PKI domains.

In order to create a signed transaction, the participants must first organize themselves as multiple

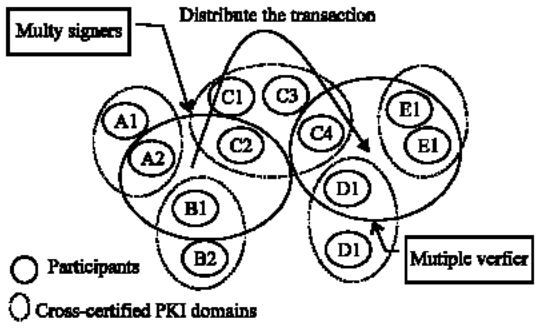

Fig. 5: Signers and verifiers of multi-party transactions

signers using MPA protocol. In our example, participants A2, B1 and C2 from three different PKI domains are grouped together as multiple signers.

After creation, a signed transaction is distributed to multiple verifiers. These verifiers may also belong to different PKI domains. Using MPA protocol they can establish a verification group. In our example, participants C4, D1, E1 and E2 from three different PKI domains are organized as multiple verifiers of the transaction.

The arrangement of multi-PaSS concept: This study describes how to arrange Multi-PaSS instances in order to perform multi-party transactions. In order to reduce complexity, we consider only two organizations (PKI domains) and assume all signers belong to one organization and all verifiers belong to the other organization. Figure 6 shows the arrangement of a Multi-PaSS instances between these two organizations.

Let's assume that users and a server in organization A need to perform financial transaction with a user and servers in organization B. Therefore, users and the server in organization A are the signers of the transaction and the user and servers in organization B are the verifiers of the transaction. As shown in Fig. 6, these two organizations have individual hierarchical certification infrastructures. These hierarchical certification infrastructures are cross-certified by a Bridge CA (BCA). Each user/server is certified by a Local CA (LCA) at the bottom level of each certification hierarchy. These registration and certification procedures are performed through Multi-PaSS proxy servers.

As shown in Fig. 6, Multi-PaSS proxy servers must be available for each user/server in their local computer environments. All users and servers must communicate with each other through Multi-PaSS proxy servers. In other words, all users/servers request all necessary security services from their local Multi-PaSS proxy servers. More than one user/server can share a single Multi-PaSS proxy server within their local environment.

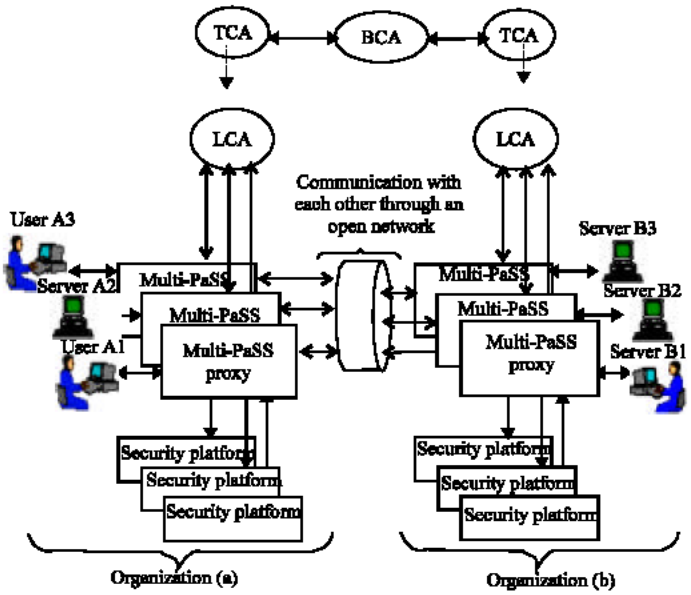

Fig. 6: Arrangement of Multi-PaSS concept between two organizations

As mentioned earlier, Multi-PaSS proxy servers perform authentication, protection of transactions and all other security services on behalf of users/servers.

After registration and certification of all users/servers, they are ready to performmulti-party transactions. Users and the server in organization A must first sign the transaction. User A1 may be the initiator of the signing process and user A3 may be the sender of the signed transaction. In order to sign the transaction, user A1 first performs MPA protocol between server A2, user A3 and himself/herself. Then he/she performs the MPT protocol and creates the transaction with multiple signatures. The recipients of the transactions are user B1, servers B2 and B3 in the organization B. Since, user A3 is the sender of the transaction, he/she sends the transaction to these multiple recipients. This distribution should be handled according to the enveloping format of the transaction. Since, these recipients are the verifiers of the transaction, they perform MPA protocols and verify the transaction. In case of real-time verification, an initiator of the verification process is user A3 in the organization A, because he/she sends the transaction to verifiers. The user and servers in organization B can also verify the transaction later without verifying it immediately. In that case, any participant, let's say server B3, can be an initiator of the verification process.

Creation of multi-party transactions: The signature creation of multi-party transactions can be based on sequential or parallel concepts. Both concepts scale to any number of participants, even in different cross-certified PKI domains. The participants perform MPA protocol for multiparty authentication before the MPT protocol for multiple signature creation. In case that MPA protocol fails, MPT protocol should not be executed. If MPA protocol fails with some non-critical participant, other participants may continue, eliminating the failed participant. In that case the rest of participants must re-execute MPA protocol and thus establish a session for signature creation. MPT protocols should be performed only after an MPA protocol establishes security session. Since all signers are authenticated before execution of an MPT protocol, an initiator can trust all signers. In addition, if a signer forges his/her signature, the initiator can detect that and prove who forged the signature.

When we consider the number of messages, sequential signing concept is more efficient than parallel signing concept. In other words, sequential signing concept has less network communication overload than parallel signing concept. However, when we consider the computational time, parallel signing concept is more efficient since everybody signs the transaction in parallel.

In a sequential signing concept, if one of the signers fails to create his/her signature, the complete signature creation process is jeopardized. If parallel signing concept is used, other signers may continue ignoring the interrupted signer, if his/her signature is not crucial. In addition, in the sequential signing concept some signers receive the information about other signers during the signing protocol. Therefore, if the sequential signing concept is used to create parallel signatures, the signatures are not conceptually independent. However, in

the parallel signing concept, only the initiator knows about all signers and vice versa. Therefore, parallel concept is suitable for transactions when signers need to keep privacy between each other. In other words, parallel signing concept is recommended to create truly independent parallel signatures.

Sequential signing concept is ideal for creation of transactions in a sequential signatures format, since sequential signatures are always dependent on the signatures of the pervious signer.

Verification of multi-party transactions: This study discusses verification process for multi-party transactions. As in the case of the creation process, verification process may also be based on parallel or sequential concepts. Both verification concepts scale to any number of participants belonging to different cross-certified PKI domains. As in the case of creation, participants must perform MPA protocol before the MPT protocol for verification. In case that MPA protocol fails, MPT protocol should not be executed. If MPA protocol fails with some non-critical participant, other participants can continue, eliminating the failed participant. In that case the rest of participants must re-execute MPA protocol and thus establish a session for verification. MPT protocol may be performed only after an MPA protocol establishes the session.

In case of the sequential verification concept verifiers receive the transaction one after the other. In other words, verifiers verify the transaction sequentially one after the other. Therefore, in this concept each verifier has a possibility to interrupt the verification and jeopardize the verification process. In addition, some verifiers receive the information about other verifiers and their verification statuses before creation of his/her verification status. Therefore, sequential verification concept does not provide truly independent verification. It conceptually and functionally depends on each verifier. However, when we consider the number of messages required, sequential verification concept is more efficient than parallel verification concept. In other words, sequential verification concept has less network communication overhead.

In the parallel verification concept, verifiers receive only information about an initiator and vice versa. In other words, verifiers do not receive information about other verifiers and their verification statuses. Therefore, parallel verification process produces truly independent verification results. In addition, in parallel verification concept, a failure of one verifier does not effect the rest of the verification process. Since, during parallel verification concept verifiers verify transactions simultaneously, this concept is computationally more efficient than sequential verification concept.

If a multi-party transaction is in a sequential envelopes format, sequential verification concept must be used. In that case, only the last verifier has the possibility to verify the signatures of the transaction. Therefore, verification status of the last verifier is critical and other verifiers have to trust the last verifier's verification status. Since, the MPA protocol is performed between verifiers before an MPT protocol, verifiers can trust each other. In case of parallel envelopes both verification concepts can be used. In that case, each verifier has a possibility to retrieve the transaction and therefore, verify signatures himself/herself. Both verification concepts can be used to verify multi-party transactions in both sequential signatures and parallel signatures formats. In the case of sequential signatures, if any of the signature verification fails, all other signatures which include that signature could not be verified. Hence, verifiers should not accept any transaction in the case of any sequential signature verification failures.

However, for some type of transactions, verifiers may accept signatures up to the failure point. Since parallel signatures are independent, if some signature verification fails, the verifier can still accept the other signatures which are verified. Therefore, in parallel signatures the transaction may be completed with the parties who had valid signatures.

As a result of this short discussion, the following recommendations are suggested for multi-party transaction verification process:

- In case when truly independent verification results are required, parallel verification concept is recommended;
- Sequential verification concept must be used if transaction is in a sequential envelopes format;

CONCLUSION

The Multi-PaSS concept dealt in this study is the new model of a security architecture for multi-party transaction. Multi-PaSS comprises theoretical concepts, security protocols and secure multi-party applications required for multi-party transactions and provides security services.The Multi-PaSS proxy server in Multi-PaSS application layer separates security services from individual applications. All participants of a transaction must communicate through Multi-PaSS proxy servers and they handle authentication and all transactions between them. This authentication is performed using sequential and parallel concept. Since, MPT protocol messages

contain corresponding timestamps, verifiers can verify creation time and verification time. Multiple signatures together with timestamps provide authenticity, integrity and non-repudiation security services for multi-party transactions. In general, Multi-PaSS provides the first set of ideas which describe all necessary concept, components and protocols required for multi-party transactions. However, research should be done to prove it statistically or mathematically. In addition, formal mathematical analysis has to be performed in order to prove our new protocols do not have any known security vulnerabilities, such as man-in-the-middle attack.

REFERENCES

Adams, A.C., P. Cain, D. Pinkas and R. Zuccherato, 2001. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC3161, The Internet Engineering Task Force, ftp://ftp.ietf.org/rfc/rfc3161.txt.

Audenaert and K. Audenaert, 1997. Clock Trees: Logical Clocks for Programs with Nested Parallelism. IEEE. Trans. Software Eng., 23: 10.

Dinning, A. and E. Schonberg, 1990. An Empirical Comparison of Monitoring Algorithms for Access Anomaly Detection. ACM Symp. Principles and Practice of Parallel Programming, pp: 1-10.

Fidge, C.J., 1991. Logical Time in Distributed Computing Systems, IEEE Computer, pp: 28-33.

Kempster, T., C. Stirling and P. Thanisch, 1999. A Critical Analysis of the Transaction Internet Protocol. The Proceedings of the 2nd International Conference on Telecommunications and Electronic Commerce (ICTEC 99), Nashville, USA.

Lamport, L., 1978. Time, Clocks and the Ordering of Events in a Distributed System. Comm. ACM., pp: 558-565.

Mattern, F., 1989. Virtual Time and Global States of Distributed Systems, Parallel and Distributed Algorithms M. Cosnard *et al.* (Eds.). Elsevier Scie. North Holland, pp: 215-226.

Netzer, R.H.B., 1993. Optimal Tracing and Replay for Debugging Shared-Memory Parallel Programs, 3rd ACM/ONR Workshop on Parallel and Distributed Debugging.

Piccinelli, G. and L. Mokrushin, 2001. Dynamic E-Service Composition in DySCo, Distributed Computing Systems Workshop, International Conference on Distributed Computing, Mesa, Arizona, pp: 88-93.

Singhal, M. and A. Kshemkalyani, 1992. An efficient implementation of vector clocks. Inf. Proc. Lett., 43: 47-52.

Zhou, J. and 1996. D. Gollmann, Observations on Non-Repudiation, Advances in Cryptology-Asiacrypt, Lecture Notes in Computer Science,.Springer-Verlag, 1163: 133-144.