A Meta-Framework for Secure Enterprise Computing

PROJECT SUMMARY

In this project, we develop a framework for secure and reliable distributed enterprise computing. With the advances in internetworking and communications, widely distributed systems have become the choice computing paradigm for enterprises and individuals. Despite their advantages, security and reliability of such systems are still suspect, at best, and is of enormous interest to the research community. In this project we address the security and reliability issues in such systems and offer solutions that are cost-effective, backward compatible, and highly scalable.

The key idea behind our approach is the real-time monitoring of behavior of protocols to ensure their secure and reliable progression. The monitoring is achieved through observing various parameter values and their temporal and spatial validity. It has been observed by the research community that security and reliability are two competing concerns and often, solutions proposed are disjoint, addressing only either one of the problems. In our approach, we use a common framework to monitor both security and reliability, which enables us to detect any unexpected behavior that may be caused by intruder intervention, communication channel errors, program compilation errors, or hardware faults.

As authentication protocols play a vital role in ensuring security of communication channels [3,4,5], our focus in security is to protect authentication protocols from intruder attacks. Our approach to reliability is based upon real-time monitoring of control-flow behaviors and application specific data value checking. The approach is highly scalable as it lends well to hierarchical deployment. It can be implemented along with any existing protocols and systems with minimal changes to the infrastructure and hence backward compatible and cost-effective. Further, it offers extreme flexibility as the meta-channel can be implemented at a later stage in the deployment of the network.

In the first part of the project, we develop the new framework based on meta-servers and channels. Detailed architecture, trust and fault models, and protocols will be developed for the framework to enable real-time monitoring and recovery. Using formal models and prototypes we conduct extensive studies on the strengths and weaknesses of the proposed techniques against all known attacks and failure scenarios.

Scientific Merit of the Proposed Work: In this project, we propose a unified approach to security and reliability of widely distributed enterprise computing environment. Though security and reliability are two essential tenets of information safety, their requirements are often contradictory; security strives to minimize accesses, whereas, reliability requires abundance of access. From the information theoretic point of view, secure systems require high entropy, leaving no room for guessing; reliable systems need low entropy with potential to reconstruct information in case of loss. Our contribution is in finding a common ground to treat security and reliability in a uniform manner. The research in this project will contribute to the body of knowledge in areas such as design of distributed protocols, authentication protocols, formal verification of protocols, concurrent error detection, distributed error recovery, and protocol simulations.

Broader Impact: The project has wide ranging implications on the general protocol and system validation. Further, the techniques developed here will enable developers to focus on the features of applications and protocols rather than worrying about fool proofing them. This will result in applications that are more imaginative. Further, the methods proposed here will positively affect the bottom lines of corporations and other entities using enterprise computing as the option is made available to deploy security and reliability features in a later stage of the design cycle.

PROJECT DESCRIPTION

A META-FRAMEWORK FOR SECURE ENTERPRISE COMPUTING

1. INTRODUCTION

Revolution in communication technology and advances in distributed management of resources have made enterprise computing a viable option to industries and organizations. Providing acceptable levels of security and reliability to such systems will determine the penetration and longevity of this paradigm. As increasingly sensitive data is generated, stored and exchanged, the threat of unauthorized access and malicious manipulation of such information increases. Recent years have seen even ordinary individuals and organizations become attractive targets for malicious activities, made easier by the proliferation of communication networks and the availability of enormous computing resources to support the attacks. With the model of transparent cooperation between computers and resources, the availability of services dependent on fault-free operation all the nodes, communication channels and the protocols that help them interact. Unfortunately, quite often the end-user may not have full control on the operation of various component systems. Reliable and secure co-ordination of operations are further hampered by complex concurrency control requirements.

A basic requirement that a good security framework should meet is authentication, the establishment of identities. Authentication essentially involves proving claims of identity or authenticity. A weak authentication protocol can seriously jeopardize the security of an information system, by failing to prevent illegitimate access to its resources. Over the years, many authentication protocols that were once believed to be foolproof have been shown to be vulnerable to a variety of attacks. Despite the advances in protocol validation and verification, skepticism still prevails on potential vulnerabilities of these protocols.

The approach to security is borne out from the observation that most of the attacks can be prevented by tightly controlling the state transitions and input conditions of even a weakly verified protocol. In the proposed work, we achieve this through real-time monitoring of the protocol footprint using a meta-framework. In the project, propose to use the same framework for monitoring the behavior of the system to capture any run-time violation caused by hardware faults, communication channel errors, or compilation errors. This proposal to use a unified framework for security and reliability monitoring is built upon the team's extensive research experience in behavior-based concurrent error detection techniques, network security, and distributed computing algorithms and architectures.

2. PROBLEM DESCRIPTION AND PREVIOUS SOLUTIONS

With proliferation of robust internetworking protocols, use of widely distributed computing connected over the Internet has become a reality. Unfortunately, with this

growth comes increasing concern about the security of transactions over the Internet. Due to the decentralized and heterogeneous architecture of the Internet, one cannot exclude the possibility that the messages between the legitimate parties go through a malicious intruder. The routing mechanisms in the Internet also do not offer any protection against malicious attacks. Various security services against known attacks include authentication services, access control services, and data confidentiality and integrity services. Network protocols that implement these services make use of mechanisms such as data encryption and digital signatures. The overall security of the service depends on both the protocol and the underlying encryption mechanism.

In the past most of the research addressing transaction security has been focused on the encryption algorithms. However, even when the underlying encryption scheme is foolproof, the protocol may still be vulnerable to intruder attacks. For instance, there are well-known chosen cipher-text attacks against implementation of RSA scheme without breaking a sweat to break the encryption. Thus, the validation of these protocols has become central to the security of the system. Existing informal approaches to validation have proven to be highly unreliable, while application of formal approaches has been limited either due to their complexity or due to their over-simplification in trying to reduce the complexity. We believe that formal or informal validation of protocols, along with run-time checking of assumptions made during the verification or validation, will allow us to deal with the complex implementations of real protocols.

Security and reliability of distributed systems are intertwined. A failure in one or more components of the system could result in security compromises and vice versa. For example if the time-stamp generator fails, the system could be exposed to a replay attack. Similarly, denial-of-service attacks in any form will reduce the availability of the system. There are attacks on security protocols that will be misconstrued as concurrency failures. If replicate data to make system fault tolerant, we may risk the confidentiality of the data. Thus, it is imperative to address the security and reliability of enterprise systems in a unified way.

2.1 Security

In one form or another, most information systems and networks support authentication of peer entities and data origin [1] [2]. Authentication is necessary to prevent misuse of resources and services, to ensure data integrity, and to establish proof of origin of data. Peer entity authentication enables an entity in a communication network to verify that a peer in a communication session is indeed the entity that it claims to be. In this way, it is assured that the peer is not trying to masquerade as another and that it is not trying to bypass authentication through a replay of earlier authenticated sessions. Data origin authentication, on the other hand, establishes the authenticity of the source of a message, thus assuring that the message really came from the claimed source.

Three broad categories of authentication techniques are widely used in information systems, the specific choice depending on the application domain.

- Proof by knowledge, where authentication is based on something the claiming

entity knows (e.g., passwords, identification numbers). By far, this is the simplest and most widely used technique in current communication networks.

- Proof by possession, where authentication is based on something the claiming entity possesses (e.g., keys, identity cards, other physical tokens). This technique is more expensive, but is used in environments where enhanced security is required.
- Proof by property, where authentication is based on some property of the claimer (e.g., biometric properties such as voice patterns, fingerprints, and facial images). This is complex and very expensive, but may be required in situations demanding utmost security.

We limit the scope of authentication in our work to proof by knowledge, since this is the most relevant form in communication networks. The `proof' in this case is established through authentication protocols that comprise special handshake messages between communicating principals. As these protocols are as vulnerable to eavesdropping as any other, cryptographic techniques are almost invariably used to protect this knowledge verification phase of communication sessions.

In benign environments, most authentication protocols function properly. However, they may fail in the presence of adversaries (intruders) capable of observing and manipulating protocols messages. The situation is complicated because, quite often, a malicious entity may itself be a valid principal in the system. Under such circumstances, it is assumed that an intruder may have the ability to observe, insert, modify, or delete messages exchanged through the network. However, according to the strong cryptography assumption, the intruder is considered incapable of extracting information from an encrypted message, unless the corresponding key is in its possession through some means (e.g., key compromise). Essentially, keys chosen through guesswork are considered virtually useless because of this assumption.

An adversary may be able to attack and defeat authentication by exploiting vulnerabilities in protocol structure, message sequences, and timing relations. Some of the well-known attacks are:

1. Key guessing attack: In this brute force approach, an intruder tries to guess the right key from the entire key space. The attack may succeed when the choice of keys is not carefully made, or where the key space is small enough for an exhaustive search.

2. Known plain text attack: Simple protocols in which parts of encrypted messages are publicly known (or can be easily deduced) are vulnerable to this attack. Using some known plain text (e.g., names, addresses) and the corresponding ciphertext, an adversary may be able to break the cryptographic system and recover the key or other secrets.

3. Chosen cipher-text attack: In this form of attack, an intruder manages to get legitimate principals to encrypt some carefully chosen data. The resulting cipher-

text is then used to help break the cryptographic code.

4. Replay attack: An intruder accumulates tables of messages between legitimate principals and replays them when the right opportunity arises. Essentially, an old message (from a past valid session) is passed off as current. Unsuspecting principals may accept such old message as fresh, potentially allowing the intruder to establish falsely authenticated sessions. Principals may even be tricked into using old compromised keys, enabling the intruder to eavesdrop on all further `encrypted' communication.

5. Oracle session attack: An intruder uses a principal as an `oracle' in its malicious actions against another. The intruder engages in authentication sessions with two principals and tricks one of them into performing the encryption and decryption required in the session with the second. Such attacks often succeed due to vulnerabilities in protocol message structure or message sequences.

6. Parallel session attack: An intruder exploits a legitimate principal to act against itself. The intruder engages in two simultaneous authentication sessions with the same principal and manipulates one session to generate messages needed in the other. One of the sessions may eventually pass authentication criteria.

Of these, key-guessing attacks, known plain-text attacks, and chosen plain-text attacks are actually attacks on the underlying cryptographic mechanism and not directly on the authentication protocol logic. Hence, based on the strong cryptography assumption, such attacks are generally not considered while analyzing authentication protocols, but are rather left to studies on cryptographic algorithms. Attacks that exploit weaknesses in the freshness assumption (such as replay of old messages) are often collectively called interleaving attack, and constitute an important class of attacks on authentication protocols [7].

A malicious entity with sufficient resources and intelligence may potentially be able to identify and exploit these weaknesses, thereby defeating the authentication mechanism to masquerade as another valid entity. Though considerable research has been done in the past on various techniques to formally verify authentication protocols, it is generally agreed upon that proving that an authentication protocol is secure is not easy or straightforward and that there is no substitute for good engineering practices in designing and deploying these protocols.

Approaching the problem from this direction, we develop an architecture designed to prevent generally known attacks on authentication protocols through a trusted third party monitoring [13]. Attention has been paid to keep the architecture simple enough so that it will not face the same problems that the protocols it tries to protect face, yet robust enough to resist most forms of attack. The use of trusted entities is not new in authentication; several authentication protocols that depend on trusted third parties have been proposed and used in the past (a familiar example is the Kerberos authentication system [10]). However, in almost all such protocols, the trusted third party is an integral

part of the authentication process. Any weakness of the trusted party can seriously damage the security of numerous entities depending on its service. Moreover, in large systems, the trusted entity can quickly become a performance bottleneck, as it needs to be directly involved in all sessions initiated among the entities in its authentication domain. Inter-domain authentication (between two entities belonging to two different domains) is also a major issue in such protocols, because of the potentially limited trust that entities in one domain may be willing to have on the trusted entity belonging to another domain.

2.2 Reliability

Computer systems that are used in high dependability and integrity applications need to be designed with the capability to detect and recover from the errors caused by hardware and software faults. Since the majority of errors are usually transient and not reproducible, off-line testing will not reliably detect them. Thus, it is imperative that the systems be designed with built-in concurrent error detection and recovery mechanisms. Behavior based error detection (BED) is touted as an inexpensive yet effective concurrent technique to support fault tolerance. Among the various approaches such as structural integrity checks, memory access checks, control-flow checks, it has been shown that the control-flow checking is most effective in its error coverage and cost.

Various control-flow checking techniques have been proposed in the past to detect processor faults. The techniques employ a watchdog processor to compute run-time signatures from the instructions and compare them with the pre-computed signatures. These techniques need either additional hardware or modification of the hardware and are invariably non-portable to various platforms. Complexity of the modern compilers is yet another source of additional control-flow faults, which cannot be handled by these methods, as they assume error free object output from the compilers. To circumvent these limitations, recently we have developed a high-level control-flow checking approach using assertions (ECCA) [13], in which branch-free intervals in a given high-level language program are identified and the entry and exit points of the intervals fortified through pre-inserted assertions. The ECCA approach is portable across architectures and requires no special hardware or database lookups to implement. It can be implemented through a pre-processor based on the syntactic structure of the language pre-processor based on the syntactic structure of the language and does not require generation and analysis of various paths in the program control-flow graph. ECCA will detect hardware or compiler induced control flow faults.

All these BED schemes had been used so far in uni-processor or tightly coupled multiprocessor environments. We observe that application of BED techniques at a higher granularity in a distributed environment system can significantly improve the error detection and correction capabilities such systems. In the project, we develop BED schemes coupled with application specific data value checking for ensuring the reliable execution of the protocols. The schemes can easily be implemented in conjunction with the security monitoring under the unified meta-frame work that we develop.

3. Approach

The key idea behind our approach is the real time monitoring of protocol behavior to ensure its secure and reliable progression. The monitoring is achieved through observing various parameter values and their temporal and spatial validity. It has been observed by the research community that security and reliability are two competing concerns and often, disjoint solutions are proposed to address these problems. In this proposal we introduce a framework, which enables non-intrusive monitoring with the capability to address both security and reliability. In what follows in this section, we describe the monitoring framework and the necessary protocols. We also present evaluation criteria for the assessment of the proposed techniques. In Section 5, we present detailed research and development plan for the project duration.

3.1 Meta-Authentication Framework

The term meta-authentication denotes an `encapsulating authentication' mechanism for general authentication protocols. This is achieved through a high level mechanism for validating the execution of underlying authentication protocols. Meta authentication operates in the context of a framework comprising an architecture and a high level validation protocol that together provide a distributed environment for monitoring and validating the execution of authentication protocols. Entities involved in an authentication session can ensure that the execution of the authentication protocol itself has been proper and devoid of any malicious tampering or manipulation. The framework does not make any assumptions on the underlying authentication protocol and is generic enough to support any protocol chosen by communicating entities. It is important to note that it is not a goal of meta-authentication to offer any assurance as to the correctness of beliefs established by the `encapsulated' protocol; this is still the responsibility of the encapsulated protocol. However, it does provide assurance that the encapsulated protocol itself runs correctly, protected from extraneous interference.

3.2 The Trust Model

Meta authentication is based on the concept of trusted third parties. All communicating entities in a domain of trust utilize the services of the trusted entity. The use of trusted entities is not new in authentication; several authentication protocols that depend on trusted third parties have been proposed and used in the past (a familiar example is the Kerberos authentication system [10]). However, in almost all such protocols, the trusted third party is an integral part of the authentication process. Any weakness of the trusted party can seriously damage the security of numerous entities depending on its service. Moreover, in large systems, the trusted entity can quickly become a performance bottleneck, as it needs to be directly involved in all sessions initiated among the entities in its authentication domain. Inter-domain authentication (between two entities belonging to two different domains) is also a major issue in such protocols, because of the potentially limited trust that entities in one domain may be willing to have on the trusted entity belonging to another domain.

In the scheme proposed here, the role of trusted third parties is limited to only helping to monitor the execution of authentication sessions. They are neither directly involved in the

execution of the protocol, nor do they play any role in establishing beliefs between communicating parties. Their service is needed only if the communicating entities wish to protect their authentication process through meta-authentication. Even without meta-authentication, they will still be able to operate any authentication protocol normally. The optional use of meta-authentication is designed to protect against security attacks, while incurring minimal overhead.

In meta-authentication, the entire user space is divided into trust domains. Each domain includes any number of ordinary communicating entities and a trusted meta-authentication server (henceforth called the meta server). A meta server establishes trust with every member in its domain, so as to function as an intermediary in intra-domain authentication. An entity need not trust any member even in its own domain, other than its meta server. Moreover, entities in one domain need not trust the meta server in another domain. However, meta-servers in different domains may establish and maintain trust on one another, so that their services can be extended to inter-domain authentication as well. As there is only one meta server in each domain, this trust is far easier to establish and manage as compared to maintaining trust among all entities in all domains, or between entities in one domain and meta servers in other domains, or even between all entities in the same domain. Essentially, meta authentication follows a hierarchical and transitive trust model. This means that, if there is trust between entity A and its meta server SA, between entity B (possibly in another domain) and its meta server SB, and between meta servers SA and SB, then eventually trust may be established between A and B.

3.3. An Architecture for Meta-Authentication

The meta authentication scheme uses public key cryptography to protect the integrity of sessions. During an authentication session, communicating members and the concerned meta servers exchange monitoring messages signed with their private keys. These signed messages, verifiable only with the respective public keys, deliver validating data to help the communicating members ascertain the integrity of the `encapsulated' run of protocol. It may however be noted that meta authentication exchanges are not confidential, i.e., exchanged messages are observable by anyone. This is because meta authentication relies only on the integrity of validation messages, not on their confidentiality. When two members of the same domain authenticate, this message exchange involves those two members and the domain's meta server. When the authentication is between two members belonging to different domains, the exchange involves the two members and the meta servers of both domains. As compared to other public key based systems, the meta authentication scheme imposes only minimal requirements, which are as follows.

1. Each member and meta server in any domain has a public/private key pair.
2. The private key of every member and meta server is kept strictly confidential, known only to the holder of the key (this is a requirement in all public key cryptographic systems).
3. All members in a domain know the public key of the meta server of the same domain.
4. A meta server knows the public keys of all members in its domain.
5. A meta server either knows the public keys of the meta servers in all other

domains, or has access to a mechanism through which such keys can be obtained (for example, based on certificates issued by a higher trusted entity).

The scheme, however, does not impose any restrictions on the members. A member need not know the public key of any other member in its own, or another, domain. Further, a member need not know the public keys of any meta server in other domains. Similarly, a meta server does not need to know the public key of any member of other domains.

Thus, the scheme allows each domain to be separately administered. The only inter-domain knowledge is limited to meta servers who need to know the public keys of one another. This not only reduces the overhead but also reduces the chance of compromised key pairs being used. The basic schemes as would be used in intra-domain and inter-domain meta authentication are shown in Figure 1(a) and Figure 1(b), respectively.

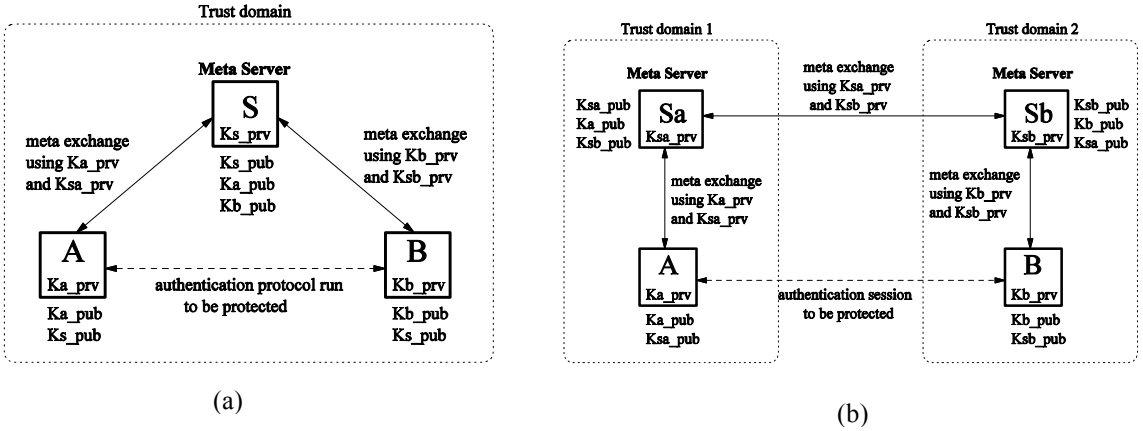


Figure 1. Inter and Intra-domain Meta Authentication

In the figure, the public and private keys of any entity X are represented as K_{x_pub} and K_{x_prv}, respectively. The authentication protocol employed by the communicating members A and B is independent of the meta authentication scheme. Members may use any authentication protocol they choose. It is also irrelevant whether the encapsulated protocols are based on public key or shared key cryptography.

3.4. Protocol for Meta-Authentication

During the run of an authentication protocol, several pieces of information are exchanged between the two communicating entities such as identities, one-time random values (nonces), time-stamps, and shared keys. If the two entities calculate an integrity check value (ICV) over these values using a pre-defined and mutually agreed upon function, the results would be identical at the two ends. The function must be irreversible, so that it is infeasible to generate a set of protocol messages that would result in that ICV and it should be collision resistant, meaning, different runs of the protocol must not result in the same ICV. It is also possible to calculate the ICV using weighted functions resulting in different values at the two ends, which prevents replay type attacks on the meta

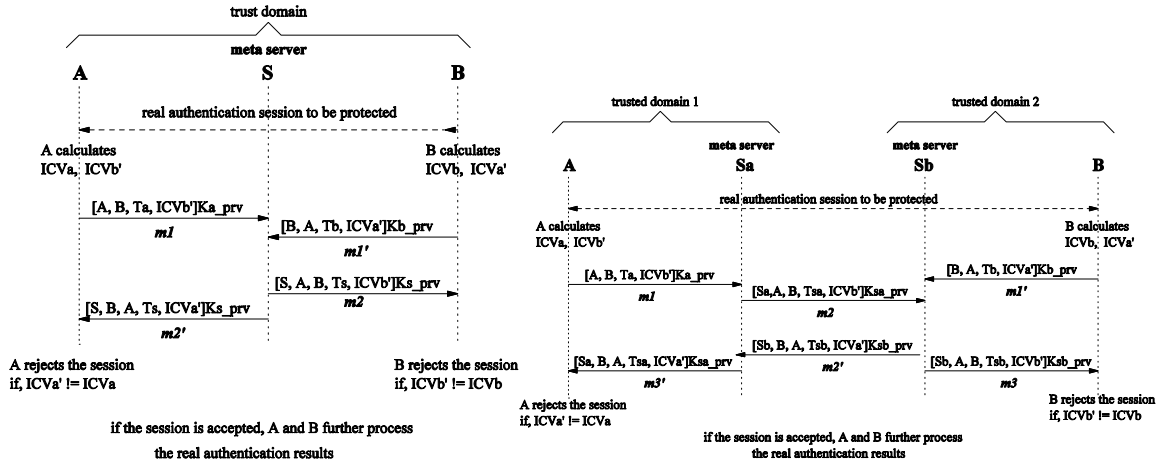
authentication itself. By virtue of encryption mechanisms used in the authentication protocol itself, many of these values may be seen only by the communicating parties and not by any third party. Thus it will be infeasible for an external entity to deduce the ICV for a fresh instance of the protocol run. This indicates that in the presence of an attack, one or more pieces of information known (sent or received) to one entity in the session may not be the same as those seen by the entity at the other end. However, there may be cases where the external intruding entity may be able to see all the pieces of information seen by legitimate participants. In such cases, simply demonstrating an ICV is not sufficient because the intruder also will be able to generate this value. This is the reason why signed messages are used in the meta authentication framework to deliver validation data. The strong cryptography assumption ensures that an intruder will not be able to forge the signature of another entity.

An authentication session protected through Meta authentication proceeds through the following steps:

1. Before initiating the authentication session, the authenticating entities agree whether to use the optional meta authentication services. If the decide not to do so, then no further protection is available and the entities proceed to step 3.
2. If meta authentication is to be used, the entities agree upon a predefined algorithm for weighted integrity check value calculation.
3. The entities run the real authentication protocol. If meta authentication is not used, the entities proceed to step 14.
4. As the authentication protocol proceeds, each entity calculates a local ICV and `deduces' the ICV that is supposedly being calculated by the other.
5. When the authentication protocol run concludes, each entity sends the 'deduced' ICV to the Meta server of its domain in a signed (signed with the entity's private key) message. This message includes the identity of the entity, the identity of the other entity (the other participant in the authentication session) to whom the information is to be delivered, and a time-stamp.
6. Each meta server verifies that the message came from an entity in its own domain, and checks the validity of the signed message received using the public key of the originating entity (this key is known to the meta sever, as the entity is a member of its domain). It also checks the time-stamp to ensure that the message is timely and not a replay.
7. Each meta server then extracts the information and determines whether the recipient entity is a member of its own domain. If this is the case (intra-domain authentication), it sends the information to the entity in a signed message (signed with the server's private key) also including the identity of this meta server and its own time-stamp. Operation then proceeds to step 9. However, if the recipient entity is not a member of its own domain (inter-domain authentication), then the meta server sends this newly formed message to the meta server of the recipient entity, and operation continues in step 8.
8. The receiving meta server verifies the received message for integrity (using the sending meta server's public key) and timeliness. It also verifies that the recipient is a member of its domain. On verification, the server extracts the relevant

information (identities of originating and receiving entities, and the `deduced' ICV value) and sends it to the recipient. The message is signed with this meta server's private key and also includes the meta server's identity and a new time-stamp.

9. On receiving this validation message, each entity verifies that it was delivered by the meta server of its own domain using the public key of the server, known to all members in the domain), and that it is timely.
10. From the validation message delivered in step 9, each entity extracts the identity of the originating entity and verifies that there is indeed an authentication session proceeding between the two. It then compares the `deduced' ICV value received in the message with the value it had computed locally, and verifes that they are identical. If both these checks are successful, then the entity proceeds to step 13.
11. If either of the checks in step 10 fails, it indicates a possible intrusion attempt and the entity immediately aborts the authentication session. After a timeout, the entity at the other end will notice the absence of response and will also terminate the session. The remaining steps are skipped in this case.
12. If an entity fails to receive the above validation message within a reasonable amount of time after the conclusion of the authentication protocol run, it assumes some foul play and aborts the authentication session. The other entity will also have to terminate the session after a timeout. The remaining steps are skipped in this case.
13. The successful checks in step 10 assure both authenticating entities that the execution of the authentication protocol itself has been proper and secure.
14. Based on the outcome of the real authentication protocol, the entities determine whether to accept the authenticity of the other entity or abort the session. The procedure described above in steps 1 through 14 constitute the meta authentication protocol



(a) (b)

Figure 2 Intra- and Inter Domain Meta-Authentication Protocol

In Figure 2 (a), the pair, m1; m2 show the path of validation message delivery from A to B, and $m1^0$, $m2^0$ show the path from B to A. Note that there is no timing relationship between the two sequences, timing is applicable only within the same sequence. Similarly in Figure 2(b), m1; m2; m3 and $m1^0$; $m2^0$; $m3^0$ show the delivery path.

3.3. Security of Meta-Authentication

Consider a high level view of the transfer of validating information in the Meta authentication model. Take *x* and *y* as the integrity check values generated locally by nodes A and B. Also *y'* is the value deduced by A and *x'* is the value deduced by B. Now the entity A accepts authentication protocol run if and only if *x'* = *x*, and entity B accepts the authentication protocol run if and only if *y'* = *y*. For given *x* or *y*, it is infeasible to generate a set of protocol messages that result in the same ICV values, because of the properties of *collision resistance* and *irreversibility*, processed by the ICV generating function.

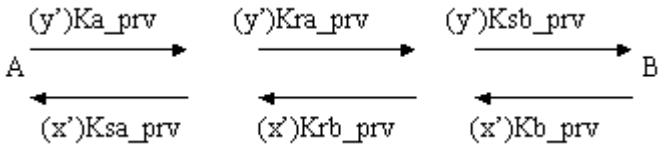


Figure 3. Trusted Paths in Meta-Authentication

Under the given assumptions, a malicious entity cannot *forge* any of the signed messages, because the required private key will not be known to it. This follows from the fact the strong cryptography assumption. An intruder cannot *replay* any of the messages in an authentication session, because *x* and *y* are time dependent since their calculation uses time-stamps as one of the inputs. It cannot substitute a message in an authentication session with a similar message from a different run of the protocol, because it is infeasible to manipulate a protocol run such that it results in a known ICV (*x* or *y*). Therefore, if a message is delivered in an authentication session, the receiver is assured of the *data origin authenticity* and *data integrity*. However, a malicious entity capable of observing and manipulating protocol sessions may be able to intercept and possibly delete a message in an authentication session. However, the receiver will detect the message loss through a timeout and abort the session. From these observations, it is clear that an encapsulated (monitored) authentication protocol is secure (in terms of integrity) if the Meta channel between the authenticating entities is safe. Thus, the Meta authentication scheme provides a robust mechanism to ensure the integrity of authentication protocols.

3.4 Reliability

We address the reliability issue at two levels. First, we want to enhance reliability through the real-time monitoring of the system behavior, through the services available through the meta-channel. Then we want to ensure that the meta-channel itself will be

protected against various failures.

For the former, we develop system level checks to detect control-flow and data value errors, caused by transient as well as permanent faults. The progress of computation as well as protocols will be reported to meta-servers for conformity checks. This is the first proposal to perform such monitoring over the Internet. The expanse of the network requires that the control-flow parameters be chosen at a higher granularity when compared to the monitoring of a uni-processor system. We do assume that the meta-channels are reliable and secure for which we will provide error detection and correction mechanisms at the data level and communication protocol level. The channels will be secured for confidentiality, authenticity, and/or integrity as the application demands.

In order to ensure the reliability of the meta-authentication system we rely upon classical fault tolerance techniques. As described in preceding sections, in its simple configuration, each domain will have only one meta-server. In order to avoid the problem a single point of failure we institute an automatic selection protocol in order to designate another met-server for the domain in case of failure of the current server. When extreme reliability and response times are critical the meta-framework could be extended to have multiple servers per domain where conformity decisions will be arrived at based on Byzantine agreement among the server nodes.

4. RESEARCH PLAN

The proposed research is planned to be finished through five parallel activities that span three years as follows.

Related Work and Impact (Year 1, Year 2, Year 3)

In this activity, an extensive review of related schemes will be conducted in Year 1. Existing solutions will be analyzed and categorized. A comparative study between the proposed framework and existing schemes will be conducted in Year 2. A study of the impact of the proposed work will be conducted in Year 3.

Framework (Year 1 & Year 2)

In this activity, a framework that enables non-intrusive monitoring with the capability to address both security and reliability will be designed. The architecture of the meta-framework will be laid out in Year 1. Enhancement to the framework will be conducted in Year 2.

Protocols (Year 1 & Year 2)

Protocols that address both security and reliability problems will be introduced during Year 1. Implementation and prototyping will be conducted in Year 2.

Evaluation (Year 2 & Year 3)

These activities include holding a number of simulation as well as real experiments to assess the proposed framework. During Year 2, an evaluation model will be developed and experiments will be designed. In Year 3, extensive simulation studies as well as experiments on real systems will be held.

Dissemination (Year 1, Year 2, Year 3)

Research results will be collected and documented as technical report(s) for publication. It is expected that a number of journal and conference articles will result from this research. In addition, we plan to distribute the protocol prototypes to a number of researchers in the community.

The research plan is summarized in Table 1.

Table 1. Research Plan

ACTIVITY	YEAR 1	YEAR 2	YEAR 3
RELATED WORK AND IMPACT	EXTENSIVE STUDIES OF RELATED SCHEMES	COMPARATIVE STUDIES	ANALYSIS OF IMPACT
FRAMEWORK	FRAMEWORK DEVELOPMENT	FRAMEWORK ENHANCEMENT	
PROTOCOLS	PROTOCOL DEVELOPMENT	PROTOCOL IMPLEMENTATION	
EVALUATION		EVALUATION MODELING	SIMULATION STUDIES & REAL SYSTEM EXPERIMENTS
DISSEMINATION	DISSEMINATION OF RESULTS, REPORT	DISSEMINATION OF RESULTS, REPORT	DISSEMINATION OF RESULTS, REPORT

5. Personnel

Prof. Suku Nair

Pertinent expertise: Network Security and Fault Tolerant Computing:

Suku Nair's current research interests are in network security, network restoration, and fault tolerant computing and is the director of High Assurance Computing and Networking Lab in the computer science and engineering department at SMU. His research in network security has been most recently supported by a grant from the Texas, Advanced Technology Program. He is an affiliate of ***eCenter,*** an independent center of excellence focusing on interactive networking. Suku Nair has been a consultant to major telecom companies such as Nortel, Alcatel, DSC Communications, and Wiltel.

Prof. Hesham El-Rewini

Pertinent expertise: Heterogeneous Parallel and Distributed Computing, Mobile Computing:

El-Rewini's current research interests are split between two main areas: *mobile computing* and *parallel and distributed processing*. He has published many papers in the above research areas. In 2001, he has been awarded $305,360 from DoD to work on the design and operation of mobile environments with hybrid backbones. He has also written several books including Introduction to Parallel Computing, Task Scheduling in Parallel and Distributed Systems, and Distributed and Parallel Computing.