3-20-96

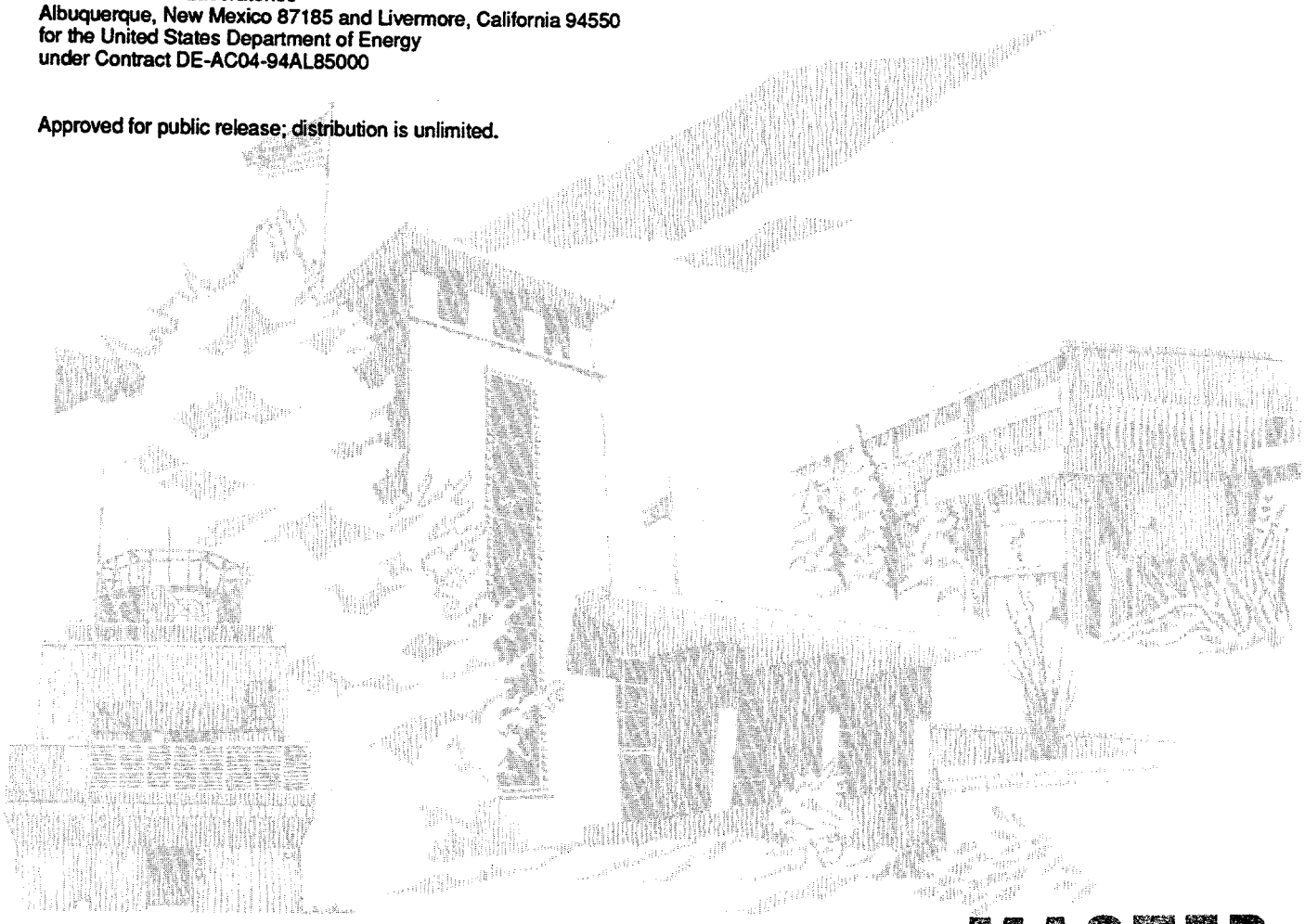
SANDIA REPORT

SAND96–0657 • UC–705
Unlimited Release
Printed March 1996

Final Report for the Protocol Extensions for ATM Security Laboratory Directed Research and Development Project

Thomas D. Tarman, Lyndon G. Pierson, Joseph P. Brenkosh,
Barbara J. Jennings, Edward L. Witzke, Marylou Brazee

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

SAND96-0657
Unlimited Release
Printed March, 1996

Distribution
Category UC-705

Final Report for the Protocol Extensions for ATM Security Laboratory Directed Research and Development Project

Thomas D. Tarman
Data Transport and Network Design Department

Lyndon G. Pierson and Joseph P. Brenkosh
Advanced Networking Integration Department

Barbara J. Jennings
Scientific Computing Systems Department

Sandia National Laboratories
Albuquerque, NM 87185

Edward L. Witzke and Marylou Brazee
RE/SPEC Inc.
Albuquerque, NM 87110

Abstract

This is the summary report for the *Protocol Extensions for Asynchronous Transfer Mode* project, funded under Sandia's Laboratory Directed Research and Development program. During this one-year effort, techniques were examined for integrating security enhancements within standard ATM protocols, and mechanisms were developed to validate these techniques and to provide a basic set of ATM security assurances. Based on our experience during this project, recommendations were presented to the ATM Forum (a world-wide consortium of ATM product developers, service providers, and users) to assist with the development of security-related enhancements to their ATM specifications. As a result of this project, Sandia has taken a leading role in the formation of the ATM Forum's Security Working Group, and has gained valuable alliances and leading-edge experience with emerging ATM security technologies and protocols.

Acknowledgments

The authors wish to thank a number of individuals and teams for their help and support throughout this project, including:

Mike Moser of Ideas, Inc. and the rest of the VINCE development team for developing and supporting a freely-available ATM signaling environment and source code. This provided us with a version of the UNI 3.1 signaling protocol which could be modified to support our security extensions. The hard work of the VINCE team allowed us to use precious project resources for the task at hand, rather than the procurement of signaling libraries from a vendor. Also, the support of the VINCE people when we experienced problems was exceptional, especially considering their "bare-bones" budget and enormous task load.

Vicky Hamilton (Department 9415) for contributing her Digital Signature Standard source code. Its straightforward organization and user-oriented APIs allowed this code to be used immediately for our application.

Jeff Ingle, project lead for NSA's Milkbush ATM encryptor prototype project, for his encouragement and willingness to provide information.

Art Hale and Kevin McCurley (Center 9200) for providing an application which gave the project a more concrete focus. Special thanks to Kevin for providing his honest opinions and suggestions.

John Naegle (Department 4616) for his help in writing the project proposal and selection of project personnel.

Finally, we would also like to thank Mike Vahle (Department Manager, 4616) and Mike Sjulin (Department Manager, 9417) for their patience, guidance, and support. Special thanks to Mike Sjulin for managing the project and tracking our use of project resources.

Contents

Figures

Tables

1. Introduction

This is the summary report for the *Protocol Extensions for Asynchronous Transfer Mode* project, funded under Sandia's Laboratory Directed Research and Development program (case # 3517.190). This project was a one year project that proposed to develop ATM security protocols and mechanisms which fit gracefully into the current ATM specifications (the proposal can be found in the appendix, Section 14.1). Specific protocols that were developed include ATM-level authentication and encryption signaling messaging, and mechanisms include low cost, software-based encryption embedded in the ATM communications protocol stack. Although not indicated in the proposal, protocols for key exchange and public key distribution were also developed and/or examined.

One of the goals for this project was to transfer our experience to the ATM technical community in a manner that would have the most impact. For this reason, we targeted the ATM Forum with our recommendations for security enhancements to the existing ATM protocols. Since the ATM Forum is a world-wide consortium of ATM product developers, service providers, and users, its mission is to develop interoperability specifications that will be used by ATM suppliers when developing their products and services. The ATM Forum has been very successful in this regard, and its success is made evident by its large membership (currently over 600 members), and the fact that its specifications are, in fact, being used in virtually every ATM product in existence. By helping the ATM Forum in recognizing the need for integrated security protocols, it was our belief that ATM could see more practical use, particularly in widely dispersed networks with users that are not necessarily trusted (such as the proposed National Information Infrastructure).

To this end, we started by looking at the assets that are expected to be connected to large-scale ATM networks, the expected threats to those assets, and the protocols and mechanisms required to mitigate those threats. Next, we examined the current specifications developed by the ATM Forum to determine how those protocols and mechanisms can be gracefully inserted into the existing specifications. After this was determined, the protocols and mechanisms were designed and implemented in our testbed to validate them with respect to criteria such as correctness, ease of implementation, and performance overhead.

Throughout this process, we presented our ideas and results to the ATM Forum, and have interacted with several other researchers and developers in this field (including NSA, IBM, Network Systems Corporation, and MCNC). Although initially there was no formal consideration of security in the ATM Forum, our contributions, along with the contributions of other organizations (such as Xerox, IBM, and the Department of Defense) clearly motivated the need for such consideration. This work culminated in the formal establishment of a Security Working Group in the ATM Forum, and Sandia has assumed a leading role in the development of its security specification as its editor. As a result of this project, Sandia has established itself as a major contributor to this field of research and development.

The remainder of the report provides a detailed account of the work performed under this LDRD project. Sections 3 and 4 introduce the reader to the concepts of network security and ATM protocols. To help the reader understand why ATM security is important, Section 5 describes the need for security at the ATM layer, and why ATM and ATM-enabled applications presents unique security problems. Section 6 describes a generic ATM security architecture which serves as a reference configuration for the protocols and mechanisms developed under this project. In order to validate our work, an ATM security testbed was constructed at Sandia. This testbed is described in Section 7. Detailed discussion of the protocols and mechanisms developed under this project is provided in Section 8. The mechanisms that were designed, but not implemented (mostly due to constraints placed upon us by outside vendors) are describe in Section 9. Section 10 provides qualitative and quantitative assessments of the protocols and mechanisms that were developed under this project, along with some of the tools that were used. A more detailed discussion of the ATM Forum, and Sandia's contributions to it is provided in Section 11. Finally, concluding remarks are rendered in Section 12.

2. Acronyms

AAL ..... Atm Adaptation Layer — the ATM protocol layer responsible for a number of adaptation functions between higher layer protocols and ATM, including SAR
ACL ..... Access Control List
AIS ..... Alarm Indication Signal — a type of ATM OAM information
API ..... Applications Programmer's Interface
ARP ..... Address Resolution Protocol — the protocol that binds a higher layer address (such as IP) to a lower layer (typically hardware) address
ASCII ..... American Standard Code for Information Interchange
ATM ..... Asynchronous Transfer Mode — a new digital communications technology based on cell switching
B-ICI ..... Broadband InterCarrier Interface
CA ..... Certification Authority — an authority which binds an entity with its public key in the form of a "certificate"
CLP ..... Cell Loss Priority — a field in the ATM cell header
CPU ..... Central Processing Unit
CRC ..... Cyclic Redundancy Check — a parity check function used to determine data corruption
CRL ..... Certificate Revocation List — a list of public key certificates that have been revoked due to compromise, loss, etc.
DSA ..... Digital Signature Algorithm — a public-key algorithm used by the DSS
DSS ..... Digital Signature Standard — a standard endorsed by NIST which uses the DSA and SHA to digitally "sign" a digital document
ENR ..... Enterprise Network Roundtable — the user's committee of the ATM Forum
FIPS ..... Federal Information Processing Standard — these standards are produced by NIST
FTP ..... File Transfer Protocol
GFC ..... Generic Flow Control — a field in the ATM cell header
GUI ..... Graphical User Interface
HEC ..... Header Error Checksum — a field in the ATM cell header that allows the receiver (either an end node or a switch) to determine if header corruption has occurred
IDEA ..... International Data Encryption Algorithm
IE ..... Information Element — a "piece" of a UNI 3.x signaling message that conveys specific information (such as called party address)
IEEE ..... Institute of Electrical and Electronics Engineers
ILMI ..... Interim Local Management Interface — a protocol which specifies management functions at the UNI (e.g. address registration)
IP ..... Internetwork Protocol — a protocol that specifies peer-to-peer communications across logical subnetworks
ITU/CCITT ..... International Telecommunication Union/The International Telegraph and Telephone Consultative Committee — an international body for standards such as ATM and network security
LDRD ..... Laboratory Directed Research and Development — a Department of Energy program that allows its national laboratories to invest funds into new, "leading edge" research programs
LFSR ..... Linear Feedback Shift Register — a device that produces a LRS
LLC/SNAP ..... Logical Link Control/SubNetwork Access Protocol — a protocol that specifies peer-to-peer communications and packet formats on a logical subnetwork
LRS ..... Linear Recurring Sequence — a sequence of numbers produced by a linear equation which is pseudo-random (i.e. appears random, but is not)
MAC ..... Marketing and Awareness Committee — a committee of the ATM Forum
MAC ..... Media Access Control — the protocol responsible for orderly access to a transmission medium

MCNC..... Microelectronics Center of North Carolina — developers of a prototype OC-12 ATM encryptor
MD5..... Message Digest 5 — a one-way function to produce a "digest" of a digital document
MDC..... Manipulation Detection Code
NIC..... Network Interface Card
NIST..... National Institute for Standards and Technology
NRL..... Naval Research Laboratory — the research institution which developed VINCE
NSA..... National Security Agency — the developers of the Milkbush prototype ATM encryption hardware
NSAP..... Network Service Access Point — an addressing convention used by most ATM devices
OAM..... Operations, Administration, and Maintenance — used in this report to refer to special ATM cells that carry maintenance information such as cryptographic resynchronizetion information
OC3c..... Optical Carrier 3, Clear channel — a specification for optical communications over 155.52 Mbps, non-multiplexed links
OSI..... Open Systems Interconnection — an international suite of networking standards
PDU..... Protocol Data Unit — a packet of protocol information (control or data)
PGP..... Pretty Good Privacy — an encryption program
PNNI..... Private Network to Network Interface
PTI..... Payload Type Identifier — a field in the ATM cell header
PVC..... Permanent Virtual Circuit — an ATM virtual circuit that is configured statically
QOS..... Quality Of Service
RDI..... Remote Defect Indication — a type of ATM OAM information
RPC..... Remote Procedure Call — a protocol used by a client to request execution of a procedure by a server
RSA..... Rivest, Shamir, and Adelman — refers to a public-key cryptosystem developed by these researchers
SAR..... Segmentation And Reassembly — the process of breaking up higher layer PDUs into ATM cells, and vice-versa
SHA..... Secure Hash Algorithm — a one-way function used by the DSS to produce a "digest" of a digital document
SNMP..... Simple Network Management Protocol
SONET..... Synchronous Optical NETwork — a common transmission medium for ATM
SPANS..... Simple Protocol for Atm Network Signaling — a proprietary signaling protocol developed by Fore Systems
SPARC..... Scalable Processor ARCitecture — the processor/hardware architecture used by the latest Sun Microsystems workstations
SSCOP..... Service Specific Connection Oriented Protocol — a "lightweight" connection oriented protocol used by ATM for signaling
SVC..... Switched Virtual Circuit — an ATM virtual circuit that is configured automatically by the network, on-demand
TC..... Technical Committee — the ATM Forum committee responsible for ATM specifications
TCP..... Transmission Control Protocol — a protocol that provides assured, in-order delivery of PDUs over IP.
TTCP..... Test TCP — a memory to memory throughput testing tool which uses the TCP protocol
UNI..... User to Network Interface
UNI 3.x..... Specifications developed by the ATM Forum which specify ATM protocols, including SVC signaling and address registration protocols, at the UNI
VCI..... Virtual Circuit Identifier — a field in the ATM cell header that provides switching information
VINCE..... Vendor Independent Network Control Entity — a freely-available ATM signaling package developed by the NRL
VPI..... Virtual Path Identifier — a field in the ATM cell header that allows virtual circuit aggregation

3. Network Security in General

3.1. Determining Security Policy and Protection Mechanisms

The first steps in securing a network are to identify the *resources to be protected* and to develop a policy that states the level of protection to be applied to each resource. Protected resources may include:

- Computing and networking equipment
- Customer premises communications equipment
- Network provider communications switching equipment
- Buildings, utilities, or other facilities required for computer and communication network operations
- Computer programs and data
- Running computer processes -- currently executing or suspended sequences of computer instructions
- Computer and communications artifacts that a running computer process may require: available memory, data structures, allocable devices, CPU (central processing unit) cycles, communication network bandwidth, etc.

Each resource or related set of resources to be protected must be well-defined before one can compile a statement of threat. Once these resources have been identified and their desired protection levels established, one can examine threats to them and develop protection measures.

The *definition of protected resources* is a statement that includes a list of the location and nature of each resource, replacement cost of associated hardware or software, cost of displaced service or utility, and cost associated with compromise of data. Any costs that are intangible or cannot be estimated should be described in narrative form.

The *security policy* is a simple statement of management intent about protecting computing or communication resources. This statement identifies to what extent important classes of resources are to be protected from a broad class of perpetrators. A security policy should briefly describe the appropriate use of computing and communication resources. It should also describe the intent of protections against compromise of need-to-know, privacy, corporate security, national security, or other applicable losses perpetrated by insiders or outsiders or caused by natural hazards. (Insiders are personnel authorized to enter physical security boundaries, while outsiders are those who are not.)

A security policy need not be formalized with a mathematical model of computer security, although formal computer security models may aid in the development of protection measures for certain resources. Computer or communication resources and threats vary widely, therefore it is unlikely that a single formal model and associated formal security policy can be used to plan adequately for all aspects of computer and communications security or for all computer networks.

A *threat* is an event (e.g. earthquake, fire) or method (e.g. IP spoofing) that can potentially cause the theft, destruction, corruption, or denial of either service, information, resources, or materials. To characterize a threat, one uses the attributes, resources, and actions required to cause the event or carry out the method. It must be emphasized that a threat is a what or how, not a who. Perpetrators are the "who" elements and may be characterized by various motivations, levels of funding, and weapons or equipment. Different kinds of perpetrators may use the same method, and one perpetrator may use many different methods to attack a network. They may employ methods from inside or outside a facility. Independent of perpetrators, categorizing threat events or methods is important because protection mechanisms are based on the various events or methods that may be used in an attack.

The *statement of threat* should define the types of threat events or methods that pertain to the network being examined. The definition of protected resources and the security policy are the components that drive the statement of threat. Protection measures and incident detection mechanisms are developed specifically, and only, for those threats defined in the statement of threat.

To develop the statement of threat, the authors use binary decomposition. We begin with the universal set of all hazards, and typically divide it into natural and non-natural (man-made) events or methods. Eventually, each subtree's decomposition ends with a general category such as other events or methods. An example threat decomposition which illustrates this concept is shown in Figure 1. The granularity of this statement of threat also governs the granularity of the documented protection mechanisms, incident detection mechanisms, and other security methodology elements.

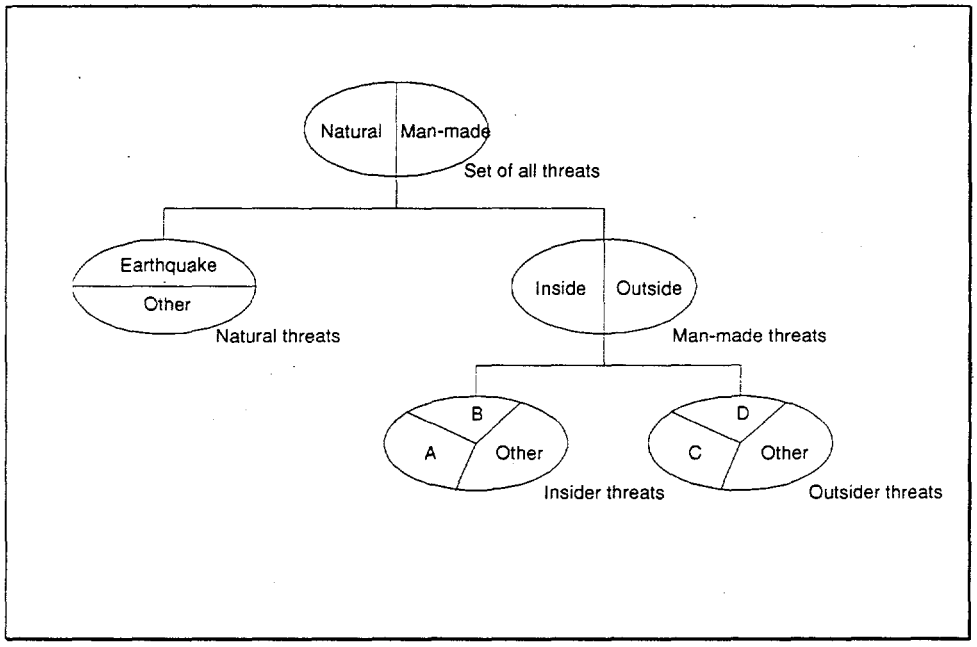


Figure 1: Example Threat Decomposition

The threat list that results should not contain overlap, except for the general other category. All threats map into a specific item identified in the statement of threat or into a nonspecific item identified as other. After more experience is gained with the computing network to be secured, planners can add more detail to items in the statement of threat and improve protection measures and incident detection mechanisms.

Protection measures are countermeasures that defeat or help defeat one or more threats. For each item in the statement of threat, one or more measures should be developed to protect the identified resource. A given countermeasure may partially negate several threats; but to negate a threat fully, more than one countermeasure may be needed.

An *incident* is an event that is judged unusual enough to warrant further investigation to determine if a threat event occurred or a threat method was attempted, and if a loss occurred or might yet occur. Incident detection mechanisms should include logging and reporting an event, and should be triggered by any attempt to circumvent the various protection measures. Another goal of incident detection mechanisms is to identify perpetrators so that they can be apprehended while a network security incident is in progress, and later prosecuted.

Further treatment of network security methodologies can be found in references [8] and [19].

3.2. Existing Protection Mechanisms

Many protection mechanisms and countermeasures that are applicable to computer systems are equally applicable to networks (groups of computers, communication facilities, and transmission media). A few of those are summarized here.

3.2.1. Physical protections

Hardening of facilities

Various techniques are available for the hardening of computing and communications facilities. These include earthquake-resistant construction practices, electromagnetic shielding and grounding, and filters for power and communication lines.

Physical access control devices

Computing and communication facilities require access control mechanisms. The variety of locking devices includes standard key, magnetic key, spin combination, push button (cipher), and twisting tumbler locks. Badging systems can range from photo badges (which must be visually inspected) to magnetic stripe and smart cards, along with their associated readers. Biometric systems have also been developed based upon fingerprint, palmprint, voiceprint, eye blood vessel, facial blood vessel, hand geometry, and written signature analysis.

Physical condition alarms

Physical intrusion alarms can be used to secure unattended or remote facilities. Alarms to detect and signal out-of-range conditions, such as heat or humidity, are also useful to protect unattended or remote facilities. To be effective, these alarms must report to a manned facility, not just sound locally.

3.2.2. Human issues

Personnel security

Personnel security measures are even more important in computer networks than with stand-alone systems. Not only computer nodes, but also routers, hubs, switches, and other communications equipment need protection from attacks based upon bribery, collusion, and "social engineering" (taking advantage of human weaknesses). Personnel security measures that can be employed here include: background investigations, security awareness training, separation of duties, and 2-person rules.

Configuration control

Configuration control is necessary to prevent unauthorized additions to, or deletions from a network. This applies to both hardware and software. Various pieces of software exist to aid in configuration tracking and control.

3.2.3. Operating System level protections

Software intrusion alarms

Software intrusion alarms can help detect attempts to penetrate individual systems on a network. Most software intrusion alarms are built into the operating systems and use surveillance, logging, and threat monitoring techniques. Some packages, such as Tripwire, will monitor the file system for unusual changes [10].

Audit/activity trails

Audits and audit/activity trails can be used to detect unauthorized production and system-oriented transactions. To be useful, audit trails and activity logs must be read and analyzed by a person or an automated process.

Security kernels

Security kernels are access control mechanisms, built into the operating systems, to check each request to ensure compliance with the systems' security policies. Security kernels are trusted code that has been verified to correctly enforce the security policies.

3.2.4. Other security measures

Error detection and control codes

Error detection/correction/control codes can improve the performance of a network in the presence of man-made or natural noise of either the burst or random variety. The two main types of coding techniques are block codes (such as simple parity checks or Hamming Codes) and convolutional codes.

Authenticated sessions

Authentication allows one or both parties involved in a communications session to be assured of the other party's identity. Depending on the authentication protocol, one party can authenticate the other, or both parties can authenticate each other. These protocols typically use a digital signature to cryptographically bind a message to its source, and use sequence numbers, time stamps, and/or "nonces" to protect against message replay attacks. Any of a number of algorithms can be used to generate the digital signature, such as DSS [14] and RSA [21].

Several additional protection mechanisms are of such significance or are sufficiently recent to merit more thorough discussion. These topics follow.

3.2.5. Communications Security Measures — Encryption

Methods of Encryption

Link encryption

Link encryption (or data link encryption) encrypts and decrypts a message at each node it goes through between the source and the destination. It requires separate keys for each physical link between network nodes (i.e. protocol processor to protocol processor). The advantages of link encryption are speed, since link encryption is performed at the hardware level, and that all information, including packet headers, is encrypted. Since all information is encrypted, messages are protected from traffic analysis, and from attempts to modulate traffic to form a covert channel. The main disadvantage of link encryption is that messages must be decrypted and re-encrypted at each routing node. The messages are processed (routing, flow control, error control) in unencrypted form. This allows traffic analysis or interception of messages at intermediate routing nodes. It can also degrade performance (in terms of delay) in large networks when a message must be routed through many intermediate nodes. Key management can be complicated, since each link at a (routing) node will most likely have distinct keys. A link *may* even have two keys; one for inbound traffic, and one for outbound traffic.

End-to-End encryption

End-to-end encryption encrypts and decrypts a message at the source and destination only. End-to-end encryption requires separate keys for each communicating peer session. The primary advantages of end-to-end encryption are that the data portion of the messages are protected in the intermediate routing nodes of a network, and that the users tend to have more control over the selection of cryptographic algorithms and keys. The main disadvantage of end-to-end encryption is that several potentially covert channels remain unprotected [15]. Also, most end-to-end encryption today is performed in software, which imposes a performance penalty.

End-to-End Encryption vs. Link Encryption

Link encryptors typically encrypt each and every bit of a synchronous communication line at one end of a leased or private circuit, and decrypt each and every bit at the other end. End-to-end encryption can be thought of as occurring at a higher layer of communication protocol, and involves identifying and passing ("in the clear") the control information associated with each data packet, while encrypting the payloads of selected data packets. Since the control information is not encrypted, this allows the processing of such packets at intermediate equipment without decryption. Since the control information (including switching information) is not encrypted, end-to-end encryption does not protect against traffic analysis of header information. Because many secure applications do not require protection from traffic analysis, the promise of less equipment and correspondingly smaller key management difficulty makes end-to-end encryption more attractive than link encryption. In ATM switched virtual circuits, encryption must take the form of end-to-end encryption unless the intermediate ATM switches are physically secured and link encryptor/decryptor pairs are to be installed on each segment of the path along the switched virtual circuit.

The encryption and decryption processes are intended to be computationally infeasible unless one holds the cryptographic "key". However, this usually causes the computations to become intensive even on behalf of the authorized "key" holder. For authorized key holders, the encryption and decryption processes are usually simpler (and faster) for link encryption than for end-to-end encryption. This is due to the fact that link encryptors have fewer decisions to make, and need not identify the beginning, end, or control information of each data packet.

Conversely, end-to-end encryption is computationally more intensive, particularly in terms of communications processing. This tends to cause the commercial availability of high speed link encryption and end-to-end encryption equipment to lag the availability of high speed communication equipment and service offerings. High speed communication equipment and service offerings become available as higher speed switching components (faster transistors) become available. In order to keep up with data rates available through high speed communication equipment, encryptors typically require a great many of the more expensive, faster switching components. Since the consumer market for encryption has been relatively small, faster encryptors typically become available only when faster transistors become "affordable".

This research applied parallel processing techniques developed in a related LDRD to scale the speed of encryption processing without requiring the availability of higher speed components. This approach still requires higher speed components to perform parallel-to-serial and serial-to-parallel conversions for transmission, but will require fewer higher speed components throughout the encryption processing function.

Algorithms

DES

The Data Encryption Standard (DES) is based upon a *private-key* (symmetric, or classical) block cipher of the substitution-permutation variety. It uses 56 key bits, operates on 64 bit blocks of plaintext, and goes through 16 rounds of substitutions and permutations. There are four modes of operation defined (in

Federal Information Processing Standards (FIPS)) for DES [13], others may be possible. A full description of the operational details of DES can be found elsewhere [12] and [22].

RSA

The algorithm commonly known as RSA, is a *public-key* (asymmetric) cryptography algorithm named after the three men (Ronald Rivest, Adi Shamir, Leonard Adleman) who first introduced it. This algorithm can be used for digital signatures (authenticity), as well as encryption (secrecy). In general terms, when using RSA, one chooses a pair of large prime numbers, p and q, and computes their product, n. One must also choose a key, d. The second key, e, is computed using a variant of Euclid's algorithm for computing the greatest common divisor, such that $e \cdot d \bmod (p-1)(q-1) = 1$. This way, e and d are multiplicative inverses $(\bmod (p-1)(q-1))$ of each other. One (and only one) of the keys (e or d) and the modulus, n are made public. The remaining key is the private key and should be protected as such. The strength of RSA public-key encryption resides in the difficulty of factoring large numbers (the modulus, n). If someone is able to factor n into p and q, he could then recover the keys, d and e, in the same manor as the original calculations. Implementation details for this cryptosystem can be found in [21] and [22].

The RSA algorithm encrypts a block of information (such as a number or series of numbers representing characters of text) by raising it to the other confidant's public key power, mod n. When the other confidant receives the message, he (and only he) can decrypt it by taking the message and raising it to his private key power, mod n. For digital signatures, the information is raised to the originator's private key power, mod n. (Since he is the only one holding this key, he must be the one "signing" this information.) When someone else wishes to verify that the originator was the one signing this information, he can raise the message to the originator's public key power, mod n, and recover the information. This property is especially useful when used in conjunction with a *manipulation detection code* (MDC) that is computed over the message being signed. Only the MDC needs to be raised to the originator's private power, mod n, for a digital signature. To verify the digital signature, compute a manipulation detection code over the message in question, and compare it with the sent (encrypted) MDC raised to the originator's public power, mod n. They should agree. If not, either someone has tampered with the message, or it did not originate with the claimed person.

Vernam

The Vernam cipher dates back to the days of Baudot code. It is a symmetric algorithm, essentially a polyalphabetic substitution cipher, and can be implemented as either a block or stream cipher. The original Vernam cipher was a 32 x 32 entry table with the 32 characters of the 5-bit Baudot alphabet across the top as plaintext and down the side as key indices. The entry at each row-column intersection is the logical "Exclusive-Or" of the binary representation of the plaintext character (at the top of the column) with the binary representation of the key index character (at the start of the row). This character can then be substituted for the appropriate plaintext encrypted by the key letter of that row. When 8-bit ASCII characters are used, this substitution table is enlarged to 256 x 256. With modern computing systems or hardware components performing the "Exclusive-Or" in real-time, substitution tables or matrices are eliminated. If the length of the key is shorter than the amount of plaintext, this system reduces to an automated form of the Vigenere cipher. If the key is truly random (not pseudo-random) and is at least as long as the plaintext it enciphers, and the key is only used to encipher one message, the system becomes a one-time pad, which is provably unbreakable.

PGP

Pretty Good Privacy (PGP) is a public-domain, privacy enhancement program created by Philip Zimmermann. It uses Zip (a popular DOS-based file compression and distribution utility) to perform message compression/decompression. It uses MD5 to create manipulation detection codes for digital

signatures. It uses RSA public-key cryptography to encrypt/verify the manipulation detection codes and to encrypt/decrypt session encryption keys. For the bulk encryption/decryption, PGP uses IDEA with a one time session key, generated by the sender. (IDEA is a symmetric, block cipher that operates on 64-bit blocks of plaintext. It uses keys that are 128 bits in length.)

PGP uses distributed key management. There are no key certification authorities. Users generate their own public key, have various *introducers* sign the public key for them, and distribute their own public key. When users receive a new public key, they examine the list of introducers who have signed the key. If they recognize one of the introducers as someone they trust, and the introducers signature is verified, then this new public key can be added to their key ring. Implementation details for PGP, IDEA, and MD5 can be found in [22] and [26].

3.2.6. Key Exchange

Diffie-Hellman Key Exchange

Diffie-Hellman key distribution is a method of distributing encryption keys -- particularly session encryption keys -- using a public-key cryptosystem developed by Whitfield Diffie and Martin Hellman. (It is also known as *exponential key distribution.*) The algorithm developed by Diffie and Hellman makes use of the difficulty of computing logarithms over finite fields with a prime number (q) of elements.

In this algorithm, each user generates a random number X between 1 and q-1. Each user keeps X secret but publishes $Y = (a^X \bmod q)$ where a and q are parameters of the cryptosystem, along with their name and address. When users i and j wish to communicate privately, they use a key $K = a^{X_i X_j} \bmod q$. User i computes this as $K = Y_j^{X_i} \bmod q$, while user j computes the same key as $K = Y_i^{X_j} \bmod q$. This key can now be used with a symmetric encryption algorithm to exchange private information or a session key for another symmetric encryption algorithm. More details can be found in [5].

Key Exchange via DSS

This is a scaleable key management method for large scale end-to-end encryption systems. It includes flexibility in the choice of public-key algorithms for exchanging session key information, dynamic generation and exchange of public-key cryptographic variables, flexibility in the selection of a symmetric, session encryption method, and dynamic generation and exchange of session key material. This method particularly lends itself to using a Diffie-Hellman type key exchange with Digital Signature Standard (DSS) keys. Public DSS keys have the form, $g^X \bmod p$, where X is the DSS private component, and g and p are system parameters. Note the similarities to the Diffie-Hellman algorithm described above. Users exchange session keys and desired encryption methods using an algorithm similar to Diffie-Hellman to ensure secure and authentic transfer of data traffic. The session keys can be exchanged in the data channel itself, or through an "out-of-band" communications channel. A detailed example of this method can be found in [30].

3.3. X.509 Key Distribution

In the operation of symmetric cryptosystems, a "key" is used to encipher information by the originator of a secret message. That same key must be used by the recipient of the encrypted message to decipher it. This type of technology requires that two individuals share the same key and infers that at some point the key must be transferred across a network. Keys that are shared and/or transported across a network have an increased risk of being compromised.

X.509 is a technology that can be used to support public key cryptography. It is a set of ITU/CCITT (International Telecommunication Union/ The International Telegraph and Telephone Consultative

Committee) Volumn VIII - FASCICLE Viii.8 recommended standards which define the framework for providing two sets of keys for authentication. One is a private key, that is known only to the individual that it is assigned to. The other is a public key that is freely published to the world. The premise of this technology is that the sender, enciphers a document with the recipient's public key. It can only be opened by the addressee when using the matching private key.

The two keys are created at the same time by a Certifying Authority (CA), and a certificate is created which binds the public key to the entity to which it belongs. The management of the keys and certificates includes: creation, revocation, presentation, and key authentication by verification against a Certificate Revocation List, CRL. When a certificate is created by the Certifying Authority, its contents are digitally signed with the CA's private key. This allows the certificate to be validated by a requesting user or node, using the CA's public key (this key should be widely available from independent sources, to protect against attacks to the system). The purpose of the CRL is to maintain a list of keys that were issued by that authority which have not yet expired but which have been revoked.

The public key is made available via a globally distributed directory, the X.500 Directory. An alternative to providing a globally accessible directory is to create a "key-ring" containing frequently used certificates. The key-ring is then provided to the individual users of the encryption application.

4. Overview of ATM Network Protocols

Although a thorough explanation of ATM protocols is beyond the scope of this report, it is necessary to provide a brief description before proceeding. The reader is encouraged to consult [2] [20] and [16] for more information on ATM protocols.

There are three types of flows that occur in an ATM network: control flows (such as signaling and OAM messages), management flows, and data flows. The basic unit of information used by all flows is the 53 byte ATM cell, as shown in Figure 2. The cell consists of a 5 byte header which contains switching and other ATM protocol information, and a 48 byte data section. These cells are then sent by encapsulating them into the transmission media's frame or packet (such as SONET). The cell is typically transmitted to an ATM switch which will examine the header and switch the cell to an outgoing port which is connected to the appropriate destination. This could be a computer or another ATM switch. The cell format and header fields is more thoroughly described in [2].

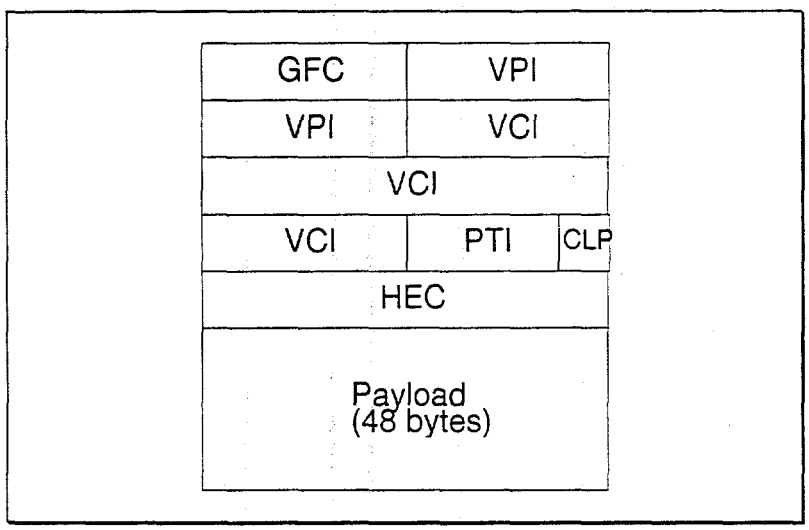


Figure 2: UNI Cell Format

There are two methods of establishing ATM data connections. These are Permanent Virtual Circuits (PVCs) and Switched Virtual Circuits (SVCs). Permanent Virtual Circuits are created manually by the user or network administrator, and provide a connection between two endpoints even if they are not actively being used to communicate with each other. Switched Virtual Circuits are created by the ATM signaling protocols, as needed, to connect to endpoints and the connection is automatically closed when no longer needed.

Since PVCs do not require signaling protocols to establish a data channel, they allow interoperability between ATM manufacturers. In fact, since the ATM signaling standards are still evolving, PVCs are often the **only** means to establish ATM data connections in a multivendor environment. However, PVCs are problematic from a network management perspective, particularly for large networks. This is true because, for each end system, the network manager must configure a PVC between it and all other nodes on the network to which it must talk. For a fully connected network of n nodes, the work required by the network administrator to manually configure this network is $O(n^2)$. Since this is unacceptable for large, fully connected ATM networks, the utility for PVCs is largely restricted applications that require a small number of long term, "nailed-down" circuits between two sites connected to an ATM service.

Since PVCs implement such "nailed-down" data circuits, they could also be used to implement a basic level of security. By disabling the ATM signaling "daemons" on nodes connected to a protected domain,

and by administratively establishing PVCs to these protected nodes, ATM layer access control can be implemented. However, the scalability concerns associated with PVCs apply here. To take advantage of the convenience offered by SVCs, and to provide realistic access controls, authentication within the SVC setup protocol is needed.

As stated earlier, for ATM networks with large numbers of users, SVCs are required. However, to establish a switched virtual circuit, a number of protocols may need to be used, depending on the types of interfaces that must be traversed. A reference configuration which shows these interfaces is illustrated in Figure 3.

This reference configuration shows two types of ATM networks: the Private ATM Network, which is typically managed by the end user's organization, and the Public ATM Network, which is managed by one or more ATM carriers. End user nodes can connect to either ATM network via the User to Network Interface (UNI), which is further distinguished into the Private UNI and the Public UNI, according to the network to which the node attaches. The Private ATM network also makes use of the Public UNI to connect to the ATM service provider.

Two protocols are provided to allow intra-network signaling and call routing. The *Private Network to Network Interface* (PNNI) provides inter-switch signaling and routing within the customer's private network, and the *Broadband Inter-Carrier Interface* is used for signaling and routing between carrier networks. *Edge switches* (switches that reside "at the edge" of the network) perform necessary protocol conversions when routing signaling messages across protocol boundaries. For example, NSAP addresses (addresses which are globally unique and are not bound to physical locations) which are used by the PNNI protocols are converted to E.164 addresses (addresses that are geographically assigned) by the carrier when entering the public network.

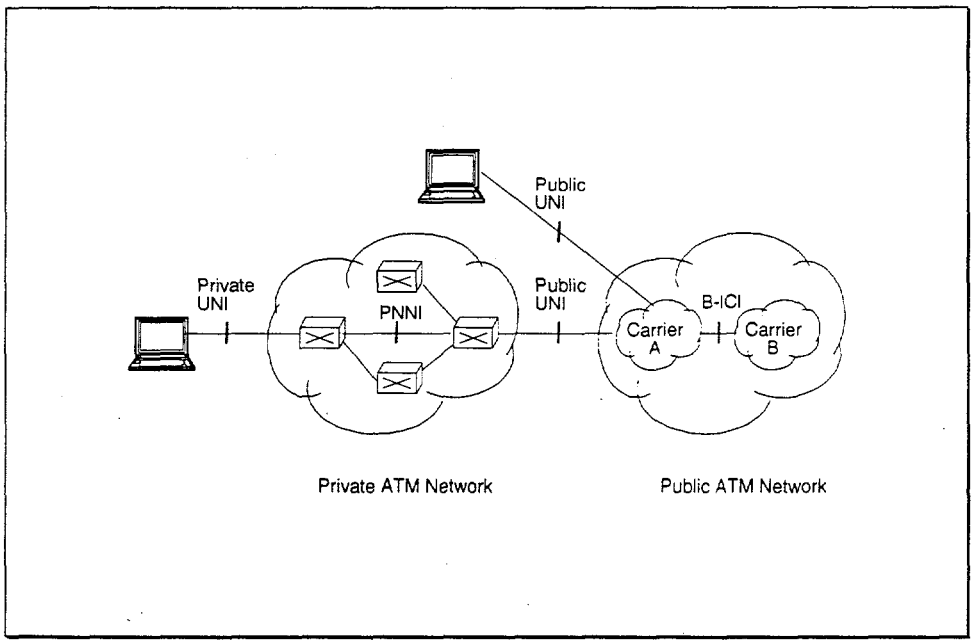


Figure 3: ATM Reference Model

A basic connection establishment protocol is illustrated in Figure 4. To establish an SVC on demand, the end user's workstation which requests the virtual circuit (the *calling party*) uses the UNI protocol and a pre-defined signaling channel (VPI=0, VCI=5) to signal its request to the other node (the *called party*). The switch, upon receiving this "setup" message from the calling party, will forward it directly to the workstation if it is directly attached (via the UNI protocol), or to the next-hop switch (using the PNNI

protocol). Upon receipt of a connection setup request, the called party will reply with a "connect" message which will propagate back to the calling party along the same path.

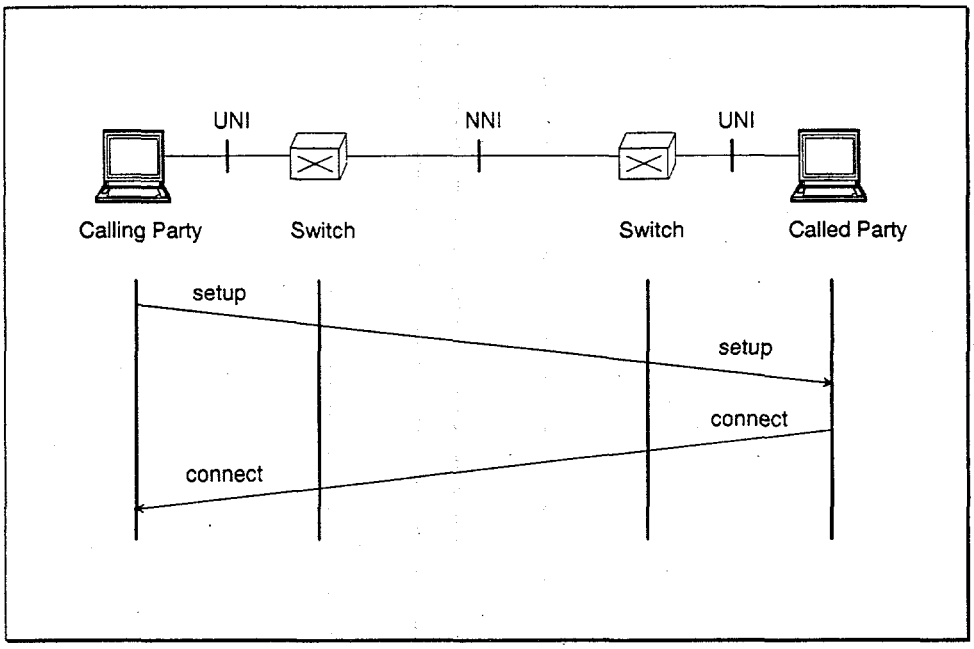


Figure 4: UNI 3.1 Connection Setup Protocol

The format of a UNI message is illustrated in Figure 5. The UNI message is simply a collection of *information elements* which are concatenated together into an AAL5 frame, segmented into cells, and transmitted across the UNI on the signaling channel. A number of information elements are defined in UNI 3.1, and depending on the message type, some information elements are required, whereas others are optional. For example, a "setup" message requires a "Called Party Number" information element, but the "Calling Party Number" is optional.

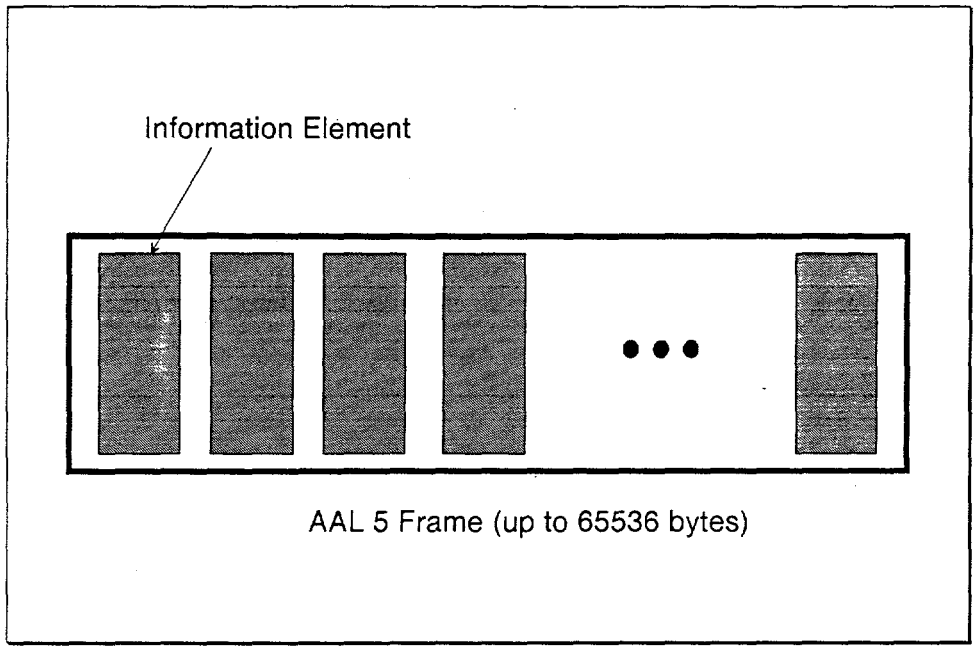


Figure 5: UNI Message Format

The work described in this report implemented SVCs according to the ATM Forum's UNI 3.1 specification. Several ATM equipment vendors such as FORE Systems provide proprietary methods for using SVCs as well. However, since one of the purposes of this LDRD was to recommend security extensions to the ATM Forum, it was decided that these extensions must be based on UNI 3.1, rather than a proprietary signaling protocol.

Finally, to properly implement SVCs in a private ATM network, a number of support services must be used to perform functions such as end system address assignment, and address resolution. These ancillary functions are described below.

In order to establish an ATM connection, each endpoint must have a unique address. UNI 3.1 uses an OSI Network Service Access Point (NSAP) address. The address has a 13 byte network-side prefix and a 7 byte user-side part. In order for a host to establish a connection to another machine, it must first know the NSAP address of that machine. The Interim Local Management Interface (ILMI) is used to discover and register those addresses automatically. When a switch boots up, it sends the 13 byte prefix to the connected hosts which fill in the 7 byte user-side part. The hosts will then notify the switch of their complete addresses.

If IP traffic is being sent, the IP address must be resolved into an ATM address, this is accomplished by ATM ARP (ATM Address Resolution Protocol). A host or switch must be a designated ARP server for an ATM LAN because current ATM standards do not support broadcasting. IP datagrams are then encapsulated using the IEEE 802.2 LLC/SNAP and are segmented into ATM cells using a ATM adaptation layer. RFC-1577 provides a standard for IP traffic over ATM.

5. Motivation for ATM Security

One of the fundamental promises of ATM is its ability to integrate voice, video, and computer data into a single architecture. One vision of the future for ATM has applications and devices connecting directly to ATM (so-called *native ATM applications*), thereby bypassing the typical higher layer protocols found today. However, network security today relies on these higher-layer protocols (such as IP) to implement access control, node identification, and lately, strong authentication [1]. To date, no standards exist which can provide more than trivial network security for native ATM applications. Rather, security for such applications must be provided by the application itself, most likely in a non-standard (possibly insecure!) fashion.

In addition to the concerns stated above, application layer security implies that the ATM switches (which operate well below the application) cannot participate in the security protocols. This capability is needed in two settings: in the network service provider setting, and in the high-performance communications setting. When switches participate in security protocols, the network can authenticate attached devices to ensure that only authorized entities can access and use the services, and to provide accurate (and non-repudiable) billing statements. In the high-performance environment where strong access control is required, an edge switch can be used to strongly authenticate "outside" devices when they wish to connect to an "inside" device, thereby implementing ATM layer access control. By implementing access control in a switch, which is necessarily a high speed and scalable device, the security services provided by such a device can also be scalable. This will allow the security device to serve an increasing number of users with no degradation in throughput.

When end-to-end data confidentiality is required in a high-performance environment, there is no choice but to implement confidentiality at the ATM layer. Since ATM is designed to switch information in hardware, it follows that an ATM layer encryptor, which is similar in many respects to an ATM switch, can be implemented in hardware as well. This makes ATM encryption more suitable than higher layer encryption (which is typically implemented in software) for high speed applications. Additionally, ATM encryption is more suitable than lower layer encryption schemes (such as link encryption) because it encrypts only the ATM payload (which carries data), leaving the ATM header in the clear (whereas link encryption encrypts both the payload and the header). This allows intermediate switches, which may belong to a public carrier, to successfully switch encrypted information across the network without the need to decrypt at each switch.

By providing security at the ATM layer, a common security framework can be achieved for all applications and protocols that use ATM services — including native ATM applications and protocol entities such as IP. Due to layering, this service can exist transparently to these applications and protocols, and will not interfere with higher-layer security protocols (such as those provided by IPv6), if such additional security mechanisms are required (for example, by site security policy).

6. ATM Security Architecture

Reference models are typically used to provide scope and terminology when new work is started; particularly new work that is standards related. A reference model was developed early in the project which provided a generalized foundation for ATM security. This model is shown in Figure 6 (note that this reference model only shows half of the architecture — the complete architecture is symmetric across the Public ATM Network).

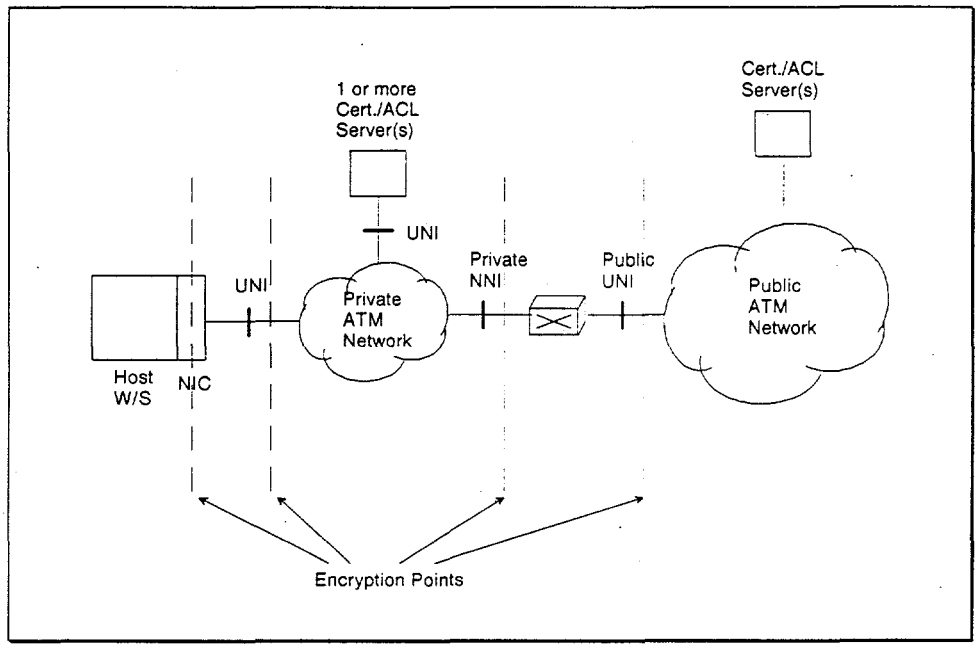


Figure 6: ATM Security Reference Model

The original purpose of this reference model was to show the possible points at which ATM end-to-end encryption may occur. As indicated in the model in Figure 6, ATM end-to-end encryption can occur at any (or all) of four points — in the ATM Network Interface Card (NIC), at the Private UNI (between the host and the switch), at the Private NNI (between private switches), and at the Public UNI. The location of ATM end-to-end encryption is largely dependent on the site's security requirements, performance needs, and budget. These tradeoffs are tabulated below:

Encryption Point	Private Network Sensitivity	Encryption Type	Cost	Per-Host Performance	Applicability
NIC (software)	Non-Sensitive	II	$	**	Clients
NIC (hardware)	Non-Sensitive	up to I	$$ - $$$	***	Clients and Servers
Private UNI	Non-Sensitive	up to I	$$$	***	Servers
Private NNI	Sensitive	up to I	$ - $$$	* - ***	Bulk Encryption
Public UNI	Sensitive	up to I	$ - $$$	* - ***	Bulk Encryption

Table 1: ATM E^3 Tradeoffs

This table assumes that the only acceptable implementations of Type I (government grade) algorithms are hardware-based. This assumption was made because Type I algorithms are currently classified, and software implementations of such algorithms could only exist in secure environments.

As indicated in the table above, if moderate performance at the host is acceptable, along with Type II (commercial grade) encryption, then software-based encryption at the host's NIC may be preferred to

provide data privacy at the lowest cost, particularly if there are a large number of end systems (e.g. for client workstations). If higher performance is required, and particularly Type I encryption, then a hardware encryptor integrated with the NIC or at the Private UNI may be required, but only if the cost is tolerable (which may be true if there are a small number of end systems that need this capability, e.g. servers). Another option is to place an ATM end-to-end encryptor at the Private NNI (for workgroup encryption) or at the Public UNI (for site network encryption). By doing so, the encryptor (and hence, its cost and performance) is shared by all of the users, however, this may require that the private network be physically secured.

In addition to encryption, the ATM security reference model also has provisions for the distribution of public key certificates. As described in Section 3.3, it may be necessary to provide public key "certificates" to the users, end systems, or processes that need them for encryption or authentication. This service could be located in the private network, the public network, or both. By providing a private network-based certificate service, public key certificates for everyone (or every node) in the organization can be made available. Additionally, the public key service could also be provide by an ATM carrier as a service to its customers, in which case, the carrier would have one or more certificate servers as well. This architecture allows communication between the private and public certificate servers to occur as well.

Finally, this architecture make provision for what is called an *Access Control List Server*. The ACL server's role is to provide ATM devices (end nodes, switches, and encryption devices) with access control lists when requested to do so. It is expected that this service will provide a convenient mechanism to distribute ACL updates to large ATM networks when security policy, and in particular, access rights change frequently. This feature provides a scalable solution to ATM network security management. Although the ACL server is logically separate from the public key certificate server in this reference model, these functions may be physically located on the same machine.

7. ATM Security Testbed Environment

7.1. Workstation and Network Environment

Early in the project, it was determined that an ATM testbed was needed to validate the protocols and mechanisms as they are implemented. The top-level architecture for this testbed is illustrated in Figure 7.

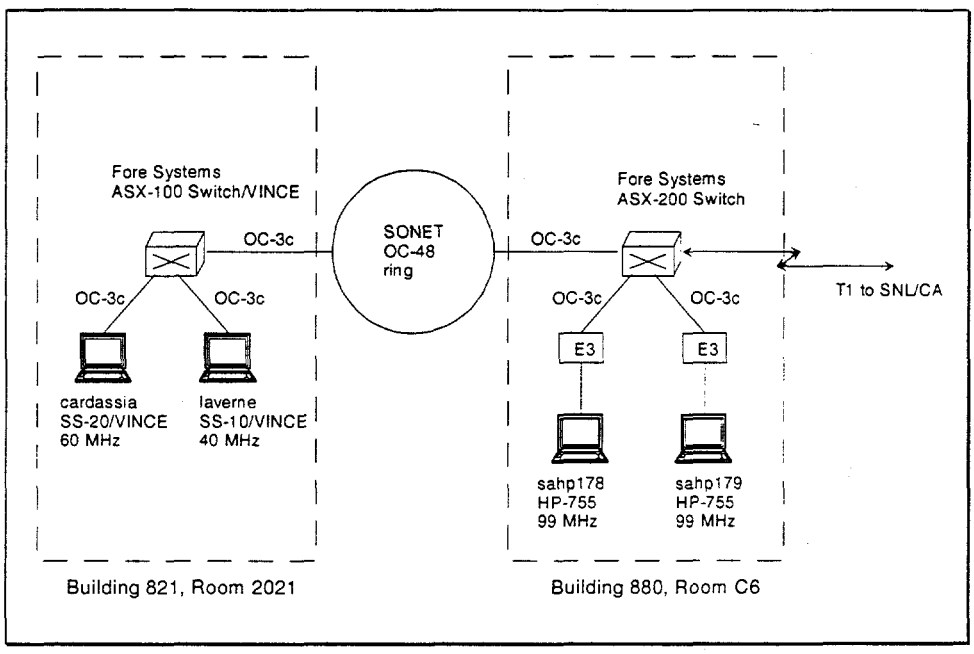


Figure 7: ATM Security Laboratory

This architecture is composed of two distinct laboratories. The Building 821 laboratory is the software development laboratory, and houses the VINCE development environment as well compilers for the SPARC and i960 (ATM interface adapter) platforms. The hardware platforms selected for this laboratory are exclusively based on Sun's SPARCstation architecture, due to VINCE's dependence on SunOS 4.1.3. The purpose of the SPARCstation 20 (60 MHz CPU) is to serve as a primary software development platform, and has compilers for both the SPARC and the i960 (which is used by the Fore Systems SBA-200 ATM adapter to perform ATM segmentation and reassembly functions) CPUs. The SPARCstation 20 has a Fore Systems SBA-200 interface adapter installed, which is used for end-system security signaling and embedded software encryption. The SPARCstation 10 (40 MHz CPU) serves as a secondary development platform and functions as an NFS file server for the ATM switch. This platform also has an SBA-200 ATM adapter installed.

Connected to both of these workstations is an ASX-100 ATM switch from Fore Systems, which allows testing of User-Network Interface (UNI) security extensions. Although the Building 821 switch is connected to the building 880 switch, security signaling at the PNNI can not be tested because VINCE did not implement the necessary signaling and routing functions, and because VINCE has not been ported to the new ASX-200 architecture used by the Building 880 switch.

The Building 880 laboratory is primarily used for development of the OC-3c encryption hardware. Since this laboratory does not use VINCE, the lab developers are less constrained with their selection of hardware. In this case, two HP-755s (99 MHz CPUs) workstations are used to provide ATM data traffic for testing, and they have Fore Systems' HPA-200 ATM interface adapters installed. These platforms are connected to a Fore Systems ASX-200 ATM switch, which in turn is connected to the Building 821

laboratory via a campus-wide OC-48 ring, and to the Sandia Livermore campus via T1 linkage. Although most of the encryption hardware testing occurs locally within Building 880, the ASX-200 is occasionally used in tests with traffic traversing the OC-48 or T1 linkages. The intent of these tests is to determine switching delay effects, speed of light delay effects, and rate adaptation effects on the operation of the hardware encryptors and associated signaling.

7.2. VINCE

The software used in this project to provide the ATM protocol functions is a publicly available package called VINCE (Vendor Independent Network Control Entity). VINCE, which was developed at the Naval Research Laboratory, provides the software tools required by this project to study various aspects of ATM networking. VINCE was originally developed to allow the introduction of experimental signaling and routing protocols to ATM networks. Because VINCE is non-proprietary, the source code is readily available, which makes it possible to modify VINCE with security extensions.

VINCE only runs on Sun SPARC workstations with SunOS 4.1.3 and Fore Systems ATM interfaces (actually VINCE can run in *simulation mode* on several other hosts, including SGI, and Hewlett Packard, but this mode was not used in this project). The only switch that is currently supported is the FORE Systems ASX-100. To execute VINCE, the vendor supplied device driver is replaced with the VINCE device driver on the hosts, and then the VINCE executable is started on the switch and the hosts. When the executable is started, it reads startup files which contain parameters denoting which ATM protocols, switching, etc. is to be used over the network. If a switch is not available, VINCE can run in the simulation mode stated earlier.

VINCE is implemented via a module-based design built around a central core. Each module implements the syntax and state manipulation for a particular protocol, and the core provides synchronization and operational context for the modules as a group. Libraries provide support routines for both the core and the modules. The libraries include support for AAL3/4, AAL5, and SSCOP.

VINCE is designed to use ports. These ports are grouped into virtual switches. Each virtual switch is considered to be a logical fabric and therefore any port contained within it can communicate to any other port. Each port has an associated signaling protocol which will maintain the status of the port and handle all of the call setup transactions for the port. The port also has a hardware module associated with it which allows VINCE to generate signaling messages to the ATM host or switch on the other side of the port. [11]

7.3. Sandia DSA Code

Early in this project, we contacted Vicky Hamilton (Sandia Labs, Department 9415) about code that she wrote for another project which implemented all of the components of the Digital Signature Standard (DSS). This code was provided to us in source form, and contained a useful *user interface library* which implemented the most common DSS functions (such as initialization, signature generation, and signature validation).

To adapt the DSS user interface library to VINCE, a "UNI Security Library" was written. These routines performed functions such as generating timestamps and sequence numbers, building authentication and key exchange information elements, adding and removing information elements from UNI messages, and checking information element fields (these routines can be found in the Appendix, Section 14.3). The DSS user interface library itself required no modifications.

8. ATM Security Mechanisms Designed and Implemented in this Project

8.1. Authentication

Early in this project, authentication was identified as a key security mechanism. Authentication is a required mechanism for encryption (more precisely, key exchange), distribution of public key certificates, and access control, to name a few. Since authentication is central to all of these ATM security services, and since ATM authentication will be used by a variety of users (with a variety of security requirements), the ATM authentication framework must be flexible.

In addition to flexibility, another requirement on the UNI authentication approach is that the approach must minimize its impact on the existing UNI signaling structures and protocols. This is required because to make to proposed extensions more acceptable to ATM standards bodies, however, it does place a number of constraints on the protocols developed in this project. For example, the ATM switched virtual circuit signaling protocol is a two-way handshake (i.e. calling party sends "SETUP" message, and called party sends "CONNECT" message). However, challenge-response protocols require at least a three way handshake (A: "I am A." B: "If you are A, then you can correctly sign this bit pattern." A: "Here is the signature you requested. You may verify its correctness."), which is incompatible with the setup protocol. Therefore, challenge-response approaches cannot be used to authenticate connection setup requests without modification to the UNI protocol. Fortunately, other mutual authentication protocols can be used which only rely on two-way handshakes, and are hence compatible. These protocols are described further in [22].

Although the UNI protocol itself is somewhat inflexible, the structure of the UNI messages which are used by this protocol is very flexible, and it allows straightforward extensibility through the definition of new information elements. This provides a natural avenue to provide extensions for signaling message authentication. The following section describes how this feature of the UNI specification is used to provide signaling message authentication.

8.1.1. The Generic Authentication Information Element

At the April, 1995 meeting of the ATM Forum, Sandia presented a contribution which outlined a proposed framework for the implementation of authenticated ATM signaling, which was subsequently modified for the August meeting [27]. These contributions proposed an "Authentication Information Element" that could be used to provide supplemental authentication information within any signaling message. The information could be used by either party in a virtual circuit to validate the claimed identity of the other party, and verify the integrity of a portion of a message's contents. These operations, while most likely to occur when a connection is being established, may be performed at any time during a connection's lifetime. Furthermore, the generic Authentication Information Element (IE) allowed authentication information to be generated by *any* signature algorithm. Since signaling messages may be exchanged between ATM endpoints at any time during a connection, it follows that an Authentication IE will allow "identity validation" to occur at any time during a connection's lifetime.

8.1.1.1. Requirements

Although a previous contribution recommended that the Digital Signature Standard be used for authentication, Sandia asserted in [17] that users need the flexibility to choose which authentication/digital signature standard they wish to use, based on the user's required security "robustness" (or strength, which is based on site security policy) and performance requirements. This need for flexibility implies the following requirements for the generic Authentication IE:

1. The Authentication IE must contain a field that identifies the authentication/digital signature standard that was used to generate the authentication information

2. The Authentication IE must be variable length

3. The Authentication IE must contain algorithm-specific information

One security threat which was seriously considered was masquerade. The ATM protocols themselves made this a problem because the ATM switch virtual circuit setup protocol is a two-way protocol, which makes mutual authentication using challenge-response techniques impossible (a mutual authentication protocol which uses challenges and responses requires a minimum of three messages). If the Authentication IE was not carefully designed with the two-way connection setup protocol in mind, one could record previous Authentication IEs, and use them in a replay attack to masquerade as another "user". Therefore, another mechanism is required to ensure that each authentication IE that is generated must be unique. To allow the receiving system to verify this uniqueness, the following were specified:

4. The Authentication IE must contain a timestamp

5. The Authentication IE should contain a sequence number

Since signaling messages are used by intermediate devices to establish routes based on quality of service (QOS) requirements, network-to-network signaling may legitimately change the contents of some of the information elements contained in the message. If the calling party calculates a digital signature across one or more of these information elements, signature validation at the called party will likely fail. In order to provide integrity assurances for information elements that are NOT modified in end-to-end signaling messages (*invariant* information elements), some means of identifying these elements, and their ordering during signature computation, must be provided in the Authentication IE. Hence the following requirement:

6. The Authentication IE must identify which components of the message were used when generating the digital signature, and the order in which they entered into the calculation of this signature.

Finally, a field must be added to the Authentication IE which positively binds the contents of the signaling message (including the Authentication IE) to the entity which generates the message. This requires that the Authentication IE must also contain:

7. A signature of (selected portions of) the signaling message

The following section summarizes these requirements in the structure of a proposed Authentication IE.

8.1.1.2. Design

The format of the Authentication IE is shown below:

8	7	6	5	4	3	2	1
Information Element ID							
ext	Coding Standard		Flag	Res.	IE Instruction IE Action Indicator		
Length							
Length (continued)							
Signature Algorithm							
Sequence Number							
Time Stamp							
Time Stamp (continued)							
Time Stamp (continued)							
Time Stamp (continued)							
IE List Length							
IE List . . .							
Algorithm-Specific Information . . .							
Signature (algorithm-specific) . . .							

Figure 8: The Generic Authentication IE

Information Element ID

This field is specified for all UNI 3.1 signaling messages (see [2], Section 5.4.5.1).

IE Compatibility Instruction Indicator

This field is specified for all UNI 3.1 signaling messages (see [2], Section 5.4.5.1).

Length

This field is specified for all UNI 3.1 signaling messages (see [2], section 5.4.5.1), and allows the Authentication IE to be variable length (per requirement 2)

Signature Algorithm

This field contains a code which identifies the algorithm that was used to generate the signature (per requirement 1).

Sequence Number/Time Stamp

These fields allow each Authentication IE to be unique (per requirements 4 and 5).

IE List Length

This field provides the length of the Information Element List.

IE List

This field contains a list of information elements identifiers that specifies the IEs over which the digital signature was computed (per requirement 6). The IE List also denotes the ordering of IEs during signature generation and validation.

Algorithm-Specific Information

This field allows supporting information that is required for signature validation (such as public key information, parameters, etc.) to be sent to the remote party (per requirement 3).

Digital Signature

This field contains the digital signature for the message (per requirement 7). The signature scope includes the IEs denoted in the IE List field, and the contents of the Authentication IE itself. This scope allows the message originator to "sign" the integrity (or "correctness") of the non-variant information contained within the signaling message. The signature operations are described in more detail in the following section.

8.1.1.3. Authentication IE Usage

As stated in earlier, the digital signature scope includes selected, *invariant* information elements in the message, as well as the Authentication IE itself (where *invariant* IEs are IEs that are not modified in transit between end systems). Since the signature of the signed message cannot be known beforehand, and the Authentication IE can occur anywhere in the signaling message, the Signature field in the Authentication IE must be filled with "placeholder" information (zeros) before the digital signature is computed. After this computation, the placeholder information is then replaced with the actual signature, and the message is sent to the remote party. (It should be noted that this is the same method as the one that is used in SNMP Version 2 for the generation of authenticated SNMP messages.)

No other information elements are required to be present in a signaling message which contains an Authentication IE. However, the remote node reserves the right to refuse a connection attempt if the originating node does not supply information which is needed by the remote node (such as a calling party ID). In this case, the node can send a cause code back to the originating node that indicates which IEs it needs. Supporting IEs have not been specified for two reasons:

1. The design of the authentication framework must support the addition of new information elements in the future.

2. By specifying a set of required information elements, signaling messages could become unnecessarily large, particularly in cases where the validating entity requires only a small subset of these "required" information elements.

Finally, if an IE list which is incomplete (from the receiving node's perspective), the receiving node may reject the message with a cause code indicating an incomplete IE list.

8.1.2. Digital Signature Standard Fields for the Generic Authentication IE

Also at the April, 1995 ATM Forum Technical Committee meeting, Sandia presented a contribution which proposed "Algorithm-Specific Information" and "Signature" fields for a Generic Authentication Information [28] which uses the Digital Signature Standard (DSS). The Digital Signature Standard, which was developed by the United States National Institute of Standards and Technology (NIST), uses the Digital Signature Algorithm (DSA) to ensure the integrity and authenticity of electronic transactions. The DSA uses a hash value of the message (computed using the Secure Hash Algorithm), the signer's private key, and a cryptographic algorithm to generate the digital signature. When used with the Generic Authentication Information Element, the integrity and authenticity of signaling messages can be validated with confidence by another party.

8.1.2.1. Requirements

Currently, the DSS specifies that the DSA modulus be 512 bits in [14]. However, several cryptanalysts have criticized this specification on the basis that this modulus, and associated parameters, are not large enough [22]. Conversely, for some applications, a 512 bit modulus may be too large. To support various levels of robustness (i.e. cryptographic strength), the DSS fields must meet the following requirement:

1. DSS-specific parameters and signature values should be variable length

The DSA uses a number of public parameters to generate and validate signatures. To minimize the time required to generate and validate signatures, these parameters, as well as the hash function identification, may be distributed beforehand to entities that are involved in these processes. Therefore, as an optimization, the following is also required:

2. Publicly known DSA parameters may be omitted from the Authentication IE

8.1.2.2. DSS-Specific Information Fields

The following diagram shows the DSS-specific format of the "Algorithm-Specific Information" field of the Generic Authentication IE. Each of these fields are optional (see requirement 2).

8	7	6	5	4	3	2	1
P Parameter ID *							
P Length *							
P * . . .							
Q Parameter ID *							
Q Length *							
Q * . . .							
G Parameter ID *							
G Length *							
G * . . .							
Y Parameter ID *							
Y Length *							
Y * . . .							

Figure 9: DSS-Specific Information

* Optional parameter

P Parameter ID

This field identifies the following parameter as the P parameter. The P parameter is a public parameter which is the prime modulus used by DSA [14].

P Length

This field contains the length of the P (see requirement 1).

P

This field contains the P parameter described above.

Q Parameter ID

This field identifies the following parameter as the Q parameter. The Q parameter is a public parameter which is the prime divisor used by the DSA [14].

Q Length

This field contains the length of the Q (see requirement 1).

Q

This field contains the Q parameter described above.

G Parameter ID

This field identifies the following parameter as the G parameter. The G parameter is a public parameter used by the DSA [14].

G Length

This field contains the length of the G (see requirement 1).

G

This field contains the G parameter described above.

Y Parameter ID

This field identifies the following parameter as the Y parameter, or the "public key" [14].

Y Length

This field contains the length of the Y (see requirement 1).

Y

This field contains the public key described above.

8.1.2.3. DSS Signature Fields

The following figure shows the DSS-specific format of the "Signature" field of the Generic Authentication IE. All of these fields are required.

8	7	6	5	4	3	2	1
R Parameter ID							
R Length							
R . . .							
S Parameter ID							
S Length							
S . . .							

Figure 10: DSS Signature

R Parameter ID

This field identifies the following parameter as the R parameter, one of two components of the DSS digital signature [14].

R Length

This field contains the length of the R parameter (see requirement 1).

R

This field contains the digital signature component described above.

S Parameter ID

This field identifies the following parameter as the S parameter, the second component of the DSS digital signature [14].

S Length

This field contains the length of the S (see requirement 1).

S

This field contains the digital signature component described above.

8.1.2.4. DSS Performance Issues

The DSA algorithm is slow, particularly for signature validation. However, prior information can be used to optimize its signature generation and validation performance. The greatest performance improvement can be realized when all authenticating entities in an ATM network use common values of the P, Q, and G parameters. This allows a one-time initialization to be used over all subsequent signature generation/validation operations.

If, by chance, another authenticating entity uses different values of P, Q, and G, then the entity which validates the signature will need to initialize another signature generator/validator with these values. This optimization still allows generation and validation of signatures with different parameters, however, this will slow authentication operations considerably.

8.2. Hardware Encryption Research Prototype

A "research prototype" encryptor/decryptor was also developed under this project. This prototype is intended only to demonstrate the viability of achieving research objectives by processing ATM cells in a SONET OC-3 payload. A "Filter Generator" was chosen for implementation in the Sandia Research Prototype. Linear Feedback Shift Registers (LFSR) produce Linear Recurring Sequences (LRS) with long periods which have good "pseudorandom" properties. LFSRs are also easily scaled to generate multiple bits of the sequence in parallel. However, crypto-analytic methods exist to utilize the linear predictability of LRS to mount a cyphertext-plaintext attack against a purely linear sequence used as a key stream [24]. Several variations of periodic sequence key stream generators exist which deter such cryptanalysis [24]. A filter generator uses a non-linear function to mask the linearity of a long linear recurring sequence generator.

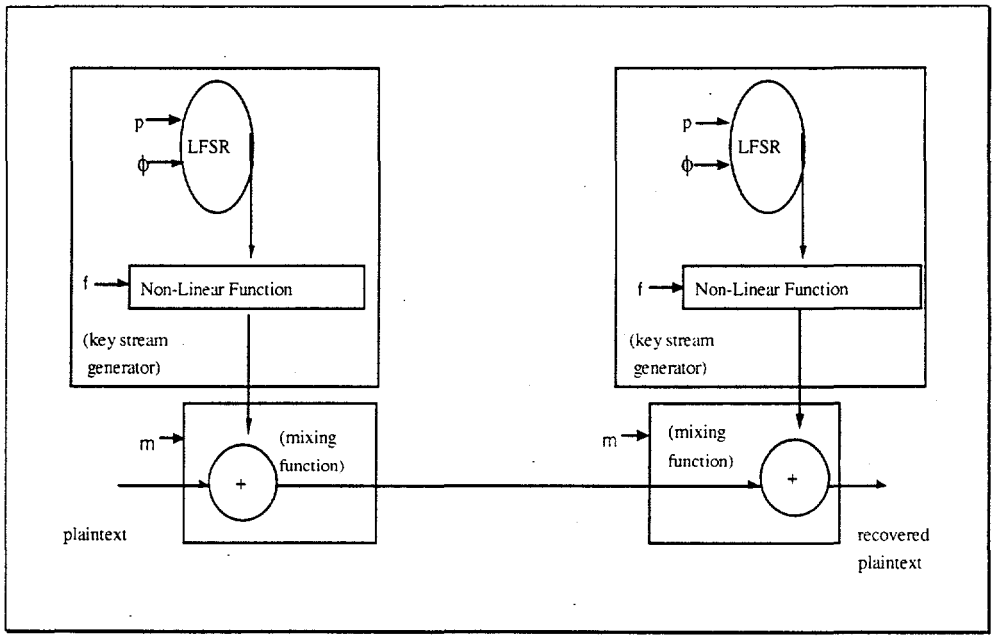


Figure 11: Filter Generator

This design involves no "feedback" around a non-linear function in order to both scale and interoperate with implementations of other scale factors. It also provides no error magnification. Single bit errors in cyphertext result in single bit decrypted plaintext errors. If the linear recurring sequence is sufficiently long, subsequent identical plaintexts are encrypted into different cyphertexts. This deters dictionary lookup and playback attacks.

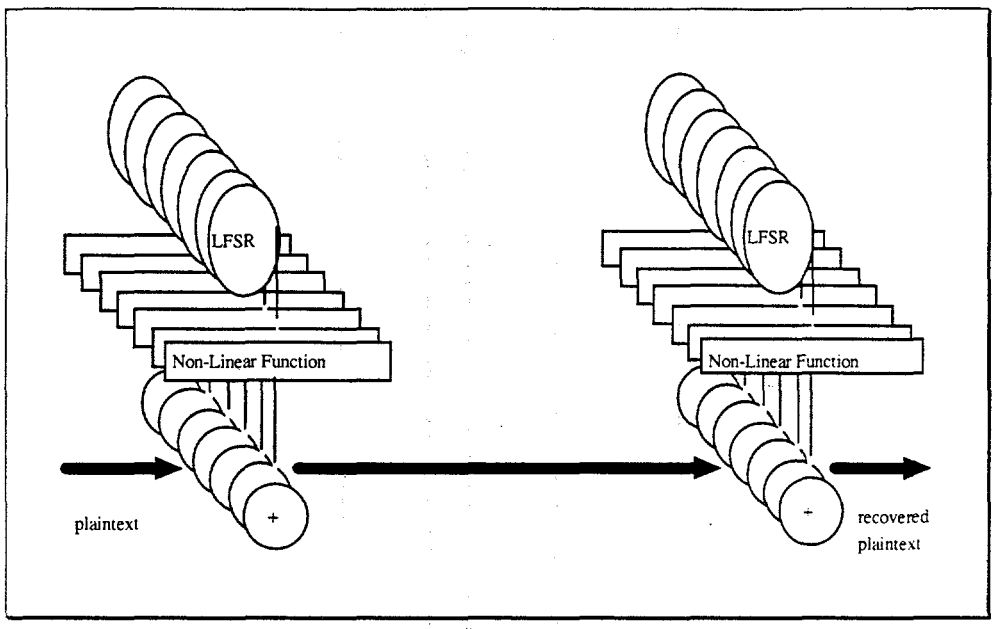


Figure 12: Parallel Filter Generator

The linear and non-linear functions were designed to add minimum traffic delay. The traffic delay through the prototype was measured to be about 2.7 microseconds. This delay is due to one clock period required to encrypt each byte plus the time required by the ATM/SONET framer to assemble one cell.

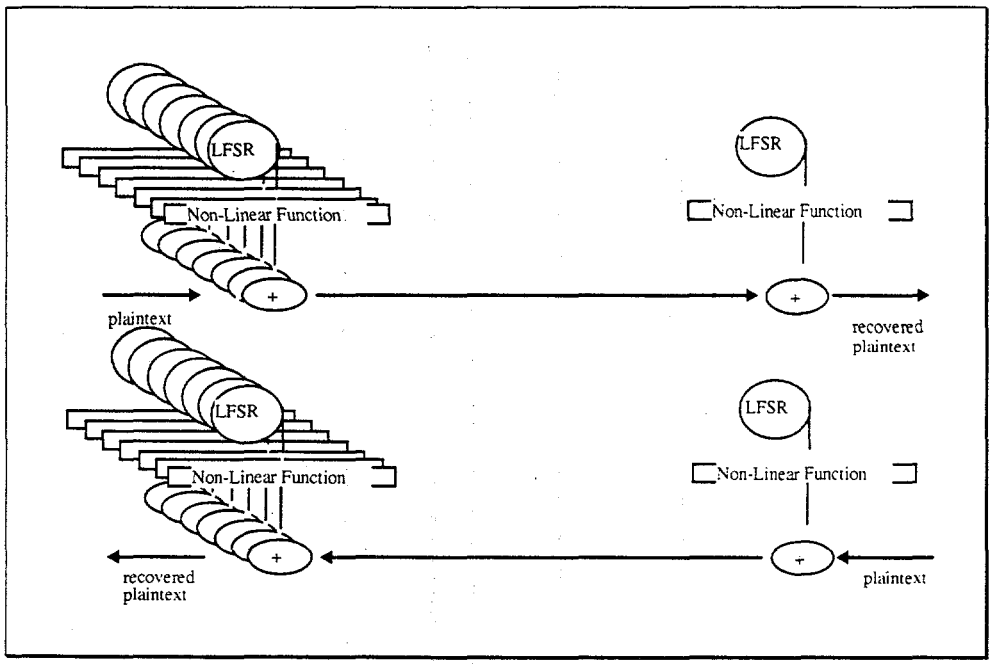


Figure 13: Interoperation of Scaled and Unscaled Filter Generators

A prototype which encrypts and decrypts 8 bits at a time was interoperated with an implementation which processed 32 bits at a time. Although this particular combination of scale factors was chosen for "proof of concept", implementations of any scale factor will also interoperate. "Key-Agile" cryptovariable context switching is done on the basis of the Virtual Path Indicator (VPI) and Virtual Circuit Indicator (VCI) in

each cell header. The prototype achieves a cryptovariable context switching time of 50 nanoseconds (one clock period). In order to rapidly demonstrate "proof-of-concept", only two cryptovariable contexts were implemented, and the prototype implements no "key management". Session keys are embedded in the prototype's Electrically Programmable Logic Devices (EPLDs).

Several crypto sync loss detection and recovery methods are under investigation. The initial testing has involved only the simplest of synchronization methods, involving implicit re-synchronization of each cell payload to an initial state. The encryptor was designed to perform synchronization upon demand, and periodic synchronization. The synchronization upon demand is implemented by signaling between the encryptor and decryptor via "Operation, Administration, and Maintenance" (OAM) cells as described in Section 8.4.

8.3. Embedded Software Encryption

Although hardware-based encryption is capable of encrypting ATM payloads at full line rate, it can be too expensive for some applications and sites. For example, the cost of putting encryption hardware at each desktop client (which may not utilize full OC-3 bandwidth) can be overwhelming. A more cost-effective solution to secure ATM client-server applications would be to provide hardware encryption for the small number of nodes that have a need for full bandwidth encryption (e.g. servers), and to provide low cost (albeit lower performance) software-based encryption for the larger population of clients.

In this project, software-based ATM end-to-end encryption was developed to provide a less expensive, but lower-performance encryption alternative. The goals for this effort were to:

- Integrate the software-based encryption with VINCE
- Off-load the encryption processing to the ATM interface adapter, if possible
- Show throughput performance exceeding 10 Mbps
- Show application transparency
- Demonstrate interoperability with the hardware version

Upon close examination of VINCE, it was evident that VINCE could be easily modified to provide ATM end-to-end encryption. The architecture of VINCE for workstations equipped with an SBA-200 ATM interface allowed the encryption algorithm to be off-loaded to the interface adapter, and provided a simple message-passing mechanism for communication between a user-level process, the VINCE device driver in the host operating system, and the i960 software running on the SBA-200 (see Figure 14). Once the message-passing mechanism was studied, and the modular organization of VINCE was determined, it was a simple matter to insert code into VINCE's `sba200_atm.c` controller module to perform encryption and encryption-related signaling in the same manner as the hardware encryptor.

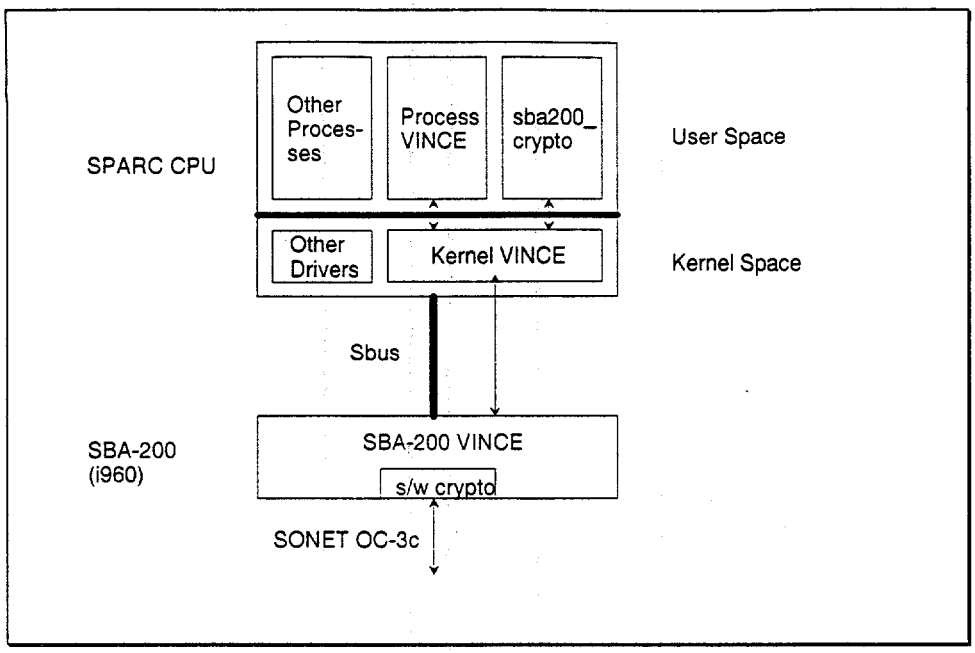


Figure 14: Software Encryptor Architecture

8.4. Signaling Support for Encryption

When encryption and decryption operations are performed on a virtual circuit, the encryptor and decryptor must exchange messages from time to time. These messages include:

- "go secure" signaling
- request for re-synchronization
- resynchronization
- key exchange
- additional authentication

In the hardware and embedded software encryption prototypes, these messages are exchanged in the data-bearing virtual circuit after it is established. Three alternative mechanisms were considered for exchanging these messages; two ATM-layer mechanisms, and one "encryption application" layer mechanism. The first proposed ATM layer mechanism took advantage of a Payload Type Indicator codepoint that was undefined by the ATM Forum. This codepoint (111 binary) would designate the cell as a "crypto-signaling" cell which would have a subtype for each of the signaling functions listed above. However, this approach's main drawback was that it usurped the only remaining "undefined" codepoint for the PTI. It was believed by the project members that this would not be received favorably by an ATM standards committee, and thus, this proposal was rejected.

One possible mechanism for getting around this difficulty would have been to establish a "control" virtual circuit between the encryptors. This would allow the encryptor/decryptor pair to exchange arbitrarily formatted messages with very little impact on the existing standards. Furthermore, the hardware encryptor architecture has an "ancillary processor" which could handle the complicated message formatting and processing functions, while leaving the simple (yet time-critical) tasks to the specialized hardware. However, one problem exists with this architecture — synchronization is required between the control channel and the data channel in order to guarantee re-synchronization of the ciphertext stream. By offloading complex message processing tasks to the ancillary processor, a large mean delay is introduced between the control channel and the data channel, with a potentially large delay variance as well. One

way around this would be to put the message processing functions in hardware, however, this has a couple of problems as well. Obviously, putting these complex functions in specialized hardware would take much work. However, even if this could be accomplished, there still remained the possibility that the control and data channels would follow separate physical paths or encounter congestion, either of which would make synchronization between the two channels difficult.

However, another solution exists for this problem. Since ATM cells are guaranteed to be switched in the order in which they are received, in-band messaging (i.e. crypto messaging that occurs in the data channel) would provide guaranteed synchronization between the control messages and the data stream. Fortunately, an extensible mechanism already exists for this messaging; Operations, Administration, and Management (OAM) cells. OAM cells can be either associated with a virtual path (F4 OAM cells), or with virtual path/virtual circuit (F5 OAM cells). Furthermore, both types of OAM cells can either have hop-by-hop (segment) significance, or end-to-end significance. However, the most fortunate aspect about OAM cells for this purpose is the fact that many OAM Type and Function Type codepoints are available. In fact, a specification already exists in ANSI that defined these types for security setup and security maintenance functions.

In this project, it was decided to use F5, end-to-end OAM cells as a framework for encryptor/decryptor messaging. This definition is illustrated below:

GFC/ VPI (4)	VPI (8)	VCI (16)	PTI (3)	CLP (1)	HEC (8)	OAM Cell Type (4)	Function Type (4)	E³ Function Type (8)	Function-Specific Fields (44 * 8)	CRC-10 (16)

Figure 15: F5 (VC-level), End-to-End OAM Cell Format

where:

PTI = 101 (end-to-end F5 flow OAM cell)
CRC-10 polynomial = $x^{10} + x^9 + x^5 + x^4 + x + 1$

The following table shows the OAM cell definition from UNI 3.1:

OAM Cell Type		Function Type	
0001	Fault Management	0000	AIS
		0001	RDI
		1000	Loopback

Table 2: F5 OAM Fields from UNI 3.1

Where:

AIS - Alarm Indication Signal
RDI - Remote Defect Indication

Table 3 shows the UNI 3.1 OAM cell definitions, along with the standard ANSI OAM Cell Type and Function Type for security setup and maintenance. These subtypes are further sub-defined for the Sandia hardware and embedded software encryption functions (indicated as shaded entries).

OAM Cell Type		Function Type		E^3 Function Type	
0001	Fault Management	0000	AIS	N/A	N/A
		0001	FERF	N/A	N/A
		1000	Loopback	N/A	N/A
1111	E^3 OAM Cell	0001	Security Setup	undef	undef
		0010	Security Maintenance	0000 0001	Request Synchrocell
				0000 0010	Synchrocell

Table 3: Augmented F5 OAM Cell Definition for E^3

The E^3 Function-Specific Field immediately follows the E^3 Function Type. The E^3 Function Specific Field for the *Synchrocell* is indicated in the following data structure:

```
struct synchrocell_fsf {
    long dPhase; /* 32 bit phase difference (phase - ref_phase) */
    unsigned char pad[41];
};
```

where `phase` is the current encryption context for the virtual circuit, and `ref_phase` is the key. Since no further information is required to request re-synchronization for the virtual circuit, the *Request for Synchrocell* does not have an E^3 Function-Specific Field.

8.5. Key Exchange Protocols

In situations where data channel confidentiality is required, it may be necessary for two ATM entities (either end stations or encryptors) to negotiate a key for the data session. An information element was designed which will allow two parties to used the ATM signaling framework to exchange key exchange information along with authentication credentials in an ATM message. As with the authentication information element, the key exchange information element is broken into two parts: a "generic" portion (which applies to all key exchange protocols), and a "protocol-specific" part. The "generic" portion is shown in Figure 16, and the "protocol-specific" portion, designed for the DSS-based key exchange protocol, is shown in Figure 17.

8	7	6	5	4	3	2	1
Information Element ID							
ext	Coding Standard		Flag	Res.	IE Instruction IE Action Indicator		
Length							
Length (continued)							
Key Exchange Protocol							
Protocol-Specific Information . . .							

Figure 16: The Generic Key Exchange IE

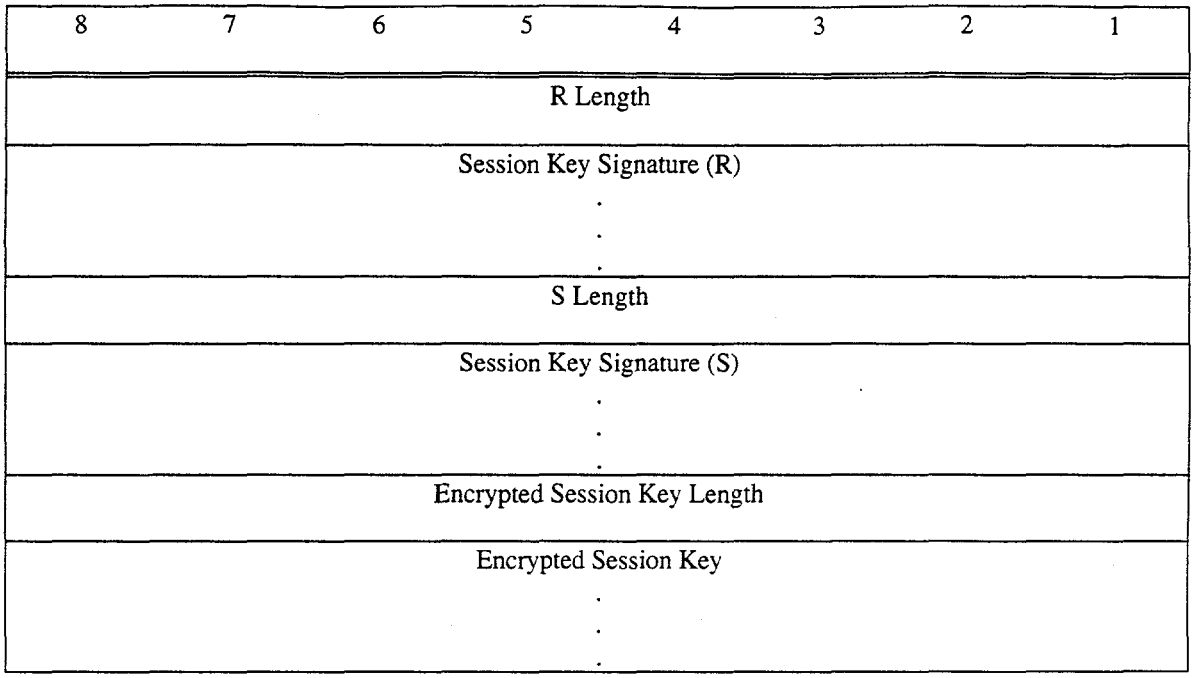


Figure 17: DSS Key Exchange Information Fields

Since this information element does not provide authentication, an authentication information element must accompany the key exchange information element. However, this is the only information element that is "required" to accompany the key exchange information element.

This mechanism was partially implemented in this project. Although the ultimate goal was to implement key exchange within ATM signaling, delays in the implementation of authenticated signaling made this impossible in the allotted time. However, this protocol was implemented using a "mock" signaling channel (a file) which was used by two processes to negotiate a key. Since the mechanism was only partially implemented, it was not presented to the ATM Forum.

9. ATM Security Mechanisms which have been Designed, but not Implemented

During this project, the need for a mechanism to distribute public key certificates became apparent. Due to time and funding constraints, a public key distribution mechanism could not be implemented. However, this is a necessary component when using authenticated signaling in medium and large scale networks, and for this reason, must be considered "future work".

9.1. Distribution of Public-Key Certificates

9.1.1. The Research

Security on the Internet ideally protects the confidentiality of messages sent across the Internet. It is dependent on levels of technology to block unauthorized access. IP source addresses can be easily forged, a recommended second level of security is available via the proof and possession of cryptographic keys.

There are many companies providing encryption and signaturing methodology; and each in a proprietary fashion with something specific to offer. The selection choice was based on knowledge and research by the electronic mail project and the computer security department. Three vendor products were researched, "Pretty Good Privacy", PGP from ViaCrypt, "Secret Agent" from AT&T, and "Entrust" from Nortel.

PGP provides a mechanism of signaturing via Rivest, Shamir, Adleman (RSA) technology. AT&T provides an enciphering application, Secret Agent, which uses the methodology required by DOE, Digital Signaturing Standard or DSS. The third application, Entrust uses RSA algorithms for encryption and signature and additionally uses DSA for digital signature. Entrust also provides "hooks" for users to implement the encryption methodology of choice.

The difficulty is not in the selection the signaturing algorithm, rather in the key management. Keys must be accessible and trustworthy. They must be issued by a Certifying Authority who can speak for the authentication of the key. The keys in turn must be validated for revocation and contain strong nonrepudiation properties.

Of the three applications under consideration, Entrust was the only one which provided key management via an easily accessible directory. The directory methodology is X.500 and utilizes a set of standards which define the keys, known as the X.509 Directory Authentication Framework. This set of standards allows information processing from different manufacturers, different software management, different levels of complexity and of different ages. This system utilizes a hierarchical system of data management and has proven to be extremely extensible as well as scalable to adapt to increased usage demands. It further allows key exchanges to be coupled with authentication. Each party then has assurance that the exchanged key had not been shared with an impostor.

The other two applications provide key management via a "key-ring" basis. This is a set of keys that is maintained at the individual workstation and whereas it may originate as corporate information, the validity of the key management is the responsibility of the user. The key-ring system of key management is primarily used for a small number of clients.

9.1.2. The Choice - Entrust

Entrust is an application which provides encryption, digital signature, and key management capabilities to a variety of computer platforms and operating systems. This is the product of Nortel, previously Northern Telecom Secure Networks in British Columbia, Canada.

The primary reason for selecting Entrust against other vendor packages was the consideration of the public key presentation. Management of the public keys via an X.500 Directory could easily be incorporated into our current X.500 Directory structure at Sandia and would allow access to the keys by all of our customers and colleagues on a global basis.

Entrust also provides the APIs (application programming interfaces) which allow the system administrators to customize the application. Since Entrust is encryption algorithm independent, the administrator can choose the encryption methodology which meets the site's security requirements. A release of Entrust with (DOE-required) DSS signaturing capability is scheduled for Quarter 1, 1996.

Entrust is created to operate within the TCP/IP protocol. When we researched utilization of RPCs, remote procedure calls, to interact with the Entrust APIs directly over the ATM network we found that the functionality is not yet being provided by the vendor. It was felt that the time it would take to develop this type of interaction would delay the project beyond the delivery date. Alternatively, key certificates were created and provided via a directory without the use of the Entrust application. These keys provided the necessary functionality to test the protocol extensions for ATM security.

10. Implementation and Performance Observations

10.1. Implementation of Custom Protocols within VINCE

Since VINCE provides source code for ATM hosts and switches which implement UNI signaling, AAL segmentation and reassembly, and the generation of OAM cells, it is a good foundation for the implementation of ATM protocol extensions for security. By modifying the VINCE signaling code to manipulate authentication information elements in UNI messages, two nodes can strongly authenticate each other at connection setup time. In addition, the segmentation and reassembly code in VINCE allowed a straightforward implementation of embedded software encryption in the ATM network interface adapter. In a similar fashion, further modifications in VINCE to support key exchange and the distribution of public key certificates could also be made.

Although VINCE provides the necessary "hooks" to implement the security extensions in end systems and switches, it did cause a number of difficulties as well. VINCE was originally developed as a tool for early adopters of ATM to implement multivendor ATM networks that use both the standard UNI signaling and Fore Systems' SPANS signaling protocols. As such, its architecture had to be flexible. This architecture caused a number of problems early in the project because the program flow was not obvious to the researchers. In addition, its flexibility necessitated the use of "startup" files which held configuration information for the endpoints and switches, and some of the key "features" of the contents of these files were undocumented. For example, the documentation and sample setup files did not accurately describe how the process level VINCE should initialize and "connect" with an SBA-200 adapter. As a result, some time was spent trying to determine the problem before one of the VINCE developers pointed out our mistake. These difficulties in the use and understanding of the VINCE architecture caused considerable delay in the development of security enhancements.

In addition to the difficulties posed by the complex architecture, a number of problems were encountered with the VINCE code. The VINCE source code was rather complicated and not commented very well. Furthermore, the executable was not very robust. On a number of occasions, minor (even seemingly insignificant) changes would cause VINCE to crash. When the time came to integrate the security extensions with VINCE, the software would repeatedly cause memory segmentation faults, dump core and stop execution. Due to VINCE complexity and time constraints, it was decided that thoroughly debugging VINCE was not prudent.

Since it was suspected that the original VINCE code had memory problems, an independent software module named the Security Server was developed. The Security Server was a separate process that had its own address space, thus eliminating any additional problems caused by the addition of calls to the UNI Security Library (summarized in Section 7.3). Additions were made to the VINCE executable so that it sent a request to the Security Server (using UNIX sockets for interprocess communication) to perform a security related function. The Security Server would perform the function and then send its status and resulting data back to the VINCE executable, which waited for the result. This provided synchronization between VINCE and the Security Server, as well as the required memory space isolation.

Altogether, VINCE provides an excellent environment in which to perform this research. Given the amount of work required of the VINCE developers to get the complex package written and running, the small VINCE development team, and the additional problems imposed upon them by evolving signaling standards, VINCE is indeed a fine package. With the development of future releases, it is expected that VINCE will provide the robustness required to more smoothly implement additional enhancements, and accurately measure the impact of these enhancements on data throughput, connection delay, etc.

To be certain, not all of the problems encountered in this project were a result of VINCE. For example, mid-way through the project, when we were finishing the implementation of embedded software

encryption, it was discovered that the Fore ASX-100 switch was calculating AAL 3/4 CRCs instead of AAL 5 CRCs when performing UNI signaling (actually, this problem manifested itself earlier, but was incorrectly attributed to VINCE). When VINCE implements UNI signaling on the ASX-100, it attempts to disable the AAL 3/4 generator (which is implemented in hardware), and enable an AAL 5 CRC generator in software. However, our ASX-100 hardware did not allow the AAL 3/4 CRC generator to be disabled, hence, any UNI signaling messages that were generated by the switch were corrupted at the cell boundaries by the incorrect insertion of AAL 3/4 CRCs. As a result, it was decided to modify the VINCE software to implement UNI signaling over AAL 3/4 instead of the standard AAL 5. Although this precludes interoperability with other UNI implementations, this modification had no effect on the implementation and performance of the security extensions because this project only used VINCE hosts and switches.

10.2. DSS-Based Authentication

The Security Server described earlier implemented three authentication functions. The first was to call a routine named `initAuth()` (see Section 14.3). This routine initialized the DSS engine for generating authentication IEs. The next function performed was to generate an Authentication IE. This was accomplished by calling the `genAuthIE()` function. The last function that the Security Server performed was to validate a message which contains an authentication information element. This was accomplished with the `validateMesg()` routine.

Once this was implemented in VINCE, the performance of the DSS-based UNI message authentication mechanism was studied. The timing results are shown in Table 4 and Table 5.

UNI Authentication Function	Elapsed Time (msec)
Initialize DSS Generator	17,942
Generate Authenticated Message	190
Validate First Message	53,185
Validate Subsequent Messages	275

Table 4: UNI Authentication Timings: SPARC 20 (60 MHz CPU)

UNI Authentication Function	Elapsed Time (msec)
Initialize DSS Generator	26,634
Generate Authenticated Message	217
Validate First Message	78,004
Validate Subsequent Messages	404

Table 5: UNI Authentication Timings: SPARC 10 (40 MHz CPU)

These timing results show that the functions that perform initialization of the DSS generator and validator are slow. Our implementation compensated for the amount of time required to initialize the DSS generator by performing this initialization during the startup phase of the UNI protocol engine. However, when these functions were used with the UNI protocol to set up authenticated switched virtual circuits, they caused the UNI protocol engine at the calling party to time out while the called party validated the first UNI message (which initialized validation engine). This timeout occurs each time a new calling party attempts to establish its first connection to the called party. However, once the called party's validation engine was initialized, subsequent authenticated connection requests were completed without timeout problems.

Although the performance of our implementation was disappointing, time constraints did not allow investigation of alternative methods to achieve strongly authenticated UNI messaging. Alternative approaches for decreasing the likelihood of timeout problems include pre-initialization of the validation engines, and investigation of alternative authentication algorithms (such as RSA).

Also note that these timings are proportional to the CPU clock rate. This indicates that these functions are largely bound by the CPU, as opposed to the I/O or filesystem of the workstation. This leads to another potential optimization: move the authentication algorithm into a special-purpose processor (perhaps an expansion board).

10.3. Hardware Encryption

After correcting initial hardware problems with the prototypes, preliminary testing in the Sandia and CalREN testbeds has shown no discernible increase in end-to-end delay (measured via "ping" between workstations), no increased cell loss rate due to encryption, and no decrease in TCP "memory-to-memory" throughput due to encryption.

These tests confirm that the "Filter Generator" design scales and inter-operates with unscaled implementations, and does not magnify the error rate to which the cyphertext is exposed. Delay of communication traffic through the encryptor has been measured to be 2.7 microseconds, essentially the time required for assembly of a single cell by the ATM/SONET cell framer. Cryptovariables were embedded in the programmable devices to speed implementation. The prototype encryptors demonstrate cryptovariable context switching between two contexts within 50 nanoseconds, demonstrating the viability of the "Key Agility" concept.

10.4. Embedded Software Encryption

Evaluation of the ATM NIC-based software encryption indicated that the concept largely worked as expected. However, the implementation of this encryption capability within VINCE made quantitative analysis impossible. This is due to the implementation of the SBA-200 device driver in VINCE Version 1.0.1, which caused the driver to lock up during sustained, high-rate requests. This circumvented the use of `ttcp` to fully quantify the degradation of memory to memory throughput due to the software-based encryption module.

In October, 1995, Mike Moser of Ideas Corporation (contractor to the NSA Milkbush encryptor project) provided us a more recent version of VINCE 1.0.1 which contained a more stable version of this device driver. Evaluation of this new driver indicated that it was indeed more stable, and allowed throughput analysis using `ttcp`. Analyses with this new code indicated that `ttcp` throughput with and without the software encryptor was limited to roughly 2 Mbps. This indicates that a bottleneck exists somewhere other than in the encryptor, and until this bottleneck is removed, encryptor performance degradation cannot be determined using this method.

Although the throughput degradation of this implementation could not be determined, we were able make other very important determinations. These include:

- The software encryptor interoperated with the hardware encryptor. This interoperability was shown when the software was operating in the 8 bit (unscaled) mode as well as the 32 bit (scaled) mode.

- The software encryptor operated transparently to the user applications. The only involvement required by the user was to use an X Windows Graphical User Interface (GUI) to submit commands (such as "Go Secure" and "Re-synchronize") to the software encryptor.

The interoperability results were particularly important for this portion of this project. The main criterion behind the algorithm selection was scalability; an algorithm that is scalable in width allows scalability in speed through increasing parallelism. However, an algorithm that is scaled to one extent must interoperate with the same algorithm scaled to a different extent (the algorithm may be differently scaled, depending on the rate of the link interface, width of data registers, microprocessor designs, cost, etc.).

10.5. Signaling Support for Encryption

When the encryptors were tested, many of the OAM cells used to support encryptor-related messaging were exercised as well. As expected, the F5 (VC-level) end-to-end OAM cell approach worked without modification to intermediate systems (e.g. switches), and allowed either encryptor to request and act upon synchronization information from the other encryptor. Since these OAM cells are embedded in-line with the encrypted data, resynchronization of the data stream was observed to be reliable.

10.6. Key-Exchange Protocols

Key exchange protocols were established to identify the means for the users to decide upon and successfully exchange a session key for an individual ATM connection. (The key exchange protocol design is explained in Section 8.5.)

The timings indicated in Table 6 were taken during runs of sample code which implemented the two different functions of the key-exchange protocol. User A decides upon the session key, encrypts it with his/her private key and prints it out to a file. The file serves as a substitute for an actual ATM signaling channel. User B reads in the encrypted session key from the file, and decrypts it using User A's public key.

It should be noted that the method for choosing a session key, is not addressed in this particular study. User A simply uses a precomputed prime number for the session key. Since computing a prime number can take a relatively large amount of CPU time, methods for choosing "good" session keys (ie. good prime numbers) should be addressed.

The timing results also do not include time required to obtain the other user's public key. It is assumed that the user's public key has previously been obtained and validated from a key server.

Finally, the method used to encrypt/decrypt the session key is a simple XOR function. This is definitely not a very secure encryption method and requires very little processing time.

Operation	Time (seconds)
Create and Sign Session Key (User A)	0.183326
Get and Decrypt Session Key (User B)	0.180548

Table 6: Key Exchange Timings : Sparc 20 (60 MHz CPU)

These results seem to indicate that the actual key exchange does not take a significant amount of time. Given all the constraints stated above, further investigation of the protocol with more extensive and secure methods is needed.

11. The ATM Forum

11.1. Organization

The ATM Forum was one of the primary "customers" for this research. The ATM Forum is a collection of ATM device manufacturers, services providers, and users which was founded on the goal to accelerate the development, deployment, and adoption of ATM. The structure of the ATM Forum, shown in Figure 18, has three major committees: the Marketing and Awareness Committee (MAC), the users' committee (the Enterprise Network Roundtable, or ENR), and the Technical Committee (TC).

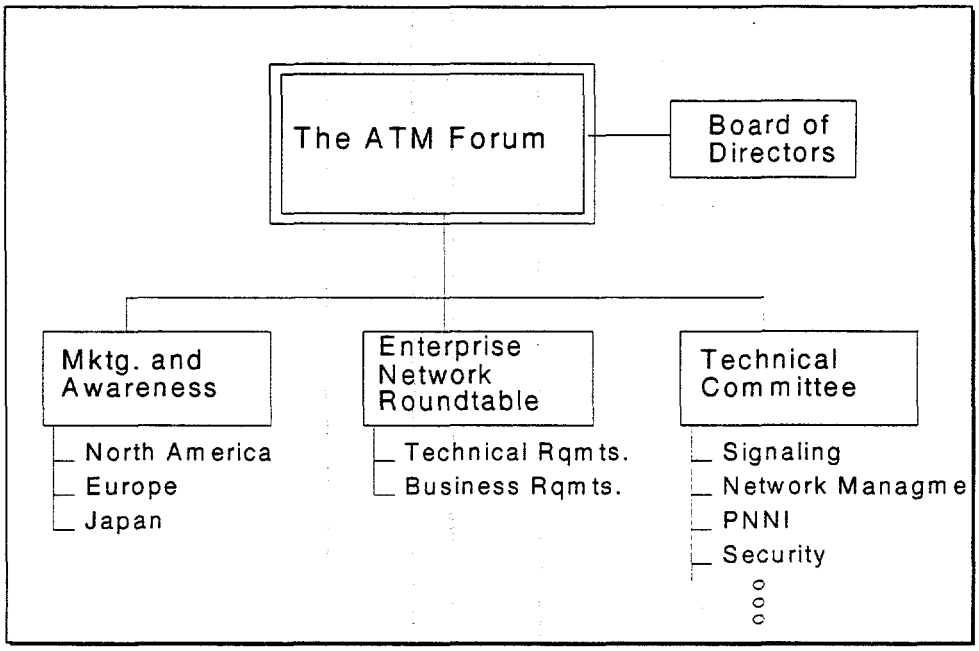


Figure 18: ATM Forum Organizational Structure

The purpose of the MAC is to give ATM visibility. Since the ATM Forum is a world-wide organization, the MAC has a number of subcommittees, each of which is associated with a particular region of the world (e.g. North America, Europe, Southeast Asia, etc.). The MAC supports its purpose of marketing and awareness through a number of channels. These channels include a monthly newsletter which provides current status on the ATM Forum and ATM technology, and the sponsorship of ATM tutorials, which provide excellent introductions into the various aspects of ATM technology (such as signaling, LAN emulation, and network management).

The ENR is the ATM Forum's users' committee. The ENR is the ATM Forum's newest committee, and was assumed by the Forum in August, 1993. The ENR is responsible for providing user input into the Technical Committee, which in turn, helps the TC to determine direction and requirements for its specifications. This responsibility is performed in a number of ways. The ENR actively writes white papers and user requirements analyses, which are submitted to the Technical Committee. In addition, at each meeting of the Technical Committee, the ENR sponsors joint ENR/TC sessions at which detailed user requirements are discussed with the appropriate working group within the TC.

Finally, the ATM Forum's Technical Committee is responsible for writing specifications for ATM equipment and protocols. The TC is not a standards committee *per se*, rather, it is a *specifications* body whose charter is to accelerate the deployment of interoperable ATM equipment. As such, it does not have ultimate authority with regard to international standards. However, the ATM Forum has a liaison

relationship the ITU, which is an accredited international standards body for ATM. Furthermore, since ATM vendors implement to many of its specifications, many of its specifications become *de facto* standards.

The ATM Forum's Technical committee is composed of a number of working groups, and its organization is partially shown in Figure 18. These working groups develop specifications for specific portions of the overall ATM architecture. For example, the signaling working group works on specifications that describe in detail message formats, protocols, and state machines to provide ATM services such as SVCs and multicast session establishment.

11.2. Sandia's Contribution to the ATM Forum

As an early adopter and major customer of ATM products and services, Sandia joined the ATM Forum as a Principle Member in 1992 to ensure that Sandia's needs as a user are known. In 1994, Sandia also became active in the Enterprise Network Roundtable, which is the users' branch of the ATM Forum. Our main goal in the ENR was to determine the level of interest with regard to ATM security, and help accelerate recognition of the users' need for security in the Technical Committee.

With the start of this project, Sandia became more directly involved in the Technical Committee with respect to security. At the same time this project started, a contribution was submitted to the Technical Committee by Xerox Corp. which outlined requirements and a methodology for authenticated ATM signaling [25]. When that contribution was introduced to the Signaling and Service Aspects and Applications (SAA) working groups, it was decided to defer consideration of security within ATM until the next meeting of the technical committee to determine if there were additional requirements from other sources (and hence, additional interest). At this next meeting (February, 1995), Sandia submitted a contribution which provided a "threat-asset" analysis, and requirements for ATM security to mitigate these threats. In the following meeting (April, 1995), Sandia presented its approach for implementing authenticated signaling. In subsequent meetings, additional contributions from other organizations (e.g. IBM, General Instrument, Network Systems, and DoD) provided additional requirements.

11.3. Security Working Group Activities

Up until June, 1995, the security work in the ATM Forum was considered a joint effort between the Service Aspects and Applications (SAA) and Signaling working groups. However, in June, IBM presented a proposal to the Technical Committee to form a new group for the security work. The rationale behind their proposal was that security is something that can affect many other working groups in the Technical Committee, and that there needed to be a focal point for security. It was decided at this meeting to form an "Ad-Hoc" working group to develop a security specification. However, in following meetings, a motion was brought forward to elevate the Security Ad-Hoc Group's status to full Working Group. After some debate, a vote was taken at the closing plenary session in the October, and the motion was passed.

As stated earlier, Sandia has made a number of contributions to the ATM Forum under this project. Our unique position within the technical committee as a user who understands the technical details of network security allowed us to contribute requirements, and back up those requirements with recommended solutions. As a result of this project, we have helped to form a "critical mass" of ATM Forum members who are willing to work on an ATM security specification. It is our intention to continue our work in the Technical Committee by contributing technically, and to continue contributing as editor of the Security Working Group's specifications.

12. Conclusion

As a result of this project, we believe that we were successful in our stated goals of developing ATM security protocols and mechanisms, transferring our knowledge to the ATM Forum, and establishing Sandia as a leader in the field of ATM security. We gained valuable experience and credibility in this field through our technical work, publications, and presentations of our results to the community. By contributing to the ATM Forum under the support of this project, we were successful (with the help of several other organizations) in the formation of the ATM Forum's Security Working Group which will incorporate our work into an industry-wide specification. We intend to continue Sandia's leading role in this field by contributing to this working group's technical specifications and operation.

13. References

[1] Randall Atkinson, Naval Research Laboratory. *Security Architectures for the Internet Protocol.* RFC 1825. August, 1995.

[2] The ATM Forum, *User-Network Interface (UNI) Specification, Version 3.1*, September, 1994.

[3] William R. Cheswick and Steven M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1994.

[4] James Arlin Cooper, *Computer and Communications Security*, McGraw-Hill Book Co., New York, 1989.

[5] Whitfield Diffie and Martin E. Hellman, *New Directions in Cryptography*, in IEEE Transactions on Information Theory, vol. IT-22, pp. 644-654, 1976.

[6] FORE Systems, Inc., *ForeRunner ASX-200 ATM Switch User's Manual*, Warrendale, PA, July, 1995.

[7] Helen Fouche Gaines, *Elementary Cryptanalysis*, American Photographic Publishing Co., Boston, 1939.

[8] Donald Graft, Mohnish Pabrai, and Uday Pabrai, *Methodology for Network Security Design*, in IEEE Communications Magazine, vol. 28, pp. 52-58, 1990.

[9] David Kahn, *The Codebreakers*, Macmillan Publishing Co., New York, 1967.

[10] Gene H. Kim and Eugene H. Spafford, *The Design and Implementation of Tripwire: A File System Integrity Checker*, in Proceedings, Second Annual Conference on Computer and Communications Security, 1994.

[11] Naval Research Laboratory, *VINCE 1.0: Application Programmers Interface (API) Manual*, Washington, DC, January, 1995.

[12] NIST, *Data Encryption Standard*, FIPS PUB 46, 1977.

[13] NIST, *DES Modes of Operation*, FIPS PUB 81, 1980.

[14] NIST, *Digital Signature Standard (DSS)*, FIPS PUB 186, May 19, 1994.

[15] M. A. Padlipsky, et al., *Limitations of End-to-End Encryption in Secure Computer Networks*, MTR-3592, Mitre Corporation, Bedford, Massachusetts, 1978.

[16] Craig Partridge, *Gigabit Networking*, Addison-Wesley Publishing Company, Reading, MA, December, 1993.

[17] Lyndon G. Pierson and Thomas D. Tarman. *Requirements for Security Signaling*. ATM Forum 95-0137. April, 1995. SAND95-0486C.

[18] Lyndon G. Pierson and Edward L. Witzke, *Data Encryption and the ISO Model for Open Systems Interconnection*, in ISE'84 Joint Proceedings, held in Albuquerque, New Mexico, May 2-4, 1984. Institute of Electrical and Electronic Engineers, New York, 1984.

[19] Lyndon G. Pierson and Edward L. Witzke, *A Security Methodology for Computer Networks*, in AT&T Technical Journal, vol. 67, pp. 28-36, 1988.

[20] Martin de Prycker, *Asynchronous Transfer Mode*, Ellis Horwood Ltd., New York, 1993.

[21] R. L. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, in Communications of the ACM, vol. 21, pp. 120 - 126, 1978.

[22] Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, New York, 1994.

[23] Winn Schwartau, *Information Warfare*, Thunder's Mouth Press, New York, 1994.

[24] Gustavus J. Simmons, (Editor), *Contemporary Cryptography - The Science of Information Integrity*, IEEE Press, New York, 1992.

[25] Ted Smith and John Stidd, Xerox Corporation. *Requirements and Methodology for Authenticated Signaling*. ATM Forum 94-1213. November 10, 1994.

[26] William Stallings, *Protect Your Privacy*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1995.

[27] Thomas D. Tarman. *A Proposed Generic Authentication Information Element*. ATM Forum 95-0460R1. August, 1995. SAND95-0665C (revised).

[28] Thomas D. Tarman. *Proposed DSS-Specific Fields for the Generic Authentication Information Element*. ATM Forum 95-0461R1. August, 1995. SAND95-0666C (revised).

[29] Edward L. Witzke, *Tools for Computer Network Security*, in Proceedings, Fifth Annual International Phoenix Conference on Computers and Communications, held in Scottsdale, Arizona, March 26-28, 1986. Institute of Electrical and Electronics Engineers, New York, 1986.

[30] Edward L. Witzke and Lyndon G. Pierson, 1994, *Key Management for Large Scale End-to-End Encryption*, in Proceedings, 28th Annual International Carnahan Conference on Security Technology, held in Albuquerque, New Mexico, October 12-14, 1994. Institute of Electrical and Electronics Engineers, New York, 1994.

14. Appendices

14.1. LDRD Proposal

LDRD PROPOSED WORK PROPOSAL COVER PAGE

Project Title: Protocol Extensions for Asynchronous Transfer Mode Security

Responsible Project Manager (PM): Michael Sjulin, 9417

Principal Investigator(s): Thomas Tarman, 9417, John Naegle, 1954

Note to Preparer: These three topics (two for new proposals) may be any length. However, they must fit in the space provided (this box).

Abstract (Nature of Work):
In support of the National Information Infrastructure (NII) initiative, which is expected to utilize Asynchronous Transfer Mode (ATM) technology, many vendors and standards committees are working hard to make ATM commercially viable. Although the ATM Forum and vendors of ATM equipment are making many technical strides toward making ATM a reality, the development of protocols and mechanisms for ATM security is conspicuously absent. This is distressing because ATM security is essential for the NII's success, since users will not utilize the NII's enormous potential if it is perceived to be a security risk. The research proposed here directly addresses this concern by developing protocols which provide mechanisms such as user-to-network authentication, security labels for ATM cells (applicable in multilevel secure ATM networks), support for end-to-end encryption over ATM networks, and the integration of light-weight encryption protocols within the ATM network adapter drivers. As these protocols are developed, they will be integrated into UNIX device drivers and switch control software, and will be evaluated to determine their effectiveness, transparency to the legitimate user, and compatibility with the existing standards. The results of this research will be documented, published, and presented to the ATM Forum and ATM vendors for adoption into the ATM standards and implementations. By developing these security protocols and providing laboratory results to back-up their effectiveness, it is expected that this research will have a significant impact on the rate of ATM deployment and its usage in the NII.

Work Proposed for Next Year:
Procurement of source code licenses and hardware for protocol development
Design and implementation of ATM security enhancement protocols
Implementation of light-weight embedded encryption
Verification of protocol performance in laboratory
Presentation of results to ATM Forum
Publication of SAND report

LDRD Data Input Form

Please ensure that an entry has been provided in every data field.

Proposal Number: _____ **Duration:** [x] 1 Year [ ] 2 Year [ ] 3 Year **Classified?:** [x] No [ ] Yes **Involves Living Subjects?:** [x] No [ ] Yes

(Leave blank, can only be assigned by the LDRD Office)

If Renewal Proposal: [ ] *2nd Year - (Go to Proposal Title)* [ ] *3rd Year - (Go to Proposal Title)*

Do you wish to give an Oral Presentation for this Proposal? [ ] No [x] Yes **Will it be classified?** [x] No [ ] Yes

Is this an NPRD proposal? [x] No - *(Select a Technology Area and Program Area)*

[ ] Yes - *(Select one)* [ ] Novel Projects [ ] Emerging Technologies

(Go to Proposal Title)

Technology Area *(Check only one)*

- [ ] Computational, Computer, & Mathematical Science
- [ ] Manufacturing Systems
- [ ] Structural Materials Development
- [ ] Solid State Sciences & Technology
- [ ] Engineering Mechanics
- [x] Information Technologies
- [ ] Engineered Systems and Devices
- [ ] Non Structural Materials Development
- [ ] Large-Scale Systems Analysis, Design, and Integration
- [ ] Environmental Sciences

Program Area *(Check only one)*

- [ ] Engineered Processes and Materials
- [ ] Microelectronics and Photonics
- [ ] Integrated Capabilities
- [ ] Electronics
- [ ] National Security Technology
- [ ] Counter-proliferation
- [ ] Biomedical Engineering
- [ ] Computational and Information Sciences
- [ ] Engineering Sciences
- [x] Information Science and Technology
- [ ] Advanced Manufacturing Technologies
- [ ] Energy & Environmental Science and Technology
- [ ] Transportation

Proposal Title: Protocol Extensions for Asynchronous Transfer Mode Security

Lead Principal Investigator: Thomas D. Tarman Org: 9417 E#: 59659

Project (Subcase) Manager: Michael R. Sjulin Org: 9417 E#: 23995

Funding Requested: (From LDRD Cost Estimate Worksheet)

	Total ($)	Manpower (FTE)	Total Labor ($)	Total DC ($)	Total SC ($)	Total Assessments ($)	Inflation ($)
FY95	536K	2.4	360K	122K		28K	26K
FY96							
FY97							
TOTAL							

Breakdown by Center for FY95:

Center #	Total ($)	Manpower (FTE)	Total Labor ($)	Total DC ($)	Total SC ($)	Total Assessments ($)	Inflation ($)
9400	325K	1.2	180K	112K		17K	16K
1900	211K	1.2	180K	10K		11K	10K

Scientific and Technical Soundness

It is expected that as the National Information Infrastructure (NII) expands, it will be used more for accessing sensitive information such as medical records, financial transactions, and business information. Since it is becoming apparent that Asynchronous Transfer Mode (ATM) will be the primary data communications technology that will make the NII a reality, there must be a set of mechanisms that will enhance the security of ATM. Without providing such security mechanisms, the productivity gains that are expected of the NII will not be realized because potential users will see the NII as a security risk. On the other hand, the normal user that expects a reasonable level of privacy does not want to purchase expensive equipment or follow complicated, awkward procedures to comply with his particular security requirements. To solve this problem, a set of simple, yet effective mechanisms are required at the ATM, ATM Adaptation Layer (AAL), and/or ATM Operations and Management (OAM) levels which provide **reasonable** levels of security in which the user may place his confidence.

The security issues associated with ATM are fundamentally different from those associated with conventional legacy networks. Where legacy networks such as Ethernet and Fiber Distributed Data Interface (FDDI) make information available to all nodes which are connected to a given segment, ATM networks employ virtual circuit switching, in which a cell of information is only seen by the switching equipment and the receiving node. Although this is a step forward toward network security, this still leaves several unresolved security questions, including "How do I know the network provider is legitimate?", "Is this user, who is connecting to my network, attempting to commit toll-fraud?", "Can I place multiple security levels of information on my local ATM network?", and "What if someone else receives this information?".

To address these questions, there are several mechanisms that should be built into the (still evolving) ATM standards. To provide adequate security for individual and corporate users of the NII, an ATM network must provide the following:

- user/network authentication
- secure ATM virtual circuit establishment
- security labels for ATM cells
- lightweight, embedded encryption within the ATM network interface device drivers

User/network authentication is considered here because a mobile user may want assurance that the network provider is who he claims to be. Conversely, user/network authentication also allows the network service provider to verify that the users that are attached to his network are legitimate, paying customers. This problem can be addressed cell-by-cell or at the beginning of each session through techniques such as private and public key encryption, and non-linear cryptographic functions similar to those pioneered in previous LDRD-funded end-to-end encryption research.

By providing additional signaling capabilities for the connection management entities in ATM switches and nodes, a user can be assured that the data path from source to destination is secure (i.e. data does not flow through a foreign country, or through an undesirable service provider's facilities). Furthermore, such signaling can be used to determine which virtual circuits carry data encrypted under certain keys. By providing this capability, encryption devices which maintain session keys can be effectively supported.

If a company wishes to use an ATM LAN to carry information at multiple levels or compartments of data security, each cell must be labeled to ensure that trusted systems only examine cells that contain data that they are authorized to see. Although labeling standards such as RFC 1108, RIPSO, CIPSO, and MaxSix exist today, these standards are designed for networks where IP packets can be sized to fit within the

Maximum Transmission Unit (MTU) of the underlying network technology (such as Ethernet or FDDI). However, ATM requires fragmentation of IP frames into the 48-byte cell size. This effectively destroys any labeling that occurs at the IP layer, as there is currently no mechanism for transferring the security label to the individual cells.

In the event that a user requires privacy as his information traverses the NII, encryption must be used. Although research efforts are currently underway which address ATM end-to-end encryption, these approaches rely on external hardware, which can be prohibitively expensive and difficult to use for users who only require a moderate amount of protection. It is believed that encryption which is integrated into the ATM network adapter device driver, and makes use of the connection management and encryption support protocols mentioned above, is easier to use and will see wider utilization by typical NII users.

To address these issues, a set of protocols which fit within the current ATM protocol suite, and software modules which implement these protocols will be developed. To implement these mechanisms, source code licenses for ATM device drivers and ATM switch control and connection management will be procured. Once these protocols are conceived, they will be implemented in the lab, evaluated for performance and utility, and documented for submission to the ATM Forum. Since these protocols will be extensions to the current ATM standards, interoperability between secure ATM and regular ATM in non-secure configurations must be considered at the protocol design phase, and validated in the laboratory.

The ATM Forum is the primary vehicle in the United States for developing ATM standards. The typical approach for defining a standard in the ATM Forum is to develop the technology using the member company's resources, and present the results of the research to the ATM Forum to include in their standards deliberations. To the authors' (who are active in the ATM Forum) knowledge, there is no work occurring in the ATM Forum which addresses ATM security at this time. Therefore, it is expected that the technical advances arising from this work will be met with much enthusiasm in the ATM vendor and user community.

Creativity and Innovation

The provision of security in ATM networks is not currently being addressed by the ATM Forum because there is very little work in industry in this area. Although some ATM equipment vendors claim security is built-into their architecture, they usually base these claims on the circuit-switching nature of ATM, and this argument is typically supplemented with some kind of auditing capability in the switch. However, these mechanisms are clearly not enough, even for users with moderate security requirements, let alone users with multilevel network security requirements.

To address the moderate security and performance needs of the typical NII user, security enhancement protocols within the existing ATM standards, and support for end-to-end encryption must be embedded in the workstation and ATM switching software. This approach is anticipated to support external end-to-end encryption devices as well as embedded software encryption. Support of software encryption (particularly below the application layer) is important for the majority of users who are constrained by cost and do not need the strength of encryption required for Type 1 (e.g. Government) data security.

It is very probable that a set of security mechanisms can be developed which will address these needs. However, these mechanisms must meet with the approval of the ATM Forum if they will see widespread use. This means that it is imperative that these protocols and mechanisms must mesh well with the current set of standards, if this research is to achieve widespread acceptance and success. At this time, it is not clear how ATM security protocols and mechanisms can be effectively designed under this constraint. Our involvement with the ATM Forum should provide much-needed guidance in this area.

Project Plan

Schedule:

1Q95	Procure source code licenses for Solaris, ATM adapter device driver, and switch control
2Q95	Design protocols for authentication, encryption support, and cell labeling.
2Q95	Implement lightweight embedded encryption.
3Q95	Implement authentication, encryption support, and cell labeling protocols.
3Q95	Verify protocol performance and modify protocols, if necessary
4Q95	Perform final performance studies and document results
4Q95	Present protocol design and performance results to ATM Forum
4Q95	Produce SAND report

Milestones:

12/31/94	Source code licenses obtained
3/31/95	Document describing protocol requirements and initial design produced
3/31/95	Embedded encryption implemented
6/30/95	Protocol implementation complete
4Q95	Performance results presented to ATM Forum
9/30/95	SAND report published

Staffing:

Tom Tarman, Lead Principal Investigator, 9417
Contribution: Project management, software design and implementation, liaison with Enterprise Networking Roundtable Security Requirements Task Group
Biography: Four years of network design experience for security-sensitive applications, chairman of the ATM Forum's Enterprise Networking Roundtable Security Task Team.

John Naegle, Principal Investigator, 1954
Contribution: Protocol design, protocol implementation, liaison with ATM Forum
Biography: Five years of design, implementation, and support of advanced corporate networks, extensive experience in security concerns of legacy (i.e. Ethernet, FDDI) networks, in depth knowledge of ATM equipment and capabilities, member and participant in the ATM Forum.

Budget:

FY95 FTEs Tarman (0.5), Naegle (0.5), 9400 staff (0.2), 1900 staff (0.2)
Total FTEs 1.4

FY95 DCs	Source code licenses	75K
	Hardware	30K
	Travel	10K
	Supplies and Misc. Support	7K
Total DCs		122K

DC Justification:

Source code licenses for the Solaris operating system, ATM switch control, and device drivers must be purchased to allow modification of protocols for the proposed security mechanisms. Hardware is required for software development, ATM interfacing, and laboratory validation of protocols. Travel is required to maintain dialog with ATM Forum, the Enterprise Networking Roundtable, and ATM equipment developers for collaboration and timely knowledge of changing standards and users' security requirements. Supplies and miscellaneous support are required for routine day-to-day operations.

Impact

The implementation of protocols that enable secure usage of ATM networks will have a profound impact on the success of ATM in applications such as the NII, which is expected to be utilized by a diverse set of users, including Sandia, in support of individual and corporate information requirements. Up until now, integrated security protocols for ATM networks have not been addressed by the ATM Forum and ATM vendors. Rather, ATM network security has been considered an "added-on option" to be addressed by the applications. However, this approach is cumbersome, and allows applications software developers to implement their own security mechanisms in different ways. This approach makes security from the user perspective awkward at best, and may cause the user to disregard these proprietary, applications-level mechanisms altogether. With its expertise with ATM network implementations, protocols, and generic network security, Sandia is appropriately postured to assume the task of developing protocols which seamlessly implement security for ATM networks. By successfully completing this proposed research, Sandia will receive much recognition for this pioneering work (due largely to the high visibility of ATM), and will have a unique position by having in-depth, valuable experience in ATM network security.

Signatures

Thomas Tarman, 9417	John Naegle, 1954
Michael Sjulin, 9417	Michael Eaton, 9400

14.2. FY 1995 Progress Report

Case Number	3517.190
Project Title	Protocol Extensions for ATM Security
Project Manager	M. R. Sjulin, 9417
Principal Investigators	T. D. Tarman, 9417, L. G. Pierson, 4616

Abstract - ***A descriptive project abstract (100-200 words)***

The purpose of this project was to develop protocols and mechanisms to provide enhanced security assurances for Asynchronous Transfer Mode (ATM) network applications, and to actively work with the ATM Forum to accelerate the development of security specifications for ATM. This project examined many aspects of ATM network security such as authentication, key exchange and management, and encryption (in hardware and software), and successfully developed mechanisms and protocols which implement these features. ATM signaling protocols and data objects that implement these security extensions were proposed to the ATM Forum for standardization, when appropriate.

FY95 Accomplishments - *(150-400 words)*

Asynchronous Transfer Mode (ATM) is a data, voice, and video communications technology that is being rapidly developed by network equipment vendors, communications providers, and specifications bodies such as the ATM Forum. Although the goal of integrated networking makes ATM attractive for applications such as the National Information Infrastructure (NII), the development of the technology with little or no regard for security will limit its trustworthiness (and hence, usefulness) in large-scale applications. At the start of this project, the ATM Forum (a specifications body for ATM) did not formally consider security when developing their specifications. This was unfortunate in two specific respects:

1. If security was required by an application, then it must be performed above the ATM layer. This excluded network security assurances for "native ATM" applications, that is, applications that directly use ATM services.
2. Further delay in the development of security extensions for ATM could mean that future security would be "added onto" rather than "integrated into" the ATM specifications as they become more mature. This could lead to a sub-optimal security solution.

The purpose of this LDRD project was to accelerate the development of security extensions into the evolving ATM protocol specifications, and lend strength to our proposals through the implementation of our solutions in working prototypes.

A number of ATM security mechanisms were implemented as a result of this project. Early in the project, authenticated ATM signaling was identified as a necessary mechanism for security services such as access control, encryption, and efficient (i.e. scalable) key management. A method which gracefully inserted authentication information into existing ATM signaling messages and protocols was developed which allowed one of a number of digital signature algorithms to be used. The initial laboratory implementation used the Digital Signature Algorithm (DSA), and its implementation considerations and performance were evaluated. Methods to provide the required signaling for ATM encryption devices were also investigated. A scheme which uses ATM Operations and Maintenance (OAM) cells was recommended and implemented in an existing hardware-base ATM encryptor, and in a device driver-based ATM encryptor which was also developed for this project. Finally, a key management framework for ATM was developed. This framework used a key management "information element" along with authenticated signaling to perform key negotiation between endpoints in a manner similar to that proposed by Diffie and Hellman. Throughout this project, Sandia has taken a leading role in the establishment of a security working group in the ATM Forum, and has contributed technically to this working group, particularly in the area of authenticated signaling.

This project was successful both in the development of ATM security mechanisms and in the acceleration of development of ATM security specifications. This project was able to show that ATM security extensions can be gracefully implemented into the existing ATM protocols and messages. In addition, Sandia's contributions to the ATM Forum regarding security requirements and implementations helped to motivate the need to establish a formal Security Working Group with the charter to develop a specification for ATM security extensions.

Refereed publications resulting from the work: Required info is as follows: Author(s), title of article, publisher, where published, volume number, page numbers, date of publication (mo/yr) *(see attached bibliographic reference sheet for more detail)*
none.

All other publications resulting from the work: Required info is as follows: Author(s), title of article, name of conference where paper was presented, date of presentation (mo/day/yr), location (city, state, country), volume number, page numbers *(see attached bibliographic reference sheet for more detail)*
Pierson, Lyndon G., and Thomas D. Tarman. 1995. "Requirements for Security Signaling (ATM Forum/95-0137)." Presented at the ATM Forum, Denver, Colorado, April 10, 1995. SAND95-0486C. L. G. Pierson and E. Witzke. 1994. "Key Management for Large Scale End to End Encryption.", *Proceedings of the 28th Annual 1994 International Carnahan Conference on Security Technology*. August 1994. L. G. Pierson. "Scalable ATM Encryption". 1995 Spring Proceeding, Cray User Group, Denver, Co., March 1995. L. G. Pierson. "Scalable ATM Encryption Test Results". 1995 Fall Proceeding, Cray User Group, Fairbanks, AK., September 1995. SAND95-2134c L. G. Pierson. "Integrating End-to-End Encryption and Authentication Technology into Broadband Networks", *SPIE International Society for Optical Engineering Photonics East '95 Symposium*, Philadelphia, PA, October 24, 1995. SAND95-2285c. Tarman, Thomas D. 1995. "Security for Asynchronous Transfer Mode Networks." Presented at the *Eighth Annual Conference on Next Generation Networks*, Washington, DC, November 7-10, 1994. SAND94-2221C. Tarman, Thomas D. 1995. "Protocol Extensions for Asynchronous Transfer Mode Security." Presented at the *IEEE Workshop on Interconnections within High Speed Digital Systems*, Santa Fe, New Mexico, May 14, 1995. SAND95-0852C. Tarman, Thomas D. 1995. "A Proposed Generic Authentication Information Element (ATM Forum/95-0460R1)." Presented at the ATM Forum, Toronto, Canada, August 6, 1995. SAND95-0665C (revised). Tarman, Thomas D. 1995. "Proposed DSS-Specific Fields for the Generic Authentication Information Element (ATM Forum/95-0461R1)." Presented at the ATM Forum, Toronto, Canada, August 6, 1995.

Number of patent disclosures: (where the invention was at least in part attributable to LDRD support)	0
Number of patent applications: (where the invention was at least in part attributable to LDRD support)	0

Number of patents: (where the invention was at least in part attributable to LDRD support)	0
Number of copyrights on computer software: (where the code was at least in part attributable to LDRD support)	0
Number of students: (if any) supported by the project	1 - Jed Greene, Northwestern University
Number of post docs: (if any) supported by the project	0
Number of permanent technical or scientific staff hired: (if any) supported by the project	0
Number of awards (and their names): by organizations outside the laboratory to an individual or team attributed at least in part to LDRD support	0
Number of new non-LDRD funded projects: their amounts and source of funding	

Your qualitative assessment about the completion of your milestones for the year in percent	100 %

Your qualitative assessment about the direction of the project as a result of research or other findings (Please place an X over the number of the statement that best describes your results)

X	Goals met, hypothesis proved
2	Goals partially met, hypothesis modified
3	Goals substantially modified, hypothesis redefined
4	Goals not met, hypothesis disproved
5	Project terminated because:

14.3. UNI Security Library

The purpose of the UNI Security Library is to provide APIs to VINCE developers and protocol designers which adapt the DSA code from Vicky Hamilton to the VINCE environment. This library consists of the following files:

vince_1.0.1/auth/libunisec.a	UNI security library
vince_1.0.1/auth/uni_auth.h	Header file containing function prototypes and macros for UNI authentication
vince_1.0.1/auth/uni_key.h	Header file containing function prototypes and macros for UNI key exchange

In addition to the files listed above, a test program can be found in "vince_1.0.1/auth/test.c". This test program shows how these APIs can be used to append key exchange and authentication information elements to a standard UNI message, and shows how to use these IEs to authenticate the UNI message and to transport session keys. The header in ".../test.c" describes how to build and run the test program.

Interfaces and descriptions of the APIs can be found on the following "man"-style pages.

NAME

initAuth - Initialize DSS engine for generating Authentication IEs

SYNOPSIS

```
#include "uni_auth.h"

void initAuth(p_buf, q_buf, g_buf, x_buf, k_buf, def_p, def_q, def_g,
              x, k, s1, s2, my_y)

unsigned char *p_buf, *q_buf, *g_buf, *x_buf, *k_buf;
mp *def_p, *def_q, *def_g, *x, *k, *s1, *s1, *my_y;
```

DESCRIPTION

initAuth() initializes the DSS signature generation engine for message authentication and key exchange, and MUST be called before genAuthIE() and genKeyIE(). initAuth() takes as input the ASCII strings p_buf, q_buf, g_buf, x_buf, and k_buf (x_buf contains the private key, and k_buf contains a random number). Upon invocation, initAuth() allocates memory for the mp structures, initializes def_p, def_q, def_g, x, and k with the values contained in the ASCII strings, inserts dummy values for s1 and s2, computes the public key, and inserts this station's public key value in my_y. (def_* should be the "default" authentication parameters for the workgroup).

To avoid memory leaks, cleanupAuth() should be called when the mp parameters are no longer needed.

RETURN VALUES

none

BUGS

Error checking should be performed, but isn't. Also assumes for the moment that the DSS parameters are standard length. (most of these routines make the latter assumption).

SEE ALSO

genAuthIE(), genKeyIE(), cleanupAuth()

NAME

genAuthIE - Generate an authentication information element in buffer (raw) format

SYNOPSIS

```
#include "uni_auth.h"

int genAuthIE(auth_ie, mesg_buf, mesg_len, elem_list, elem_list_len,
              def_p, def_q, def_g, my_y)

unsigned char *auth_ie, *mesg_buf;
int mesg_len;
int * elem_list;
int elem_list_len;
mp *def_p, *def_q, *def_g, *my_y;
```

DESCRIPTION

genAuthIE() builds a DSS authentication information element, converts it to raw format suitable for ATM signaling, and returns it in auth_ie parameter. genAuthIE() takes as input the UNI message buffer, in raw format, to be authenticated (mesg_buf), the length of the UNI message buffer (mesg_len), a list of information element codes which identify the IEs that are to be included in the digital signature (elem_list), the length of the element list (elem_list_len), and the DSS parameters to be used in computing the signature (def_p, def_q, def_g, my_y).

To initialize the digital signature engine, initAuth() must be called first, using the same def_p, def_q, def_g, and my_y which are used here. Also, it is assumed that auth_ie is already malloc'd with sufficient space to handle the contents of the IE generated by this routine.

RETURN VALUES

genAuthIE() returns the length of the auth_ie data on success. On failure, genAuthIE returns -1.

BUGS

SEE ALSO

initAuth()

NAME

genKeyIE - Generate a key exchange information element in buffer (raw) format

SYNOPSIS

```
#include "uni_key.h"

int genKeyIE(key_ie, pt_key, ct_key, key_len, def_p, def_q, def_g,
             my_y)

unsigned char *key_ie;
unsigned char *pt_key, *ct_key;
int key_len;
mp *def_p, *def_q, *def_g, *my_y;
```

DESCRIPTION

genKeyIE() builds a key exchange information element, converts it to raw format suitable for ATM signaling, and returns it in the key_ie parameter. genKeyIE() takes as input the plaintext and encrypted keys (pt_key and ct_key, respectively), the length of the keys (key_len), and the DSS parameters to be used in computing the signature of the plaintext key (def_p, def_q, def_g, my_y).

To initialize the digital signature engine for key exchange, initAuth() must be called first, using the same def_p, def_q, def_g, and my_y which are used here. (this is usually the case, as it is a good idea to use UNI message authentication when doing key exchanges). Also, it is assumed that key_ie is already malloc'd with sufficient space to handle the contents of the IE generated by this routine.

RETURN VALUES

genKeyIE() returns the length of the key_ie data on success. On failure, genKeyIE returns -1.

BUGS

SEE ALSO

initAuth()

NAME

validateMesg - Validate a raw UNI message which contains an authentication information element.

SYNOPSIS

```
#include "uni_auth.h"

int validateMesg(mesg_buf, mesg_len, elem_list, elem_list_len, p, q,
                 g, their_y)

unsigned char *mesg_buf;
int mesg_len;
int *elem_list;
int elem_list_len;
mp *p, *q, *g, *their_y;
```

DESCRIPTION

validateMesg() parses the raw UNI message in mesg_buf (which is mesg_len bytes long) to the last authentication information element in the message. If the authentication information element is found, then the contents of the IE list in the authentication information element are compared against the IE list in elem_list (which has elem_list_len elements). If this checks out, then DSS validation is performed on the UNI message, and the result is returned to the calling routine.

An authentication information element is allowed to convey DSS parameters (p, q, g, and y) along with the authentication message. However, any (or all) of these parameters may be omitted. If so, then the parameters supplied in p, q, g, and their_y are used. Conversely, if any of these supplied parameters are not initialized, validateMesg() will alocate space for them, and initialize them with the parameter data in the authentication IE. NOTE: a problem can occur if the parameter is not initialized, AND the authentication IE does not contain this parameter (see BUGS below).

Finally, if multiple authentication elements are present in the UNI message, the last one is significant. This is done to support nested authentication. This is not necessarily contrary to UNI 3.1, Section 5.5.6.6.2 (which states that if an IE that should not be replicated has multiple instances, then the first instance is significant), because multiple authentications IEs are allowed.

RETURN VALUES

validateMesg() returns -1 if a validation processing error occurs. Such errors could indicate that the message does not contain an authentication information element, or the contents of elem_list are not consistent with the IE list in the authentication information element.

If no processing error occurs, then validateMesg() returns the Message Authentication Code (MAC). The MAC is non-zero if the message does not validate, or zero if the validation is successful.

BUGS

Currently, this routine does not function properly if it is called with an uninitialized parameter, and that parameter is missing from the authentication IE. Until this is fixed, make sure that authentication IEs carry ALL DSS parameters (genAuthIE() includes all DSS parameters in the authentication IEs it generates).

Error codes should be used to differentiate between processing errors.

NAME

validateKeyIE - Validate the contents of the key exchange IE.

SYNOPSIS

```
#include "uni_key.h"

int validateKeyIE(key_ie, key_ie_len, pt_key, key_len, p, q, g,
                  their_y)

unsigned char *key_ie;
int key_ie_len;
char *pt_key;
int key_len;
mp *p, *q, *g, *their_y;
```

DESCRIPTION

validateKeyIE() validates the plaintext key (pt_key) and the contents of the key exchange information element (key_ie) using the DSS parameters for the remote node, contained in p, q, g, and their_y.

Before calling validateKeyIE(), validateMesg() must be called first to validate the UNI message. This is required in order to properly initialize the DSS validation engine with the proper DSS parameters.

RETURN VALUES

validateKeyIE() returns -1 if a validation processing error occurs.

If no processing error occurs, then validateKeyIE() returns the Message Authentication Code (MAC). The MAC is non-zero if the message does not validate, or zero if the validation is successful.

BUGS

Error codes should be used to differentiate between processing errors.

SEE ALSO

validateMesg()

NAME

cleanupAuth - Deallocate multi-precision DSS variables.

SYNOPSIS

```
#include "uni_auth.h"

void cleanupAuth(p, q, g, x, k, s1, s2, y)

mp *p, *q, *g, *x, *k, *s1, *s2, *y;
```

DESCRIPTION

This routine deallocates the multi-precision DSS variables p, q, g, x, k, s1, s2, and y. To avoid memory leaks, this routine should be called when these variables are no longer needed.

RETURN VALUES

none

BUGS

SEE ALSO

NAME

appendIEtoMesgBuf - Append a "raw" information element to a "raw" UNI message.

SYNOPSIS

```
#include "uni_auth.h"

int appendIEtoMesg(mesg_buf, mesg_len, ie, ie_len)

unsigned char *mesg_buf;
int *mesg_len;
unsigned char *ie;
int ie_len;
```

DESCRIPTION

This routine appends the raw information element in ie to the raw UNI message in mesg_buf. The lengths of mesg_buf and ie are indicated by *mesg_len and ie_len, respectively. Once the IE is appended to the message, *mesg_len is updated with the length of the new message.

RETURN VALUES

appendIEtoMesg currently only returns zero. mesg_len is updated accordingly.

BUGS

SEE ALSO

NAME

extractAuthIEfromMesgBuf - Remove authentication information element(s) from a "raw" UNI message buffer, and return the last IE.

SYNOPSIS

```
#include "uni_auth.h"

int extractAuthIEfromMesgBuf(mesg_buf, mesg_len, auth_ie)

unsigned char *mesg_buf;
int *mesg_len;
unsigned char *auth_ie;
```

DESCRIPTION

This routine removes all authentication information elements from the "raw" UNI message in mesg_buf, and inserts the last one in auth_ie. The length of mesg_buf is indicated by *mesg_len. After removal of the authentication IE(s), *mesg_len is updated with the length of the new message.

RETURN VALUES

extractAuthIEfromMesgBuf() returns the length of auth_ie if mesg_buf contains at least one authentication IE, and returns -1 if no authentication IEs were found. mesg_len is updated accordingly.

BUGS

SEE ALSO

NAME

extractKeyIEfromMesgBuf - Remove key exchange information element(s) from a "raw" UNI message buffer, and return the last IE and encrypted key.

SYNOPSIS

```
#include "uni_key.h"

int extractKeyIEfromMesgBuf(mesg_buf, mesg_len, key_ie, ct_key,
                            ct_key_len)

unsigned char *mesg_buf;
int *mesg_len;
unsigned char *key_ie, *ct_key;
int *ct_key_len;
```

DESCRIPTION

This routine removes all key exchange information elements from the "raw" UNI message in mesg_buf, and inserts the last one in key_ie. The length of mesg_buf is indicated by *mesg_len. After removal of the key exchange IE(s), *mesg_len is updated with the length of the new message.

In addition, the encrypted key is extracted from the last key exchange IE, and is placed in ct_key, with ct_key_len denoting the length of the key in bytes.

RETURN VALUES

extractKeyIEfromMesgBuf() returns the length of key_ie if mesg_buf contains at least one key exchange IE, and returns -1 if no key exchange IEs were found. mesg_len is updated accordingly.

BUGS

SEE ALSO

14.4. VINCE Modification Log

```
------------------------------------------------------------------------
6/12/95

Modified ~/sba200/kontrol.c to support calls to encryption module.
------------------------------------------------------------------------
6/12/95

Modified ~/sba200/sba200_atm.c extensively to provide encryption and
encryption control.
------------------------------------------------------------------------
6/12/95

Wrote ~/kernel/sba200crypto.c to control crypto functions on sba200
------------------------------------------------------------------------
6/12/95

Modified ~/kernel/Makefile to build sba200crypto
------------------------------------------------------------------------
6/12/95

Added ~/sba200/encrypt.c
------------------------------------------------------------------------
6/19/95

Created ~/auth directory and put working test.c and Makefile files there
(note: to compile testq, chdir to ~/process/uni and do 'make test')
------------------------------------------------------------------------
7/6/95

Modified ~/uni/uni_interpret.c to build signaling stack on top of aal4
instead of aal5. This was done because our asx, for some reason,
generates CRC-10s for each cell, and overwrites the cell trailers with
this CRC (even when instructed not to do so).
------------------------------------------------------------------------
7/6/95

Modified ~/ilmi/ilmi_main.c to build signaling stack on top of aal4
instead of aal5.
------------------------------------------------------------------------
7/6/95

Modified ~/arp1577/arp1577.c to build signaling stack on top of aal4
instead of aal5.
------------------------------------------------------------------------
7/7/95

Modified ~/host/host_hardware.c to do aal5 crc32 in software instead of
hardware.
------------------------------------------------------------------------
8/2/95

Modified ~/uni/uni_interpret.c to initialize uni authentication in
vince.
------------------------------------------------------------------------
8/2/95
```

Modified ~/auth/pmssldef.h to comment-out the 'USER' macro (USER is used somewhere else in vince).

8/2/95

Modified ~/auth/syscnst.h to comment-out the 'DEBUG' macro (DEBUG is used somewhere else in vince).

8/2/95

Renamed FreeMessage() in ~/auth/{mem.c, memdef.h} to FreeMessage1() because FreeMessage conflicts with a routine in vince.

9/15/95

Modified ~/auth/secrequest.c and ~/auth/secserver.c. In both of these files, the fdopen() calls were removed, and read() calls were used instead of fgetc().

secrequest.c was modified to wait for confirmation from the server before moving on. This provides interprocess sync.

Since things are stream oriented, EOM is now determined by receipt of null character, whereas before these mods, EOM was determined by EOF.

9/27/95

Modified ~/uni/uni_interpret.c (specifically, send_uni_message) extensively. These modifications allow, in the case where an auth IE is generated, the use of a new storage buffer for the authenticated messages.

NOTE: this mod MAY cause a memory leak (see in-line comments for details).

9/27/95

Modified ~/uni/uni_decode.c, ~/uni/uni_interpret.c, and ~/uni/uni_open.c to allow the fore switch to pass an authentication IE from the input port to the output port. The code fragment in uni_decode.c is responsible for extracting the auth. IE. The code fragment in uni_open.c is responsible for setting a flag (pass_auth_ie) to tell the output port to get the auth. IE. The code fragment in uni_interpret.c is responsible for checking the state of 'pass_auth_ie', and obtaining the auth IE from the global buffer (rather than generating it) if true.

NOTE NOTE NOTE NOTE NOTE NOTE NOTE

Due to a bug _somewhere_, on cardassia, all references to pass_auth_ie, and code fragments which are conditional on pass_auth_ie being true must be disabled.

9/28/95

Modified method for obtaining 'obuf' in ~/uni/uni_decode.c

9/28/95

Modified ~/process/Makefile to simplify building of fore_vince, and version of fore_vince that induces auth errors.

9/28/95

Modified ~/nsscop/sscop.h to increase INTERVAL_no_response from 60000 msec (60 sec) to 600000 msec (600 sec). This was done because laverne takes more than 60 sec. to initialize the validation engine.

DISTRIBUTION:

1		Jeff Ingle National Security Agency Attn: R222, R&E 9800 Savage Rd. Ft. Meade, MD 20755-6000
1	MS 0431	S. G. Varnado
1	MS 0449	S. K. Fletcher
1	MS 0451	V. A. Hamilton
1	MS 0451	J. H. Moore
5	MS 0451	M. R. Sjulin
1	MS 0655	T. J. Draelos
1	MS 0806	Marylou Brazee (c/o Ed Witzke)
1	MS 0806	J. P. Brenkosh
1	MS 0806	S. A. Gossage
1	MS 0806	J. H. Naegle
5	MS 0806	L. G. Pierson
1	MS 0806	M. O. Vahle
5	MS 0806	E. L. Witzke
1	MS 0807	B. J. Jennings
1	MS 0811	C. D. Brown
1	MS 1109	A. L. Hale
1	MS 1110	K. S. McCurley
1	MS 1436	C. E. Meyers
1	MS 9011	P. W. Dean
2	MS 0100	Document Processing for DOE/OSTI, 7613-2
1	MS 0619	Print Media, 12615
5	MS 0899	Technical Library, 4414
1	MS 9018	Central Technical Files, 8523-2
10	MS 0451	T. D. Tarman