

Location-Independent Information Object Security

John Lowry

BBN Technologies

Abstract

Users are mobile and use multiple platforms with differing applications over time. These users have no security service applications which are independent of location or computing environment. The IOS project has developed syntax and applications which provide these services. The IOS Tools allow one or more users to apply security services to documents. The project has also investigated third-party security services and has developed an Electronic Signature Timestamp Server (ESTS) which provides the services of document registration (timestamping), validation, and non-repudiation.

1: Purpose and Motivation

This paper presents concepts developed as part of the Information Object Security (IOS) project. The IOS project investigates techniques for protecting information (e.g., files and documents) independently of any one application, protocol, or platform.¹ Current security approaches focus on protecting communications or protecting information in the context of a particular application association. IOS is considering techniques which allow a user to protect information independently of a protocol or application. Further, IOS defines a comprehensive approach for providing non-repudiation services including the use of trusted third-party services.

Workplace and home computing are moving rapidly toward a distributed environment. This movement is possible because of increasingly cheap and reliable networking computing power, most importantly in the

desktop and portable computing arena. In addition, people are cooperating and sharing in the creation and distribution of information.

One example where security services provide significant additional value is a work-group cooperating in the creation of a document. During the creation phase, members of the group may want to restrict knowledge of the document contents to members of the group and yet make it available to the members through a public mechanism like anonymous FTP. Once the document is complete, the members may want to publish the document with strong guarantees of the report's origin and integrity. They may also need to establish unambiguously the date and time for certain ideas within the document as well as the document itself.

The project has, to date, investigated and defined concepts critical to this purpose, developed a set of data definitions and techniques to provide security services, and developed a set of prototype application user tools and a third-party service provider.² The set of tools is collectively referred to as the IOS Tools. The data definitions are referred to as the IOS Components. The third-party service application is called the Electronic Signature Timestamp Server or ESTS.

¹The information presented here also appears in the first project quarterly report. (ARPA/IOS-1994/01 Quarterly Technical Report: Location Independent Information Object Security)

²Design and development tasks began in early 1994. This paper is being reviewed in November 1994 and is intended for presentation in February 1995. The most significant project milestone will occur in December 1994/January 1995 when the applications and services will be made available on the Internet. The paper assumes a time context of February 1995 and may present some things that are currently in development as if they were complete.

1.1: Application-layer Security Tools

The focus of the project is with the user. Several security mechanisms currently exist or are defined to protect information carried in lower layer protocols or as part of a particular application exchange. The security services provided by these mechanisms exist only in a limited context. The IOS Tools enable the user to act directly on data objects (documents) individually or in combination with other users in a fashion that is independent of any platform, location, or application protocol.

An important requirement is to support the individual user who is operating in a portable or nomadic environment. This user needs to be able to protect a document while in one location and be able to access that document or provide different protections at another location.

A survey of technologies on the Internet shows that there are no security applications geared to the goals stated above. Most of the security applications provide a particular service, for example, Privacy Enhanced Mail (PEM)³ for electronic mail security or secure FTP for file transfer security. Removing the data object from these application environments removes the associated security services as well.

Most applications available to the typical user (e.g., document generation, image processing, electronic mail, and file transfer applications) do not supply security services. The user can apply security services to his documents using the IOS Tools and still use untrusted applications for information processing and transfer.

1.2: Third-party Security Services

The goals of the IOS effort included the investigation of trusted third-party security services. The IOS project has developed a third-party service provider called the

³PEM - Privacy Enhancement for Internet Electronic Mail; RFCs 1421 to 1424 describe the syntax, semantics, and procedures for providing a large number of security services in the Internet electronic mail environment.

Electronic Signature Timestamp Server (ESTS). The ESTS is an application entity that operates on IOS Components and provides a variety of services. These services include time stamping, authentication and validation, and non-repudiation. Non-repudiation is the inability to deny having participated in a transaction. The definition implies that there are three participants, an originator, a recipient, and an adjudicator. The originator or recipient is seeking to demonstrate the repudiability (or non-repudiability) of a document to the adjudicator. The ESTS supports such a service through time and Certificate validation functions which determine whether or not the purported originator did sign the document. The time stamp function is a combination of a reliable and trusted time-source and digital signature technology. The ESTS requires a good time source (currently a Global Positioning System (GPS) receiver) as well as procedural and technical safeguards to ensure a trusted time. Document registration occurs when a user submits a document and requests the registration service. The ESTS registers the document by providing a time stamp value, associated control information, and a digital signature. The document as submitted and the time-stamp data are returned to the originator.

2: Requirements

2.1: Environment

The IOS Tools and the ESTS are intended to operate in the Internet environment in the context of distributed applications. Users may employ high-powered workstations, home personal computers, or nomadic notebook computers. The security mechanisms within IOS require access to cryptographic technology. The encryption services could be provided in software or in integrated or peripheral hardware (e.g., a PCMCIA card).

The IOS Tools obtain cryptographic services through an API that hides the details of the underlying cryptographic system. This feature enhances the portability of the IOS Tools to other systems and other cryptographic modules.

The initial implementation of IOS uses a software implementation of RSA and DES (RSAREF⁴).

2.2: System Requirements

The goals of the IOS Project include encouraging experimentation and use among a wide number of users. Consequently, the interfaces are as simple and portable as possible. The IOS Tools perform atomic security operations and transformations and are designed to be used sequentially either as part of a pipeline or through individual invocations to create complex objects of interest to the user. The tools can be called from within a UNIX™ shell-script or other batch processing language to ensure consistent generation and handling of complex objects.

The IOS Components are designed as building blocks. This allows combination of components to achieve multiple sets of security services. The flexibility of this approach frees the designers from attempting to identify all the uses and service combinations that users may require in the future.

An important system requirement is to make the software publicly and freely available. The software is a combination of sources written in C, C++, ASN.1 and shell-scripts. All of the software is compiled using GNU C(gcc) and C++(g++) and comes with an ASN.1 to C++ compiler that is itself compiled using C++⁵. The software and supporting documentation will be freely available from an Internet host.

2.3: Security Services

The IOS Tools provide security services and mechanisms ranging from confidentiality, integrity, and basic digital signatures, to more complex services like multiple signatures with annotations, and non-repudiation.

⁴RSAREF is a library of cryptographic software provided with a license for non-commercial use by RSA Data Security Inc. and is freely available on the Internet.

⁵Features and design of the ASN.1 compiler *asn_gen* are presented in the third project quarterly report. ([ARPA/IOS-1994/03 Quarterly Technical Report: Location Independent Information Object Security](#))

These services are built on a number of underlying primitives.

2.3.1: Key Management

Key management is used by IOS for keys used to provide confidentiality services. Key management is based on techniques found in PEM⁶ and other technologies. Data encrypting keys (DEKs) are generated internally to the application tools and cannot be specified or supplied by the user. DEKs are protected inside a token (a combination of data consisting of the DEK, a date, a data hash, and integrity information) to maintain both confidentiality and integrity. An integrity check-value of the token contents ensures token integrity. Symmetric or asymmetric key algorithms provide token confidentiality. Signature keys are asymmetric.

2.3.1.1: Certificate-based Authentication

Asymmetric keys are carried in certificates as specified in X.509⁷. X.509 certificates provide the necessary binding between a key and a named individual as well as a method to identify a key uniquely. Certificate chains are supported both in a hierarchical model as espoused in PEM, and transitive trust models (non-hierarchical cross-certification) as are currently being developed in X9.30⁸. Strong security services require employment of a strong trust model. Therefore, while other trust models can be accommodated within the IOS Component syntax (e.g., so-called "web-of-trust"), their use precludes the services of non-repudiation and data-origin authentication.

⁶ Key management techniques described in PEM are used and are similar to techniques used in other protocols like CCITT X.500 and X.400.

⁷X.509 - [CCITT Recommendation X.509, ISO/IEC 9594-8](#); Both the 1988 and 1993 versions of *Certificate* are acceptable as are most of the variants currently under consideration since the *Certificate* syntax and semantics from succeeding versions are compatible with previous versions.

⁸X9.30 - [Public Key Cryptography using Irreversible Algorithms for the Financial Services Industry, N18-94](#); The committee has, through a number of draft documents, been working to develop and refine alternative certificate trust models, in particular, in consideration of the needs of the wholesale banking community.

2.3.1.2: Algorithm Independence

The key management syntax accommodates RSA and DES and most other asymmetric and symmetric algorithms. Public keys are carried in X.509 certificates. Individual certificates for confidentiality and signature purposes can be accommodated as well as certificates that have subjectPublicKeys that combine the public keys used for confidentiality and signing.

2.3.2: Canonicalization and Data Integrity

All the data, both management information and the user data, are canonically encoded in order to provide a single representation and to provide platform independence. ASN.1⁹ meets that purpose and therefore that notation was chosen to define the IOS Components. Use of the Distinguished Encoding Rules¹⁰ (DER) is mandatory.

The IOS project has developed an ASN.1 to C++ compiler, sources to which are delivered and are available for distribution. All the ASN.1 specified objects in the IOS tools and ESTS were created using this compiler. A key design consideration of the compiler was to generate software which separates detailed consideration of the encoding requirements from the security services being explored and developed. The significant amount of work involved in encoding and decoding has been a major obstacle encountered in other projects using ASN.1 to specify data formats. This compiler generates object-oriented definitions which contain the functions necessary to encode and decode any defined object. Encoding a Certificate, for example, is as simple for the developer as making the statement *Certificate.encode()*; All the components of a definition are addressable using a rule set similar to standard C or C++ structure definitions. Consequently, the subjectPublicKey portion of a Certificate can be obtained with the statement *Certificate.tbs.subjectPublicKey.read()*; These features

⁹ASN.1 - Abstract Syntax Notation Number 1; CCITT Recommendations X.208 and X.209 (1988) specify the syntax and basic encoding rules for this notation.

¹⁰DER - Distinguished Encoding Rules as specified in CCITT X.509 (1988) § 8.7 are the constraints on the Basic Encoding Rules (BER) defined in X.209 which result in a canonical encoding.

and others allow programmers and application designers who are inexperienced with ASN.1 to work quickly and efficiently without getting bogged down in the minutiae of canonical encoding.

User data is encapsulated in a DataComponent. Data integrity is assured through the use of a unique identifier for each Component object generated and through the use of hash algorithms. The unique identifier allows the user to determine exactly which components should be evaluated. Hash values are protected through encryption or by application of a digital signature algorithm. The SignatureComponent defines the syntax for a signed object. Specified Components are input to the hash function and the hash is covered with a digital signature. All parts of the SignatureComponent (collectively called the signature control information) except the signature value are also protected with the signature hash. ConfidentialityComponents contain tokens generated for each intended recipient of the data. Tokens contain a hash of the plaintext encrypted data. The contents of the token (consisting of a DEK, hash and date) are also covered with an integrity check value which is inserted into the token prior to token encryption.

2.3.2.1: Signature Generation

Signatures are constructed by first identifying the IOS Components to be signed. The list of unique identifiers becomes part of the signature control information and identifies for the recipient (the signature verifier) the components covered by the signature and the order in which they should be evaluated. The method used to calculate the final hash value is critical to the ability to provide multiple parallel and sequential signatures. This method hashes each identifiable component and yet associates groups of components in the order they are evaluated. The hash used in the signature is calculated by hashing the first component and prepending the resulting hash value to the data of the second component to be hashed. Each component is hashed in turn, each with the hash value of the preceding component as the first data evaluated in calculating its hash. The final hash value is calculated as a function of the preceding hash value and

the signature control information. The final hash is input to the appropriate digital signature algorithm.

2.3.3: Authentication / Validation

Authentication is based on the user possessing a unique name which has been previously established. Each IOS Component contains a unique identifier, a part of which is a name form. A name form consists of a formal name and/or an informal name. Use of the formal name is encouraged and the syntax is defined as a Name as specified in X.500. This name form satisfies the requirements of uniqueness and canonical form. It also allows a simple matching of identifier and Certificate. The alternative form is an informal name which consists of any arbitrary string of octets. In some cases, in particular for the ESTS protocol header, the informal form is expected to be an RFC-822¹¹ e-mail address.

The name as presented in a SignatureComponent is used to help identify a suitable public key for signature validation purposes. Again, the DN is favored, and if there might be ambiguity, for example where the same DN appears in multiple Certificates, the originator has the option of including a CertificateComponent which contains a CertificationPath and a Certificate Revocation List (CRL).

Multiple signatures are a feature of this design. Signatures can be applied sequentially and in parallel. A sequential signature covers the data and all existing signatures in a nested fashion. The order of signature application is preserved. Initial signatures cannot be removed without detection unless all subsequent signatures are removed as well. Sequential signatures are well suited to a hierarchical authorization mechanism. Parallel signatures are applied only to the original data and do not cover other SignatureComponents.

Each SignatureComponent contains an optional time value which is intended to establish the date and time the signature was applied. This time value is part of the data included in calculating the signature hash. The time value

is included to help establish non-repudiation. Signatures can be repudiated because of claims that the keying material was compromised and because the certificate-based key management system is time sensitive, e.g., certificates have a validity period and revocations are time-based. Binding a time value to the signed data is a critical component to establishing non-repudiation. Originators would be expected to supply and verify the time that a document was signed before releasing the document. The time value field is where the ESTS places its time when registering a document.

2.3.4: Annotation

Annotations are carried in an AnnotationComponent. This component is analogous to a DataComponent but has a different syntax and is meaningful only when used with a SignatureComponent. The AnnotationComponent carries its own unique identifier, the identifier of the DataComponent it annotates, and the annotations. Annotations can be general or specific. General annotations are arbitrary strings wherein the context (if any) is established as part of content of the annotation. An example of a general annotation might be the comment "Take a look at paragraph three and explain why ...". Specific annotation formats are expandable in type and form and are used to express such references as specific row/column offsets into a document or references to digitized data, for example a digitized photograph.

2.3.5: Confidentiality

Confidentiality services are achieved through the use of symmetric and asymmetric algorithms. Data encryption is expected to be achieved through use of symmetric algorithms although various types and modes can be accommodated. Both block cipher and stream ciphers (with appropriate IV data) can be used and syntax exists to carry any algorithm parameters. Public key algorithms are too inefficient to use for bulk data encryption and are not explicitly supported in the syntax. In addition, each ConfidentialityComponent is expected to use a unique data encrypting key (DEK) which makes the difficulties of

¹¹RFC-822 - [Standard for the Format of ARPA Internet Text Messages](#), David H. Crocker, 1992.

using a public key algorithm almost insurmountable. The IOS Tools currently support DES.

Tokens are constructed by encrypting integrity information and the data encrypting key (DEK). Tokens can be encrypted using symmetric or asymmetric algorithms. Public key (asymmetric) algorithms are preferred but systems using secret keys, in particular those systems relying on mechanisms specified in X9.17¹² can also be supported.

Confidentiality is applied to ComponentLists. A ComponentList is a sequence of components identified by the originator. The identified components are constructed into a list and encrypted using a symmetric algorithm like DES. The encryption function gathers a hash (invisibly to the caller) which it stores in an internal register. When the originator is ready to generate tokens for the recipients, the token contents (data hash, DEK, date, token ICV) are encoded and then encrypted on a per-recipient basis. When the recipient decrypts the token, the token ICV is checked, the date is returned to the recipient, and the data hash and DEK are made available to the decryption function. When the recipient decrypts the data, a hash is recalculated and compared with the data hash obtained from the token. The recipient is presented with the result of the comparison when the last block of data is returned.

2.3.6: Access Control

Identity-based access control is supported. It is based on identity and enforced through use of confidentiality and certificate-based key management. The syntax is defined in the ConfidentialityComponent and the CertificateComponent. Identity-based access control (IBAC) has the flexibility to allow the controlled release of confidentiality tokens, either computed at the time the object was created and retained by the originator for dissemination later or computed later when the originator determines that another entity should be granted access.

Access control can be based on membership in a group which has knowledge of a secret key. An example is users

of X9.17-based systems. The ConfidentialityComponents using this key management system contain tokens encrypted with a symmetric algorithm.

Finally, the AccessControlComponent explores the area of non-cryptographic access control. The syntax allows specification of access control lists which both include and exclude either specific entities or groups having common elements in their names. The syntax supports specification of an access time period during which the information should be made available to the intended recipient. The originator can also express access permissions, e.g., read-only, execute, destroy, etc. These functions are of interest in operating systems and other third-party applications wherein a trusted third-party would enforce the access control rules by requiring an entity requesting access to submit a signed request which contains identifying information, i.e., a Certificate.

3: Non-repudiation Issues

The IOS Project addresses several issues in non-repudiation. The IOS applications share many characteristics with store-and-forward technology like PEM. There is a Certificate-based key management system, syntactic and semantic rules, and the same basic set of cryptographic services, in particular, digital signatures. Non-repudiation has the characteristic that it is intended as a proof, to be compelling evidence, to a third-party that one or more individuals engaged in a transaction. Using Certificate-based key management and digital signatures does not completely solve the problem.

3.1: Key Management

The first difficulty is that the user typically controls access to his private signature component and has the ability to carelessly, accidentally or purposely compromise that key. This is partially compensated with the periodic issuing of a Certificate Revocation List (CRL) which revokes the binding between a key and a named entity. Accurate CRLs depend on timely reporting of known or suspected key compromise. CRLs must also be accessible to all users who need them. CRL delivery can follow the

¹²X9.17 - [ANSI X9.17- 1985 Financial Institution Key Management \(Wholesale\)](#).

push or pull model but there is no explicit requirement that a CRL be issued whenever a new entry is available. Nor is there a requirement that, should a non-scheduled CRL be issued, all users be notified. Notification of all users may be impossible anyway.

The ESTS solves this problem by managing and maintaining a database of all valid Certificates and CRLs for CAs. It does not store user Certificates. One of the difficulties in deploying a Certificate-based key management system is the problem of Certificate and CRL distribution. The ESTS uses three methods to obtain these. The first is the Certificate and CRL retrieval methods specified in PEM and eventually in PEM/MIME. For the PEM approach, the ESTS requires that an RFC-822 e-mail address be associated with a CA. Insertion of CA Certificates which resolve to a PEM Policy Certification Authority (PCA) or the IPRA cause the ESTS operator to be queried for an e-mail address. The default e-mail address is the one associated with the PCA but can be overridden with an explicit entry. The second method is to use an X.500 directory service. The ESTS is capable of invoking a Directory User Agent (DUA) interface. The DUA is not managed by the ESTS but has its own configuration information. The third approach is called the "encounter" method. Certificates and CRLs are viewed as rare, precious, and difficult to obtain and when encountered are evaluated to determine if they are suitable for retention. Certificates and CRLs are encountered as a direct result of servicing user requests which include one or more Certificates or CRLs. A newly deployed ESTS typically has a sparse database and users can help fill that database by passing along any Certificates or CRLs they have with their requests.

The ESTS always responds with a status response to a request for non-repudiation. Non-repudiation will always require that the *next* CRL for each CA be obtained before validation and consequently there will always be a delay. The status response states whether the ESTS believes it can establish non-repudiation and when it believes it will be able to do so. The request is stored in a database along with a time when it should be finally evaluated. The ESTS then attempts to retrieve any and all Certificates and CRLs

needed to complete the servicing of the non-repudiation request.

Certificates suitable for non-repudiation must be valid at the time specified in the request. Any CRL must be the CRL issued after the time specified in the request to establish whether a Certificate was revoked. The time that the last CRL is expected to be available is the time the ESTS uses to determine when to finish servicing the non-repudiation request. There is a potential window of vulnerability which could lead to a false non-repudiation assertion. This condition exists when a Certificate expires prior to issuance of the next CRL. If the Certificate is revoked just prior to its expiration, the CA may choose not to list it on his CRL. Two approaches were identified to deal with this problem, the first is to require issuing of new Certificates before the old ones expire and to have the start date of the new Certificate be prior to the expiration date of the old Certificate. The Certificate holder is then expected to use the new Certificate as soon as it becomes valid. The ESTS accommodates this by storing any such "new" Certificates with a state of "not-yet-valid". The second approach is to require CAs to carry revoked Certificates, even ones that have expired, on the CRL issued after they expire. The ESTS accommodates this in its rule set for establishing non-repudiation by looking for the CRL issued after Certificate expiration and by warning the originator of the non-repudiation request about this condition in the response. The ESTS returns all Certificates and CRLs it used in making its determination with the non-repudiation response.

3.2: Time Context Establishment

The second difficulty is that most documents do not have a time context established. Non-repudiation services can be applied with or combined with proofs of origin, submission, delivery, and receipt. Each of these proofs requires a binding of a time context to a particular activity. Proof of origin requires a time stamp to register the document's existence. Proofs of submission, delivery, and receipt involve a transmission of the data and require that a time stamp be associated with that transmission. X.400

attempts to provide proof of submission and delivery in the Message Transfer Agent (MTA) on the assumption that the MTA is being operated by a trusted third-party (a national telephone company) rather than being under the control of the subscribers, which is how MTAs tend to be deployed in practice.

The IOS Components all carry room for a time value which can be used to make a claim about the time a document was originated or a signature was applied. The claim in this case is made by the originator of the document and the time value may be accidentally or maliciously inaccurate. The ESTS provides a trusted time stamp service for documents. This service is called registration because it registers a document or a document's hash as of a particular point in time. The originator of a document cannot obtain a time stamp in the past from the ESTS. The ESTS makes use of a Global Positioning System (GPS) receiver to obtain an accurate time which has little likelihood of being tampered with. The ESTS provides additional safeguards by associating a sequence number, a hash of the data, and the time stamp with each transaction. This information is stored in a list which is periodically signed and submitted to other ESTSs with a request for registration and non-repudiation.

Non-repudiation with proof of origin is the basic non-repudiation service offered by the ESTS. The originator of a request submits a digitally signed document and a time value to be used for the context. The ESTS uses the time submitted or defaults to the current time.

Non-repudiation involving a transaction (i.e., proofs of submission, delivery, and receipt) require that the ESTS be involved in the transaction. These services are experimental but work as follows. The originator of a request supplies an additional list of e-mail addresses to a registration request. The ESTS attaches a time stamp and Certificates and CRLs necessary to establish its non-repudiability and signs the document. The signed document is then forwarded to each of the recipients named by the originator. Finally, a copy of the document and a report on the processing is returned to the originator.

3.3: Document Semantics

The third-party seeking to establish non-repudiation requires that the semantics of the document be appropriate. An informal message, as is typically the content of e-mail, is not the same as a contract. A contract requires not only that the signatures be validated and a time context be established, but that it be apparent that the document is a contract. This semantic is supported by the syntax of a DataComponent which allows specification of the document type. The basic forms are pre-defined and specify encodings like binary data, and document syntax like PEM. The syntax allows expansion to support a wide range of formats from EDI to compressed video.

Digital signatures form a critical component of the semantic for many documents. In the case of the contract mentioned above or a purchase order or any number of business or official documents, multiple signatures must be accommodated. The IOS Tools allow the attachment of multiple sequential and parallel signatures and the attachment of annotations bound to an individual signer. These services are provided in the SignatureComponents and AnnotationComponents mentioned above.

4: Conclusions

The IOS Tools and ESTS supply security services to users engaged in a wide range of activities. These range from simple establishment of ownership of a document to complex activities like controlled release of confidential information. The user can take advantage of these tools on a wide variety of platforms.

Ongoing work includes continuing investigation of security services that are useful to the user. Among these are non-repudiation with proofs of submission, delivery, and receipt. The requirements of access-control in a distributed environment are being investigated, particularly in regards to the role a trusted third-party might play.

The software for the IOS Tools, the ESTS, and the ASN.1 compiler are expected to be available in January 1995 via anonymous ftp from the host *ftp.bbn.com*. The quarterly reports are currently available on this host.

The Electronic Signature Timestamp Server process is available at the email address *ests@bbn.com*.

5: Acknowledgments

The original concept for location-independent objects was developed by David Solo and are presented in the first and second quarterly reports; ARPA/IOS 1994/01 and ARPA/IOS 1994/02.

The original ideas regarding the necessity of establishing a time context for non-repudiation and development of other elements critical to non-repudiation are the work of Dr. Stephen Kent. Significant additional information on this topic is available in the first quarterly report; ARPA/IOS 1994/01.

This paper is sponsored by the Advanced Research Projects Agency / CSTO under ARPA Order No. A787 issued by ESC/ENS under contract #F198628-93-C-0192.