Telecommunication Standardization Sector AVC - 1018
Original: English

Experts Group for Video Coding and Systems in ATM and Other Network Environments **September 24-27, 1996**

STUDY GROUP 15
CONTRIBUTION

Source: Jim Toga, Intel Corporation
email: jtoga@ibeam.intel.com
voice: +1 (503) 264-8816
fax: +1 (503) 264-3485

Prakash Iyer, Intel Corporation
email: Prakash_Iyer@ccm.jf.intel.com
voice: +1 (503) 264-1815

Anand Rajan, Intel Corporation
email: Anand_Rajan@ccm.jf.intel.com
voice: +1 (503) 264-7547

Title: Proposal for H.323 Security

Date: September 16, 1996

Framework For Security Extensions To H.323-based Networked Multimedia Applications

0.0 Scope

This document proposes enhancements within the framework of the ITU H.323 specification, to incorporate security features such as *Authentication* and *Data Encryption*. The proposal does not describe specific H.323 PDU format changes, but suggests where and how they may be implemented. The proposed scheme is scaleable for multi-point conferences, but details have not been worked out.

1.0 Introduction

Secure communication over insecure networks generally involves two major areas of concern – *privacy* and *authentication*.

Privacy indicates the desire to keep anyone except the intended recipient from being able to read the message. Accomplishing this usually involves some type of data encryption/decryption, otherwise known as cryptography.

Authentication is the need to know the entity with whom you are communicating is, in fact, who you think it is. Perhaps the most common method of accomplishing this is through the use of *certificates*. A certificate is a set of data that completely identifies an entity, and is issued by a *Certification Authority* (CA) only after that Authority has verified that the entity is who it says it is.

2.0 Mechanisms To Implement Authentication And Data Encryption

The three main components of security are : Authentication, Key Exchange and Encryption algorithms. There are several implementations of schemes to address each of these areas. In this section are brief descriptions of some of the mechanisms available to implement support for these components.

2.1 Authentication Mechanisms

A very important step in establishing secure network communications is authentication of the entity at the other end of the communication. Many authentication mechanisms are available. Authentication mechanisms fall into two categories of strength - weak and strong. Several mechanisms are available to enforce and implement authentication. Examples include:

2.1.0 Digital Certificates / Digital Signatures

Digital signatures, such as the *Digital Signature Standard* (DSS) and the *Rivest-Shamir-Adleman* (RSA) signature, are public key based strong authentication mechanisms. When using public key digital signatures each entity requires a public key and a private key. Certificates are an essential part of a digital signature authentication mechanism. Certificates bind a specific entity's identity (be it host, network, user, or application) to its public keys and possibly other security-related information such as privileges, clearances, and compartments. Authentication based on digital signatures requires a trusted third party or certificate authority to create, sign and properly distribute certificates. For more detailed information on digital signatures, such as DSS and RSA, and certificates see [Schneier].

2.1.0.0 Certificate Authorities

Certificates require an infrastructure for generation, verification, management and distribution. The Internet Policy Registration Authority (IPRA) [RFC-1422] has been established to direct this infrastructure for the IETF. The IPRA certifies Policy Certification Authorities (PCA). PCAs control Certificate Authorities (CA) which certify users and subordinate entities. Current certificate related work includes the Domain Name System (DNS) Security Extensions [DNSSEC] which will provide signed entity keys in the DNS. The Public Key Infrastructure (PKIX) working group is specifying an Internet profile for X.509 certificates. There is also work going on in industry to develop X.500 Directory Services which would provide X.509 certificates to users. The U.S. Post Office is developing a (CA) hierarchy. The NIST Public Key Infrastructure Working Group has also been doing work in this area. The DOD Multi Level Information System Security Initiative (MISSI) program has begun deploying a certificate infrastructure for the U.S. Government. Verisign and GTE are also active

in this space. Alternatively, if no infrastructure exists, the PGP Web of Trust certificates can be used to provide user authentication and privacy in a community of users who know and trust each other.

2.1.0.1 Entity Naming

An entity's name is its identity and is bound to its public keys in certificates. The CA MUST define the naming semantics for the certificates it issues. See the UNINETT PCA Policy Statements for an example of how a CA defines its naming policy. When the certificate is verified, the name is verified and that name will have meaning within the realm of that CA. An example is the DNS security extensions which make DNS servers CAs for the zones and nodes they serve. Resource records are provided for public keys and signatures on those keys. The names associated with the keys are IP addresses and domain names which have meaning to entities accessing the DNS for this information. A Web of Trust is another example. When webs of trust are set up, names are bound with the public keys. In PGP the name is usually the entities e-mail address which has meaning to those, and only those, who understand e-mail. Another web of trust could use an entirely different naming scheme.

2.1.1 Passwords

Passwords are an example of a mechanism that provides weak authentication. The reason passwords are considered weak is the fact that most users pick passwords that are easy to guess and when used over an unprotected network are easily read by network sniffers.

2.1.2 Kerberos

There are other authentication systems available which rely on a trusted third party called a key distribution center (KDC) to distribute secret session keys. An example is Kerberos, where the trusted third party is the Kerberos server, which holds secret keys for all clients and servers within its network domain. A client's proof that it holds its secret key provides authentication to a server. Kerberos is not scaleable and should not be considered for implementation of an authentication mechanism.

2.2 Key Exchanges

The two common methods of using public key cryptography for key establishment are key transport and key generation. Several schemes exist that allow for secure key exchanges between two communicating end-points. Examples include:

2.2.0 RSA Key Exchange

An example of key transport is the use of the RSA algorithm to encrypt a randomly generated session key (for encrypting subsequent communications) with the recipient's public key. The encrypted random key is then sent to the recipient, who decrypts it using his private key. At this point both sides have the same session key, however it was created based on input from only one side of the communications. The benefit of the key transport method is that it has less computational overhead than the following method.

2.2.1 Diffie-Hellman Key Exchange

The Diffie-Hellman (D-H) algorithm illustrates key generation using public key cryptography. The D-H algorithm is begun by two users exchanging public information. Each user then mathematically combines the other's public information along with their own secret information to compute a shared secret value. This secret value can be used as a session key or as a key encryption key for encrypting a randomly generated session key. This method generates a session key based on public and secret information held by both users. The benefit of the D-H algorithm is that the key used for encrypting messages is based on information held by both users and the independence of keys from one key exchange to another provides perfect forward secrecy. Detailed descriptions of these algorithms can be found in [Schneier].

The exchange between two end-points is as described below. (N is a large prime, G is primitive mod N, R_1 and R_2 are random large integers).

Caller		Callee
Generate number pair (G, N)	⟶	
	Send (G, N) to Callee (may be insecure)	
Generate random number, R_1		Generate random number, R_2

Compute $K_A = G^{R}{}_1 \bmod N$ | Compute $K_B = G^{R}{}_2 \bmod N$

Exchange K_A and K_B

Compute key,
$K = K_B{}^{R}{}_1 = G^{R}{}_1{}^{R}{}_2 \bmod N$

Compute key,
$K = K_A{}^{R}{}_2 = G^{R}{}_1{}^{R}{}_2 \bmod N$

After these exchanges, the Caller and Callee come up with the same key. It is important to note that other endpoints *snooping* the initial insecure exchanges, will not be able to generate the key K because random numbers R_1 and R_2 are not transmitted.

2.3 Encryption Algorithms

The encryption algorithms considered for H.323 conferences must be capable of handling lossy media streams delivered over unreliable transport protocols such as UDP. Another factor to be considered in the selection of these algorithms is processor usage and their generality enabling them to be used across different media types and different data rates for a media type. Algorithms may also be Coder-specific, meaning that more than one algorithm may be in use in a H.323 conference session at the same time. Examples include:

2.3.1 RC5

The RC5 encryption algorithm is a fast, symmetric block cipher with a variable-length secret key, providing flexibility in its security level. It was invented by Ron Rivest and analyzed by RSA Laboratories.

RC5 is a parameterized algorithm, and a particular RC5 algorithm is designated as *RC5-w/r/b*. The parameters are as follows :

- *w* is the word size, in bits. The standard value is 32 bits; allowable values are 16, 32, and 64. RC5 encrypts two-word blocks: plaintext and ciphertext blocks are each 2*w* bits long.
- *r* is the number of rounds. Allowable values are 0,1...255.
- The number of bytes in the secret key *K*. Allowable values of *b* are 0, 1...255.

A variety of parameter settings allows users to select an encryption algorithm whose security and speed are optimized for their application, while providing an evolutionary path for adjusting their parameters as necessary in the future. For example, RC5-32/16/7 is an RC5 algorithm with the number of rounds and the length of key equivalent to DES. Unlike unparameterized DES, however, an RC5 user can upgrade the choice for a DES replacement to an 80-bit key by moving to RC5-32/16/10.

RC5 uses three mathematical operations : XOR, addition, and rotations. Rotations are constant time operations on most processors and variable rotations are a nonlinear function. A distinguishing feature of RC5 is its heavy use of data-dependent rotations – the amount of rotation performed is dependent on the input data, and is not pre-determined. The use of variable rotations should help defeat differential and linear cryptanalysis since bits are rotated to *random* positions in each round.

2.3.2 IDEA

IDEA is a block cipher; it was invented by Xuejia Lai and James Massey. IDEA operates on 64-bit plaintext blocks and the key is 128 bits long. The same algorithm is used for both encryption and decryption. As with most block ciphers, IDEA uses both confusion and diffusion. The design philosophy behind the algorithm is one of *mixing operations from different algebraic groups*. The three groups of operations are XOR, Addition modulo 2^{16}, Multiplication modulo $2^{16} + 1$.

IDEA is based on some impressive theoretical foundations and, although cryptanalysis has made some progress against reduced-round variants, the algorithm still seems strong. Its current claim to fame is that it is part of PGP.

2.3.3 Data Encryption Standard (DES)

DES is a block cipher that encrypts data in 64-bit blocks. A 64-bit block of plaintext (un-encrypted text) goes in one end of the algorithm and a 64-bit block of ciphertext (encrypted text) comes out the

other end. The key length is 56 bits. The key is expressed as a 64-bit number, but every eighth bit is used for parity checking and is ignored. These parity bits are the least-significant bits of the key bytes.

Detailed descriptions of these encryption algorithms can be found in [Schneier].

3.0 Authentication And Data Encryption Proposal For H.323

In this section we propose and strongly recommend the use of Digital Certificates for authentication. The generation of Digital Certificates occurs out-of-band with the H.323 protocol exchange and is beyond the scope of this document. The authentication, key exchange and encryption algorithm exchange occurs via the Secure Sockets Layer (SSL) protocol, which is currently a Working Draft in the IETF. SSL is a layered protocol that expects a reliable transport underneath. It is however independent of the reliable transport protocol itself, lending itself to implementation and interoperability across a variety of platforms.

SSL provides connection security between communicating end-points, that has essentially three properties.

1. The connection is private. Symmetric encryption is used for data encryption after the initial handshake.
2. The peer's identity can be authenticated by using asymmetric or public key cryptography.
3. The connection is reliable.

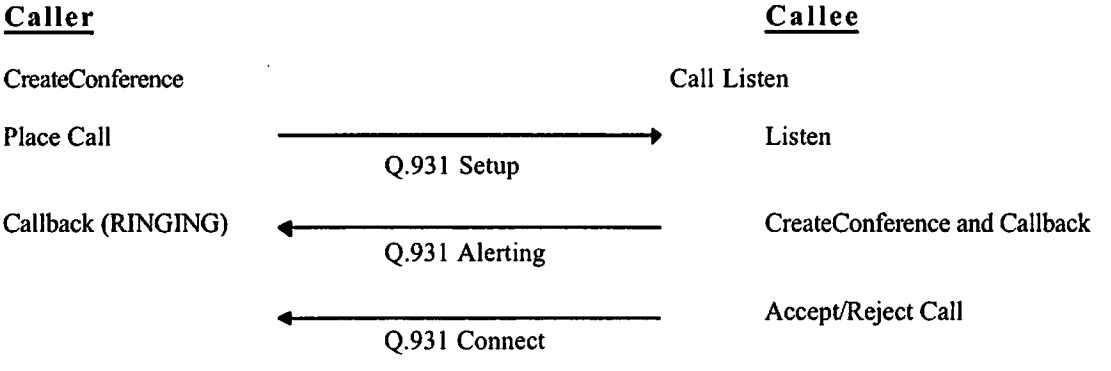
The secure WinSock implementation utilizes the Secure Sockets Layer (SSL) protocols and implementation. The SSL protocol support one or more secure sessions between a pair of communicating parties. Parties may also create multiple simultaneous SSL sessions. Setting up a communication session includes negotiation of the SSL *cipher-suite*. The cipher-suite defines the level of security provided for all connections created within the SSL session. It also defines which *Cryptographic Service Providers* (CSPs) should be used to support connection authentication, encryption, and message authentication. The communicating parties must use the same negotiated CSPs to support the client and server actions for each function. The cipher suite may be re-negotiated during the SSL session. Authentication is performed transparent to the application and independently for each connection within the SSL session.

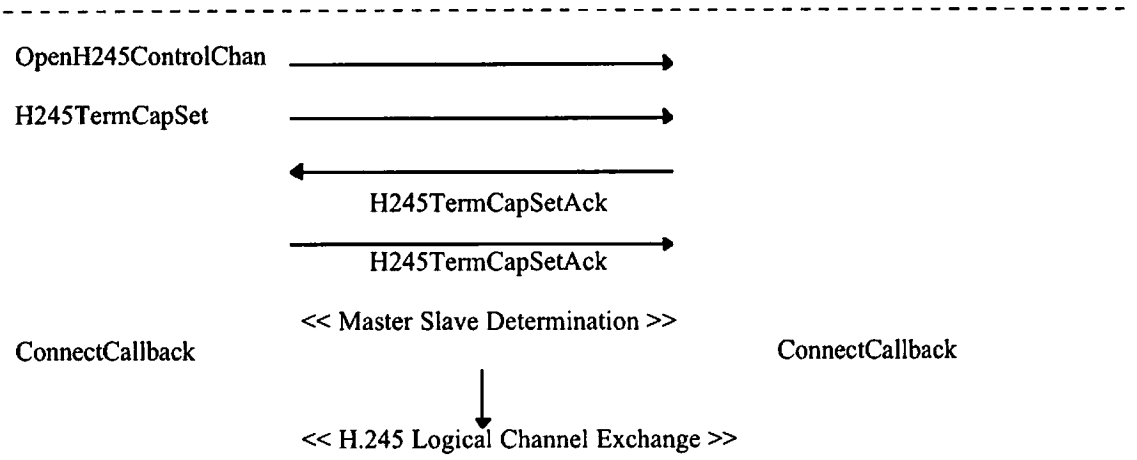
Once a connection is established, the communicating application processes begin to exchange messages of application-specific content. SSL takes each application created message, breaks the message content into blocks, compresses the content, applies an authentication hash code, encrypts the content and sends it to the destination system and process. At the receiving system, the received block is decrypted, the identity of the sender is verified, and content is decompressed, and the new content block is placed in a reassembly buffer for deliver when all the message components have arrived at the destination system.

All detected errors in the SSL protocol such as failure to authenticate, decrypt or decompress result in termination of the effected connection. Other connections within the SSL session containing the failed connection may continue, but the session is constrained such that new connections can not be created in that session.

3.1 How Does SSL Fit Into The Current H.323 Call Establishment Process

A typical Q.931/H.245 exchange (as mandated by H.225) between two communicating end-points during call establishment is as follows.




OpenH245ControlChan
H245TermCapSet
H245TermCapSetAck
H245TermCapSetAck
<< Master Slave Determination >>
ConnectCallback
ConnectCallback
<< H.245 Logical Channel Exchange >>

The Q.931 and H.245 protocol exchanges occur on two different reliable transport sockets. In each case, a WinSock modeled call sequence consists of the following steps.

Caller		Callee
socket () Initialize SOCKADDR_IN bind ()		socket () Initialize SOCKADDR_IN bind () listen ()
connect ()	⟶	accept ()
<< Association created; either side can send or receive >>		

SSL implemented as a Layered Service Provider (LSP), augments the handshake above, as follows.

Caller		Callee
socket () Initialize SOCKADDR_IN bind ()		socket () Initialize SOCKADDR_IN bind () listen ()
connect () **SSLConnect ()**	⟶	accept () **SSLaccept ()**
<< Association created; either side can send or receive >>		

SSLConnect() internally enables security via the following steps.

- Authentication using digital certificates
- Negotiates encryption algorithm
- Key Generation and Master Secret transfer
- Key Exchange
- Encryption (privacy) and integrity

The other control Q.931 and H.245 protocol messages can be exchanged via the secure channel. In order to support data encryption for the media streams, the encryption algorithms supported and the key exchange can occur over the secure channel. Two mechanisms are possible.

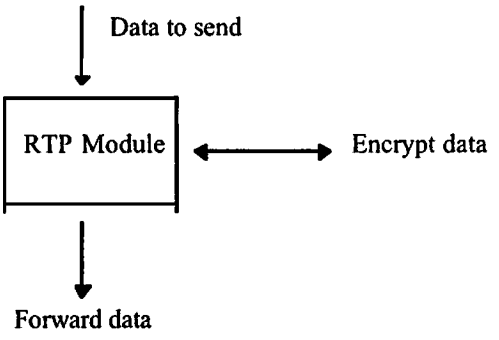
1. On the SSLConnect call, a Master Secret byte stream (32 bytes) is exchanged between the two end-points on the secure channel. The Master Secret stream exchanged during the H.245 protocol can be used to derive encryption keys by the two end-points, guaranteeing uniqueness and without requiring extra steps to exchange keys in a symmetric key based encryption system.
2. In this scheme the encryption algorithm and the key can be optionally passed along via the OpenLogicalChannel PDU, in the forwardLogicalChannelParameters sequence, immediately following the multiplexParameters entry. The format of these new fields is TBD.

Since the data streams may have characteristics demanding specifically tailored encryption algorithms, we need a mechanism to exchange and negotiate multiple supported encryption algorithms. It is also possible to identify an algorithm that supports media streams with different lossy attributes via mode settings. The algorithms and their modes can be exchanged via the H.245 Capability structure. A new structure EncryptionCapability could be defined for this purpose following the ConferenceCapability in the H.245 Capability structure. The format of these new fields is TBD.

3.2 Encryption Of Media Streams Over RTP

RTP supports encryption of data streams. The default encryption algorithm is DES. But the RTP spec allows for other encryption algorithms to be specified dynamically for a session by non-RTP means.

The scheme described above can be used to negotiate the encryption key and algorithm for a session. These parameters can then be passed along to the RTP stack during its initialization. A simplified view of the data flow once encryption is enabled is as follows.



4.0 Extensions For Multi-Point Conferences

This section is incomplete.

5.0 References

RTP

- RFC 1889 - RTP: A Transport Protocol for real-time applications.
- RFC 1980 - RTP Profile for Audio and Video Conferences with Minimal Control.

Security

- Applied Cryptography, Second Edition Protocols, Algorithms, and Source Code in C, Bruce Schneier: John Wiley & Sons, Inc., 1996
- SSL Protocol Version 3.0 Alan O. Freier, Philip Karlton, Paul C. Kocher, March 1996 Netscape Communications Corporation

ITU-T / IETF Documents

- ITU-T Recommendation Q.931.
- ITU-T Recommendation H.323.
- IETF Working Draft - SSH Transport Layer Protocol.