DRAFT

A Proposal for Unclassified Public Key Signature and Key Exchange Standards

20 October 1989

Table of Contents

Introduction

The purpose of this paper is to describe cryptographic algorithms and protocols which can be used to provide unclassified standards for public key exchange and digital signatures. These designs arose out of a request from NIST (the National Institute of Standards and Technology, formerly the National Bureau of Standards) for NSA help in developing unclassified public key signature and key exchange standards.

In this document we define algorithms for digital signatures and key exchange and discuss some ideas on how to authenticate the public keys generated in the system. The specification for a hashing function will be provided separately.

We envision that there would exist decentralized system administration for the overall signature system. There could exist enclaves of terminals who would build their own trusted centers for certification of their public signature keys and manage their own secure system using the algorithms specified by the standard. Alternatively, a national center for public key certification could be established to allow greater secure connectivity. Both structures can be accommodated. We sought to develop algorithms which would allow users to administer their networks and security policy as they see fit, while maintaining a base of secure cryptography.

The algorithms were selected to allow for common processing components where possible. As a result we perform all of the arithmetic operations modulo a single large composite integer N.

The Signature Process

The purpose of a public key signature is to authenticate the source and integrity of data. Through the signature verification process, the recipient of the data will obtain a high degree of assurance that the data was sent by only those parties possessing certain private information. The signature verifying party will possess related public information. To understand the signature process consider the following. Let:

- N = A 512-bit composite number which is the product of two primes chosen so that factoring is not feasible. This is common for all terminals.
- x_i = A terminal unique 79-bit integer known only to the signer (fixed for all messages). The value for terminal i will be denoted x_i.
- k = A 208-bit integer (with msb = 1) known only to the signer (changed for each message).
- g = An integer of large order in the multiplicative group mod N.
- r = $g^k \bmod N$.
- m = The message to be transmitted.

Also let h(a) be a one-way hashing function on arbitrary length inputs (a) to 128-bit integers. Define h(a,b) to be the hash of a data stream corresponding to the concatenation of the data stream representing the inputs a and b.

With these definitions in mind we can proceed with the definition of the signature process. Prior to any exchange of messages the sender's public key is emplaced at the receiver or this key is sent in an authenticated manner during the transmission of the message. This public key is created by first computing $y_i = g^{x_i} \bmod N$. The integer y_i is the public key. When the sender wants to transmit a message, he chooses a random k and computes r, as specified above, and then computes the following:

$s_1 = h(r,m)$ (the result is a 128-bit integer)

$s_2 = (k - x_i s_1)$ (This is an operation over the integers. The value s_2 will be positive, since x_i is 79 bits and s_1 is 128 bits their product will be less than 2^{207}. The value k, however, is

greater than or equal 2^{207} since we specified that the msb of k be 1.

The values s_1 and s_2 constitute the signature of the message . These are transmitted along with the message to the recipient. Let m' be the received message. It may or may not match the signed message. To verify the signature, the receiver computes:

$$r' = (y^{s1})(g^{s2}) = (g^{x(s1)})(g^{k-x(s1)}) = g^k = r \pmod N$$

and

$$s_1'' = h(r', m')$$

If ($s_1' = s_1$) then the signature is verified and the receiver can have high confidence that the message was sent by only parties holding the secret key x_i.

Besides the message itself the overhead for a signed message is 128 bits for s_1. and 208 bits for s_2 for a total of 336 bits.

Authenticating Signature Keys

A previous section of this paper dealt with the signature process for messages. It did not cover how user's public keys are authenticated and distributed for use by the receivers of messages. The public key certification procedure involves the use of a trusted center in the system. The role of the center is to bind (via public key signature) a users identification (ID) and their public key $g^x \bmod N$. The system start-up will work as follows:

1. A user will generate his secret integer x and produce $g^x \bmod N$ as described in the signature description.

2. The user goes to the trusted center and presents his public key and some identification information. The center verifies the ID and forms a message consisting of the ID concatenated with the public key. This will be called the ID/Public Key Message.

3. The center then uses the signature process and its own secret information to sign the message. Then the signed message (consisting of the message itself and the signature parameters r and s) is returned to the user along with the public key for the trusted center.

The trusted center's public key allows the user to verify the authenticity of other user's public keys. To send a signed message, instead of sending only the message itself and its corresponding signature parameters ($s1_{mess.}$, $s2_{mess.}$) the user also sends his ID/Public Key Message and its corresponding signature parameters created by the trusted center ($s1_{tc}$, $s2_{tc}$). This allows any recipient to verify that the public key does indeed belong to the party identified in the ID and to verify that the message came from the identified party. The entire signed message now consists of the following information:

(ID/Public Key Message) ($s1_{tc}$) ($s2_{tc}$) (MESSAGE) ($s1_{mess.}$) ($s2_{mess.}$)

At time of enrollment, all users are given the trusted centers public key to verify the ID/Public Key Message. We should note that if the public key of the sender is already in place at the receiver then only the message dependent paramters need to be sent.

It is also important to note that the trusted center does not have access to a user's secret information at any time in the process. All one relies on the trusted center to do is:

1. Maintain the security of its own secret information.

2. Accurately identify the users seeking enrollment and sign their public keys.

This can be done at one centralized location or can be done separately by any enclave of terminals who share a common trusted authority and will trust that authority to certify their keys.

The Key Exchange Process

The purpose of a key exchange is for two parties (A and B in our example) to exchange some cryptographically derived data in such a way that at the end of the exchange the parties hold a shared secret key. This key is known to the two parties but to no one else. The key can then be loaded into a standard key generator and the parties can securely communicate. To understand the key exchange process consider the following. Let:

- N = A 512-bit composite integer. It is chosen to be the product of two large primes and in such a way that both the discrete log problem mod N and the factorization of N is difficult.
- g = An element of large order in the multiplicative group mod N.
- x_i = A secret 80-bit integer known only to user i (changed at user discretion).
- y_i = $g^{x_i} \bmod N$; This is the public key exchange value for user i.

To securely establish an authenticated key, parties A and B do the following:

1. A and B exchange public values (y_A and y_B). This can be done through extraction from a public directory or via the exchange of signed versions of the public values from a trusted authority.
2. A computes: $(y_B)^{x_A} = g^{x_A x_B} \bmod N$
3. B computes: $(y_A)^{x_B} = g^{x_A x_B} \bmod N$
4. A and B select a subset or hash of the resultant value to be the traffic key.

Authenticating Key Exchange Parameters

As described so far the key exchange process in the system has no user authentication. Crucial to any automated key distribution system is an ability to identify the other party on the end of a link. In the key exchange context this implies knowing who has generated the half of the key exchange which we are using to establish a secret key. The authentication process we are suggesting is straight forward. The data to be authenticated is a key exchange packet $g^{xA} \bmod N$ for user A. In order to have other users trust that when they use this data in a key exchange, they will be communicating with A, user A first signs the key exchange packet as it would any other message. A would then transmit (or leave in some sort of electronic mailbox) the following information:

(ID/Public Key Message) $(s1_{tc})$ $(s2_{tc})$ $(g^{xA} \bmod N)$ $(s1_{key})$ $(s2_{key})$

where $s1_{key}$ and $s2_{key}$ correspond to the signature parameters for $g^{xA} \bmod N$. This data will then allow users to extract A's ID and public key, verify its authenticity, and then verify that the key exchange packet received is indeed associated with user A. After this key exchange packet verification, user's can have confidence that if they use the packet in a key exchange that they will be establishing a key with user A and no one else. We again note that if user A's public key is already in place at the receiver only the key dependent parameters need to be transmitted.