NTP Security Algorithms

David L. Mills
University of Delaware
http://www.eecis.udel.edu/~mills
mailto:mills@udel.edu

Sir John Tenniel; Alice’s Adventures in Wonderland, Lewis Carroll
Symmetric key and public key cryptography

- Public key cryptography
  - Encryption/decryption algorithms are relatively slow with highly variable running times depending on key and data
  - All keys are random; private keys are never divulged
  - Certificates reliably bind server identification and public key
  - Server identification established by challenge/response protocol
  - Well suited to multicast paradigm

- Symmetric key cryptography
  - Encryption/decryption algorithms are relatively fast with constant running times independent of key and data
  - Fixed private keys must be distributed in advance
  - Key agreement (Diffie-Hellman) is required for private random keys
  - Per-association state must be maintained for all clients
  - Not well suited to multicast paradigm
Message propagation time budget

We want $T_3$ and $T_4$ timestamps for accurate network calibration

- If output wait is small, $T_{3a}$ is good approximation to $T_3$
- $T_{3a}$ can’t be included in message after cryptosum is calculated, but can be sent in next message; use $T_{3b}$ as best approximation to $T_3$
- $T_4$ captured by most network drivers at interrupt time; if not, use $T_{4a}$ as best approximation to $T_4$

Largest error is usually output cryptosum

- Private-key algorithms (MD5, DES-CBC) running times range from 10 $\mu$s to 1 ms, depending on architecture, but can be predicted fairly well
- Public-key algorithms (RSA) running times range up to 100 ms, depending on architecture, but are highly variable and depend on message content
**MD5 message digest computations**

- Measured times to construct 128-bit hash of 48-octet NTP header using MD5 algorithm in RSAREF
MD5/RSA digital signature computations

- Measured times (s) to construct digital signature using RSAREF
- Message authentication code constructed from 48-octet NTP header hashed with MD5, then encrypted with RSA 512-bit private key
Certificates

- A private/public key pair and self signed host certificate are required for each host.
  - Certificates are in X509 version 3 format valid for one year.
  - The serial number is the NTP seconds of generation to insure uniqueness.
- Extension fields are used to convey identity parameters and whether the certificate is private or trusted.
  - The required Basic Constraints field contains the string “critical,CA:TRUE”, indicating the host can act as a certificate authority.
  - The required Key Usage field contains the string “digitalSignature,keyCertSign”, indicating the certificate is valid for digital signatures and to sign other certificates.
  - The optional Extended Key Usage field contains the string “private” indicating a private certificate (PC identity scheme) or the string “trustRoot” indicating a trusted certificate. By definition, private certificates are trusted.
  - The optional Subject Key Identifier field contains the public key for the GQ identity scheme.
Signature operations

- Public keys, certificates and leapseconds files can be read from local files or sent over the net using the Autokey protocol.
- Cryptographic values are signed only when the host is synchronized.
  - Filestamps record the NTP seconds when the file was created. These are proventic data and provide a reliable total ordering of creation epochs.
  - Timestamps record the NTP seconds when the data were last signed. These are proventic data only when the sender is synchronized and provide only a partial ordering of signing epochs.
- Cryptographic values derived from files and received over the net are signed only when they are created or changed and in addition at refresh intervals of about one day.
- Autokey values are signed when the key list is regenerated, about once per hour.
- Cookie values are signed when sent.
- Identity values are signed when sent.
# Identification exchange

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nonce $r$ and send</td>
<td>Challenge Request</td>
</tr>
<tr>
<td>Verify response and signature</td>
<td>Send response and signature</td>
</tr>
</tbody>
</table>

- This is a challenge-response scheme
  - Client Alice and server Bob share a common set of public parameters and a private group key $b$.
  - Alice rolls random nonce $r$ and sends to Bob.
  - Bob rolls random nonce $k$, computes a one-way function $f(r, k, b)$ and sends to Alice.
  - Alice computes some function $g(f, b)$ to verify that Bob knows $b$.
- The signature prevents message modification and binds the response to Bob’s private key.
- An interceptor can see the challenge and response, but cannot determine $k$ or $b$ or how to construct a response acceptable to Alice.
Private certificate (PC) identity scheme

- TA generates a certificate marked private and transmits it by secure means to all servers and clients.
- The certificate is never divulged outside the group and never presented for signature.
- An identity exchange is not necessary.
- Refreshing certificates is a major problem.
Each certificate is signed by the issuer, which is one step closer on the trail to the trusted host (TH).

The trusted host certificate is self-signed and self-validated.

This scheme is vulnerable to a middleman masquerade, unless an identity scheme is used.

A trusted authority (TA) generates the group key (if used) which has the same name as the TH subject name.
TA generates the IFF parameters and keys and transmits them by secure means to all servers and clients.

Only the server needs the group key; the client key derived from it is public.

IFF identity exchange is used to verify group membership.
Schnorr (IFF) identification scheme operations

- Schnorr (IFF) scheme is based on DSA principles.
  - Public parameters include 512-bit prime $p$, 160-bit prime $q$ that divides $p-1$ and generator $g$ of $p$ such that $g^q = 1 \mod p$.
  - TA rolls private random group key $b$ and distributes to all servers in the group using secure means.
  - TA computes public $v = g^q - b \mod p$ and distributes to all clients in the group using insecure means.
  - Client Alice rolls random nonce $r$ ($0 < r < q$) and sends to server Bob.
  - Bob rolls random nonce $k$ ($0 < k < q$), computes $y = k + br \mod q$ and $x = g^k \mod p$, then sends ($y$, hash($x$)) to Alice.
  - Alice computes $g^y v^r \mod p$ (which is $g^k \mod p$ without revealing $k$), then verifies hash($g^k$) matches hash($x$).

- If the parameters or group key are changed, all group members must be updated.
Guillou-Quisquater (GQ) scheme

- TA generates the GQ parameters and keys and transmits them by secure means to servers and clients.
- Server generates a GQ private/public key pair and certificate with the public key in an extension field.
- Client uses the public key in the certificate as the client key.
- GQ identity exchange is used to verify group membership.
Guillou-Quisquater (GQ) identity scheme operations

- Guillou-Quisquater (GQ) scheme is based on RSA principles.
  - Public parameters include 512-bit modulus $n$ a product of two large primes $p$ and $q$.
  - TA rolls private random group key $b$ and distributes to all group members using secure means.
  - Each group member rolls random private nonce $u$ ($0 < u < n$) and computes public $\nu = (u^1)^b \mod n$, then saves both for future reference. The $\nu$ is conveyed in an extension field of the member’s public certificate.
  - Alice rolls random nonce $r$ ($0 < r < q$) and sends to Bob.
  - Bob rolls random nonce $k$ and computes $y = ku^r \mod n$ and $x = kb \mod n$, then sends $(y, \text{hash}(x))$ to Alice.
  - Alice computes $y^b \nu^r \mod n$, which simplifies to $k^b \mod n$, then verifies $\text{hash}(k^b)$ matches $\text{hash}(x)$.

- If the parameters or group key are changed, all group members must be updated; however, a member can refresh $u$, $\nu$ and certificates at any time.
Mu-Varadharajan (MV) identity scheme – setup I

- Mu-Varadharajan (MV) identity scheme is based on DSA principles.
- TA generates private parameters and server coefficient $A$.
  - TA generates $n$ distinct primes $s_1, \ldots, s_n$, their product $q$, prime $p = 2q + 1$ and generator $g$ of $p$ such that $g^q = 1 \mod p$. These parameters are generated by a probabilistic algorithm such that $p$ has approximately 500 significant bits. Note that the multiplicative group $\mathbb{Z}_q^*$ includes only those elements $x$ where $\gcd(x, q) = 1$.
  - TA generates $n$ roots $x_1, \ldots, x_n$ of the polynomial $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \mod q$, then solves for $a_0, \ldots, a_n$ using a fast recursive algorithm.
  - TA computes functions $g_{ij}(a_i, x_j)$ ($i = 0, \ldots, n; j = 1, \ldots, n$) mod $p$ as the matrix $G$ with $i$ rows corresponding to coefficients $a_i$ and $j$ columns corresponding to roots $x_j$. By construction, the product of all elements of $G$ is unity. The functions $g_{ij}$ are described elsewhere.
  - Let $S$ be the submatrix $g_{ij}$ ($i = 0, \ldots, n-1; j = 1, \ldots, n$); i.e., all but the last row, and $C$ the vector $g_{nj}$ ($j = 1, \ldots, n$); i.e., only the last row. The server coefficient is $A$ computed as the product of all elements of $S$ mod $p$; this need be computed only once; $S$ will not be used again.
Mu-Varadharajan (MV) identity scheme – setup II

- TA generates private server encryption and client decryption keys.
  - TA rolls private random group key $b$ ($0 < b < q$) and computes its inverse $b^{-1} \mod q$.
  - For each $s_i$, TA computes $s'_i$ such that $s'_i s_i = s_i \mod q$; i.e., $s'_i = (q + s_i)/s_i$. These are used as enabling keys to activate or revoke client decryption keys.
  - For each $g_{nj}$ of $C$, TA generates corresponding $x_{\bar{j}} = b^{-1} \Sigma x_i^p \mod q \; (i = 1, \ldots, n, i \neq j)$ and $x_{\hat{j}} = s'_j x_j^n$. Each tuple $(p, x_{\bar{j}}, x_{\hat{j}})$ $(j = 1, \ldots, n)$ is a private client decryption key for the $b$ group and can be activated and revoked independently of each other. The $j$th key is distributed to each member of the $j$th client subgroup by secure means.
  - TA determines which client subgroups are to be enabled and computes the product $s$ of the associated $s_j$. Then it computes the server private encryption key $E = A^s \mod p$ and public decryption keys $g_{\bar{b}} = g^s \mod p$ and $g_{\hat{b}} = g^{sb} \mod p$. The tuple $(p, q, E, g_{\bar{b}}, g_{\hat{b}})$ is distributed to the server group by secure means. All other data are private to the TA.
Mu-Varadharajan (MV) scheme

- TA generates MV parameters, group key, server key and client keys.
- TA transmits private encryption and public decryption keys to all servers using secure means.
- TA transmits individual private decryption keys to each client using secure means.
- TA can activate/deactivate individual client keys.
- The MV identity exchange is used to verify group membership.
Mu-Varadharajan (MV) identity scheme operations

- Client Alice verifies server Bob knows the secrets of the scheme identified with the $b$ group and $j$ subgroup.
  - Alice rolls random nonce $r$ ($0 < r < q$) and sends to Bob.
  - Bob rolls random nonce $k$ ($0 < k < q$) and computes $y = rE^k$, and public decryption keys $ybar = gbar^k$ and $yhat = ghat^k$, then sends (hash($y$), $ybar$, $yhat$) to Alice.
  - Alice computes $F = ybar^{\text{yhat}} yhat^{xbar}$, which by construction is the inverse of $E^k$. She computes $x = rF^{-1}$, then verifies that hash($x$) matches hash($y$).

- As a practical consideration, this scheme is limited to $n$ less than about 30 with $p$ in the order of 500 significant bits. This is because the number of distinct primes $s_j$ become harder to find as the number of significant bits of $s_j$ diminish.
Key generation

- Key files are generated using the ntp_keygen utility.
  - Most files are generated and used on the same host; only the identity values need to be securely distributed in advance.
  - `hostname` is provided by the Unix gethostname() routine.
  - `filestamp` is the NTP seconds when the file was created.
  - All files are in PEM-encoded printable ASCII suitable as MIME extensions

- `ntpkey_key_hostname.filestamp`
  - Public/private encryption key

- `ntpkey_cert_hostname.filestamp`
  - X.509 version 3 certificate

- `ntpkey_sign_hostname.filestamp`
  - Public/private signature key; must agree with certificate key

- `ntpkey_scheme_hostname.filestamp`
  - Identification scheme IFF, GQ or MV
Key management

- Keyspace is relatively small, so keys must be refreshed frequently
  - Keys are refreshed automatically and without management intervention
  - Session key list is regenerated about once per hour
  - Server private cookie is regenerated about once per day
  - Public keys and certificates are regenerated by scripts about once per month
  - Autokey protocol automatically handles key refreshment and recovery

- Autokey protocol enforces partial ordering for file creation and use
  - NTP timestamp is appended to the name of every cryptographic data file
  - Filestamps accompany the data as it is moved from place to place
  - Certificate and certificate requests include filestamp as sequence number
  - Dependency graph is created for public keys, certificates and data dependent on them
  - By induction, the graph includes all cryptographic data in the network derived from the trusted primary servers at the root of the graph
Further information

  - Current NTP Version 3 and 4 software and documentation
  - FAQ and links to other sources and interesting places
- David L. Mills: http://www.eecis.udel.edu/~mills
  - Papers, reports and memoranda in PostScript and PDF formats
  - Briefings in HTML, PostScript, PowerPoint and PDF formats
  - Collaboration resources hardware, software and documentation
  - Songs, photo galleries and after-dinner speech scripts
- FTP server ftp.udel.edu (pub/ntp directory)
  - Current NTP Version 3 and 4 software and documentation repository
  - Collaboration resources repository
- Related project descriptions and briefings
  - See “Current Research Project Descriptions and Briefings” at http://www.eecis.udel.edu/~mills/status.htm