Autokey Version 2 Protocol Model and Implementation

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

Neanderthal Training Protocol

- We all know what NTP stands for, but NTP is only an implementation vehicle providing a distributed chassis to test and exercise something here called the Autokey Version 2 Protocol. Chassis, Autokey, get it?
- Rather than have me redraw a bunch of slides for this presentation, please interpret “NTP” in the following as the Neanderthal Training Protocol.
- Any mention of “time” should be interpreted as somewhere in the ages of the dinosaurs when clocks hadn’t been invented yet.
- We also have a project for timekeeping in the Interplanetary Internet in which this stuff might appear. There it’s the Neptune Terraforming Project.

NTP authentication - approach

- Authentication and synchronization protocols work independently for each peer, with tentative outcomes confirmed only after both succeed.
- Public keys and certificates are obtained and verified relatively infrequently using X.509 certificates and certificate trails.
- Session keys are derived from public keys using fast algorithms.
- Each NTP message is individually authenticated using session key and message digest (keyed MD5).
- A proventic trail is a sequence of NTP servers each time synchronized and cryptographically verified to the next and ending on one or more trusted time servers.
- NTP and Autokey run in parallel to synchronize time and construct proventic trails.
- When both time and at least one proventic trail are verified, the peer is admitted to the population used to synchronize the system clock.

Neanderthal distributed sensor model

- A Neanderthal sensor network consists of a hierarchical forest of servers and server/clients.
  - Some servers are declared primary (stratum 1) and are self-certifying using means outside the protocol.
  - Other servers automatically configure in a forest at increasing stratum or authentication flow toward the leaves.
  - The Autoconfigure protocol does configuration, but that's another briefing.
- Each stratum authenticates from the preceding stratum using the Autokey protocol described later in this briefing.
  - Until recently, this scheme was vulnerable to a middleman attack.
  - A rogue could pretend to be a server and terrorize dependent server/clients without being detected.
  - So, a step has been added where a client must provide secure identification TBD to the server, which then is then convinced to sign the client certificate.
  - Thus, each server becomes a trusted certificate authority for dependent clients.

More on Neanderthals

- There are three steps to the authentication process for a server at stratum *N*.
  - Server N obtains the certificate from server *N* – 1. This certificate is signed by server *N* – 2, except for servers *N* = 1, which are self-signed.
  - Server *N* obtains the certificate for server *N* – 2, which sever *N* – 1 has previously obtained, and uses it to authenticate the server *N* – 1 certificate.
  - Server *N* proves authentic to server N – 1 using a scheme TBD and asks it to sign its certificate. This is provided to the client at stratum *N* + 1 on demand.
- This scheme works as long as servers *N* can prove identity to servers *N* – 1.
- For the most secure, servers could cache all certificates to the root, but this could burn lots of network bandwidth and sensor memory.
- The identification scheme will use the extension fields of the X.509 version 3 certificate.

NTP protocol header and timestamp formats

NTP Protocol Header Format (32 bits)

LI	VN	Mode	Strat	Poll	Prec
Root Delay					
Root Dispersion					
Reference Identifier					
Reference Timestamp (64)					
Originate Timestamp (64)					
Receive Timestamp (64)					
Transmit Timestamp (64)					
Extension Field 1 (optional)					
Extension Field 2… (optional)					
Key/Algorithm Identifier					
Message Digest (64 or 128)					

Cryptosum

Authenticator (Optional)

LI	leap warning indicator
VN	version number (4)
Strat	stratum (0-15)
Poll	poll interval (log2)
Prec	precision (log2)

NTP Timestamp Format (64 bits)

Seconds (32)	Fraction (32)

Value is in seconds and fraction since 0^h 1 January 1900

NTP v4 Extension Field

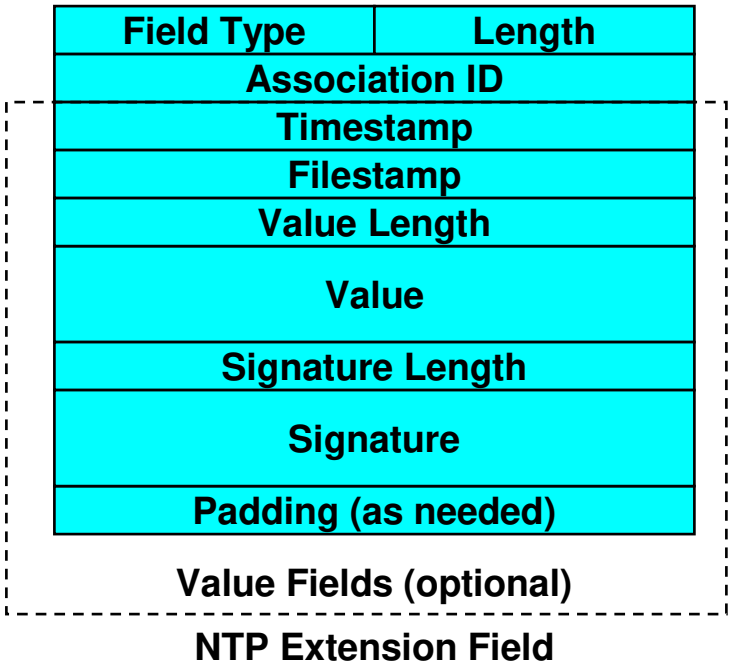
Field Type	Length
Extension Field (padded to 32-bit boundary)	

Last field padded to 64-bit boundary

NTP v3 and v4
NTP v4 only
authentication only

Authenticator uses MD5 cryptosum of NTP header plus extension fields (NTPv4)

NTPv4 extension fields



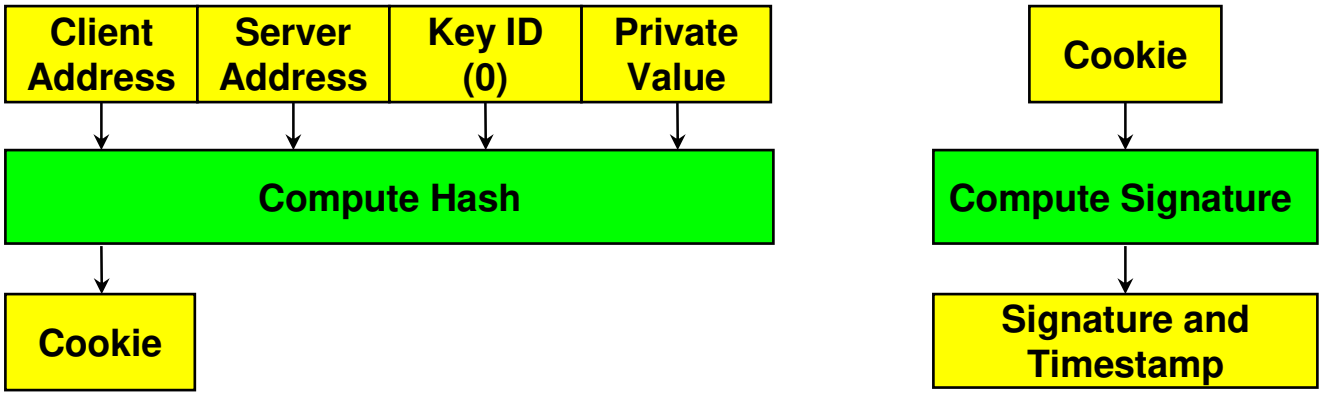
- New extension fields format defined for NTP Version 4
  - Fields are processed in order.
  - Requests may be transmitted with or without value fields.
  - Last field padded to 64-bit boundary; all others padded to 32-bit boundary.
  - Field length covers all payload and padding.

Session keys

NTPv4 Session Key

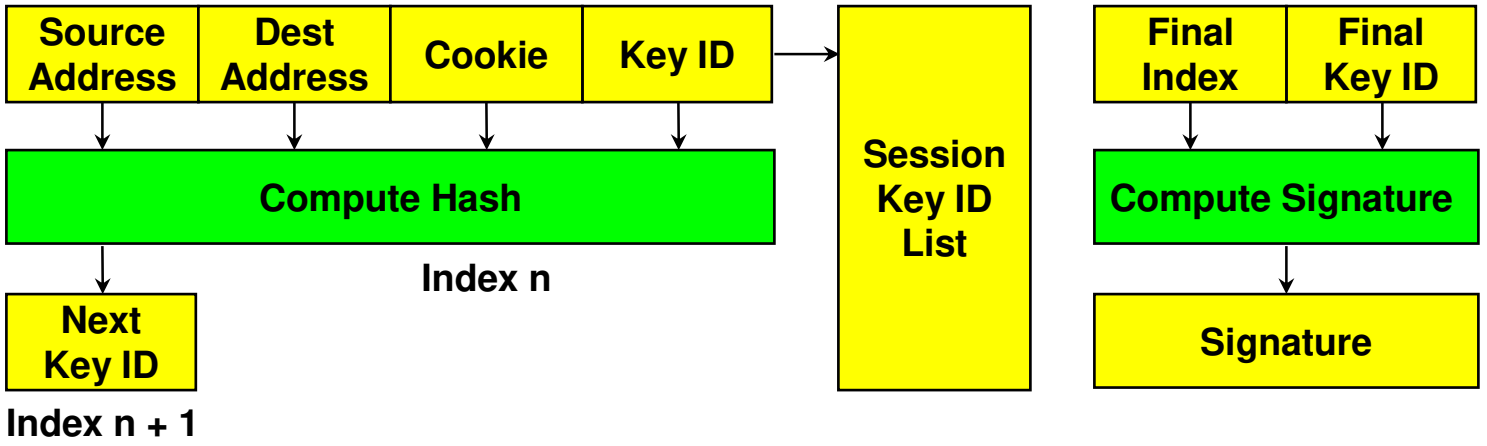
- NTPv4 session keys have four 32-bit words (16 octets total).
- The session key value is the 16-octet MD5 message digest of the session key.
- Key IDs have pseudo-random values and are used only once. A special key ID value of zero is used as a NAK reply.
- In multicast mode and in any message including an extension field, the cookie has a public value (zero).
- In client/server modes the cookie is a hash of the addresses and a private value, optionally blinded by a stream cipher.
- In symmetric mode both peers derive the cookie from the Diffie-Hellman agreed key.

Computing the cookie



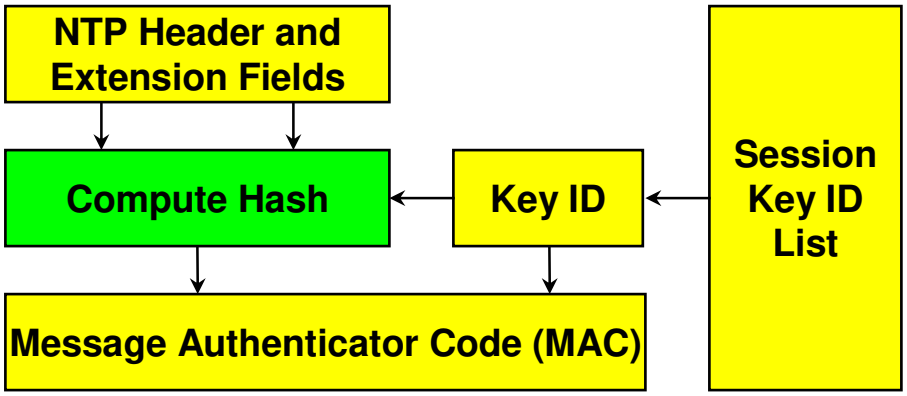
- The server generates a cookie unique to the client and server addresses and its own private value. It returns the cookie, signature and timestamp to the client using an extension field.
- The cookie is blinded by a one-time stream cipher generated by a Diffie-Hellman agreement.
- The server uses the cookie to validate requests and construct replies.
- The client uses the cookie to validate the reply and checks that the request key ID matches the reply key ID.

Generating the session key list



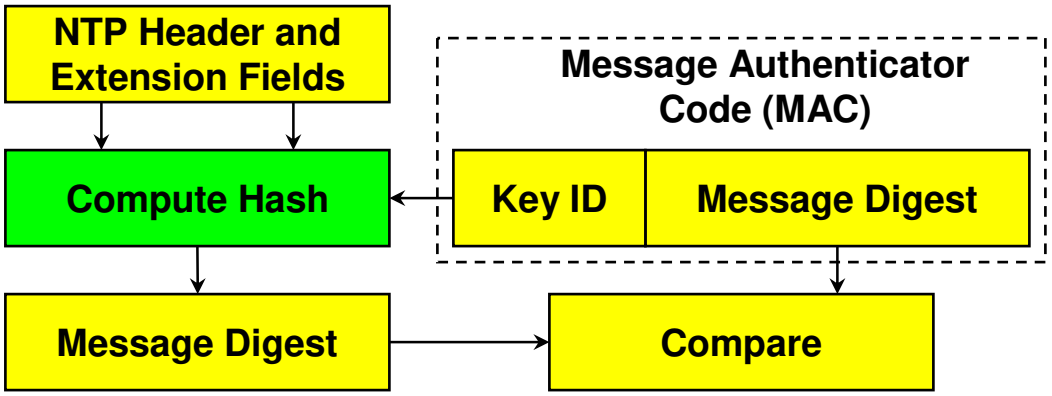
- The server rolls a random 32-bit seed as the initial key ID and selects the cookie. Messages with a zero cookie contain only public values.
- The initial session key is constructed using the given addresses, cookie and initial key ID. The session key value is stored in the key cache.
- The next session key is constructed using the first four octets of the session key value as the new key ID. The server continues to generate the full list.
- The final index number and key ID are provided in an extension field with signature and timestamp.

Sending messages



- The message authenticator code (MAC) consists of the MD5 message digest of the NTP header and extension fields using the session key ID and value stored in the key cache.
- The server uses the session key ID list in reverse order and discards each key value after use.
- An extension field containing the final index, key ID and signature is included in the first message from the list.
- This extension field can be provided upon request at any time.
- When all entries in the key list are used, a new one is generated.

Receiving messages



- The intent is not to hide the message contents, just verify where it came from and that it has not been modified in transit.
- The MAC message digest is compared with the computed digest of the NTP header and extension fields using the session key ID in the MAC and the key value computed from the addresses, key ID and cookie.
- If the cookie is zero, the message contains public values. Anybody can validate the message or make a valid message containing any values.
- If the cookie has been determined by secret means, nobody except the parties to the secret can validate a message or make a valid message.

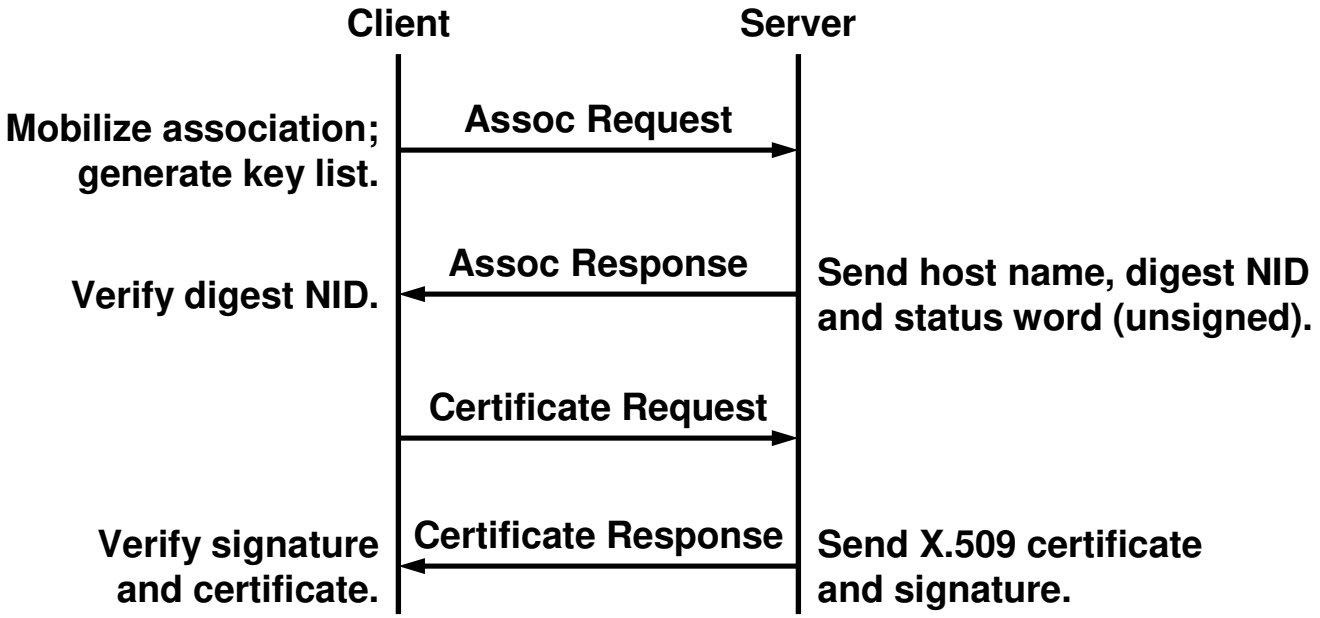
Autokey messages

- Association (1)
  - Send request; receive association parameters and host name. Not signed.
- Certificate (2)
  - Send request; receive X.509 certificate with issuer name and validity period.
- Cookie (3)
  - Send public key; receive encrypted cookie.
- Autokey (4)
  - Send request with autokey values; receive autokey values.
- Leapseconds (5)
  - Exchange leapseconds tables, each peer keeping the latest version.
- Sign (6)
  - Send client certificate; receive signed client certificate.

Signature operations

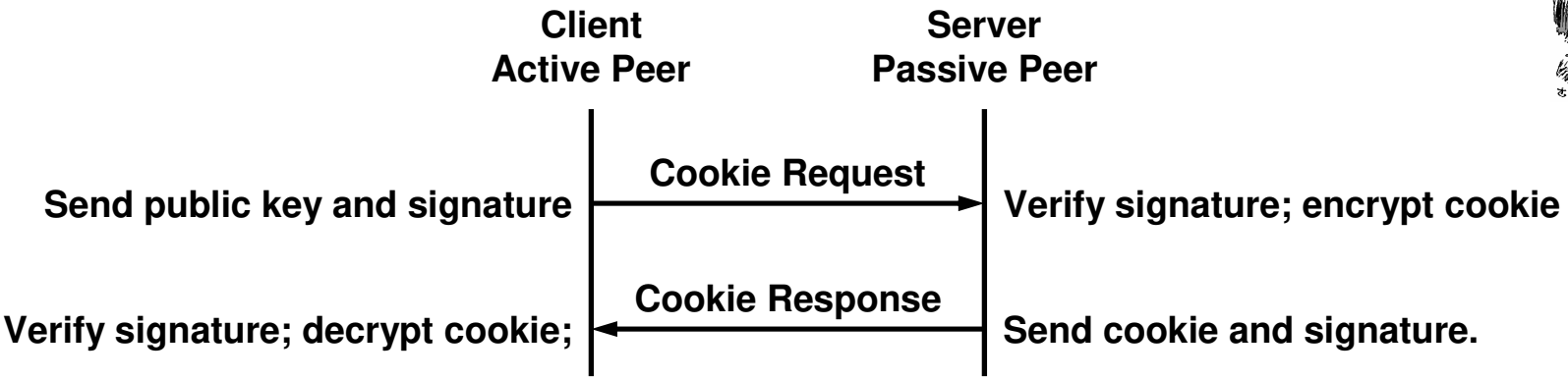
- Public keys, certificates and leapseconds files can be read from local files or sent over the net using the Autokey protocol.
- Cryptographic values are signed only when the host is synchronized.
  - Filestamps record the NTP seconds when the file was created. These are proventic data and provide a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

Association and certificate messages (all modes)



- Initial exchange specifies server message digest and signature encryption algorithm, which can be any supported by OpenSSL.
- The Certificate Request/Response cycle repeats as needed.
- Primary (stratum 1) certificate is self-signed and ipso facto valid.
- Secondary certificates are signed by the next lower stratum server and validated with its public key.

Cookie agreement (client/server and symmetric modes)

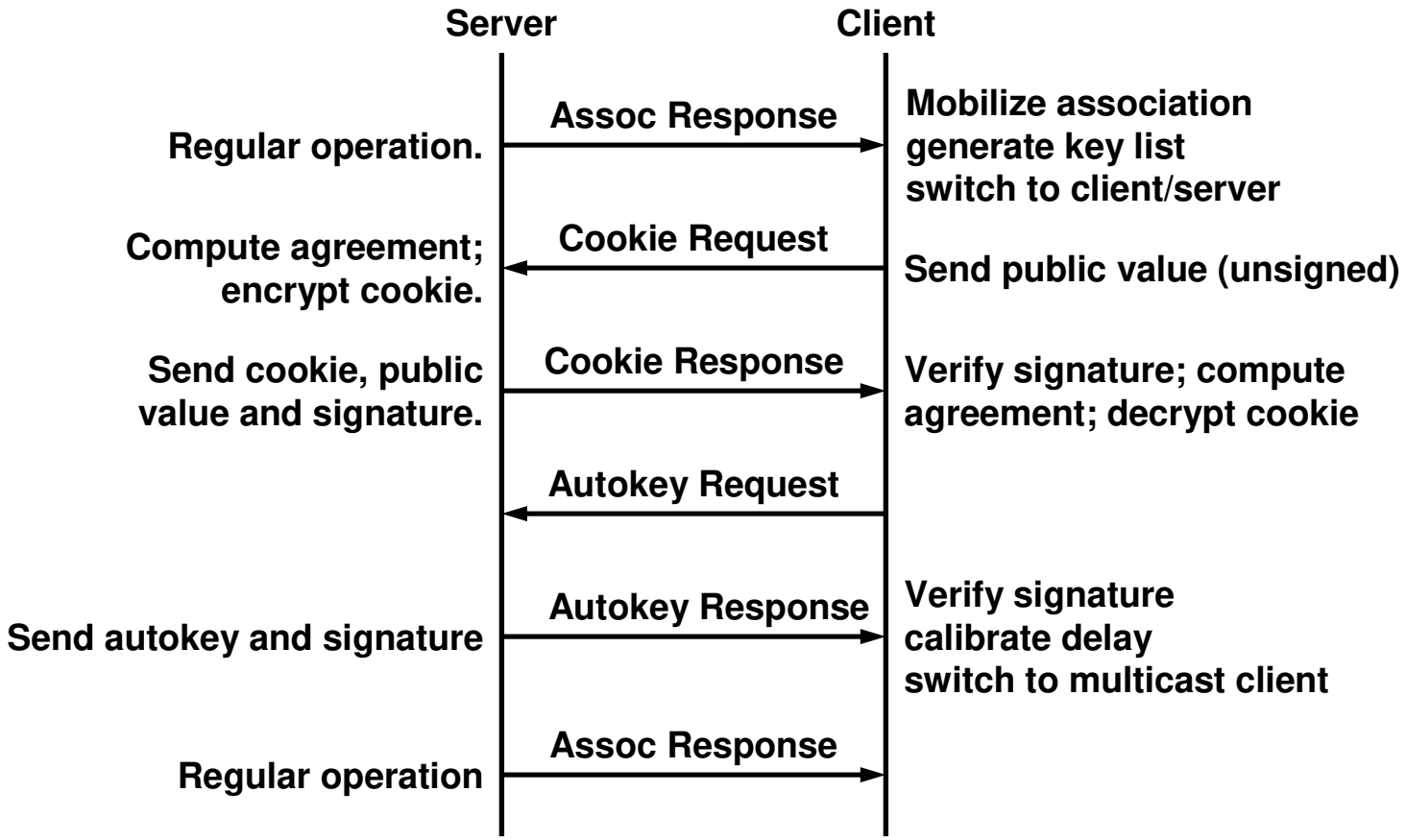


- Client sends public key to server without signature when unsynchronized.
- Symmetric active peer sends public key and signature to passive peer when synchronized.
- Server cookie is encrypted from the hash of source/destination addresses, key ID and server private value.
- Symmetric passive cookie is a random value.

Broadcast/multicast mode

- The server sends messages at fixed intervals.
  - The first message sent after regenerating the key list includes the autokey values and signature.
  - Other messages include the server association ID, but no signature. This is used as a handle for clients to request the autokey values if necessary.
  - These messages are considered public values, so the cookie value is zero.
- When a multicast client receives the first message, it temporarily switches to client/server mode in order to calibrate the network propagation delay and authenticate the server.
- The client first obtains the certificate, cookie and signature as in client/server mode, then obtains the autokey values and signature.
- When the propagation delay is calibrated, the client switches back to multicast client mode and makes no further transmissions.

Broadcast/multicast mode protocol

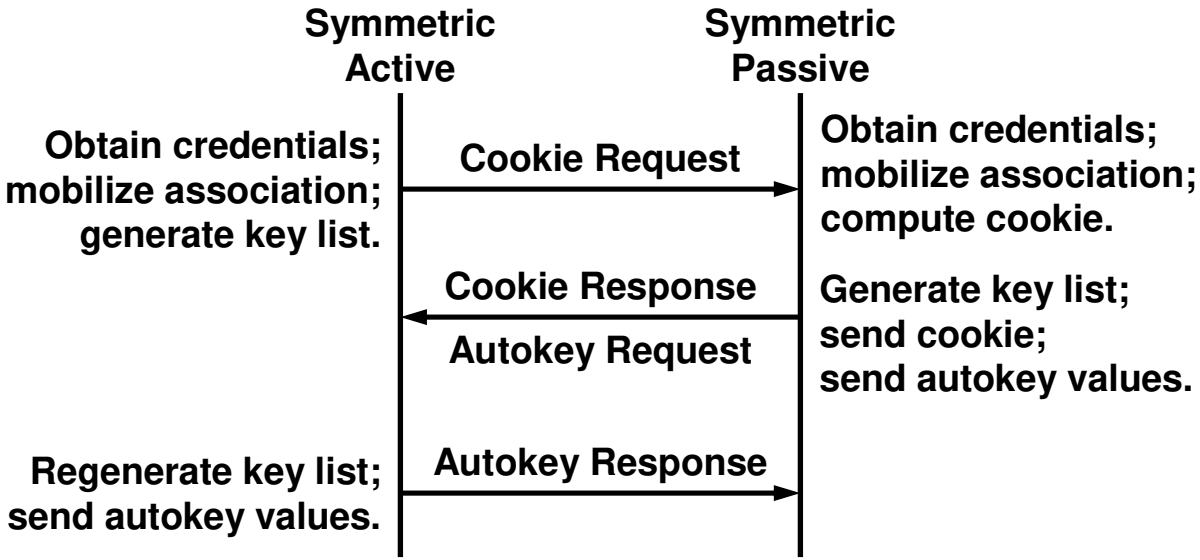


- Note: Association and Certificate messages not shown for clarity.

Symmetric mode

- Symmetric peers can each synchronize the other, depending on which one has the lowest synchronization distance.
  - One of the peers must be active; the other can be active or passive.
  - The passive peer must mobilize an association, verify the active peer credentials and compute the cookie
- Each peer computes a cookie as in client/server mode.
- Each peer generates a key list independently of the other.
- The peers verifies the other peer as in multicast client mode.

Symmetric mode protocol

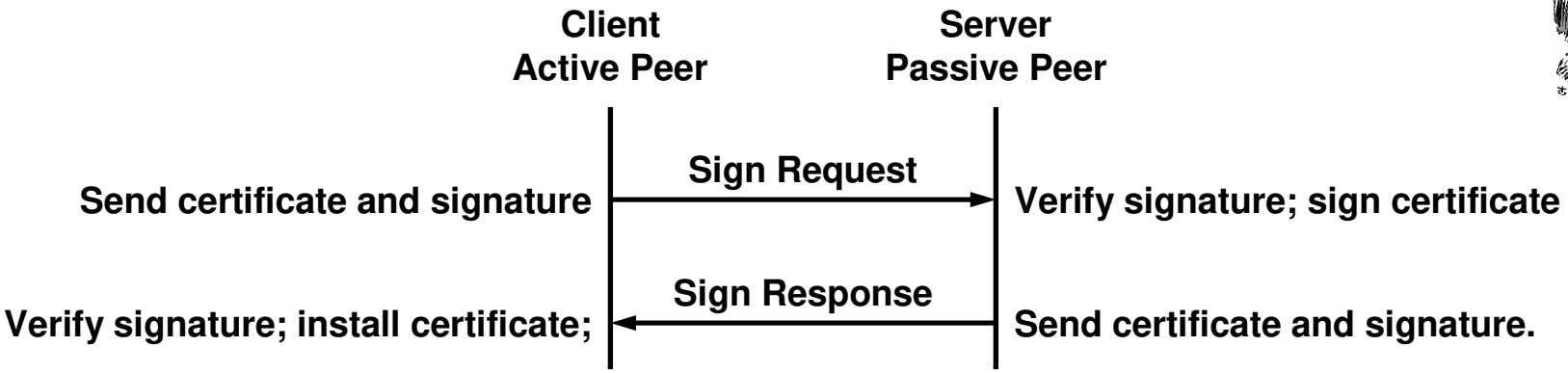


- Note: Association and Certificate messages not shown for clarity.

TAI leapsecond table

- The UTC leapsecond table contains the historic epoches, in NTP seconds, of leapsecond insertions since UTC began in 1972
  - An authoritative copy is on NIST NTP servers in `pub/leap-seconds`
  - It can be retrieved directly from NIST using FTP
  - It can be retrieved from a server or peer during the Autokey dance
  - If both peers have the table, only the most recent is used
  - NTP provides the seconds offset relative to TAI to the kernel
- Application program interface
  - The ntp_gettime() system call returns the current time and seconds offset relative to TAI
  - Currently, only FreeBSD, Linux and locally modified SunOS and Tru64 (Alpha) have modified kernels to support this interface

Sign certificate (client/server and symmetric modes)



- This is used to authenticate client to server, with server acting as de facto certificate authority using encrypted credential scheme TBD.
- Client sends certificate to server with or without signature.
- Server extracts request data and signs with server private key.
- Client verifies certificate and signature.
- Subsequently, client supplies this certificate rather than self-signed certificate, so clients can verify with server public key.

Key generation

- Key files are generated using the ntp_genkeys utility.
  - Normally, these files are kept at the generating host and need not be 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 MIME extensions
  - Filesets are generated for all OpenSSL digest/signature schemes.
- ntpkey_key_*hostname.filestamp*
  - Public/private encryption key
- ntpkey_cert_*hostname.filestamp*
  - X.509 certificate, either version 1 or version 3
- ntpkey_sign_*hostname.filestamp*
  - Public/private signature key; must agree with certificate key

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

Current progress and status

- NTP Version 4 architecture and algorithms
  - Backwards compatible with earlier versions
  - Improved local clock model implemented and tested
  - Multicast mode with propagation calibration implemented and tested
  - Distributed multicast mode protocol implemented and tested
- Autonomous configuration *autoconfigure*
  - Span-limited add/drop heuristic metric implemented and in test
  - Static stratum assignment hierarchy implemented and in test
- Autonomous authentication *autokey*
  - Completed integration with OpenSSL cryptographic library
  - Version 2 implemented in NTP Version 4 and tested
  - Automatic certificate trail search design in study
- IP Version 6 version nearing completion at Viagenie and UDel

Future plans

- Complete *autoconfigure* and *autokey* implementation in NTP Version 4
  - Complete and test dynamic certificate validation in Autokey
  - Design, implement and test dynamic stratum assignment in Manycast
- Complete IP Version 6 integration in NTP Version 4
- Deploy, test and evaluate NTP Version 4 in CAIRN testbed, then at friendly sites in the US, Europe and Asia
- Revise the NTP formal specification and launch on standards track
  - Participate in deployment strategies with NIST, USNO, others
  - Prosecute standards agenda in IETF, ANSI, ITU, POSIX
- Develop scenarios for other applications such as web caching, DNS servers and other multicast services

Further information

- Network Time Protocol (NTP): http://www.ntp.org/
  - 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