Survivable Real-Time Network Services

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

Distributed, real-time sensor networks

- Ad-hoc, wireless networking
  - Self organizing network infrastructure
  - Redundant sensors resist loss of data
  - Diversity paths resist jamming
- Autonomous system model
  - Sensors loosely deployed on battlefield or planetary surface
  - Sensors can be lost or destroyed or added during the mission
- Working assumptions
  - Lowest level is network connectivity and routing
  - Next level is security and time management
  - Higher levels define applications
- Game plan
  - Nearest equivalent sandbox is the Internet
  - Nearest equivalent sensor network is the Network Time Protocol (NTP)

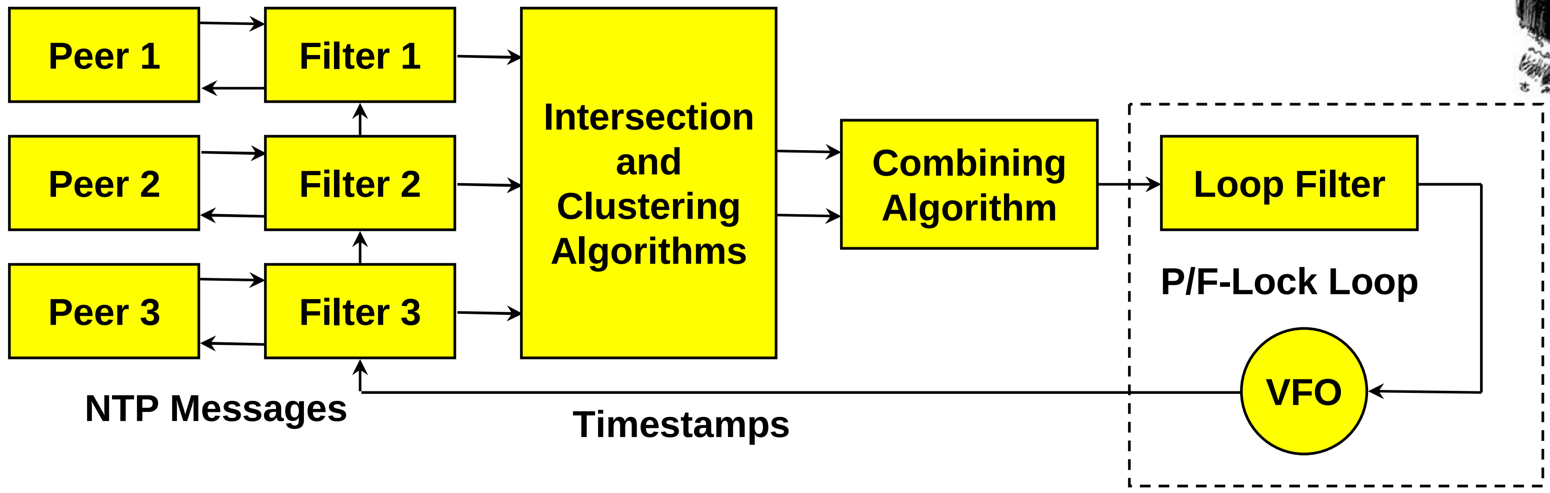
Autonomous system model

- Fire-and-forget software
  - Single software distribution can be configured, built and installed automatically on most host architectures and operating systems
  - Run-time configuration can be automatically determined and maintained in response to changing network topology and server availability
- Autonomous configuration
  - Survey nearby network environment to construct a list of suitable servers
  - Select best servers from among the list using mitigation algorithms and a statistical metric
  - Reconfigure the subnet for best accuracy with overhead constraints
  - Periodically refresh the list in order to adapt to changing topology
- Autonomous authentication
  - For each new server found, fetch and verify its cryptographic credentials
  - Authenticate each message received using an engineered protocol
  - Regenerate keys in a timely manner to avoid compromise

Goals and non-goals

- Goals
  - Robustness to many and varied kinds of failures, including Byzantine faults, destruction, malicious attacks and implementation bugs
  - Maximum utilization of Internet multicast services and protocols
  - Depend only on public values and certificates seeded in the network itself
  - Cryptographic authentication based on established public key infrastructure
- Non-goals
  - Administrative scoping – use it as it is, but provide TTL fences
  - Access control - this is provided by firewalls and address filtering
  - Privacy - all protocol values, including time values, are public
  - Protection against attacks on data values - this is provided by the NTP protocol
  - Non-repudiation - this can be provided by a layered protocol if necessary

NTP as sensor network



- Anatomy of a sensor
  - Multiple peers provide redundancy and diversity
  - Data filters select best from a window of offset/delay samples
  - Intersection algorithm discards falsetickers using Byzantine agreement principles
  - Clustering algorithm picks the best subset of truechimer peers
  - Combining algorithm, loop filter and variable frequency oscillator (VFO) implement hybrid phase/frequency-lock feedback loop which determines the system time

NTP autonomous systems

- Configuration and authentication and synchronization are inseparable
- Autonomous configuration (Manycast)
  - Centralized configuration management does not scale to large networks
  - Finding optimal topologies in large subnet graphs under degree and distance constraints is NP-hard
  - Formal methods may not produce good topologies in acceptable time
  - Span-limited, expanding-ring search strategies may be a good place to start
- Autonomous authentication (Autokey)
  - Centralized key management does not scale to large networks
  - Symmetric key cryptosystems require pairwise key agreement and persistent state in clients and servers
  - Servers cannot maintain persistent state for possibly thousands of clients
  - Public-key cryptosystems are too slow for good timekeeping
  - Solution may involve a combination of public and symmetric key cryptosystems

Autonomous configuration

- Manycast dynamic server discovery scheme
  - Clients discover servers using span-limited, expanding-ring search
  - To minimize overhead, clients send polls at intervals depending on the number of servers found
  - To minimize implosion, servers reply only if equal or lower stratum
  - Clients mobilize potential server associations for later mitigation
- Automatic quasi-optimal mitigation algorithms
  - NTP synchronization distance metric is used as a quality indicator
  - Distance is dominated by stratum, which is normally controlling
  - Byzantine agreement algorithm is used to detect and discard falsetickers
  - Clustering algorithm discards outlyers in repeating rounds until no more than a fixed minimum number of survivors remain

Static and dynamic stratum assignment

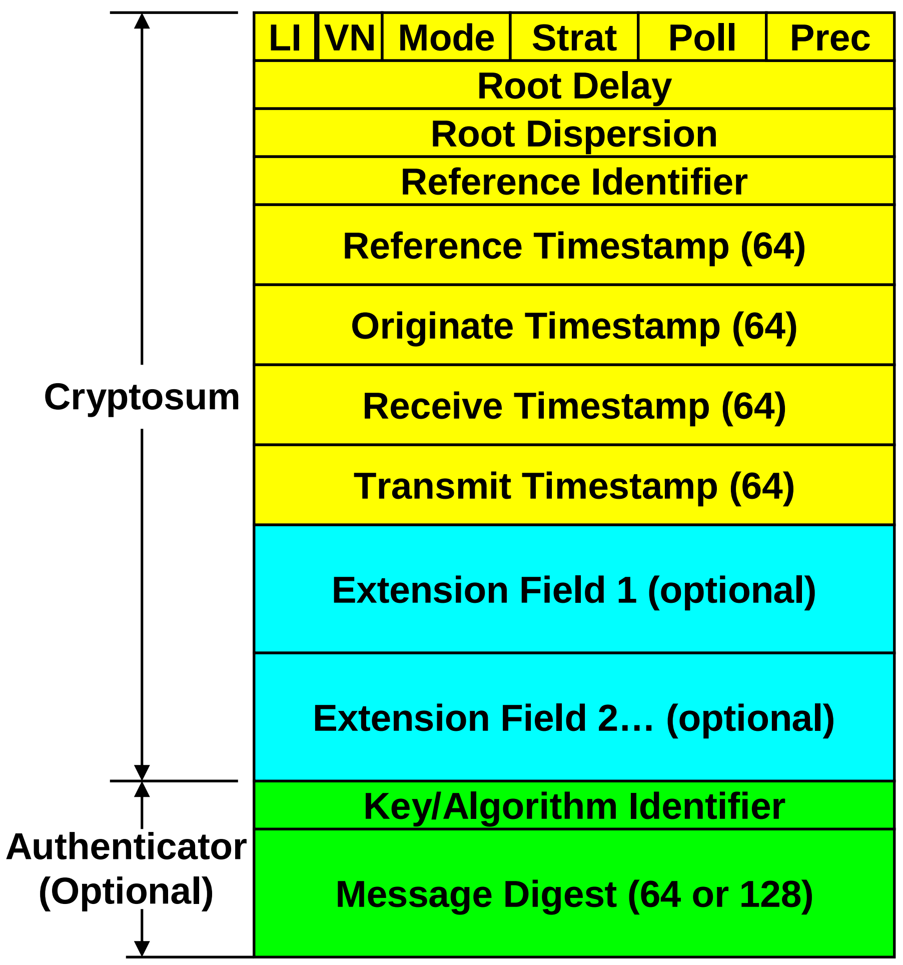
- Manycast subnet topology is automatically determined and maintained
  - Nodes are identically configured as both Manycast servers and clients
  - In addition, primary servers are configured for external references
  - In operation, primary servers peer with each other, secondary servers peer first with primary, then with secondary servers at the same stratum
- However, this scheme results in only two strata, so we assign each node static floor and ceiling strata
  - Peer distance below the floor is increased by the floor
  - Peer distance above the ceiling is increased to the maximum
  - Node stays between floor and ceiling, unless insufficient servers are found
  - Manycast search strategy stays the same
- However, the scheme needs to dynamically determine quasi-optimum floor and ceiling for each node
  - Ongoing research…

Autonomous timekeeping

- Autokey authentication and NTP synchronization protocols work independently for each peer
  - 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), but cryptographically bound do public key
- 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 primary 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
- Further research: trust metric based on number of proventic trails

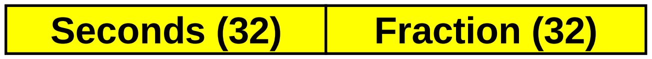
Autokey packet format

NTP Protocol Header Format (32 bits)



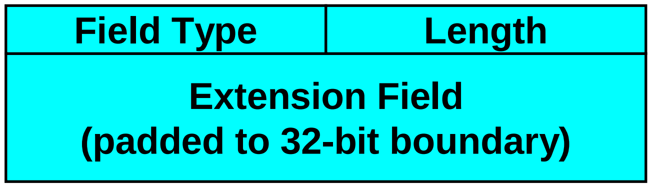
LI	leap warning indicator
VN	version number (4)
Strat	stratum (0-15)
Poll	poll interval (log2)
Prec	precision (log2)

NTP Timestamp Format (64 bits)



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

NTP v4 Extension Field

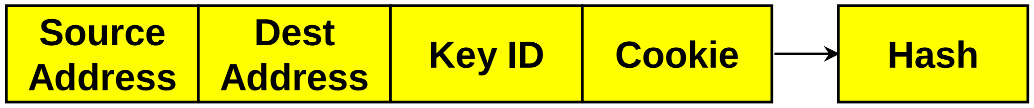


Last field padded to 64-bit boundary

🟨	NTP v3 and v4
🟦	NTP v4 only
🟩	authentication only

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

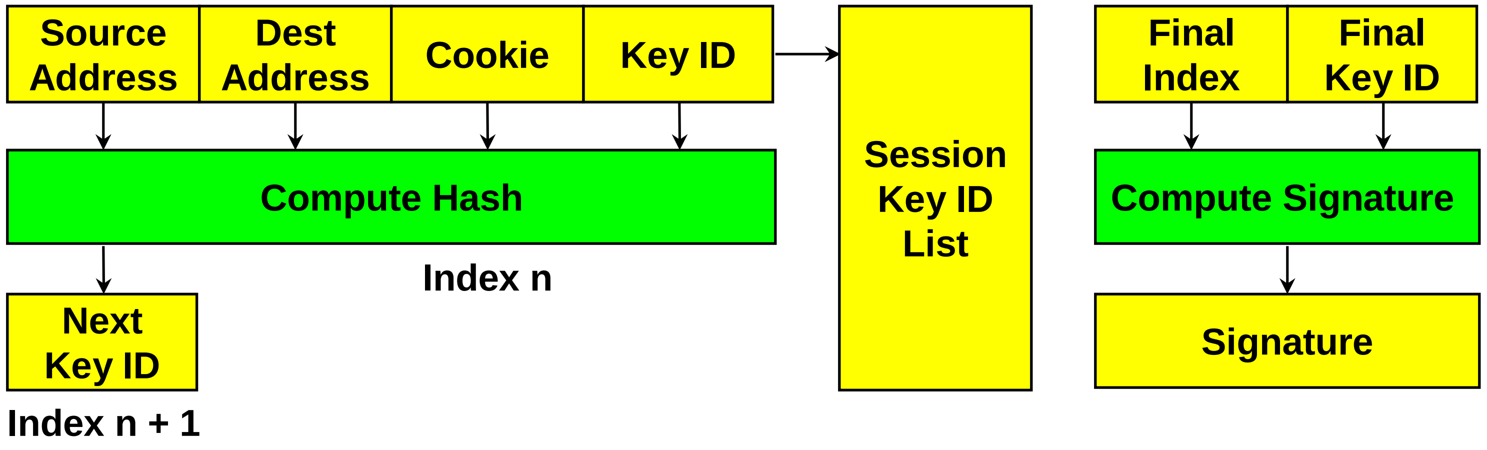
Session keys



NTPv4 Session Key

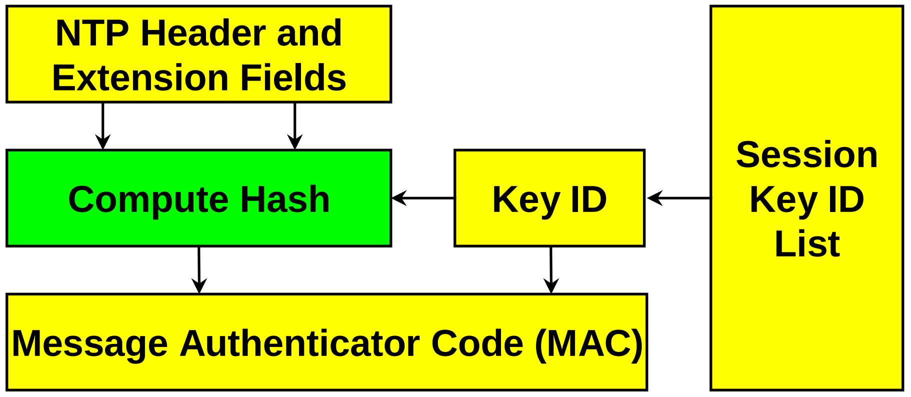
- NTPv4 session key includes four 32-bit words; NTPv6 session key includes ten 32-bit words
- Session key value is the 16-octet MD5 message digest of the session key
- Key IDs have pseudo-random values and are used only once. A special key ID value of zero is used as a NAK reply
- In any message including an extension field, the cookie has a public value (zero)
- In other cases the cookie is a hash of the addresses and a private value encrypted by the client public key

Generating the session key list



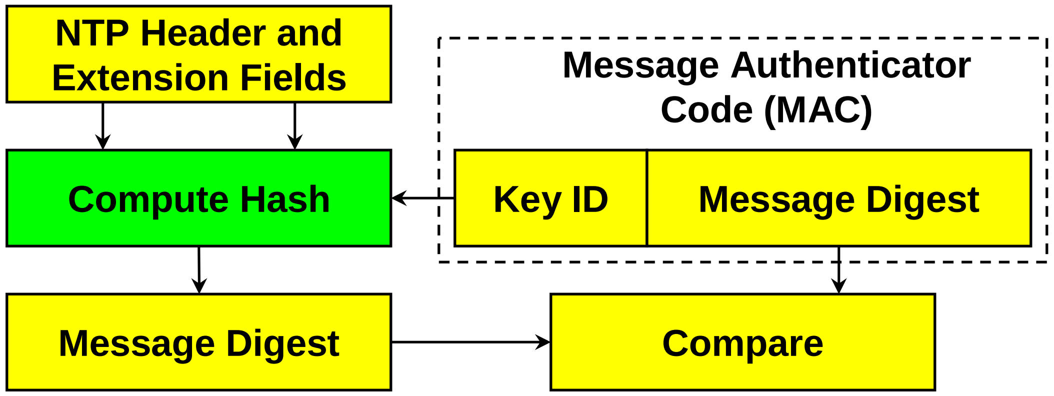
- Server rolls a random 32-bit seed as the initial key ID and selects the cookie. Messages with a zero cookie contain only public values
- Initial session key is constructed using the given addresses, cookie and initial key ID. The session key value is stored in the key cache
- Next session key is constructed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

Sending messages


NTP Header and Extension Fields
Compute Hash
Key ID
Session Key ID List
Message Authenticator Code (MAC)

- 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.
- Server uses the session key ID list in reverse order and discards each key value after use.
- Extension field containing the final index, key ID and signature is included in the first message from the list.
- 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


NTP Header and Extension Fields
Message Authenticator Code (MAC)
Compute Hash
Key ID
Message Digest
Message Digest
Compare

- Intent is not to hide the message contents, just verify where it came from and that it has not been modified in transit
- 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

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 filestamp 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

Lessons learned

- Authentication and time synchronization must presume nothing about the network other than network routing
  - Authentication and time synchronization are interdependent
  - Every value, public and private, in the network must have a proventic timestamp cryptographically bound to trusted time and certificate servers
  - Key refreshment means must be an integral component of the security model and authentication scheme
- Autonomous configuration must presume nothing about the network other than multicast routing and authentication
  - Everything moves in every direction, anything can lie and everything gets old
  - The protocols are most vulnerable to a clogging attack from the time a new player has been discovered until receiving its certificate
  - It is really difficult to balance effective clogging defense and rapid response to topology changes

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
  - NTP Version 4 with Manycast and Autokey is available for download from www.ntp.org
- Manycast autonomous configuration
  - Manycast mode with static floor/ceiling implemented and tested
  - Deployed for UDel and CAIRN testbeds in operational environment
- Autokey autonomous authentication
  - Autokey protocol with self-signed certificates implemented and tested
  - Key refreshment automated and tested under several intrusion scenarios, including replay and clogging
  - Deployed for UDel and CAIRN testbeds in operational environment

Future plans

- Refine Manycast implementation in NTP Version 4
  - Define dynamic floor/ceiling metric based on NTP stratum and distance
  - Implement dynamic configuration scheme based on this metric
- Refine Autrokey implementation in NTP Version 4
  - Define multiple-component, nonbinary trust metric
  - Implement dynamic certificate trail validation scheme based on this metric
- Deploy, test and evaluate NTP Version 4 daemon 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