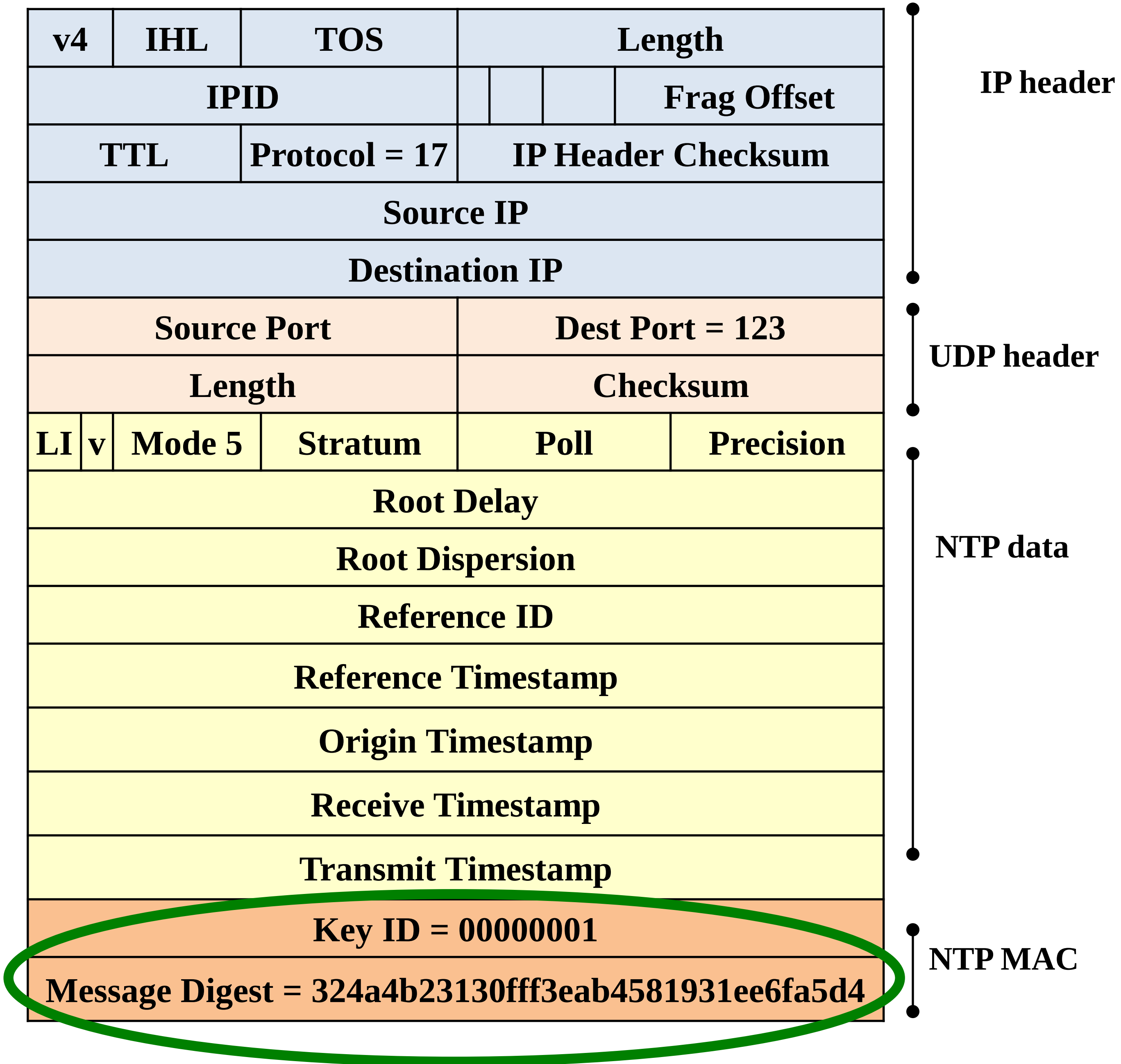
October 14, 2016

draft-aanchal4-ntp-mac-02

NTS F2F

Aanchal Malhotra
Sharon Goldberg

NTP Packet



v4
IHL
TOS
Length
IPID
Frag Offset
TTL
Protocol = 17
IP Header Checksum
Source IP
Destination IP
Source Port
Dest Port = 123
Length
Checksum
LI
v
Mode 5
Stratum
Poll
Precision
Root Delay
Root Dispersion
Reference ID
Reference Timestamp
Origin Timestamp
Receive Timestamp
Transmit Timestamp
Key ID = 00000001
Message Digest = 324a4b23130fff3eab4581931ee6fa5d4
IP header
UDP header
NTP data
NTP MAC

Why is MD5 (key||message) insecure?

RFC 5905 suggests MD5 (key||message) for NTP authentication.

Why is this bad?

- RFC 6151 says not to use MD5 for authentication this way.

- MD5 as a hash function is not collision resistant
  - Can find x1, x2 so that MD5(x1)=MD5(x2) in < 1sec
  - Using e.g. https://marc-stevens.nl/p/hashclash/

- MD5 (key||message) is vulnerable to length extension attack
  - Given y = MD5 (key || m1)
  - Can construct MD5 (key || m1 || m2) without knowing key!
  - https://en.wikipedia.org/wiki/Length_extension_attack

Updating NTP’s MAC: Potential Algorithms

Algorithm	Input Key-Length (bytes)	Output Tag Length (bytes)
Legacy MD5	16	16
HMAC-MD5 [RFC 4868]	16	16
HMAC -SHA224 [RFC 4868]	16	28 (truncated to 16)
CMAC (AES) [RFC 4493]	16	16
GMAC (AES) [RFC 4543]	16	16
Poly1305 (ChaCha20) [RFC 7539]	16	16

We include these just for performance comparison

NTP’s Performance Requirements for its MAC

1. **Constant Computational Latency:**
    - fewer clock cycles for computation is better
    - this directly translates to a reduction in jitter

2. **Throughput:**
    - NTP servers can deal with thousands of requests per second
    - NIST's NTP stratum 1 servers cater to 28,000 requests/second/server on an average

We perform two different benchmarks once with **AES-NI enabled** and the other time **disabled** on an x86_64, Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz with one core CPU.

Performance: Latency in Clock Cycles per Byte

Algorithm	with AES-NI	w/o AES-NI
Legacy MD5	16.0	15.7
HMAC –MD5	18.2	18.1
HMAC -SHA224	39.4	39.0
CMAC (AES)	6.6	11.3
GMAC (AES)	3.0	10.8
Poly1305-ChaCha20	14.4	15.0

Latency in terms of number of CPU cycles per byte (cpb)
when processing a 48-byte NTP payload.

Performance: Throughput in NTP packets per second

Algorithm	with AES-NI	w/o AES-NI
Legacy MD5	3118K	3165K
HMAC (MD5)	2742K	2749K
HMAC (SHA-224)	1265K	1267K
CMAC-AES	7567K	4388K
GMAC	16612K	4627K
Poly1305-ChaCha20	2598K	2398K

throughput in terms of number of 48-byte NTP payload processed per second

NTP-Specific Constraints with using GMAC

- NTP servers are stateless
- Symmetric key is shared by many servers (typically at the same stratum)

Why is this a problem?

Nonce Reuse vulnerability of GMAC : can recover authentication key

Nonce length = 96 bits
High probability of collision after **2^48** messages (birthday bound)

NTP server is stateless - does not know when to refresh keys for a client

An MiTM can replay messages and exhaust this number very fast

Recommendations

- GMAC - best performance but is surrounded by several security issues
- HMAC - poor performance (lack of h/w support), but better security
- CMAC - reasonable choice between performance and security requirements

We recommend CMAC for now!

Algorithm	Performance	Security
GMAC	best	weak
CMAC	medium	good
HMAC	poor	good

Other potential MAC candidates with nice features

- **SipHash** - Optimized to work with short messages
- **GCM-SIV** (still an internet draft) - Nonce misuse resistant
- Other **CAESAR AEAD** competition candidates