IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

LATe: A Lightweight Authenticated Time Synchronization Protocol for IoT

Renzo E. NAVAS, Laurent TOUTAIN

Table of contents

Why do we need time synchronization?

- Timestamp measurements (application data)
- Validate cryptographic credentials (e.g. OAuth tokens)
- Is a way to assure *freshness* of transactions

why do we need secure time synchronization?

Modified from Source: Ben Stansall / AFP - Getty Images file.

What happens if the source of time of a system is not secure?

- None of the aforementioned use cases could be guaranteed (i.e. can be attacked).

Security bootstrapping problem

- Many security services rely on synchronized time.
- How to securely synchronize time?
- A *leap of faith* needed... (make it short)

State of the Art: Standards

- Patches to well-known standards:
  - *Annex K* for Precision Time Protocol (PTP). Network Time Protocol (NTP) *symmetric key authentication scheme* and *Autokey*.
- IETF Network Time Security (NTS) [1] work-in-progress.
- Current Standards are **not** optimized for IoT.
  - e.g. NTS at least 4 messages (2 cookie + 2 sync). Not compact representations. Focused on precision.
- Work done for Wireless Sensor Networks:
  - Similar constraints. Lack of standard, will compare later.

LATe Synchronization Protocol

- **Functional Goal**: Provide a Time Client with the time representation from a *trusted* Time Server.
- **Non-goal**: Precise time synchronization.

- **Security Goals**:
  - Data Authentication/Integrity
  - Freshness (i.e. no replay attack)
- **Design Goals**:
  - Lightweight (minimize **energy**).
  - Agnostic to underlying layers
  - Cryptographic agility
  - Built upon standards

LATe Messages Exchange

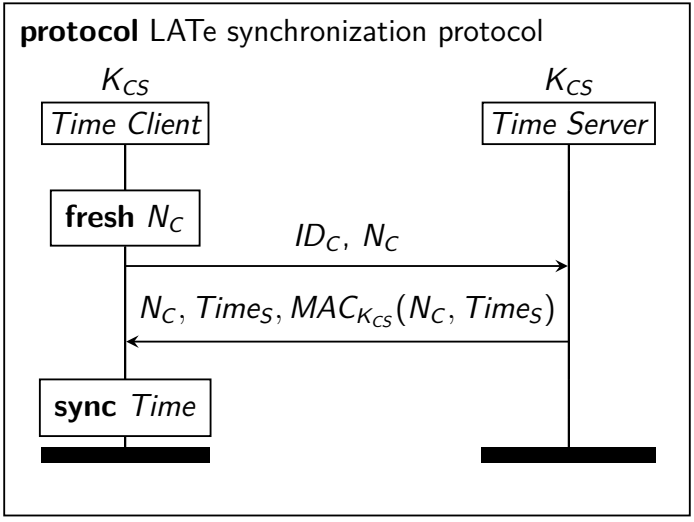


Figure: LATe Synchronization Protocol Diagram.

Time Synchronization Calculation

- $RTT = T_{Msg2} - T_{Msg1}$

$$Time_{Client} = Time_S + \frac{RTT}{2}$$

- Uncertainity $\pm\frac{RTT}{2}$

Message Encoding: IoT Standards!

- **CBOR**: Concise Binary Object Representation [RFC7049] for Data representation
- **COSE**: CBOR Object Signing and Encryption [RFC8152] for Security Services (i.e. the MAC'ed response)

Message Encoding: IoT Standards!

- Application: Two new CBOR Maps (Key-Value pairs)
  - TIC Information
  - TOC Response
- Security: TOC Response will be authenticated using a COSE_Mac0 structure

Parameter name	CBOR Key	Value Type	Description
nonce	4	binary string	A random nonce
kid	5	binary string	Key-ID is an opaque value and identifies the cryptographic key to be used in the response
alg (optional)	6	int	Identifies the cryptographic algorithm to be used in the response
server (optional)	7	string	Identifies the intended Server for time synchronization (Absulute URI)

Table: CBOR Map "TIC Information" object definition

```
{
  nonce:h'73616E206C6F7265',
  kid   :h'0001',
  alg   :4  /*HMAC w/SHA-256 truncated to 64 bits*/
}
```

Listing 1: *TIC Information* on CBOR diagnostic notation.

```
D83B                    # tag(59) (TIC Info.)
 A3                      # map(3)

  04                     # unsigned(4) (=nonce)
  48                     # bytes(8)
  73616E206C6F7265       # Nonce Value

  05                     # unsigned(5) (=kid)
  42                     # bytes(2)
  0001                   # Key-ID Value

  06                     # unsigned(6)  (=alg)
  04                     # unsigned(4)
```

Listing 2: *TIC Information* CBOR object (19 Bytes).

... what about the security goals?

Formal Method ~~(vs. provable secure)~~

- Scyther tool
- Dolev-Yao attacker model, black box cryptography
- Automatic proofs

Formal Method Verification: Results

Claim			Status		Comments
LATe	I	LATe,I1 Nisynch	Ok	Verified	No attacks.
		LATe,I2 Niagree	Ok	Verified	No attacks.
		LATe,I3 Alive	Ok	Verified	No attacks.
		LATe,I4 Weakagree	Ok	Verified	No attacks.

Figure: Scyther Results

- Data authentication-integrity: OK.
- Freshness?
  - Not enough to prove it!
  - Must prove *injective synchronization* property.

.

From a model to real world:

- simplifications, abstractions, generalizations.
- a model does not map 1 to 1 with reality
- can the use case live with that?
  - NASA Apollo Moon Missions where ok with newtonian physics and simplifications (only one gravitational body considered; jupiter, venus off) 
- can security?
  - your system is secure.. with 99% provability.
  - how a *proof* on a model translates to reality? means that axioms and logic of deduction are valid (leap of faith?).
- epistemology, philosophy of science [2] (inductivism, falsifiability)

Is never enough... real world

- Real Nonces/Crypto
  - Finite length: birthday attack, pre-play attack.
  - True Random? ~~YES~~/NO
  - Avoid randomness altogether (auth. 1st msg. LATe v2)
- Real attackers
  - real humans, AI-powered cyberattacks.
  - attacker model enough?
- Real systems
  - software, implementations, bugs
  - hardware, internal time representation, bugs
  - side-channel attacks

why is LATe lightweight?

Comparison to other protocols I

Protocol	Nr. of Msg.	Avg. msg. size (Bytes)	Total Bytes	Crypto Ops. at Node
SPS [6]	2	21	41	$1 \times MAC$ $1 \times Nonce$
E-SPS [6]	3	17	50	$1 \times MAC$ $1 \times Nonce$
TinySeRSync [7]	2	21	42	$2 \times MAC$
Guo et al. [8]	3	39	116	$2 \times Signature$ $1 \times MAC$
E-SPBS [9]	3	35	104	$1 \times Signature$ $1 \times Nonce$
LATe	2	15	30	$1 \times MAC$ $1 \times Nonce$
LATe v2	2	15	30	$2 \times MAC$ $1 \times Nonce$

Figure: Secure Time Synchronization protocols baseline comparison

- **Standards**: NTS 268 Bytes; PTP-K 512 Bytes.
- **WSN Best**: SPS 41 Bytes.
- **LATe**: 30 Bytes. 25% less TX/RX than SPS.
  - Minimizing energy consumption is our priority.
  - Radio TX/RX is the most energy consuming activity.
  - $\mapsto$ Minimizing Radio TX/RX is our priority.
- Trade-off energy vs time precision.

- Gap protocol designers and cryptography
  - Computer verif tools shortens the gap. TLS 1.3 design.
- IoT needs time synchronization, but no standard exists.
- LATe offers authenticated end-to-end, coarse-grained solution.
- Go for an IoT secure time sync. open protocol?
  - IETF-draft `https://datatracker.ietf.org/doc/draft-navas-ace-secure-time-synchronization/`

Thank you!
Questions?

Appendix

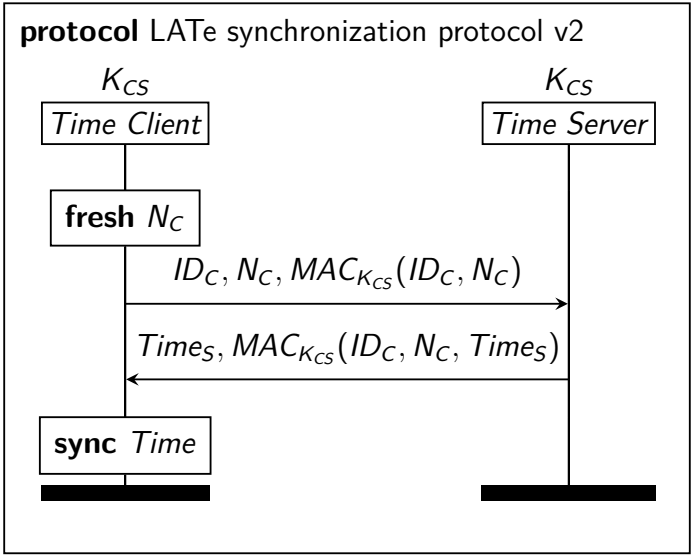


Figure: LATe Synchonization Protocol V2.

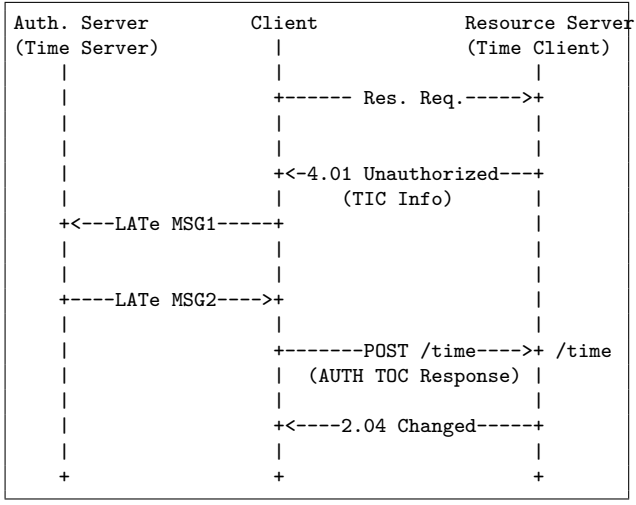


Figure: LATe on IETF ACE Scenario 1

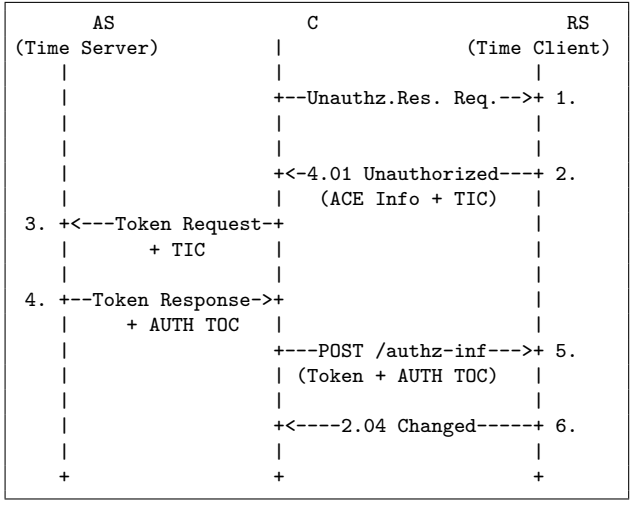


Figure: LATe on IETF ACE Scenario 2

Parameter name	CBOR Key	Value Type	Description
time	3	unsigned int	Time representation information
nonce	4	binary string	A random nonce

Table: CBOR Map "TOC Response" object definition

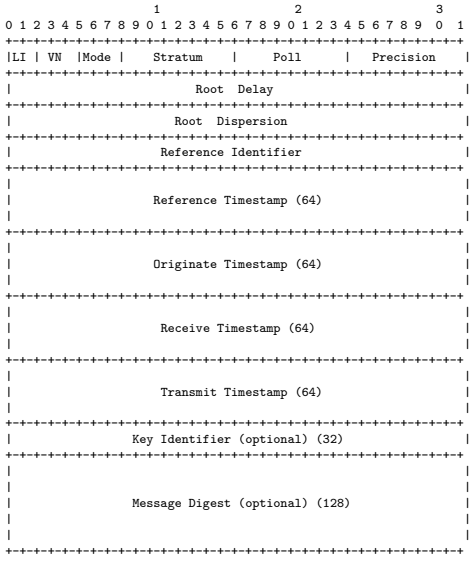
```
Header: Changed (Code=2.04)
Content-Type: "application/late+cose;
cose-type=cose-mac; late-type=toc"
Payload:
{
        protected  : {
                kid: h'0001',
                alg: 4 /* HMAC w/ SHA-256 truncated to 64 bits */
        },
        payload    : {
                time  : 1477307841,
                nonce : h'73616e206c6f7265'
        },
        tag        : h'36f5afaf0bab5d43'
}
```

Figure: COSE-MACed 'TOC Response' in CBOR diagnostic notation

Baseline comparison assumptions

No overhead for metadata, and we assume the following data sizes: *Timestamp* representation is 4 bytes, *Node Identity* is 2 bytes, a *Nonce* is 8 bytes, and a *MAC* is 8 bytes. In E-SPBS an ECDSA signature is 48 bytes; In Guo et al. we assume an Unspecified Signature being of 16 bytes, and non-cryptographic hash 16 bytes; In Ganeriwal2008 and Sun2006 syn-ack information of 1 byte.

An NTP Packet Header



- 44 Bytes Minimum
- ~~Batteries~~ Security not included

Cryptographic Recommendations

- **MAC**: COSE_Mac0 message recommended algorithms:
  - `HMAC w/SHA-256 truncated to 64 bits` (256-bit pre-shared-key) (MUST)
  - `AES-CBC-MAC` (128-bit key)
  - `AES-CMAC` (128-bit key)
- **Nonce**: At least 64-bits. TRNG or good seed for pseudo-RNG.
  - 64-bit probability of collision around 2^{-32} for 2^{16} (65 536) uses of the protocol
  - **50%** for 2^{32} uses (4 294 967 296)
- **Real-Time Clocks (RTC)**: The Time Client must have a RTC. (Disclaimer: Raspberry Pi does not have)

Authentication Notions Cremers [3] I

Synchronization

A security property requiring that all protocol messages occur in the expected order with the values as expected.

Injectivity

Requires that each run of an agent executing the initiator role corresponds to a *unique* run of its communication partner running the responder role.

Injective Synchronization

An Initiator I considers a protocol injectively synchronizing if the protocol (non-injective) synchronizes and each run of I corresponds to a *unique run* of Responder R

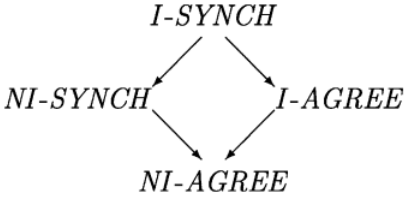


Figure: Hierarchy of security properties

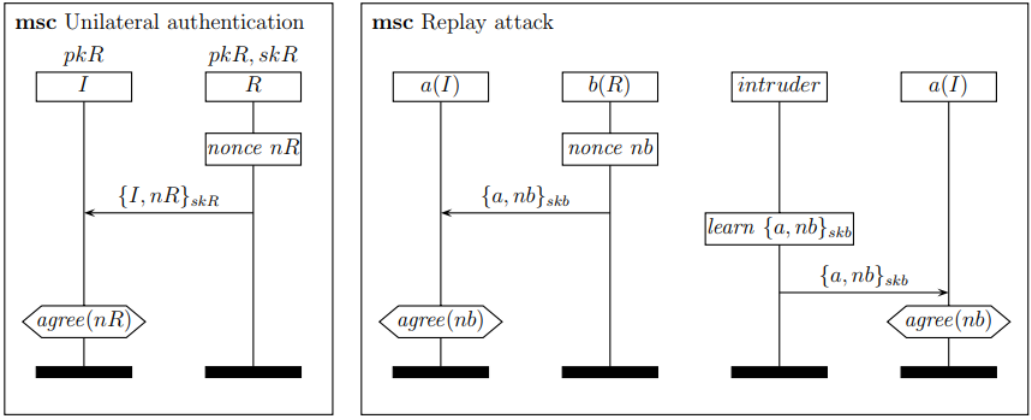


Fig. 1. An authentication protocol that is vulnerable to a replay attack.

Further Reading I


[1] Daniel Fox Franke , Dieter Sibold and Kristof Teichel
Network Time Security for the Network Time Protocol
IETF, draft-ietf-ntp-using-nts-for-ntp-11, March 05, 2018.


[2] Cormac Herley ; P. C. van Oorschot
SoK: Science, Security and the Elusive Goal of Security as a Scientific Pursuit
Security and Privacy (SP), 2017 IEEE Symposium on, 22-26 May 2017 .


[3] Cremers, C. J. F. et. al.
Injective synchronisation: An extension of the authentication hierarchy
Theoretical Computer Science Journal no. 1-2, 2006.