Securing IoT Communication: The Path from SSL to DTLS & Compact TLS

Hannes Tschofenig

hannes.tschofenig@arm.com

<*Technology X*> was never designed with <*Feature Y*> in mind

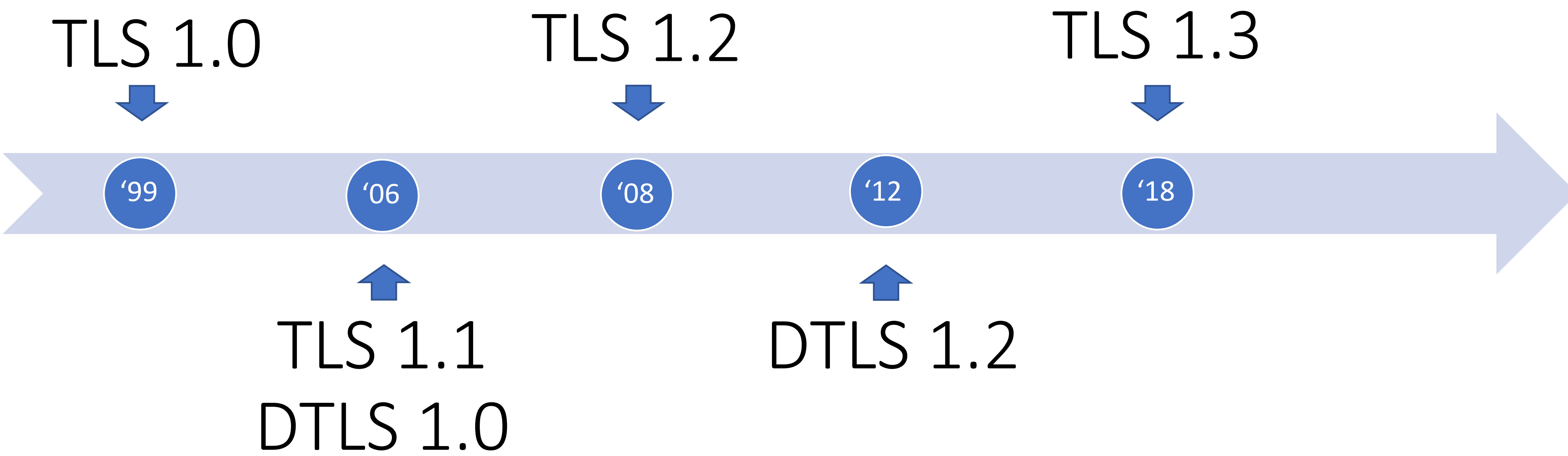
Design of SSL

- SSL 2.0 released early 1995. SSL 3.0 released in ‘96. SSL 1.0 never released.
  - Acorn Computers made their ARMv3 RISC computer available at that time.
  - Most users access the Internet using a slow, dial-up modem.
  - Nokia 8110 launched in ‘96.
- SSL provided communication security and used asymmetric crypto for authentication to secure web-based communication.

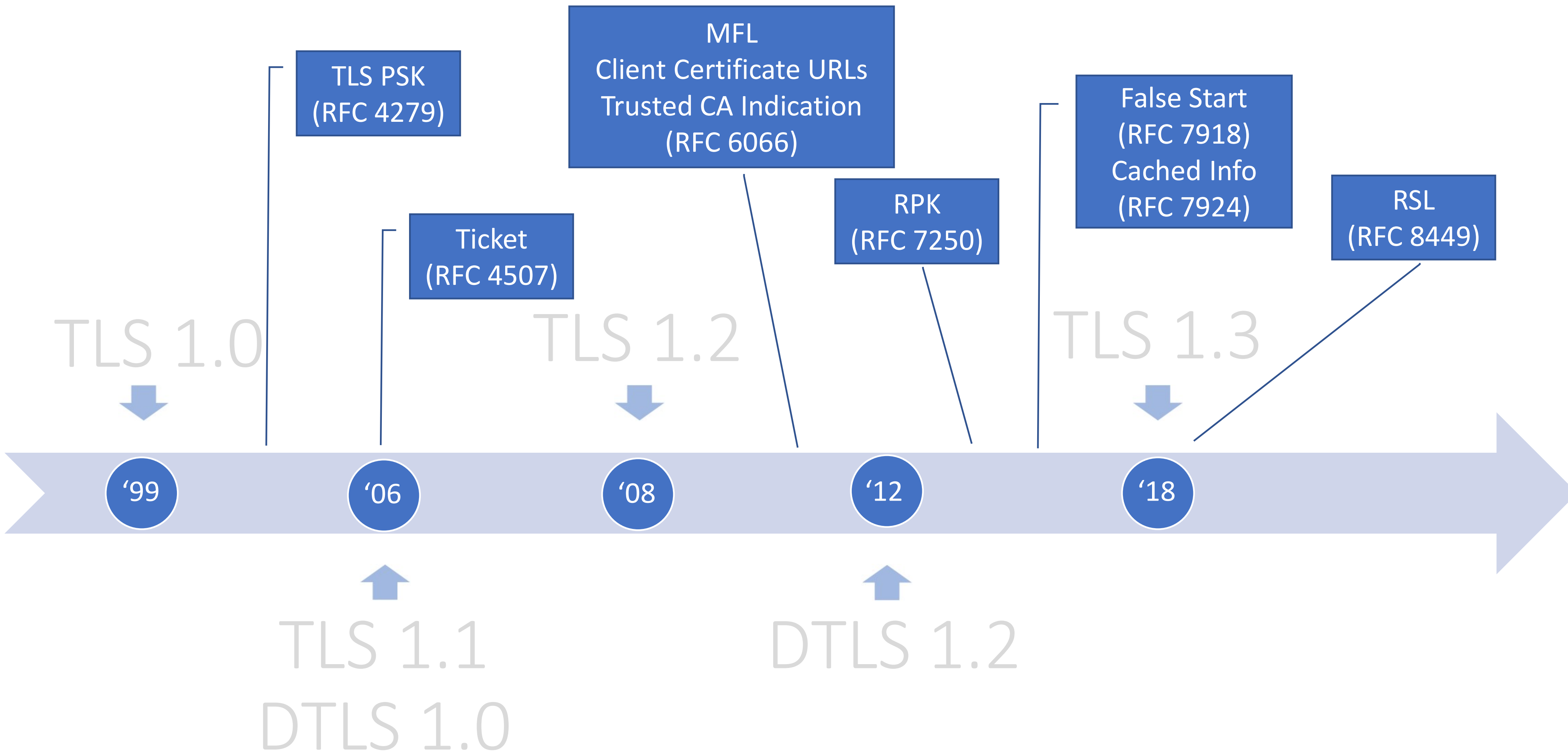
BeBox used two PowerPC 603 processors running at 66 or 133 MHz

Pictures from https://www.computerhistory.org/timeline/1995/

Timeline of IETF TLS/DTLS Specifications



Timeline of IoT-relevant Extensions



TLS became a target of attacks

- TLS 1.0, 1.1, and 1.2 fixed security problems and added new cryptographic algorithms ➔ Foundation unchanged.
- With the success of TLS, the interest in attacking it increased.
- With RFC 7925 and RFC 7525 we have TLS & DTLS profiles that exclude problematic algorithms and configuration.

Internet Engineering Task Force (IETF)
Request for Comments: 7925
Category: Standards Track
ISSN: 2070-1721

H. Tschofenig, Ed.
ARM Ltd.
T. Fossati
Nokia

July 2016

**Transport Layer Security (TLS) /
Datagram Transport Layer Security (DTLS)
Profiles for the Internet of Things**

Abstract

A common design pattern in Internet of Things (IoT) deployments is the use of a constrained device that collects data via sensors or controls actuators for use in home automation, industrial control systems, smart cities, and other IoT deployments.

This document defines a Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) 1.2 profile that offers communications security for this data exchange thereby preventing eavesdropping, tampering, and message forgery. The lack of communication security is a common vulnerability in IoT products that can easily be solved by using these well-researched and widely deployed Internet security protocols.

Why TLS 1.3?

Value-add:

1. **Performance** improvement, and
2. better **privacy** protection
   (see BCP 188 ‘Pervasive Monitoring Is an Attack’)

The Register®
Biting the hand that feeds IT

Security

Hurrah! TLS 1.3 is here. Now to implement it and put it into software

Which won't be terrifyingly hard: it's pretty good at making old kit like the way it moves

By Richard Chirgwin 27 Mar 2018 at 00:58 16 SHARE ▼

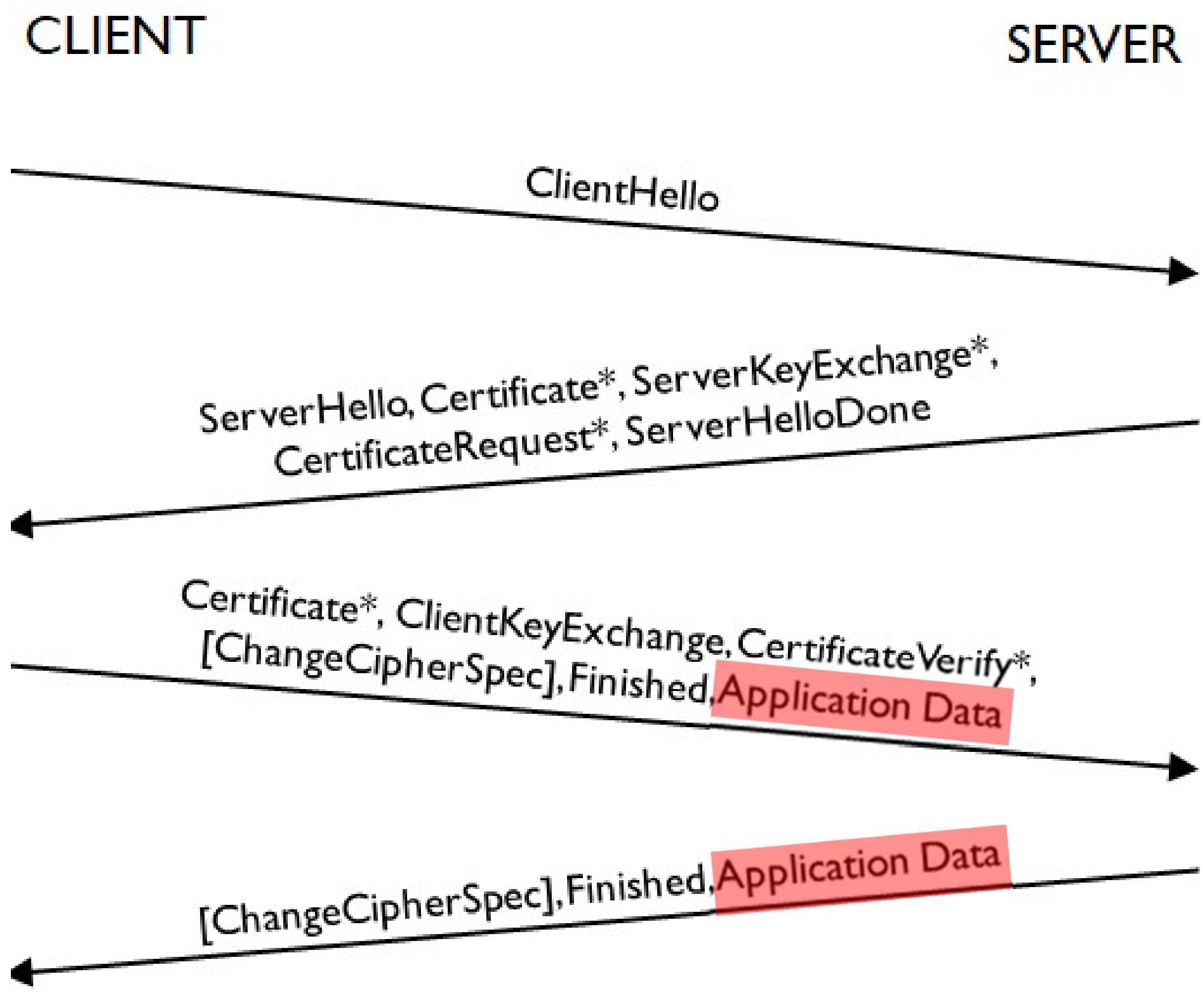
Comparing TLS/DTLS 1.2 vs 1.3

- **Roundtrips** ↓
- **Features** ↑
- **Message sizes** ↓
- **Code Size** ↑
- **Energy** ↓
- **Cryptographic operations** ↑
- **Memory** =

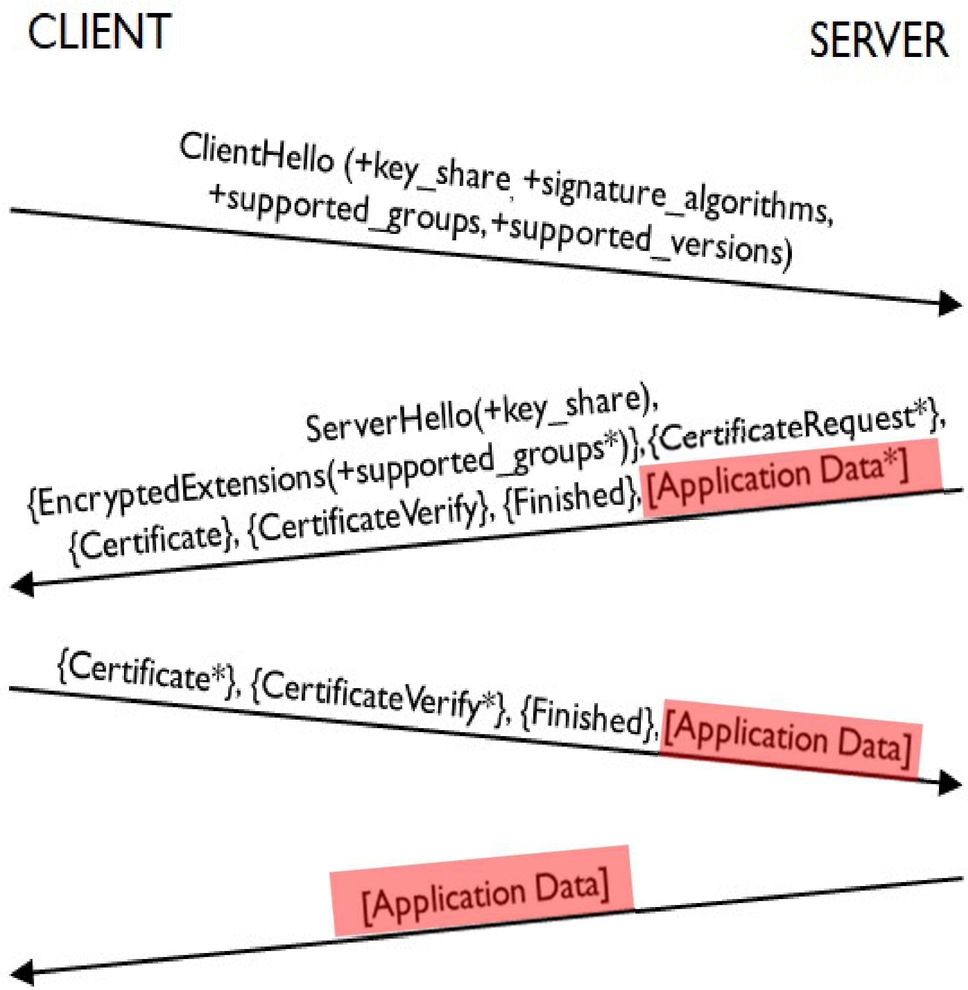
Thanks to my collaborators Emmanuel Baccelli and Gabriele Restuccia for their help with this investigation.

Performance

TLS 1.2 Full Handshake



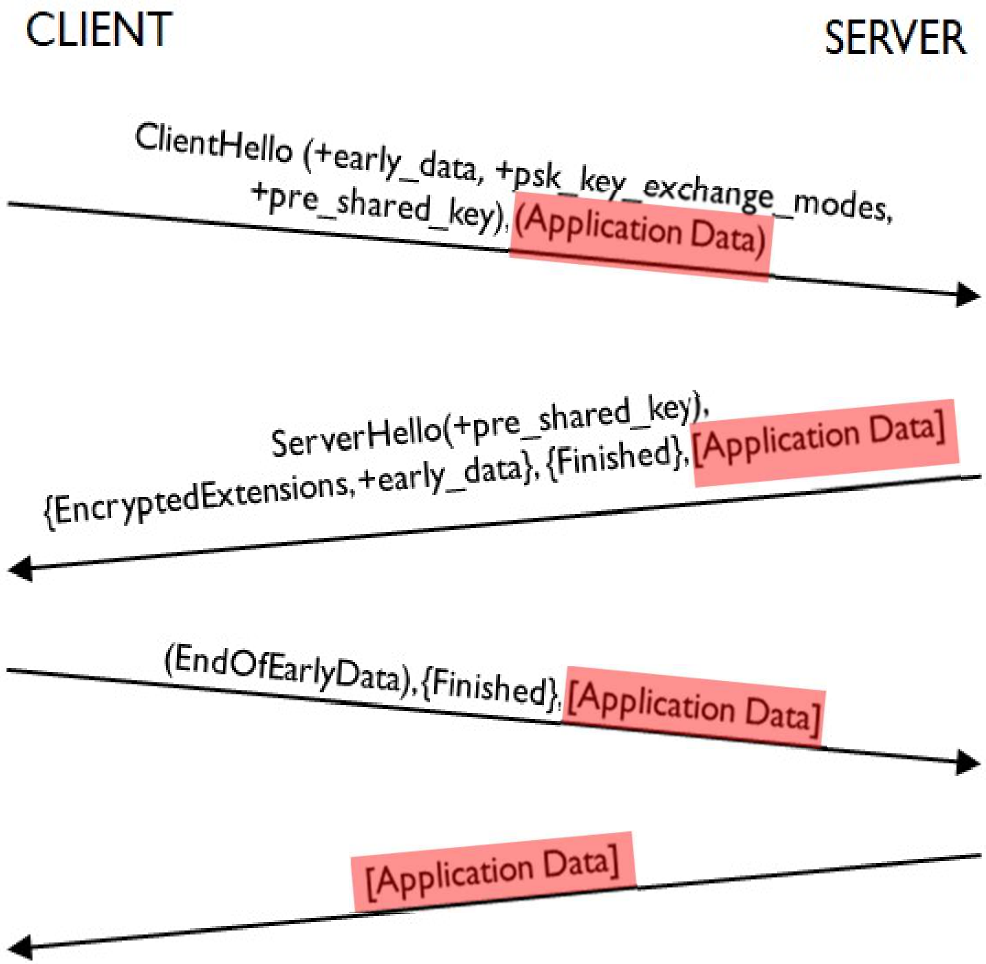
Optional messages indicated via (*). Finished and application data msgs are encrypted.



TLS 1.3 Public Key based Authentication

Legend: *: optional message, []: Not a handshake message, {}: Encrypted message

TLS 1.3 0-RTT



Legend:
*: optional message
(), {}, and [] indicates messages protected using different keys

What should be optimized for?

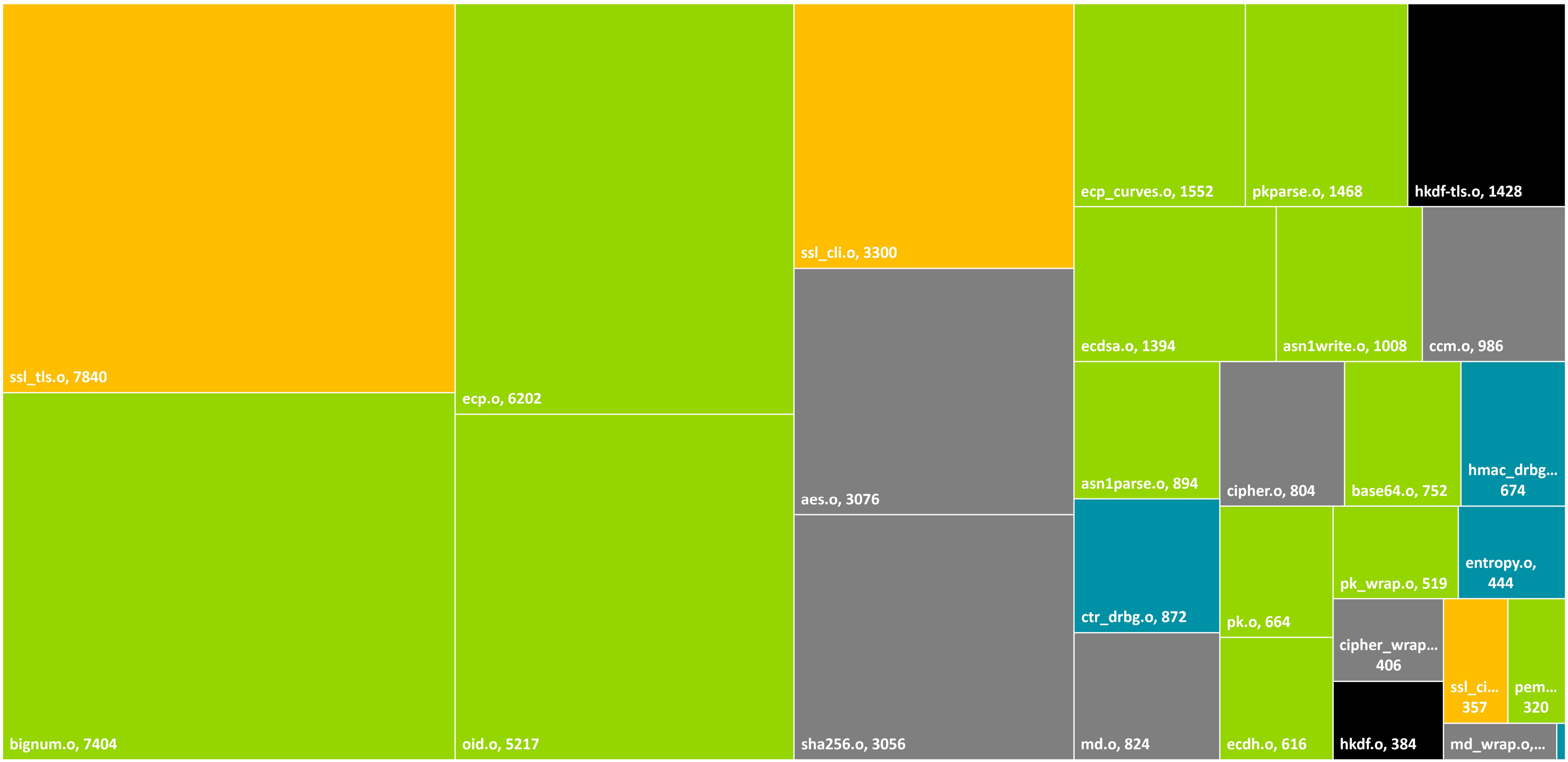
- Latency
- Code size
- RAM utilization
- CPU Performance
- Power consumption
- Over-the-wire bandwidth
- Cost

Unfortunately, there are **tradeoffs**.

Examples:

- Optimizing crypto for CPU speed typically increases RAM utilization and code size.
- Adding a new compression algorithm adds code size, might require more RAM, requires more CPU cycles and adds development cost but reduces the over-the-wire overhead.

Flash Size in Mbed TLS: TLS 1.3, ECDSA-ECDHE (P2561), AES-128-CCM



Almost exclusively used by AES implementation.

	MbedTLS Heap	MbedTLS Stack	WolfSSL Heap	WolfSSL Stack
TLS 1.2 PSK AES-128-CCM	5749	8772	3496	12
TLS 1.2 ECC AES-128-CCM	13879	8786	7162	12
TLS 1.2 ECC AES-256-GCM	20603	8780	7922	12
TLS 1.3 PSK AES-128-CCM	6757	8764	6224	12
TLS 1.3 ECC AES-128-CCM	12914	8778	9458	12
TLS 1.3 ECC AES-256-GCM	14366	8780	10250	12
DTLS 1.2 PSK AES-128-CCM	5975	8772	5340	12
DTLS 1.2 ECC AES-128-CCM	14414	8786	8540	12
DTLS 1.3 PSK AES-128-CCM	6934	8764	N/A	N/A
DTLS 1.3 ECC AES-128-CCM	13248	8778	N/A	N/A

RAM Utilization

baremetal lowers the RAM requirements to less than 10 Kb for DTLS with ECDHE-ECDSA with AES-128-CCM using TinyCrypt, combined with a more efficient management of send and receive buffers, as well as an improved handling of certificates and of the DTLS retransmission buffers.

Energy Measurements
(Values in Millicoulomb)

	1.2	1.3	Diff
Mbed TLS - TLS with PSK, AES-128-CCM	2.7	2.3	0.4
Mbed TLS - TLS with ECDHE-ECDSA, AES-128-CCM	89.6	63.4	-26.2
Mbed TLS - DTLS with PSK, AES-128-CCM	2.0	5.3	3.3
Mbed TLS - DTLS with ECDHE-ECDSA, AES-128-CCM	87.5	73.3	-14.2
WolfSSL - TLS with ECDHE-ECDSA, AES-128-CCM	76.3	77.5	1.2
WolfSSL - DTLS with PSK, AES-128-CCM	1.9	N/A	N/A
WolfSSL - DTLS with ECDHE-ECDSA, AES-128-CCM	77.0	N/A	N/A

The DTLS 1.2 implementation allows multiple DTLS records to be packed into a single datagram thereby reducing the required bandwidth, which leads to lower energy consumption.

Bandwidth

- The biggest contribution to the handshake size is coming from certificates.
- Contributors to the size include:
  - Long Subject Alternative Name field.
  - Long Public Key and Signature fields.
  - Can contain multiple object identifiers (OID) that indicate the permitted uses of the certificate
  - Many intermediate certificates

- Lots of solutions available:
  - Sensible configuration and deployment options.
  - ECC instead of RSA certs
  - Client Certificates URLs
  - Caching Certificates
  - Compressing Certificates
  - Suppressing Intermediate Certificates
  - Raw Public Keys
  - New Certificate Types (e.g. CBOR Web Token, Weave digital certificates)

Privacy Protection

Privacy Protection

TLS 1.2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TLSv1.2	183	Client Hello
2	0.000449	127.0.0.1	127.0.0.1	TLSv1.2	162	Server Hello
3	0.000675	127.0.0.1	127.0.0.1	TLSv1.2	366	Certificate
4	0.000776	127.0.0.1	127.0.0.1	TLSv1.2	111	Certificate Request
5	0.000808	127.0.0.1	127.0.0.1	TLSv1.2	75	Server Hello Done
6	0.013619	127.0.0.1	127.0.0.1	TLSv1.2	366	Certificate

TLS 1.3

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TLSv1.3	354	Client Hello
2	0.000332	127.0.0.1	127.0.0.1	TLSv1.3	194	Server Hello
3	0.000535	127.0.0.1	127.0.0.1	TLSv1.3	140	Application Data
4	0.000630	127.0.0.1	127.0.0.1	TLSv1.3	200	Application Data
5	0.000875	127.0.0.1	127.0.0.1	TLSv1.3	200	Application Data

+PFS, -key transport, +padding, +various unlinkability properties

Not everyone is happy…

- Eavesdropping and intercepting TLS handshakes became much more difficult.
- Claimed to cause [problems for enterprise network management](#).
- Resulted in delayed publication of the TLS spec and polarized IETF engineering community.
- Additional extensions are being developed that even [encrypt the Server Name Indication (SNI)](#).

Security

World celebrates, cyber-snoops cry as TLS 1.3 internet crypto approved

Forward-secrecy protocol comes with the 28th draft

By Kieren McCarthy in San Francisco 23 Mar 2018 at 21:53 57 SHARE ▼

Article reference: https://www.theregister.co.uk/2018/03/23/tls_1_3_approved_ietf/

TLS was primarily used for protecting protocols running on top of TCP, like HTTP ...

but what about IoT protocols?

Eclipse IoT Developer Survey 2019

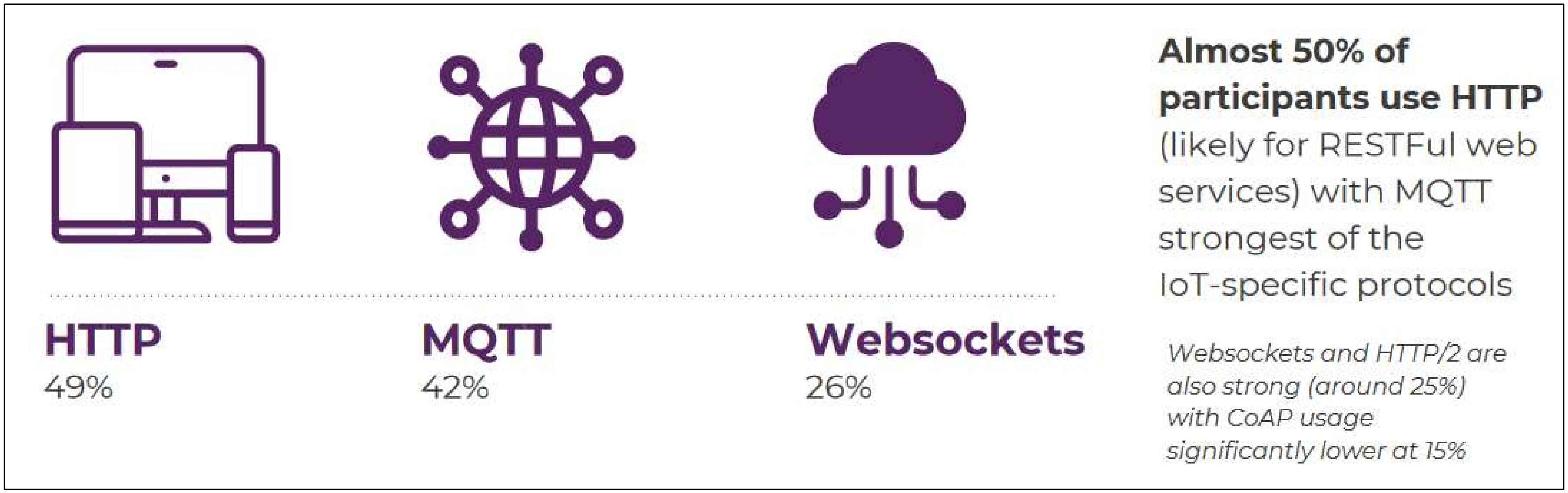


Figure copied from https://iot.eclipse.org/community/resources/iot-surveys/

Note: The survey may be biased due to the size of the poll and the way it is advertised.

The IoT standards community is split when it comes to protocols

CoAP vs. MQTT vs. HTTP

Trend: Protocol developments have made all three very similar

All three use TLS/DTLS for communication security

CoAP was initially designed to run over UDP and DTLS was used to secure it.

According to [HomeGateway], the mean NAT binding timeouts is 386 minutes for TCP and 160 seconds for UDP.

Shorter timeout values → more keepalive messages

IoT devices that sleep a lot, handshake needs to be repeated.

[HomeGateway] Haetoenen, S., et al., "An experimental study of home gateway characteristics", Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, November 2010.

How can we skip the handshake? Connection ID (CID)

- If possible, handshakes should be avoided.
- CID is a new field in the record layer that allows untangling the security context lookup from the 5 tuple.
- Handshake extension to negotiate feature, i.e., optional to use.
- Specification available for DTLS 1.2 and DTLS 1.3.
  - DTLS 1.2 is close to publication as an RFC.
  - The DTLS 1.3 CID solution offers better unlinkability capabilities.
- Performance improvements are significant (for a certain class of IoT devices).

From Standards to Implementations

Code

- Support for TLS 1.3 is already pretty good.
- Certs and PSKs are well supported.
- Many of the IoT performance improving extensions are not implemented.
- Note: Server-side support for an extension is required as well.

Feature	Mbed TLS	Tiny DTLS	WolfSSL	Matrix SSL	CycloneSSL	axTLS	BearSSL
TLS 1.2							
TLS 1.3							
DTLS 1.2							
DTLS 1.3							
TLS 1.2 PSK							
TLS 1.2 RPK							
TLS 1.2 Cert							
OCSP stapling							
TLS/DTLS 1.2 ATLS							
DTLS 1.2 CID							
TLS 1.2 Ticket							
MFL							
RSL							
TLS Cached Info							
Client Cert URLs							
Trusted CA Ind.							
False Start							

Table shows implementations that are officially released; not prototyping code.

More Standards in the search for more “lightweightness”

LAKE and cTLS

Compact TLS (cTLS)

A compression of the TLS/DTLS handshake (+ record layer):

- Change encoding of integers
- Omit fields that are used only for backwards compatibility.
- Define profiles of configuration settings
  (i.e. ciphersuite concept extended to extensions and other parameters)
- New certificate compression scheme

Security properties of TLS unchanged.

	ECDHE		
	TLS	CTLS	Overhead
ClientHello	132	50	10
ServerHello	90	48	8
ServerFlight	478	104	16
ClientFlight	458	100	11
Total	1158	302	45

Work in progress IETF draft: draft-ietf-tls-ctls

Outlook

- Most engineering is cost minimization, given constraints
- But hard for networking
  - cost data not available (proprietary)
  - very little economics in our network teaching
  - improvements are in operations and management more than protocols and algorithms
- Would require better software skills in carrier work force
  - and willingness to develop own software
  - and get rid of legacy systems and services

Henning Schulzrinne, “Networking Research - A Reflection in the Middle Years”, URL: https://arxiv.org/abs/1809.00623