TLS FOR IOT

Illya Gerasymchuk

illya@iluxonchik.me

INESC-ID Investigação e Desenvolvimento, Rua Alves Redol 9, 1000-029 Lisbon, Portugal
Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, 1049-001 Lisbon, Portugal

Motivation:
- Internet of Things (IoT) lacks security
- (D)TLS provides connection security
  - Too heavy for IoT devices
    - Code size, resources, energy
- Existing work focuses on DTLS
  - New standards like *"CoAP over TLS"* call for TLS optimization

Goals:
- Profile most important features of TLS
- Associate costs with each feature
  - e.g. *How much does PFS cost?*
- Build a framework that suggests a TLS configuration
  - Based on the needs and limitation of the environment
    - e.g. required security services, available power, memory and processing speed
  - Fully compatible with "vanilla" (D)TLS
- Focused on TLS 1.2
  - Taking into account TLS 1.3 and its features

The Transport Layer Security (TLS) Protocol

```
     Client                                          Server

     ClientHello                  -------->
                                                ServerHello
                                               Certificate*
                                      ServerKeyExchange*(1)
                                        CertificateRequest*
                                  <--------      ServerHelloDone
     Certificate*
  (1)ClientKeyExchange
  (2)CertificateVerify*
     [ChangeCipherSpec]
  (3)Finished                     -------->
                                         [ChangeCipherSpec]
                                  <--------        Finished(3)
     Application Data             <------->    Application Data
```

(1) - Server to client authentication (depending on the key exchange method).
(2) - Client to server authentication
(3) - MITM protection guarantee

- **Datagram TLS (DTLS)** – adaptation of TLS that runs on top of an **unreliable transport protocol** (e.g. **UDP**)
  - While **TLS** runs on top of a **reliable transport protocol**, such as **TCP**
- The majority of TLS features is also available in DTLS

Wednesday, April 18th 2018 15:00h

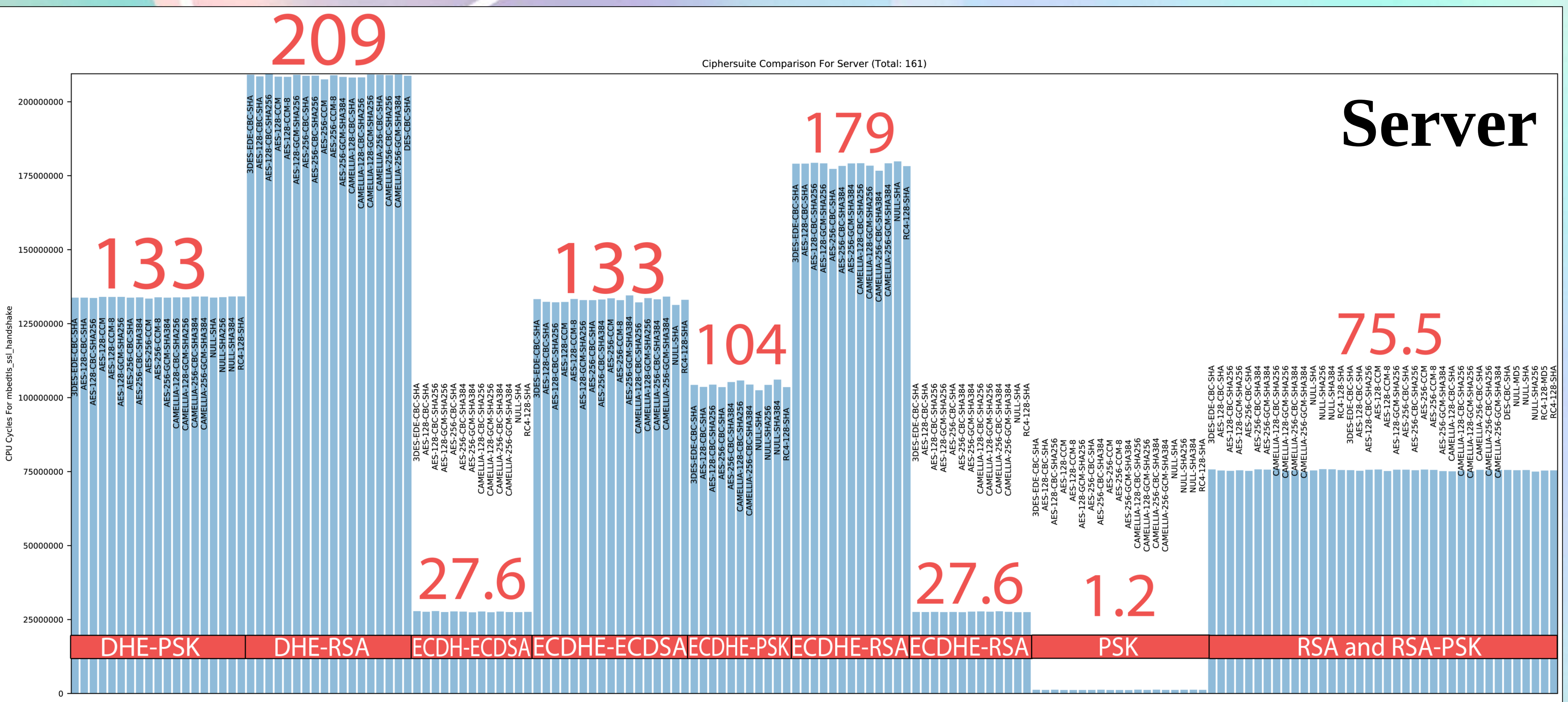
- Powers **HTTPS** (HTTP over TLS)
- **Client-Server** protocol
- **Connection-oriented** and **reliable**
- Main goals:
  - **Data confidentiality** and **integrity**
- Two phases:
  - Negotiate security parameters (**Handshake Protocol**)
  - Exchange data securely (**Record Protocol**)

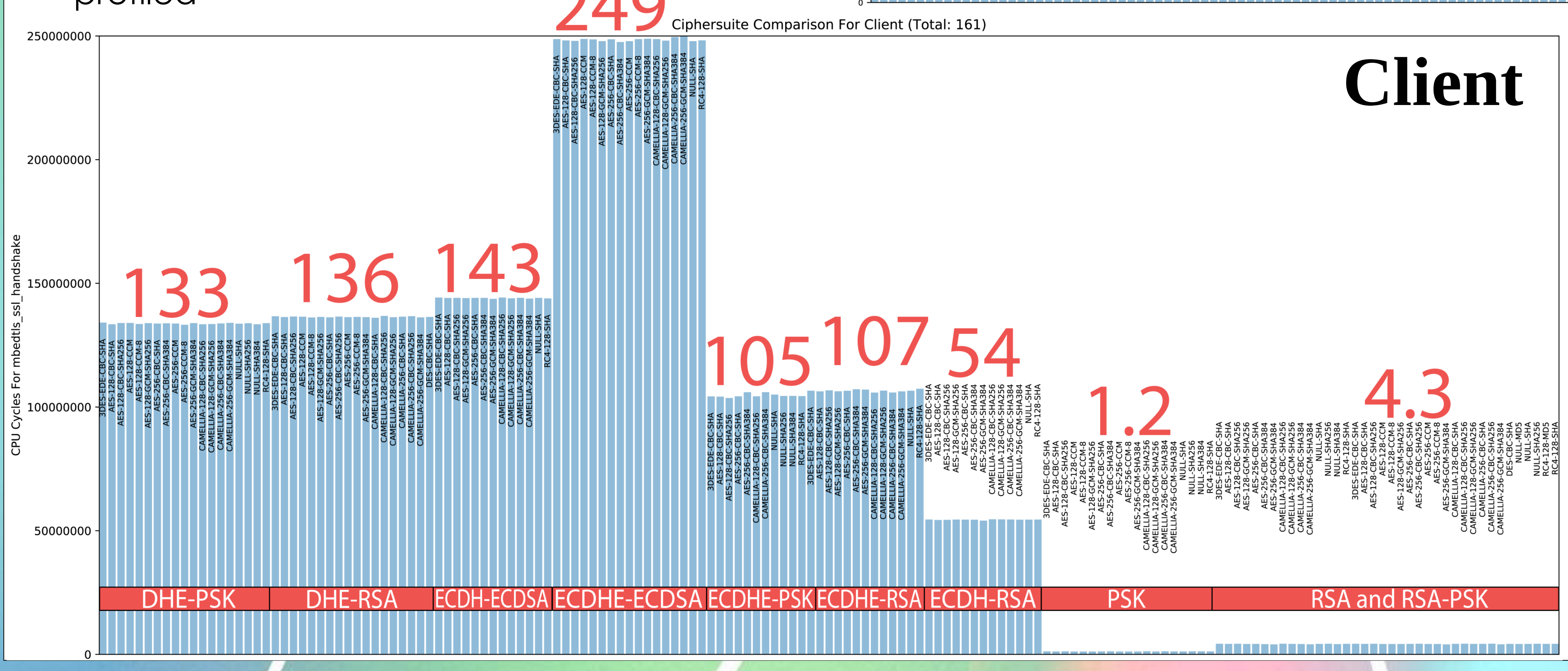
Using

- Security Services:
  - **Authentication**
  - **Confidentiality**
  - **Integrity**
  - **Replay Protection**
  - **Perfect Forward Secrecy**

Profiling TLS Ciphersuites

- **Ciphersuite** = key exchange alg. + authentication alg. + encryption alg. + PRF (TLS 1.2)
- **TLS Handshake profiled** with each one of the **ciphersuites** (depicted in graphs)
  - For both, client and server
  - Using a tool written for that purpose
- mbedTLS's library TLS implementation
  - TLS library for embedded devices
- Data **encryption** and **MAC** algorithms also profiled





Legend

- Red **rectangles** group by **key exchange** and **authentication**
- **Text** on bars specifies **encryption** algorithm and **MAC** function
- **Numbers** in red represent the number of **CPU cycles** in **millions**
  - *valgrind* estimates

Analysis

- Different results for client and server
  - Client needs to verify certificate chain (public key operations)
  - Public key operations are faster than private key ones
    - Because public RSA exponent is usually small
- ECDSA's signature verification is slower than RSA's
- TLS with Pre-Shared Keys (PSK) is used a lot in constrained devices
  - Graphs show why
- PFS is costly
- ECDH(E) ciphersuites use less CPU cycles than DHE(E)

Future Work

- Associate **cost** with each **security service**
- **Profile** remaining **TLS features**
  - e.g. session resumption
- Profile with actual CPU cycles measures
- **Profile** on **relevant architectures**
- Measure **power consumption**
- Build a **framework** that **suggests** a **TLS configuration**

CRYPTACUS