Attestation use case:

Computational Offloading for Mobile Devices

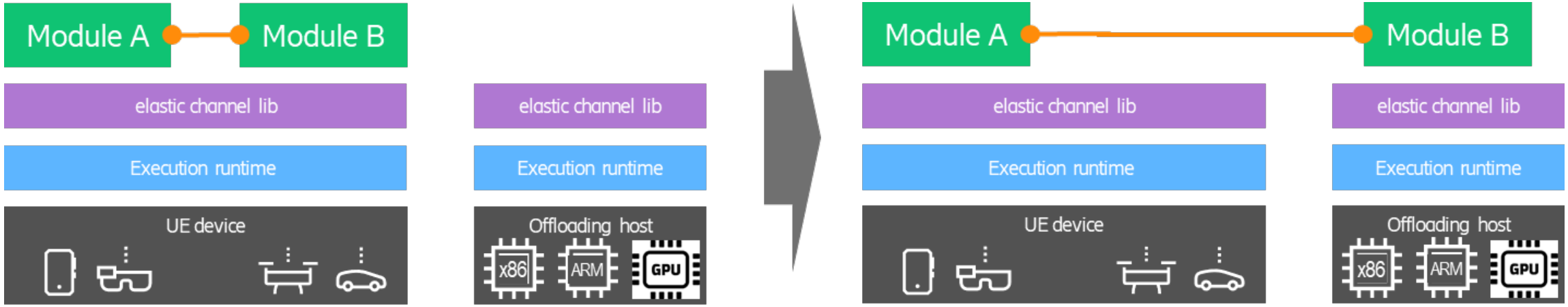
Jimmy Kjällman,
Ondrej Smid

Ericsson Research

2024-05-16

Dynamic Computational Offloading

Background and basic idea: Extending the functionality of mobile devices through dynamic use of remote compute resources

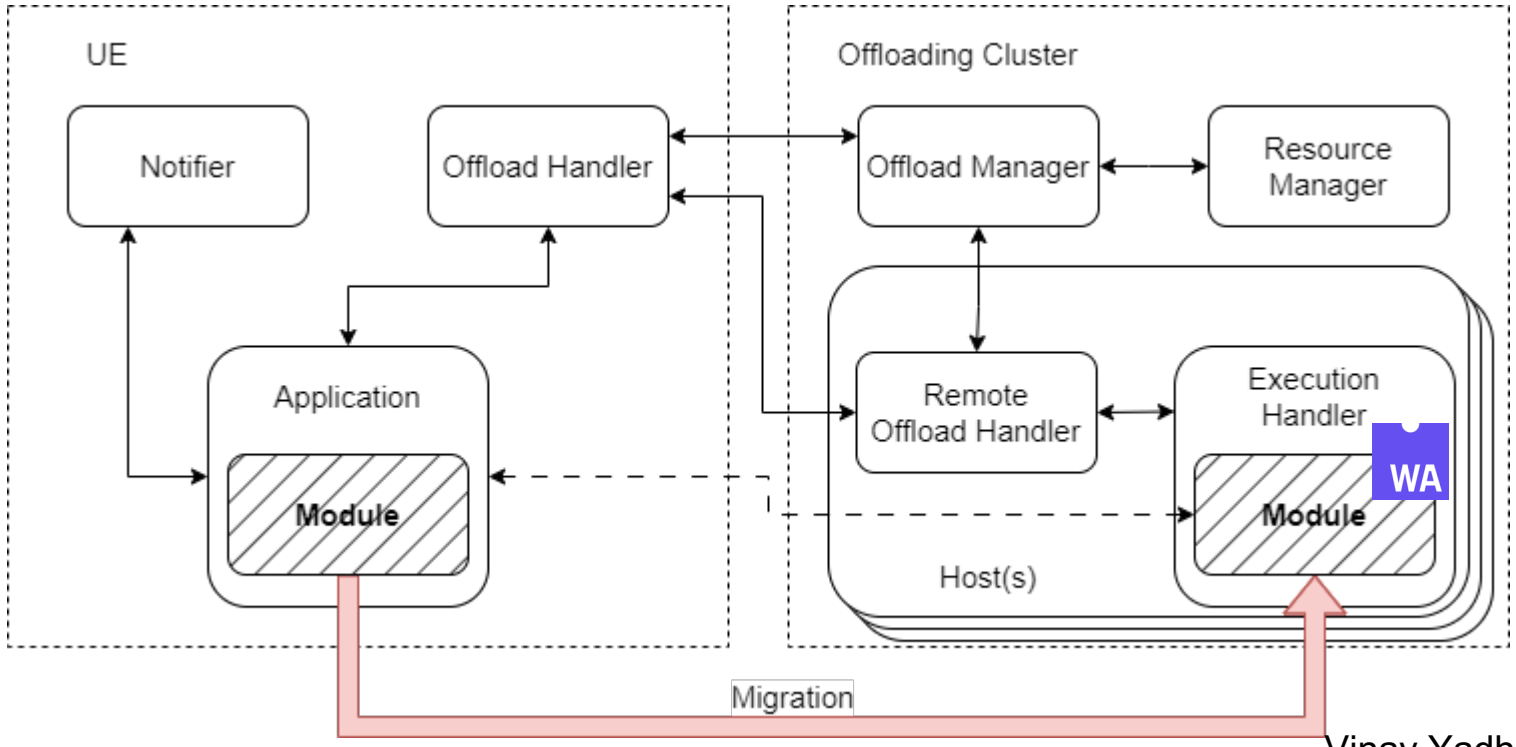


Vinay Yadhav, et al. “Benefits of Dynamic Computational Offloading for Mobile Devices”, CLOSER 2024. https://www.scitepress.org/PublicationsDetail.aspx?ID=ccKzjw4ElfA=&t=1

Dynamic Computational Offloading

System Design

- Requirements: **platform independent**, **lightweight**, **secure**, **polyglot**

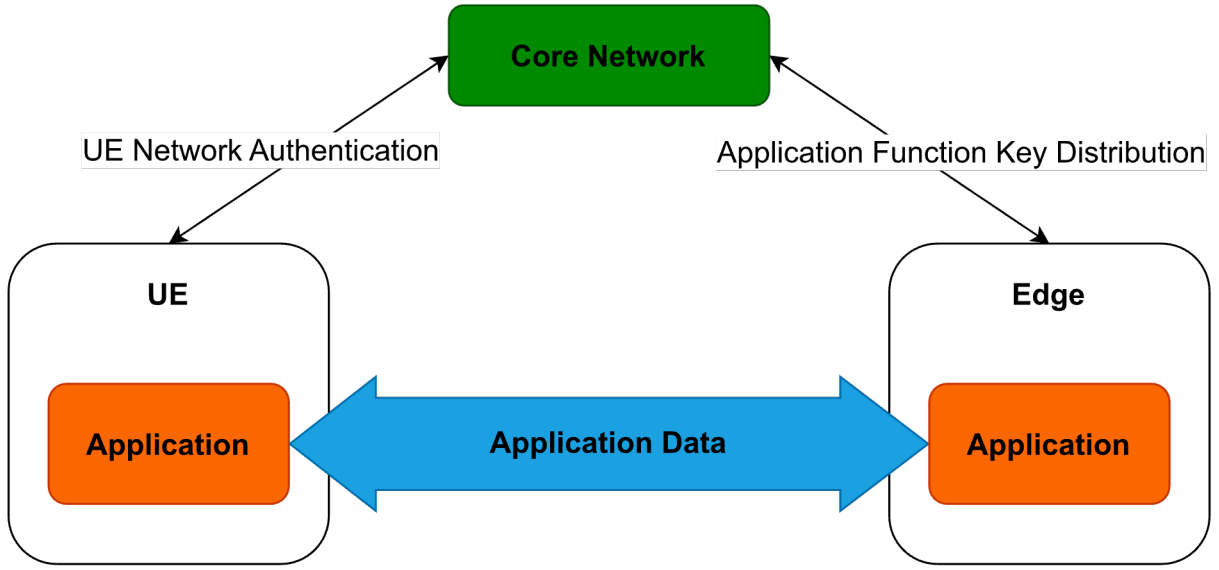


UE = User Equipment

Vinay Yadhav, et al. “Benefits of Dynamic Computational Offloading for Mobile Devices”, CLOSER 2024. https://www.scitepress.org/PublicationsDetail.aspx?ID=ccKzjw4ElfA=&t=1

Authentication & Network Integration

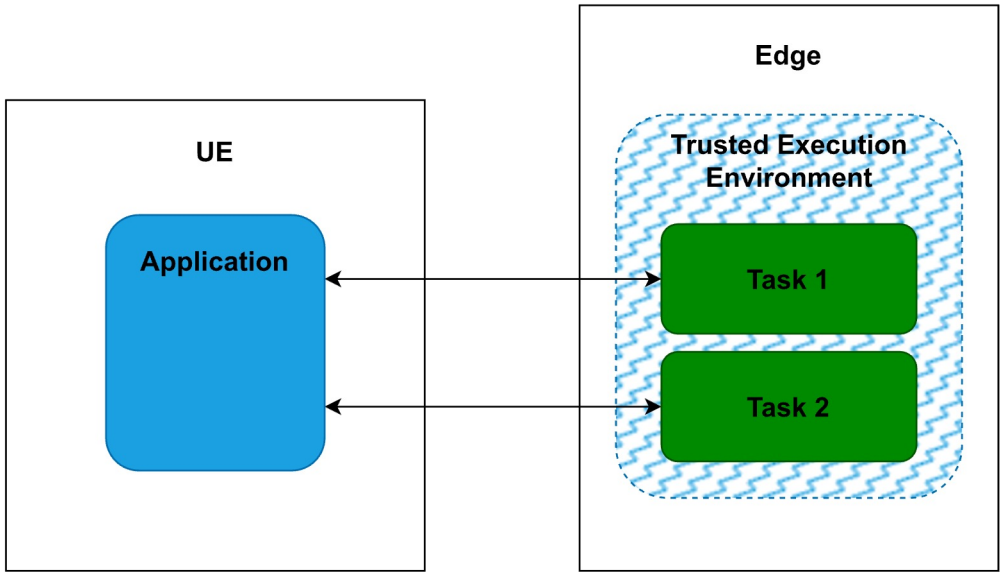
- Offloading provided as a **mobile network-integrated** service
  - Authentication and Key Management for Applications (AKMA) used for authentication of a device based on SIM credentials
    - **Pre-shared key** (PSK) provisioned to the device and offloading service
  - Compute service at the Edge for (e.g.) low latency
- Mobile/dynamic use cases, may open new connections often



UE = User Equipment

Confidential Computing

- Offloading host / execution environment run in **TEEs** for protecting workloads
  - Needs to be **attested** to the device side before doing module migration
    - Also mutual attestation should be supported
  - We need to be able to use multiple types of TEEs (in a broad sense)
  - Execution environment possibly spawned on demand



Attested TLS

- **TLS** connections between the device (client) and offloading service (server)
  - **PSK**(-DHE) **+ attestation** in the handshake
    - …taking place as **quickly**, in as few milliseconds and round trips, as possible
  - tls-attestation draft used as the starting point
    - PSKs alongside attestation, mutual attestation, passport model

Questions around TLS Attestation

- Is the tls-attestation draft going to include, e.g., detailed mutual attstation and passport model examples?
- Can evidence generation and/or verification be parallellized within the handshake? Other optimizations?
- Certificate message used also with TLS PSK?
- Runtime attestation with long-term connections?

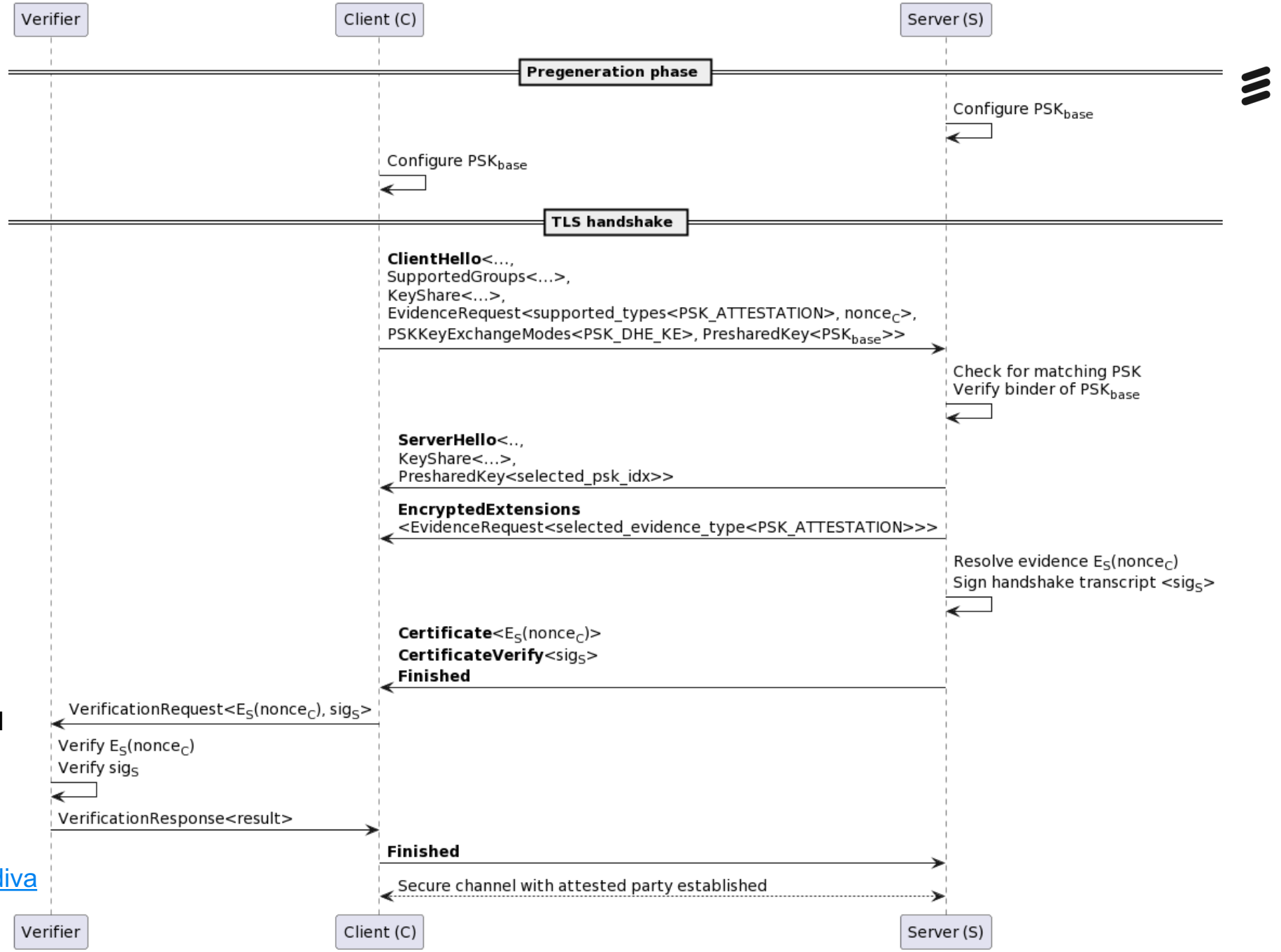
Mutual attestation & passport / background check model

- Mutual attestation – models combinations: client - server
  - Background check – Background check
    - Client needs to wait for its own attestation
  - Passport – Passport
    - Server needs to wait with prepared passport for the client
  - Background check – Passport
    - Server needs to wait with prepared passport for the client
  - Passport – Background check

Implementation of TLS-Attestation using PSK in rustls

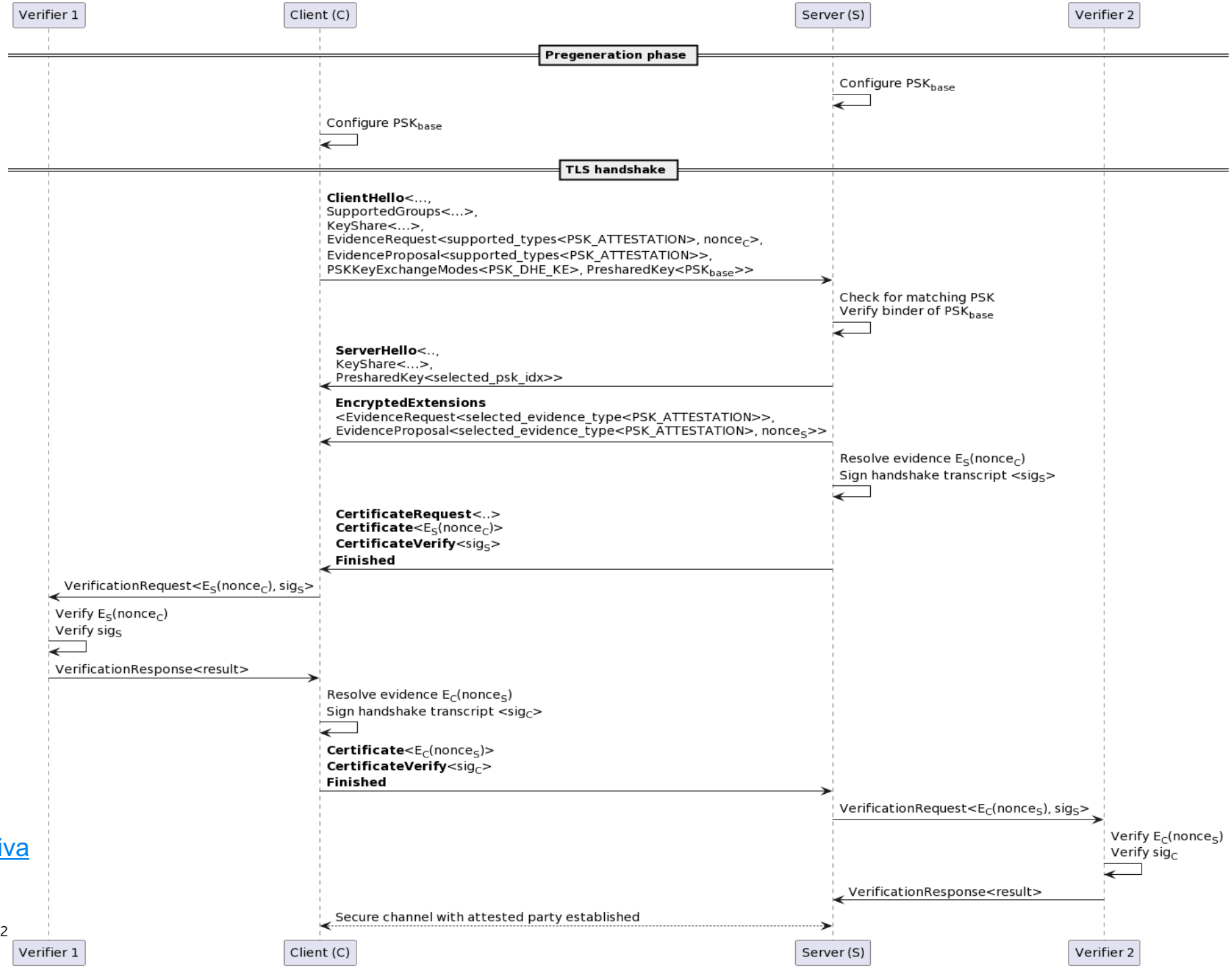
- Rustls - https://github.com/rustls/rustls
- Attestation + PSKs
  - (For PSKs, our implementation requires “Importing External Pre-Shared Keys (PSKs) for TLS 1.3” RFC 9258)
- Token generation and verification is not yet implemented

PSK + attestation exchange (an initial draft)



Leon Hejdenberg Philip, Establishment of Secure Channel Binding with Remote Party Attestation, BSc thesis, Uppsala University, 2024 http://uu.diva-portal.org/smash/record.jsf?pid=diva2%3A1852751&dswid=-5478

PSK + mutual attestation exchange (an initial draft)



Leon Hejdenberg Philip, Establishment of Secure Channel Binding with Remote Party Attestation, BSc thesis, Uppsala University, 2024
http://uu.diva-portal.org/smash/record.jsf?pid=diva2%3A1852751&dswid=-5478

TLS with integrated attestation

PSK + attestation	Original	
Task	**Mean execution time (ms)**	**% of total**
Initialize Client	**0.7729**	**15.178**
→ Prepare ClientHello	0.2109	4.142
→ Rest of task	0.5620	11.036
Handshake	**4.2451**	**83.365**
→ (S) Generate evidence	0.0084	0.165
→ (S) Prepare ServerHello	1.1976	23.518
→ (S) Prepare CertificateVerify	0.0215	0.422
→ (C) Verify evidence	0.0016	0.032
→ (C) Verify signature	0.0058	0.113
→ Rest of task	3.0103	59.115
[Other] (calculated)	**0.0742**	**1.457**
All	**5.0922**	**100.000**

Actual evidence generation and verification time is not included.

PSK + attestation	Example	
Task	**Calc. execution time (ms)**	**% of total**
Initialize Client	**0.7729**	**0.858**
→ Prepare ClientHello	0.2109	0.234
→ Rest of task	0.5620	0.624
Handshake	**89.2452**	**99.060**
→ (S) Generate evidence	50.0084	55.508
→ (S) Prepare ServerHello	1.1976	1.329
→ (S) Prepare CertificateVerify	25.0215	27.773
→ (C) Verify evidence	5.0016	5.552
→ (C) Verify signature	5.0058	5.556
→ Rest of task	3.0103	3.341
[Other]	**0.0742**	**0.082**
All	**90.0923**	**100.000**

E.g., on AWS EC2, AMD SEV-SNP attestation report generation seems to take very roughly ~50 ms and verification ~5 ms. The CertificateVerify signing and verification (w/ Identity Key) numbers used here are only examples.

TLS with integrated mutual attestation

PSK + mutual attestation	Original	
Task	**Mean execution time (ms)**	**% of total**
Initialize Client	**0.7957**	**14.934**
→ Prepare ClientHello	0.2134	4.005
→ Rest of task	0.5823	10.929
Handshake	**4.4673**	**83.841**
→ (S) Generate evidence	0.0087	0.164
→ (S) Prepare ServerHello	1.1938	22.404
→ (S) Prepare CertificateVerify	0.0199	0.373
→ (C) Verify evidence	0.0016	0.029
→ (C) Verify signature	0.0100	0.187
→ (C) Generate evidence	0.0018	0.035
→ (C) Prepare CertificateVerify	0.0137	0.256
→ (S) Verify evidence	0.0030	0.055
→ (S) Verify signature	0.0036	0.067
→ Rest of task	3.2114	60.271
[Other] (calculated)	**0.0653**	**1.226**
All	**5.3283**	**100.000**

Actual evidence generation and verification time is not included.

PSK + mutual attestation	Example	
Task	**Calc. execution time (ms)**	**% of total**
Initialize Client	**0.7957**	**0.454**
→ Prepare ClientHello	0.2134	0.122
→ Rest of task	0.5823	0.332
Handshake	**174.4675**	**99.509**
→ (S) Generate evidence	50.0087	28.523
→ (S) Prepare ServerHello	1.1938	0.681
→ (S) Prepare CertificateVerify	25.0199	14.270
→ (C) Verify evidence	5.0016	2.853
→ (C) Verify signature	5.0100	2.857
→ (C) Generate evidence	50.0018	28.519
→ (C) Prepare CertificateVerify	25.0137	14.267
→ (S) Verify evidence	5.0030	2.854
→ (S) Verify signature	5.0036	2.854
→ Rest of task	3.2114	1.832
[Other]	**0.0653**	**0.037**
All	**175.3285**	**100.000**

E.g., on AWS EC2, AMD SEV-SNP attestation report generation seems to take very roughly ~50 ms and verification ~5 ms. The CertificateVerify signing and verification (w/ Identity Key) numbers used here are only examples.