HTTPA/1 Protocol

Presenter: Hans Wang, AI Research Scientist, SPR lab, Intel Labs.

Contributor: Gordon King, Cloud Software Architect, SST.

Tuesday, April 26, 2022

Legal Disclaimers

Intel provides these materials as-is, with no express or implied warranties. All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice.

Intel processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://intel.com.

Some results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries. *Other names and brands may be claimed as the property of others. © Intel Corporation 2022

*Other names and brands may be claimed as the property of others. © Intel Corporation 2022

Purposes

- Share an idea for an L7 protocol, HTTPA/1, to establish trusted communication channel between HTTP endpoints using remote attestation.
- Invite feedback, contribution or collaboration.
- Notice: HTTPA/2 is still in legal review.

Outline

- Motivation
- Background
- Problem statements
- HTTPA/1
- Summary

Motivation

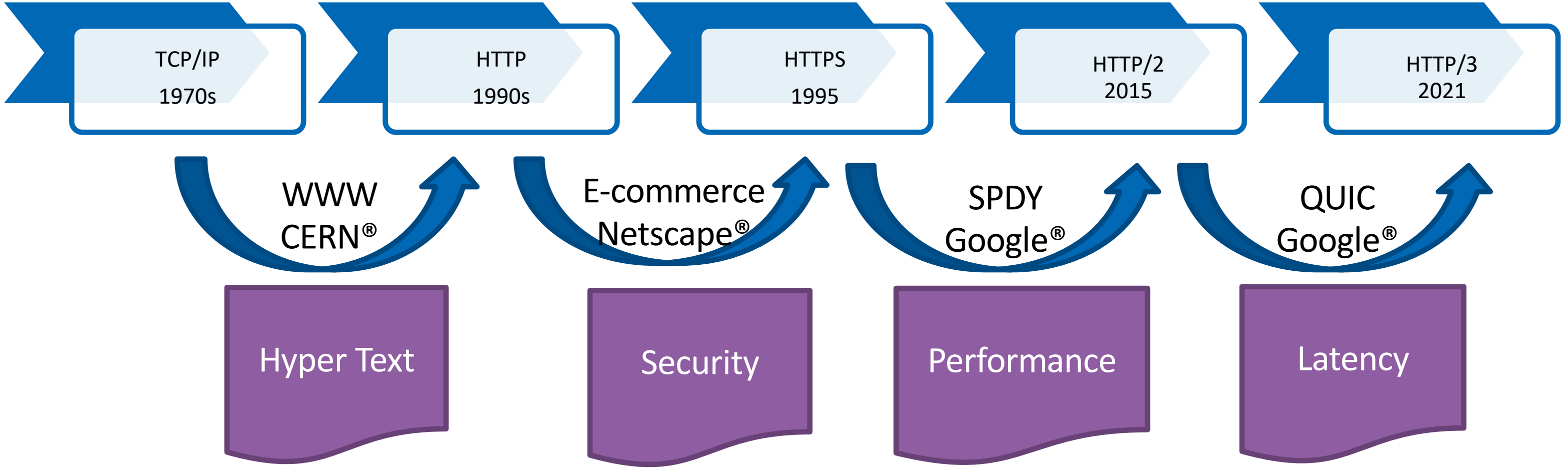
- Trusted federated learning requires building trust with remote parties.
  - How can we trust or attest the remote AI service?

- Customer's feedback to request a standard way to use attestation with better usability.

- Multiple confidential computing platforms need unification for communication protocol.

Background

Internet Protocol Evolution



TCP/IP
1970s

HTTP
1990s

HTTPS
1995

HTTP/2
2015

HTTP/3
2021

WWW
CERN®

E-commerce
Netscape®

SPDY
Google®

QUIC
Google®

Hyper Text

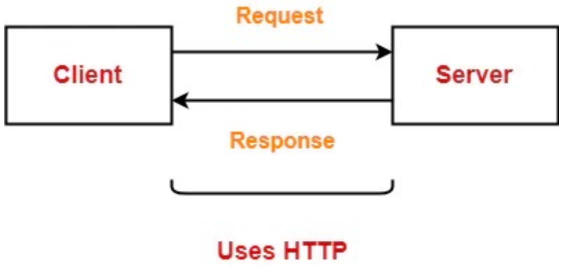
Security

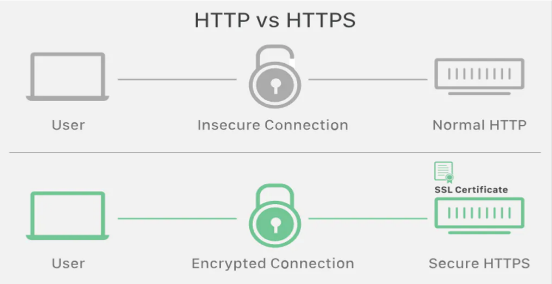
Performance

Latency

Background

- HTTP
  - A protocol to define communication of request/response between a client and a server.
- HTTPS
  - HTTP over TLS/SSL
  - **Not** a separate protocol from HTTP.
  - TLS/SSL handshake to establish a secure connection:
    - TLS/SSL certificate for server authentication
    - TLS/SSL encryption for confidential communication
    - TLS/SSL message digest for authentic data integrity





https://docs.microsoft.com/en-us/windows-server/security/tls/transport-layer-security-protocol#:~:text=The%20TLS%20(and%20SSL)%20protocols,data%20to%20the%20transport%20layer.
https://satyampariyar.hashnode.dev/what-is-web-http-and-https

Problem statements

Problem statements

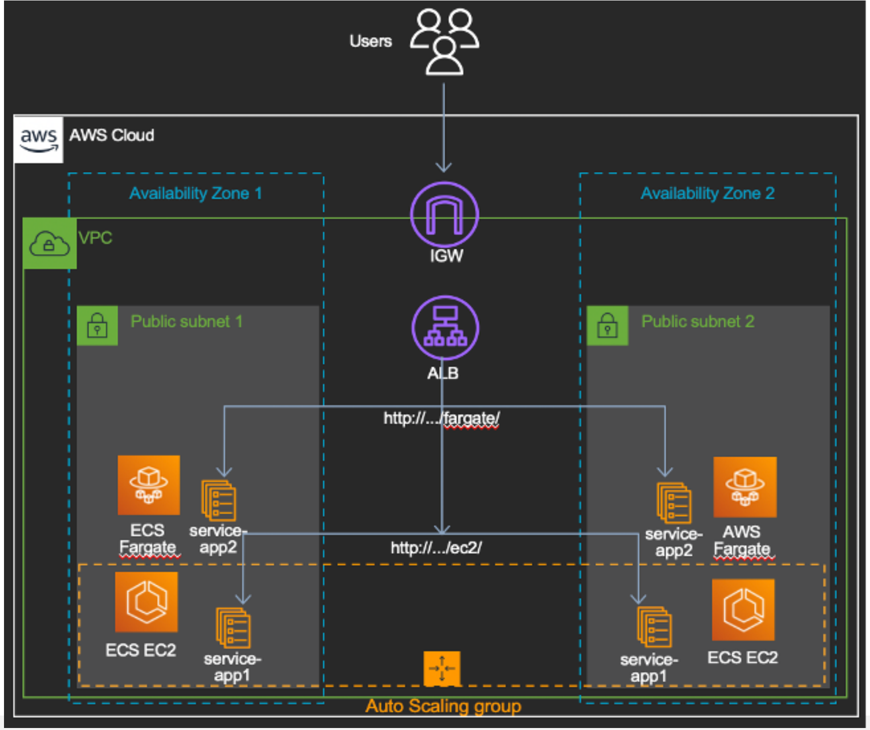
- Assumption: TEE-aware services will be popular in the future.
- Context:
  - A website can host multiple web services, many of them relying on third-party software vendors.
  - Webservices are vulnerable to the host, and vice versa.
  - Increasing demands for **microservices**, and **L7 load balancers**, and etc.
- **In most HTTPS communication, TLS terminates at the application gateway or L7 load balancer. This imposes security concerns of vulnerabilities, leading to lack of confidence in trustworthiness.**
- **Modern web services are lacks of trust.**
- **What's current solutions?**
  - **HTTPS** alone providing **secure channels** is not enough protecting web services for building trust.
  - Existing remote attestation mechanism adds high complexity for development and requires high learning curve for non-security developers, e.g., AI experts, data experts, software engineers, and etc.
  - Big players are proposing their own solutions to solve this problem.
- We need a standardized approach by simply extending existing protocol to build trust for using web services: **HTTPA**.
  - HTTPA aims to establish trusted end-to-end communication with the attested end point.

Industrial URL Routing and Load Balancing (LB)

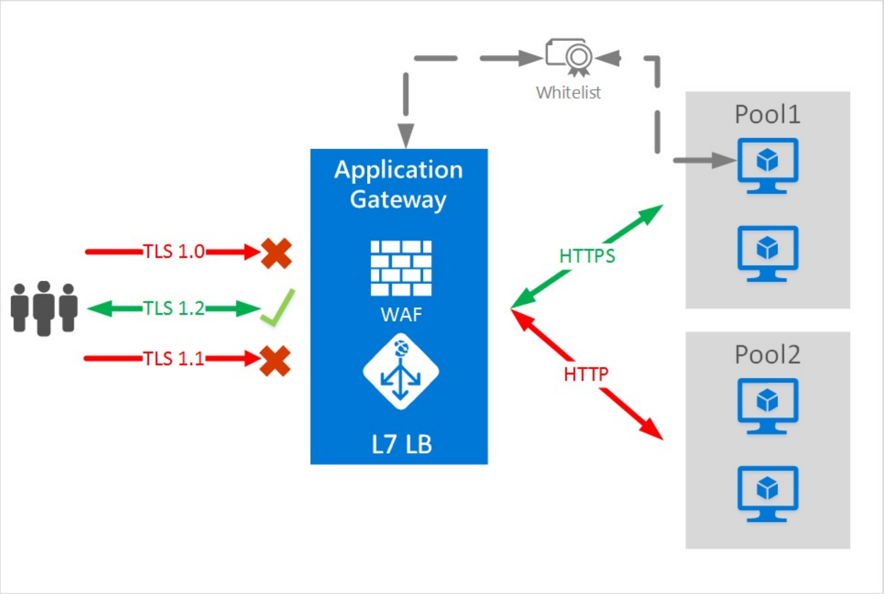
“Application Gateway supports **TLS termination at the gateway**, after which traffic typically **flows unencrypted** to the backend servers” – Microsoft Azure

“Application Gateway then **initiates a new TLS connection** to the backend server and **re-encrypts data** using the backend server's public key certificate before transmitting the request to the backend. ” – Microsoft Azure

Amazon AWS



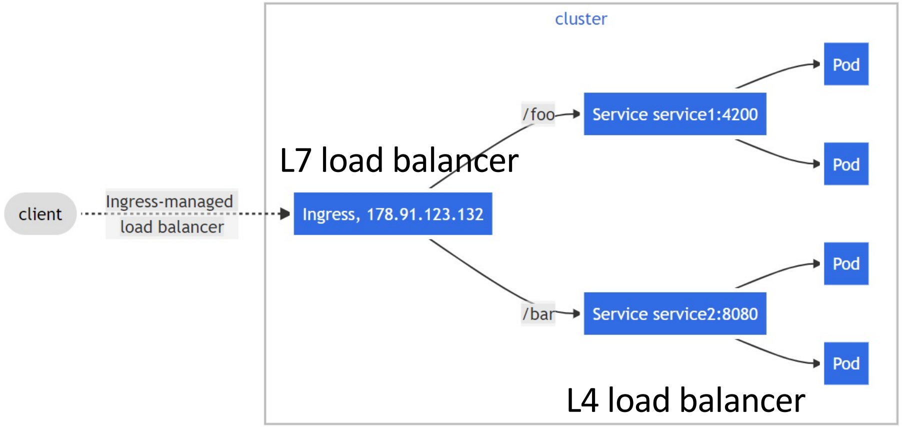
Microsoft Azure



https://aws.amazon.com/blogs/containers/using-aws-application-load-balancer-path-based-routing-to-combine-amazon-ecs-launch-types/
https://docs.microsoft.com/en-us/azure/application-gateway/ssl-overview

Kubernetes Ingress

- “The Ingress resource only supports a single TLS port, 443, and assumes **TLS termination** at the ingress point (traffic to the Service and its Pods is in plaintext).”



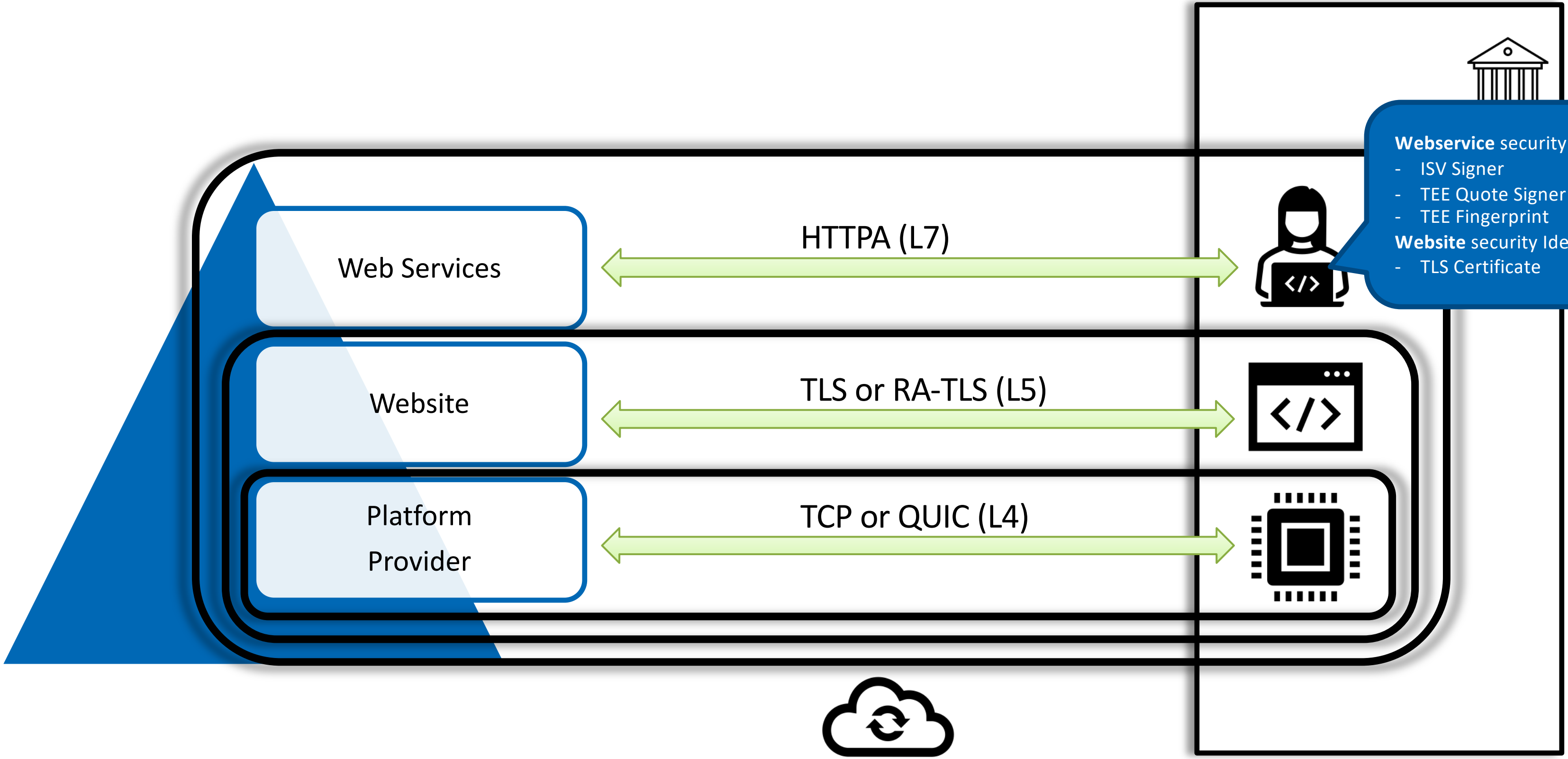
L7 load balancer

L4 load balancer

https://kubernetes.io/docs/concepts/services-networking/ingress/

HTTPA/1

HTTPA Stack for Internet



Webservice security identities:

- ISV Signer
- TEE Quote Signer
- TEE Fingerprint

Website security Identities

- TLS Certificate

What is HTTPA?

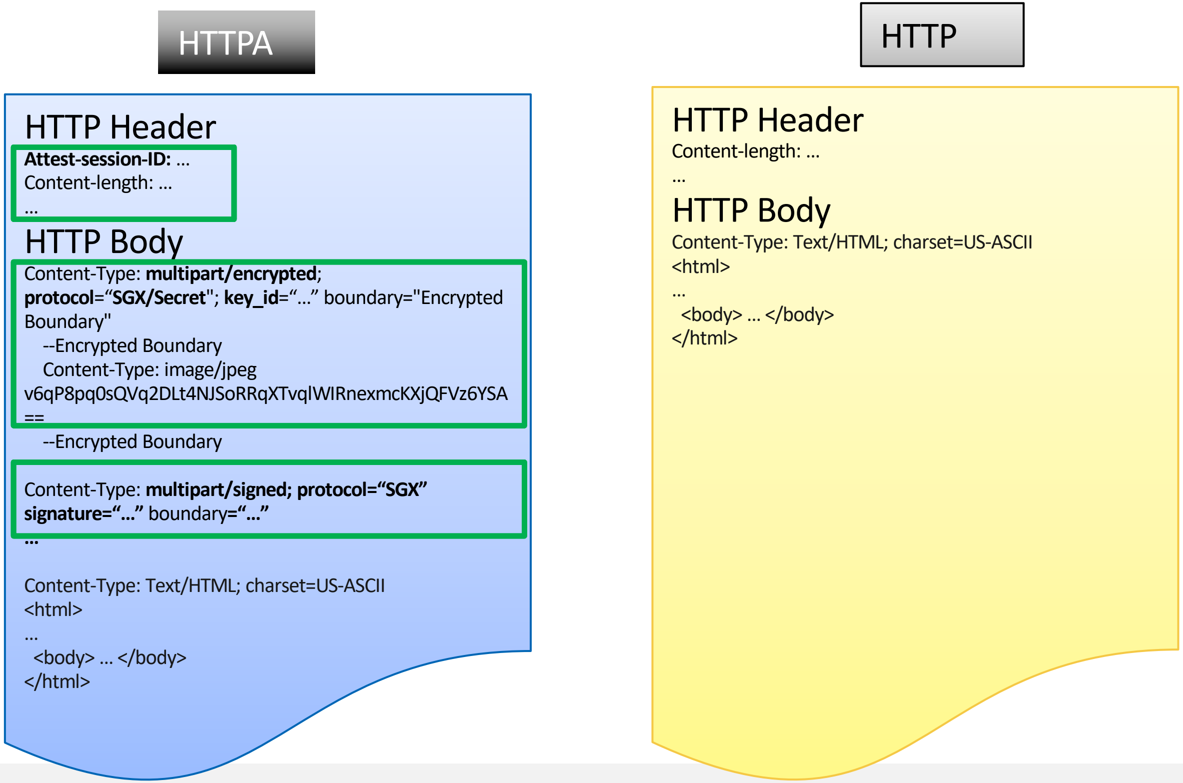
Both HTTP and HTTPA reuse original untouched TLS to secure traffics.

HTTP-Attestable (HTTPA) is an L7 handshake protocol using HTTP extension.

1. Attest-session-ID

2. Encryption by shared ephemeral keys for TEE sensitive data

3. Signing messages by the ephemeral key



HTTPA

HTTP Header

Attest-session-ID: …
Content-length: …
…

HTTP Body

Content-Type: **multipart/encrypted**;
protocol="**SGX/Secret**"; **key_id**="…" boundary="Encrypted Boundary"
  --Encrypted Boundary
  Content-Type: image/jpeg
v6qP8pq0sQVq2DLt4NJSoRRqXTvqlWIRnexmcKXjQFVz6YSA
==
  --Encrypted Boundary

Content-Type: **multipart/signed; protocol="SGX"**
signature="…" boundary=**"…"**
…

Content-Type: Text/HTML; charset=US-ASCII
<html>
…
  <body> … </body>
</html>

HTTP

HTTP Header

Content-length: …
…

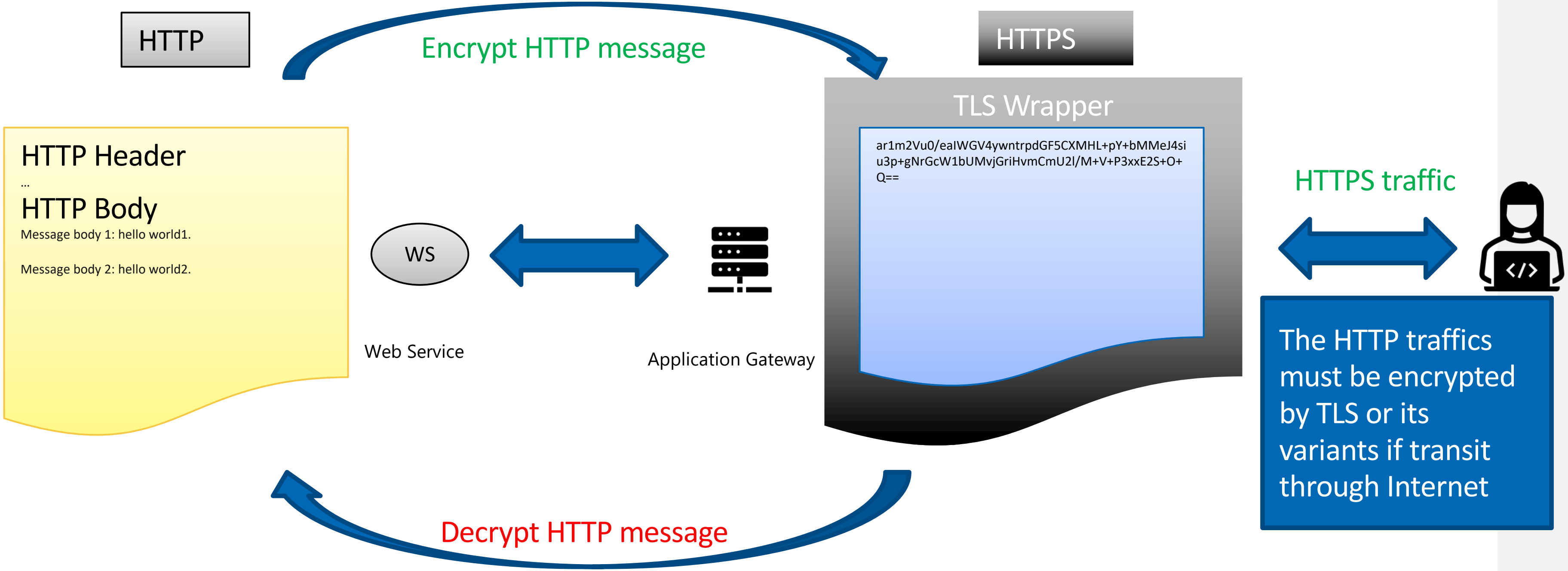
HTTP Body

Content-Type: Text/HTML; charset=US-ASCII
<html>
…
  <body> … </body>
</html>

https://datatracker.ietf.org/doc/html/rfc1847
https://docs.microsoft.com/en-us/previous-versions/office/developer/exchange-server-2010/aa493937(v=exchg.140)

HTTP over TLS (HTTPS)

“Application Gateway supports TLS termination at the gateway, after which traffic typically flows unencrypted to the backend servers” – Microsoft Azure

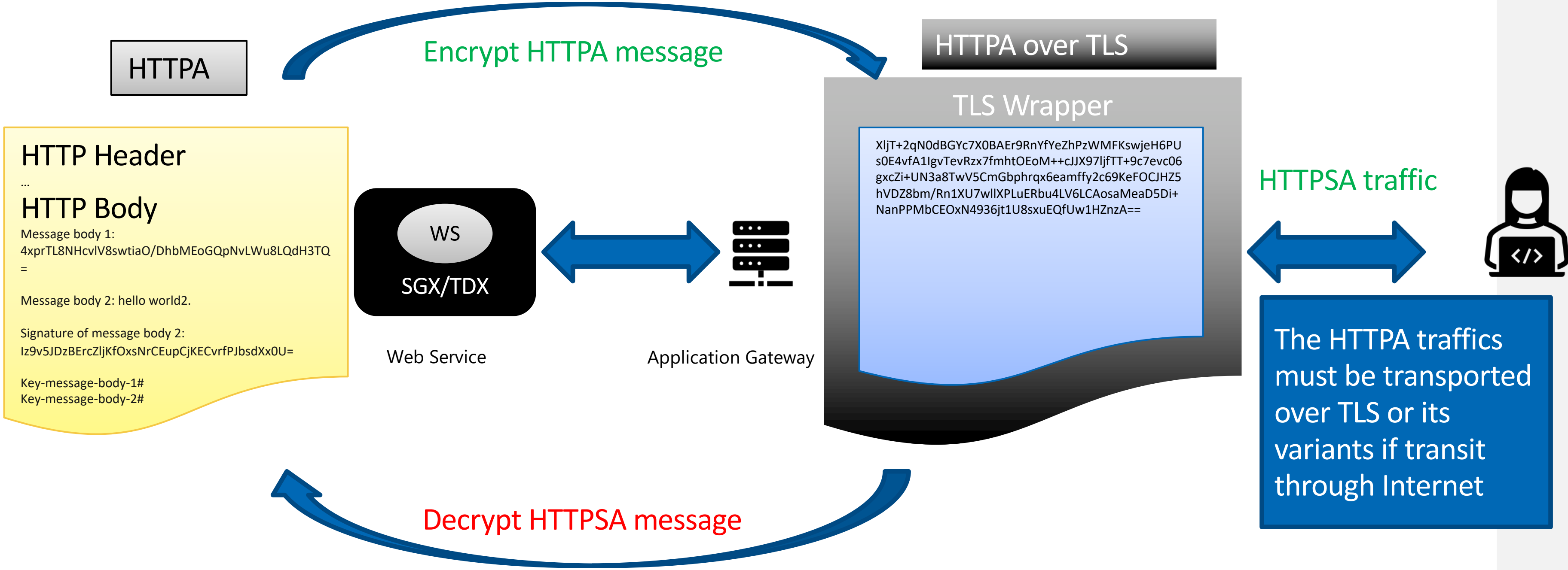

HTTP
Encrypt HTTP message
HTTPS
TLS Wrapper
HTTP Header
...
HTTP Body
Message body 1: hello world1.
Message body 2: hello world2.
WS
Web Service
Application Gateway
ar1m2Vu0/ealWGV4ywntrpdGF5CXMHL+pY+bMMeJ4si
u3p+gNrGcW1bUMvjGriHvmCmU2l/M+V+P3xxE2S+O+
Q==
HTTPS traffic
</>
Decrypt HTTP message

The HTTP traffics must be encrypted by TLS or its variants if transit through Internet

https://docs.microsoft.com/en-us/azure/application-gateway/ssl-overview

HTTPA over TLS (HT.TPSA)

TEE-encrypted data can go through L7 load balancers witho
leaking sensitive information!



Message body 1:
4xprTL8NHcvlV8swtiaO/DhbMEoGQpNvLWu8LQdH3TQ
=

Message body 2: hello world2.

Signature of message body 2:
Iz9v5JDzBErcZljKfOxsNrCEupCjKECvrfPJbsdXx0U=

Key-message-body-1#
Key-message-body-2#

```
XljT+2qN0dBGYc7X0BAEr9RnYfYeZhPzWMFKswjeH6PU
s0E4vfA1IgvTevRzx7fmhtOEoM++cJJX97ljfTT+9c7evc06
gxcZi+UN3a8TwV5CmGbphrqx6eamffy2c69KeFOCJHZ5
hVDZ8bm/Rn1XU7wllXPLuERbu4LV6LCAosaMeaD5Di+
NanPPMbCEOxN4936jt1U8sxuEQfUw1HZnzA==
```

The HTTPA traffics must be transported over TLS or its variants if transit through Internet

https://docs.microsoft.com/en-us/azure/application-gateway/ssl-overview

HTTPSA

TEE-encrypted data can go through L7 load balancers without leaking sensitive information!

HTTPA

HTTP Header

…

HTTP Body

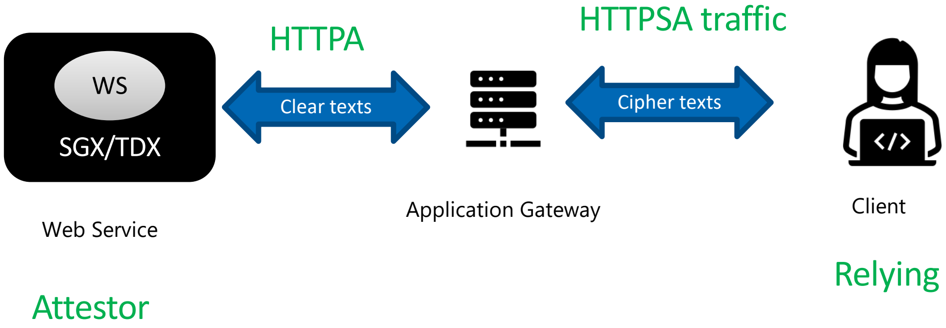
Message body 1:
4xprTL8NHcvlV8swtiaO/DhbMEoGQpNvLWu8LQdH3TQ
=

Message body 2: hello world2.

Signature of message body 2:
Iz9v5JDzBErcZljKfOxsNrCEupCjKECvrfPJbsdXx0U=

Key-message-body-1#
Key-message-body-2#

HTTPA over TLS



WS

SGX/TDX

Web Service

Attestor

HTTPA

Clear texts

Application Gateway

HTTPSA traffic

Cipher texts

Client

Relying party & Verifier

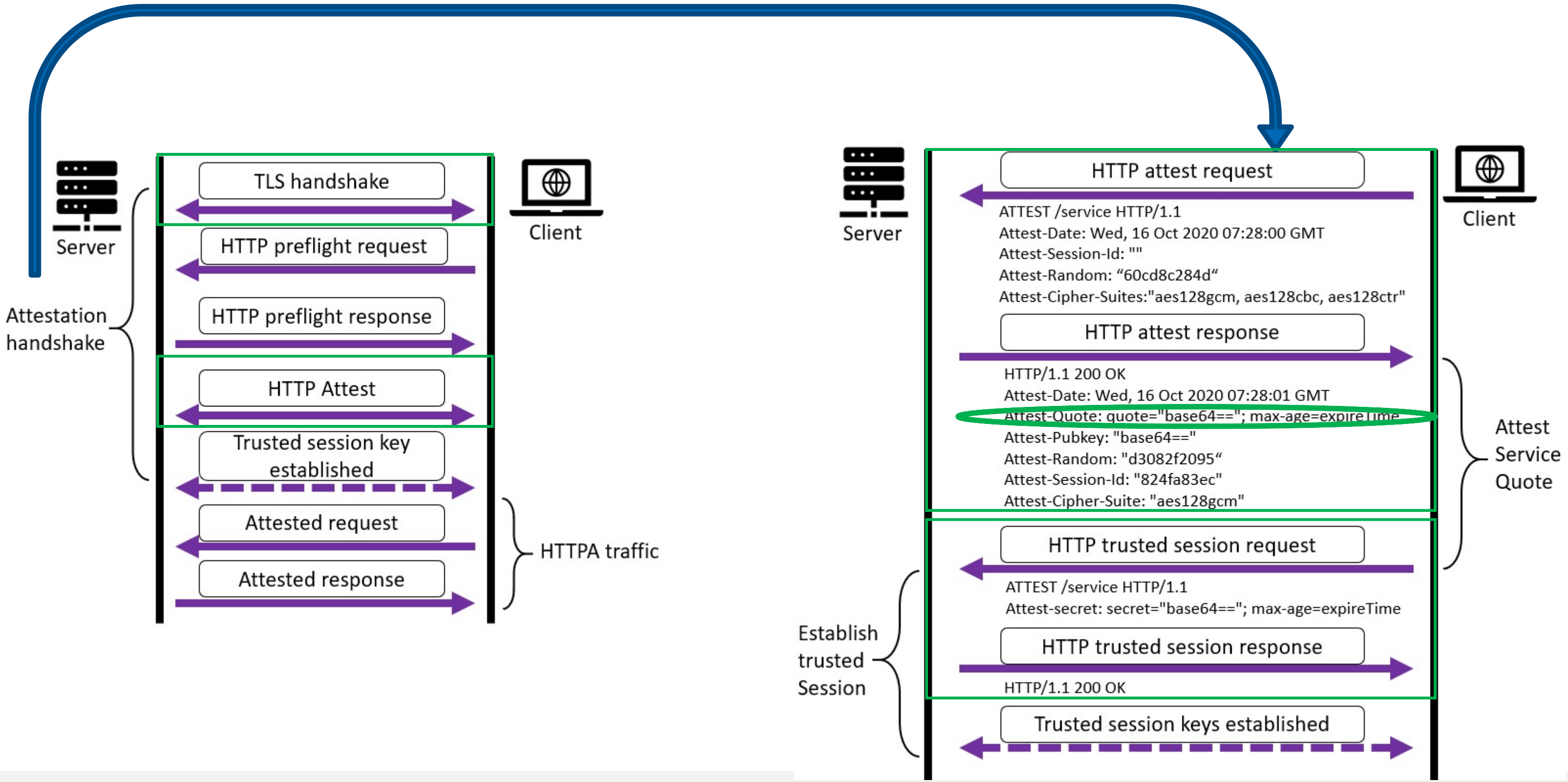
The HTTPA traffics must be transported over TLS or its variants if transit through Internet

https://docs.microsoft.com/en-us/azure/application-gateway/ssl-overview

HTTPSA handshake

Goal:

1. HTTPA attestation process
2. HTTPA key exchange
3. HTTPA trusted session establishment
4. HTTPA secret provisioning



```
ATTEST /service HTTP/1.1
Attest-Date: Wed, 16 Oct 2020 07:28:00 GMT
Attest-Session-Id: ""
Attest-Random: “60cd8c284d“
Attest-Cipher-Suites:"aes128gcm, aes128cbc, aes128ctr"
```

```
HTTP/1.1 200 OK
Attest-Date: Wed, 16 Oct 2020 07:28:01 GMT
Attest-Quote: quote="base64=="; max-age=expireTime
Attest-Pubkey: "base64=="
Attest-Random: "d3082f2095“
Attest-Session-Id: "824fa83ec"
Attest-Cipher-Suite: "aes128gcm"
```

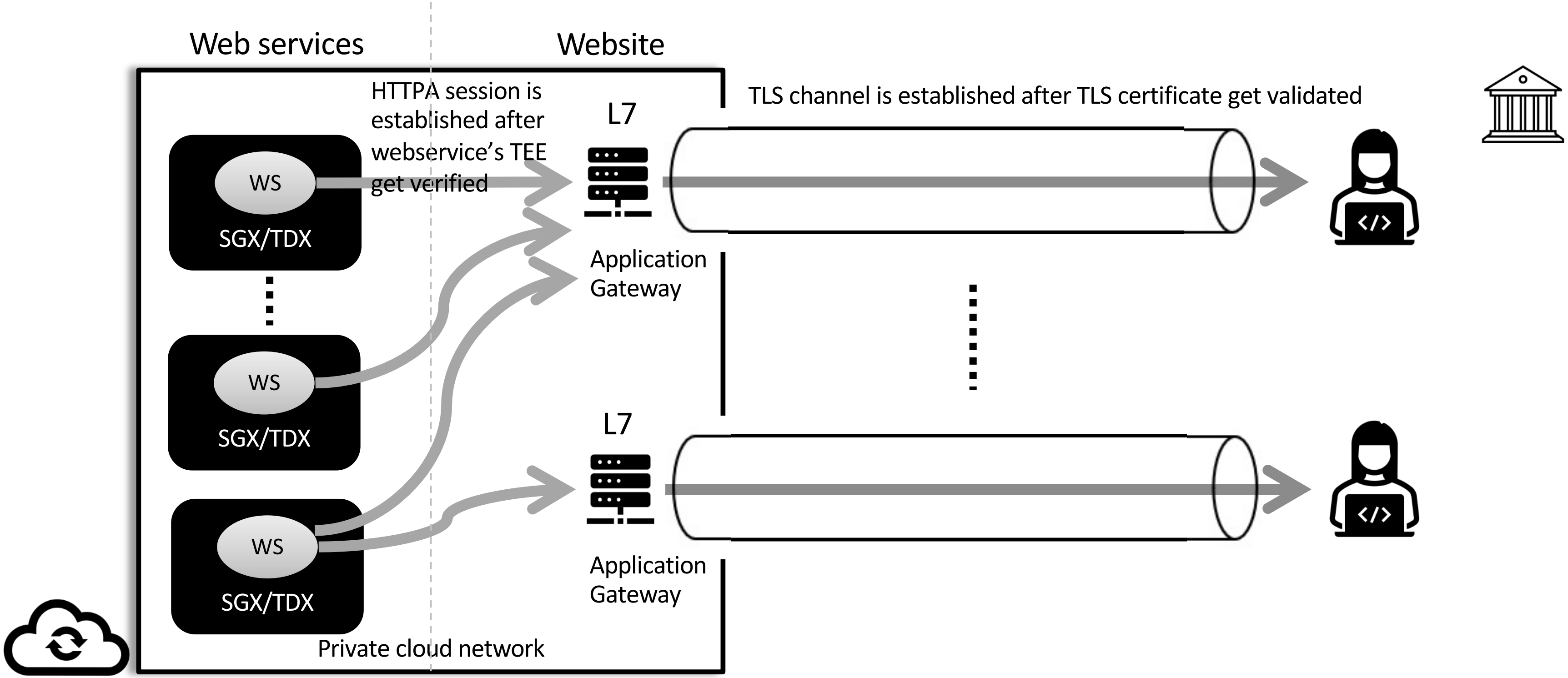
```
ATTEST /service HTTP/1.1
Attest-secret: secret="base64=="; max-age=expireTime
```

```
HTTP/1.1 200 OK
```

https://arxiv.org/pdf/2110.07954.pdf

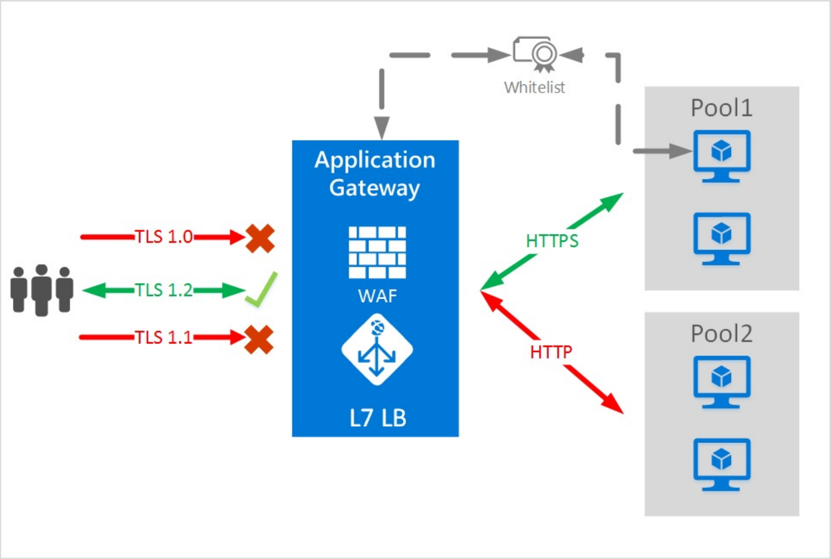
HTTPSA

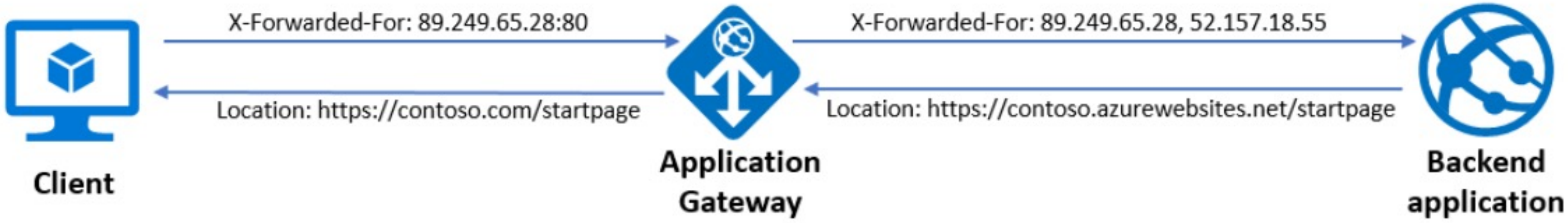
Trusted end-to-end channel


Web services
Website
HTTPA session is established after webservice's TEE get verified
L7
TLS channel is established after TLS certificate get validated
WS
SGX/TDX
WS
SGX/TDX
WS
SGX/TDX
Application Gateway
L7
Application Gateway
Private cloud network
</>

TLS termination is popular

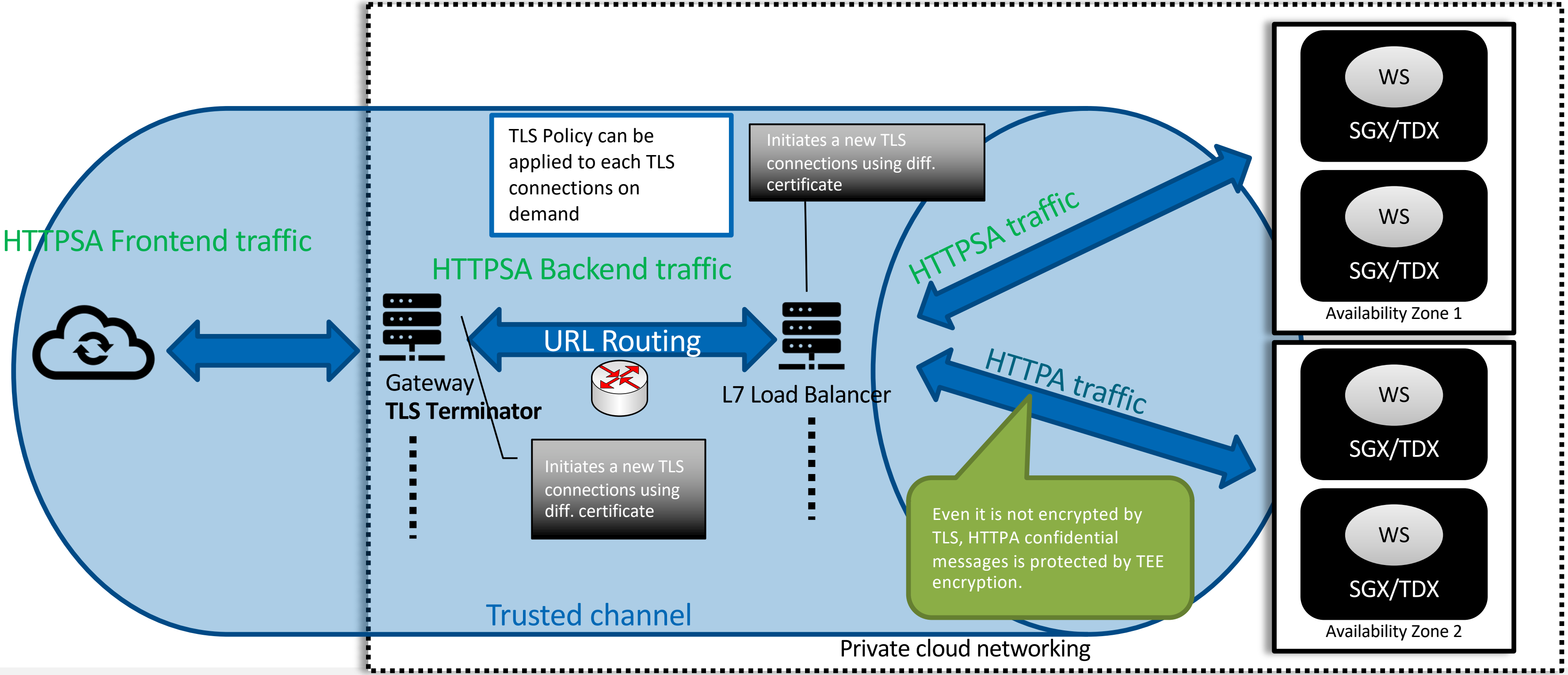
- Better performance.
- Better utilization of backend servers.
- Better routing.
- Better certificate management.




X-Forwarded-For: 89.249.65.28:80
Location: https://contoso.com/startpage
X-Forwarded-For: 89.249.65.28, 52.157.18.55
Location: https://contoso.azurewebsites.net/startpage
Client
Application Gateway
Backend application

https://docs.microsoft.com/en-us/azure/application-gateway/ssl-overview
https://docs.microsoft.com/en-us/azure/application-gateway/rewrite-http-headers-url

Application gateway and LB with HTTPA



TLS Policy can be applied to each TLS connections on demand

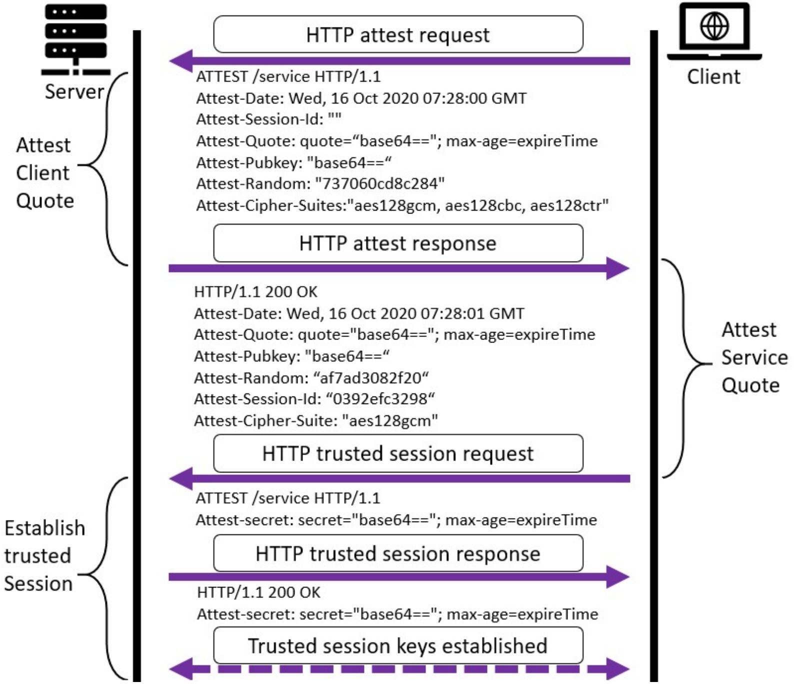
Initiates a new TLS connections using diff. certificate

Initiates a new TLS connections using diff. certificate

Even it is not encrypted by TLS, HTTPA confidential messages is protected by TEE encryption.

Mutual-HTTPA (mHTTPA)

HTTPA supports mutual attestation remotely


Server
Client
Attest Client Quote
Attest Service Quote
Establish trusted Session
HTTP attest request
HTTP attest response
HTTP trusted session request
HTTP trusted session response
Trusted session keys established

```
ATTEST /service HTTP/1.1
Attest-Date: Wed, 16 Oct 2020 07:28:00 GMT
Attest-Session-Id: ""
Attest-Quote: quote=“base64=="; max-age=expireTime
Attest-Pubkey: "base64==“
Attest-Random: "737060cd8c284"
Attest-Cipher-Suites:"aes128gcm, aes128cbc, aes128ctr"
```

```
HTTP/1.1 200 OK
Attest-Date: Wed, 16 Oct 2020 07:28:01 GMT
Attest-Quote: quote="base64=="; max-age=expireTime
Attest-Pubkey: "base64==“
Attest-Random: “af7ad3082f20“
Attest-Session-Id: “0392efc3298“
Attest-Cipher-Suite: "aes128gcm"
```

```
ATTEST /service HTTP/1.1
Attest-secret: secret="base64=="; max-age=expireTime
```

```
HTTP/1.1 200 OK
Attest-secret: secret="base64=="; max-age=expireTime
```

How does a webservice start using HTTPA?

- Use HW-TEE to generate HTTPA quotes
- Implement HTTP extension to support HTTPA
- Implement web client plug-ins to support HTTPA

HTTPA benefits

- Allow users to verify identities:
  - Software service
  - Software vendor
  - TEE
- Establish a TEE-based end-to-end trustworthy channel
  - HTTPS connection + remote attestation + secret provisioning
- Establish TEE for remote confidential computing
  - Execution security
- Allow users for more freedom to control the process over their secrets/data and reject remote services if they do not trust
- Unify remote attestation in a simple way

Feedback of HTTPA/1 from customers

- Customers want perfect forward secrecy.
  - We recommend ECDHE for HTTPA/1
- Customers want both high security and good performance.
  - We recommend ECDHE
- Complexity:
  - Reduce current complexity:
    - If you have secure end-to-end communication at L7, why do we need TLS which increases another layer of overhead?
- Scalability:
  - Current, usage model for TEE is not scalable for customers.
- To some extent, HTTPA/2 has addressed most concerns above.

HTTPA vs. RA-TLS vs. HTTPS (Cont.)

#	Differentiators	HTTPA	RA-TLS	HTTPS
1	OSI layer	Application layer	Session layer	Session layer and up
2	TEE awareness	Generic TEE (e.g. SGX, TDX, TPM)	SGX enclave	No awareness
3	Remote attestation	Yes	Yes	No
4	Workload type	HTTP web services	TLS workloads	HTTP website
5	Modified HTTP handling codes	Yes	No; Yes(if you use provisioned secrets)	No
6	Encryption in transit	Yes, if enabling TLS; encrypted messages	Yes	Yes
7	Website/Gateway CA authentication	Yes, if enabling TLS	No (self-signed)	Yes
8	Allow TLS gateway	Yes	No (skip TLS gateway if RA-TLS is bound with workload)	Yes
9	Allow TLS inspection appliance	Yes (not for TEE-encrypted HTTP messages)	No (if RA-TLS is bound with workload)	Yes

https://wentzwu.com/2020/08/21/what-osi-layer-does-tls-operate-and-why/

HTTPA vs. RA-TLS vs. HTTPS

#	Differentiators	HTTPA	RA-TLS	HTTPS
10	TLS Lib. required for workload	No	Yes, if it is bound with workload	No
11	Certificate signing required	Yes, if using TLS; No, if no using TLS	Yes (self-signed)	Yes
12	CA issued certificate	Yes, if enabling regular TLS	No, because of self-signed	Yes
13	A single webservice to use multiple TLS connections simultaneously (multiple connections)	Supported	No	No
14	Multiple webservices to share single TLS connection (multiplexing)	Supported	No, if bound with workload	No
15	TLS configurable before or during negotiation	No Limitation	Bound to each workload TCB separately, needs ISVs to get involved	No Limitation
16	TLS upgradable/replaceable	No Limitation	Subjected to each workload TCB separately, needs ISVs to get involved	No Limitation

https://wentzwu.com/2020/08/21/what-osi-layer-does-tls-operate-and-why/

Summary

- HTTPA is an L7 protocol which defines an HTTP extension to build a trusted communication channel over the **Internet**.
- HTTPA facilitates attesting web services based on TEE.
- HTTPA ties secure communication to attestation context for L7 applications.
- We envision HTTPA as an industry standard, such as RFC, to enable confidential applications and accelerate transformation towards **trustworthy Internet**.

Thank you

Hans Wang: hans.wang@intel.com

Gordon King: gordon.king@intel.com

intel®

Amazon’s approach

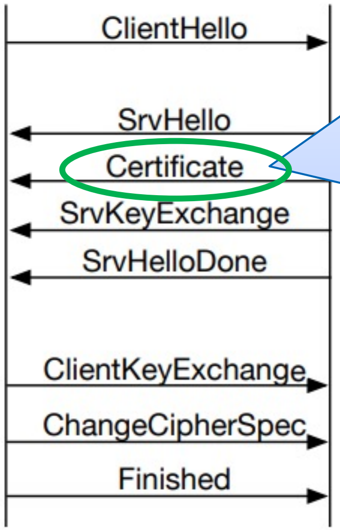
- Title: HTTP Message Signatures
- Issue:
  - Integrity of HTTP message is **not** guaranteed across multiple TLS connections
    - TLS-terminating gateway
    - TLS inspection appliance
- Key summary:
  - Describe a mechanism for signing/verifying digital signatures over components of an HTTP message.
  - Require web application-level signing key to be **separate** from any TLS certificate.

https://httpwg.org/http-extensions/draft-ietf-httpbis-message-signatures.html

RA-TLS

Client

Sever



“propose to include additional information into the X.509 certificate exchanged during a TLS handshake” – section 3

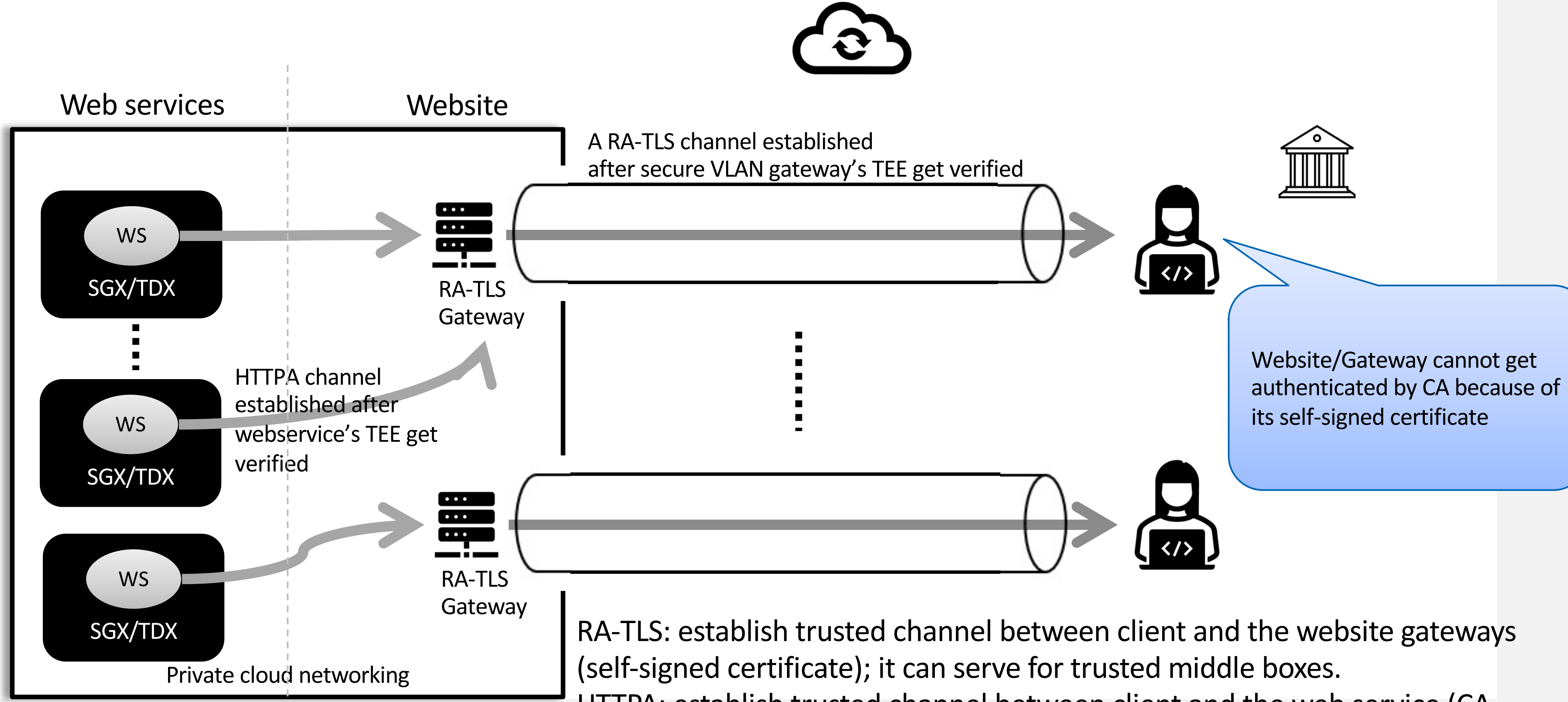
“propose to **embed** the attestation evidence as custom X.509 extensions in the server’s certificate” – section 3.2

“Extending the certificate traditionally requires resigning it by a CA. However, since we propose to use Intel SGX as a **trust root**, we can simply **self-sign** the certificate. “ – section 3.2

Figure 2: TLS 1.2 Handshake Messages.

https://arxiv.org/ftp/arxiv/papers/1801/1801.05863.pdf

How can we combine HTTPA and RA-TLS?

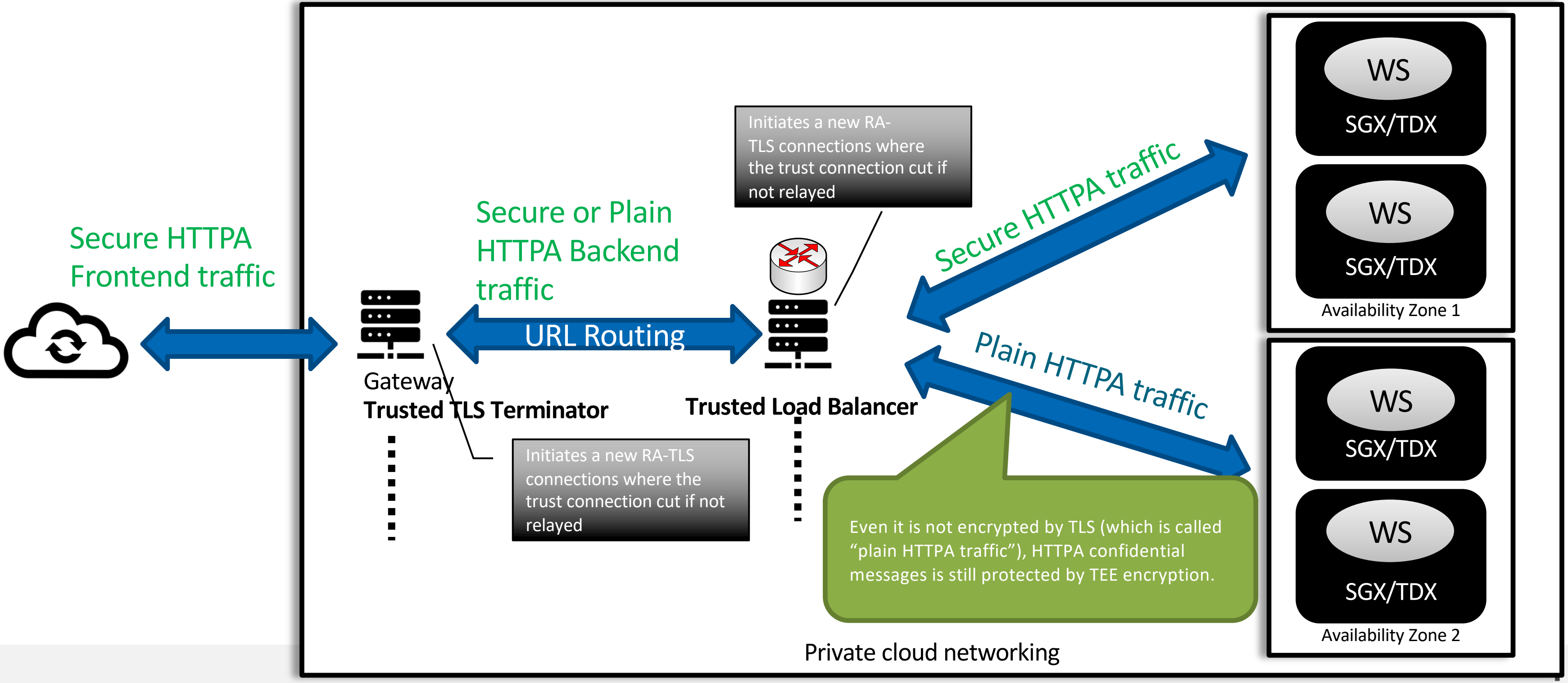

Web services
Website
WS
SGX/TDX
WS
SGX/TDX
WS
SGX/TDX
RA-TLS Gateway
RA-TLS Gateway
HTTPA channel established after webservice's TEE get verified
Private cloud networking
A RA-TLS channel established
after secure VLAN gateway's TEE get verified
</>

Website/Gateway cannot get authenticated by CA because of its self-signed certificate

RA-TLS: establish trusted channel between client and the website gateways (self-signed certificate); it can serve for trusted middle boxes.
HTTPA: establish trusted channel between client and the web service (CA-signed certificate); it can serve for L7 trusted end points.

HTTPA Using RA-TLS for Trusted URL Routing

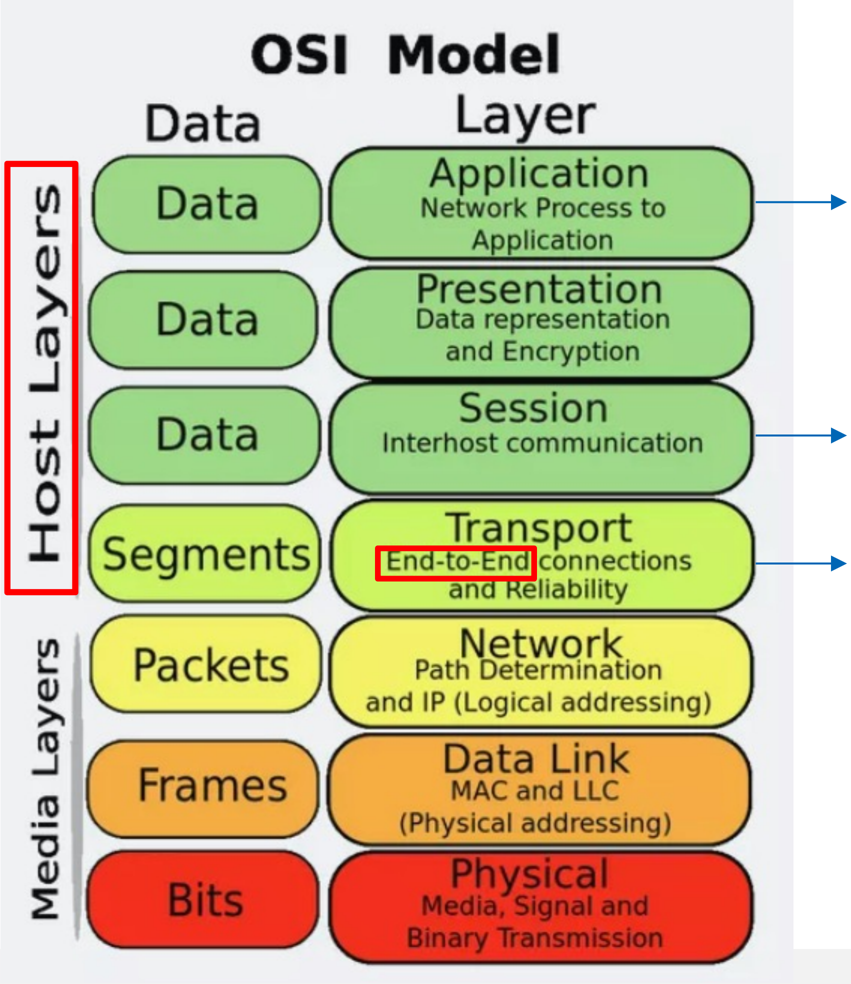
HTTPA and RA-TLS can co-exist to serve for trusted communication.



Initiates a new RA-TLS connections where the trust connection cut if not relayed

Initiates a new RA-TLS connections where the trust connection cut if not relayed

Even it is not encrypted by TLS (which is called “plain HTTPA traffic”), HTTPA confidential messages is still protected by TEE encryption.


OSI Model
Data
Layer
Host Layers
Data
Application
Network Process to Application
Data
Presentation
Data representation and Encryption
Data
Session
Interhost communication
Segments
Transport
End-to-End connections and Reliability
Media Layers
Packets
Network
Path Determination and IP (Logical addressing)
Frames
Data Link
MAC and LLC (Physical addressing)
Bits
Physical
Media, Signal and Binary Transmission

Service-to-service
e.g., HTTP(S), HTTPA

Edge-to-edge
e.g., TLS, RA-TLS

Host-to-host
e.g., TCP

https://www.lifewire.com/osi-model-reference-guide-816289