Lecture #5: IoT Device Security

Cristian Hesselman, Antonia Affinito, Savvas Kastanakis
Etienne Khan, Ting-Han Chen, and Pascal Huppert

University of Twente | May 27, 2025

UNIVERSITY OF TWENTE.

SIDN LABS

Schedule

Lecture	Date	Contents
R1	Apr 25	Course introduction
G1	Apr 30	How the core of the Internet works (recorded)
R2	May 9	Principles of IoT Security
R3	May 16	Internet Core Protocols
R4	May 23	IoT Botnet Measurements
R5	**May 27**	**IoTLS and Q&A Group Assignment**
G2	Jun 6	Guest Lecture – PQC in IoT
R6	Jun 13	IoT Security Vulnerabilities
R7	Jun 20	IoT Forensic

Today's learning objective

After the lecture, you will be able to:

- understand passive and active measurements
- identify vulnerabilities and poor security practices
- assess IoT TLS implementation in practice
- understand the importance of secure configurations and protocol evolution

Contributes to SSI learning goal #1: "Understand IoT concepts and applications, security threats, technical solutions, and a few relevant standardization efforts in the IETF"

Background – Purpose of SSL/TLS

SSL (Secure Sockets Layer) and **TLS (Transport Layer Security)** are cryptographic protocols designed to:

- **Encrypt** communication between two endpoints (e.g., IoT device and a firmware / telemetry / content / cloud server)

- Ensure **data integrity (no altering)** and **authentication (proper identity):**
  - "Set temperature to 21°C" but not "Set temperature to 35°C"
  - "assistant.google.com" but not "assistant.g88gl3.com"

- Prevent eavesdropping, tampering, and impersonation.

Background – From SSL to TLS

TLS evolved from **SSL**, first developed by Netscape around 30 years ago. The protocol has gone through multiple versions:

- **SSL 2.0** (1995), **SSL 3.0** (1996) – now deprecated
- **TLS 1.0** (1999), **TLS 1.1** (2006) – also deprecated
- **TLS 1.2** (2008) – widely used
- **TLS 1.3** (2018) – current standard with enhanced security and performance

Background – From SSL to TLS

TLS evolved from **SSL**, first developed by Netscape around 30 years ago. The protocol has gone through multiple versions:

- **SSL 2.0** (1995), **SSL 3.0** (1996) – now deprecated
- **TLS 1.0** (1999), **TLS 1.1** (2006) – also deprecated
- **TLS 1.2** (2008) – widely used
- **TLS 1.3** (2018) – current standard with enhanced security and performance

Why do protocols change? Why don't they stay the same forever?

Background – From SSL to TLS

TLS evolved from **SSL**, first developed by Netscape around 30 years ago. The protocol has gone through multiple versions:

- **SSL 2.0** (1995), **SSL 3.0** (1996) – now deprecated
- **TLS 1.0** (1999), **TLS 1.1** (2006) – also deprecated
- **TLS 1.2** (2008) – widely used
- **TLS 1.3** (2018) – current standard with enhanced security and performance

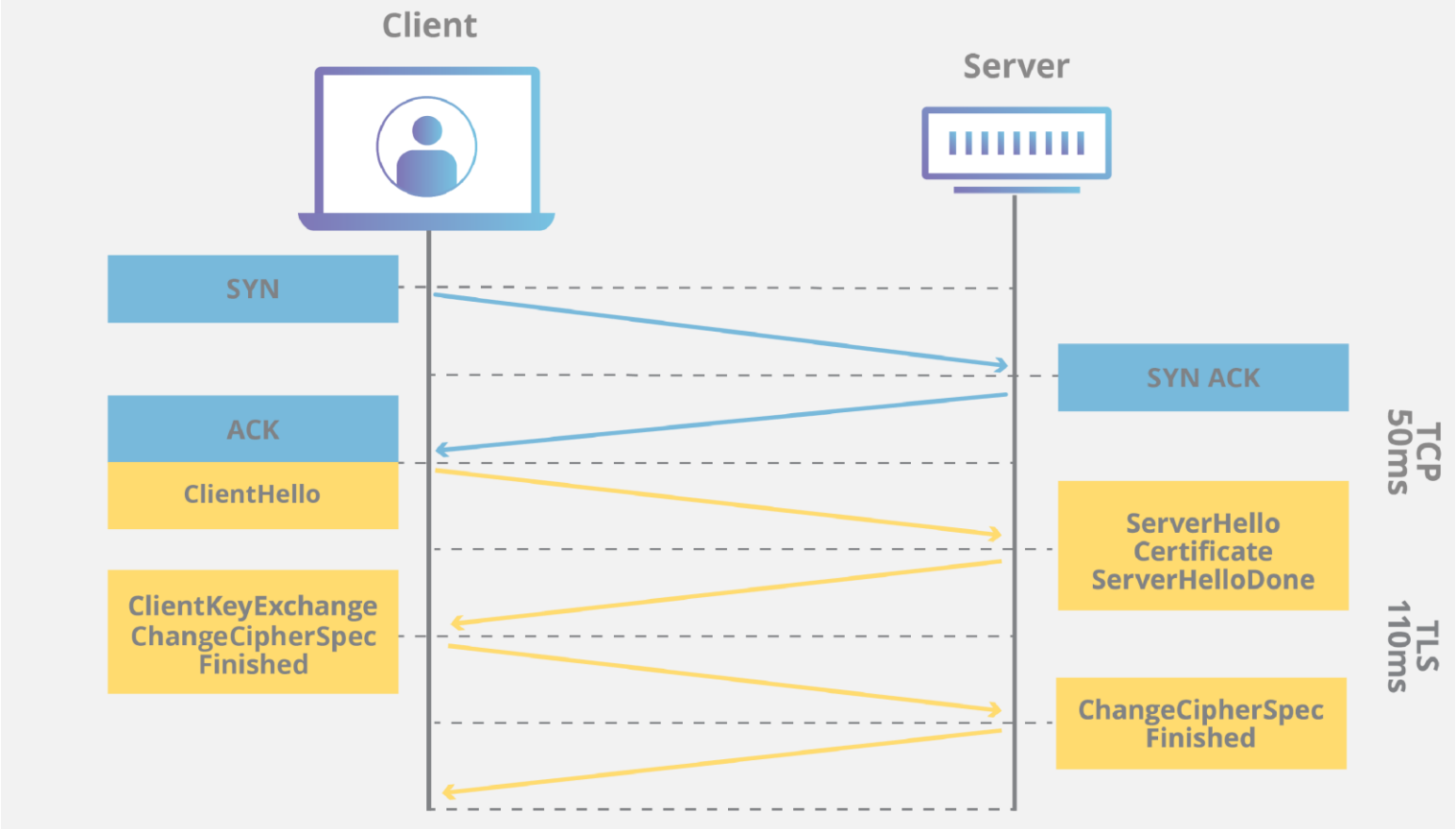
Why do protocols change? Why don't they stay the same forever?

Protocol evolution is driven by a variety of factors, including emerging vulnerabilities, cryptographic advancements, performance optimizations, and the need to support modern Internet infrastructure.

Background – TLS-101

Blue Block: Establish Connection

- **SYN** (Client → Server)
  - Client sends a request to initiate a TCP connection.
- **SYN-ACK** (Server → Client)
  - Server acknowledges and responds.
- **ACK** (Client → Server)
  - Client confirms. Now the TCP connection is open.



https://www.cloudflare.com/en-gb/learning/ssl/what-happens-in-a-tls-handshake/

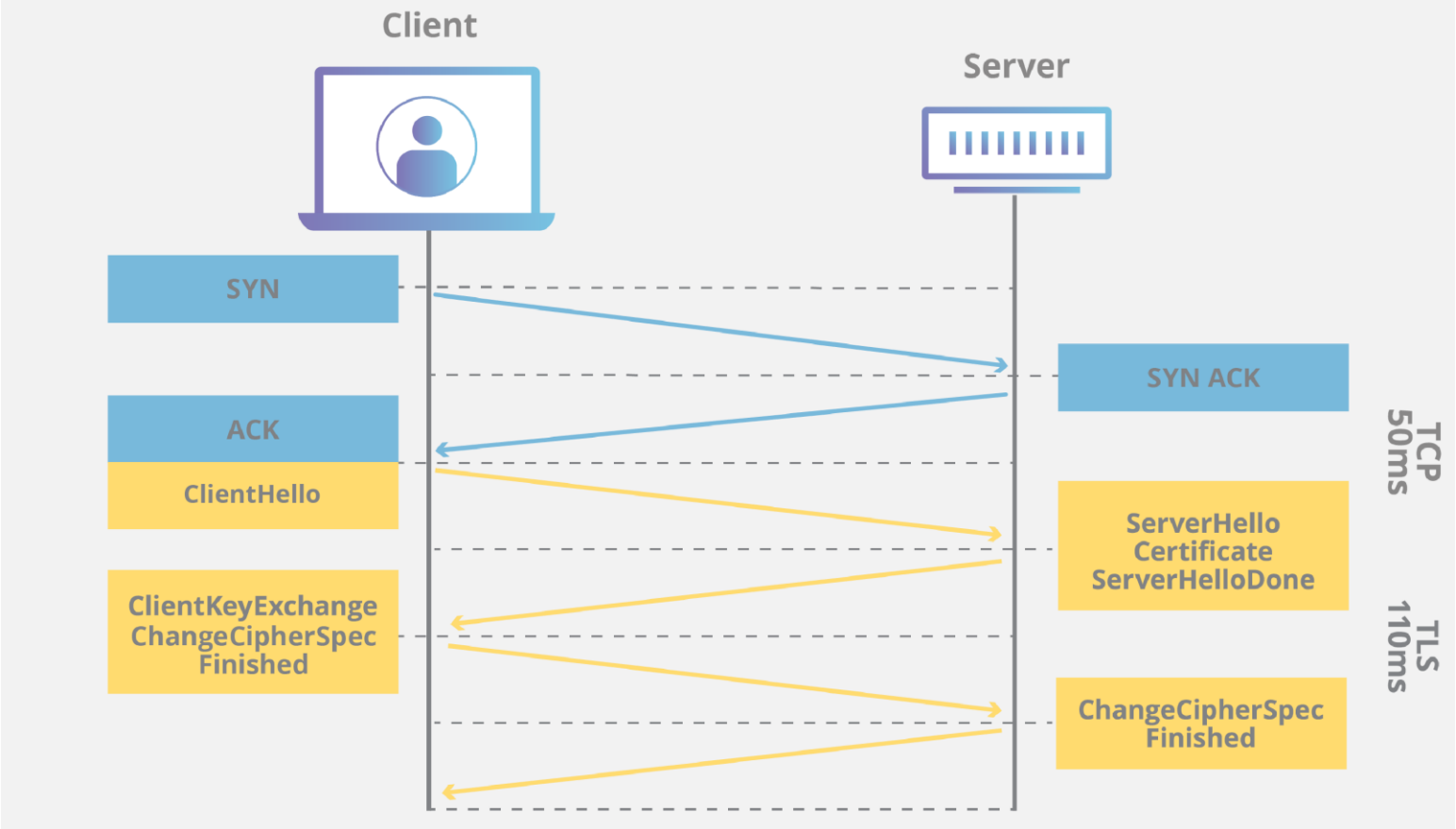
UNIVERSITY OF TWENTE.

SDN LABS

Background – TLS-101

Blue Block: Establish Connection

- **SYN** (Client → Server)
  - Client sends a request to initiate a TCP connection.
- **SYN-ACK** (Server → Client)
  - Server acknowledges and responds.
- **ACK** (Client → Server)
  - Client confirms. Now the TCP connection is open.



https://www.cloudflare.com/en-gb/learning/ssl/what-happens-in-a-tls-handshake/

Notice that **TLS runs on top of TCP**, since it needs a **reliable, ordered connection**—which TCP provides—to securely exchange handshake messages.

UNIVERSITY OF TWENTE. SDN LABS

Background – TLS-101

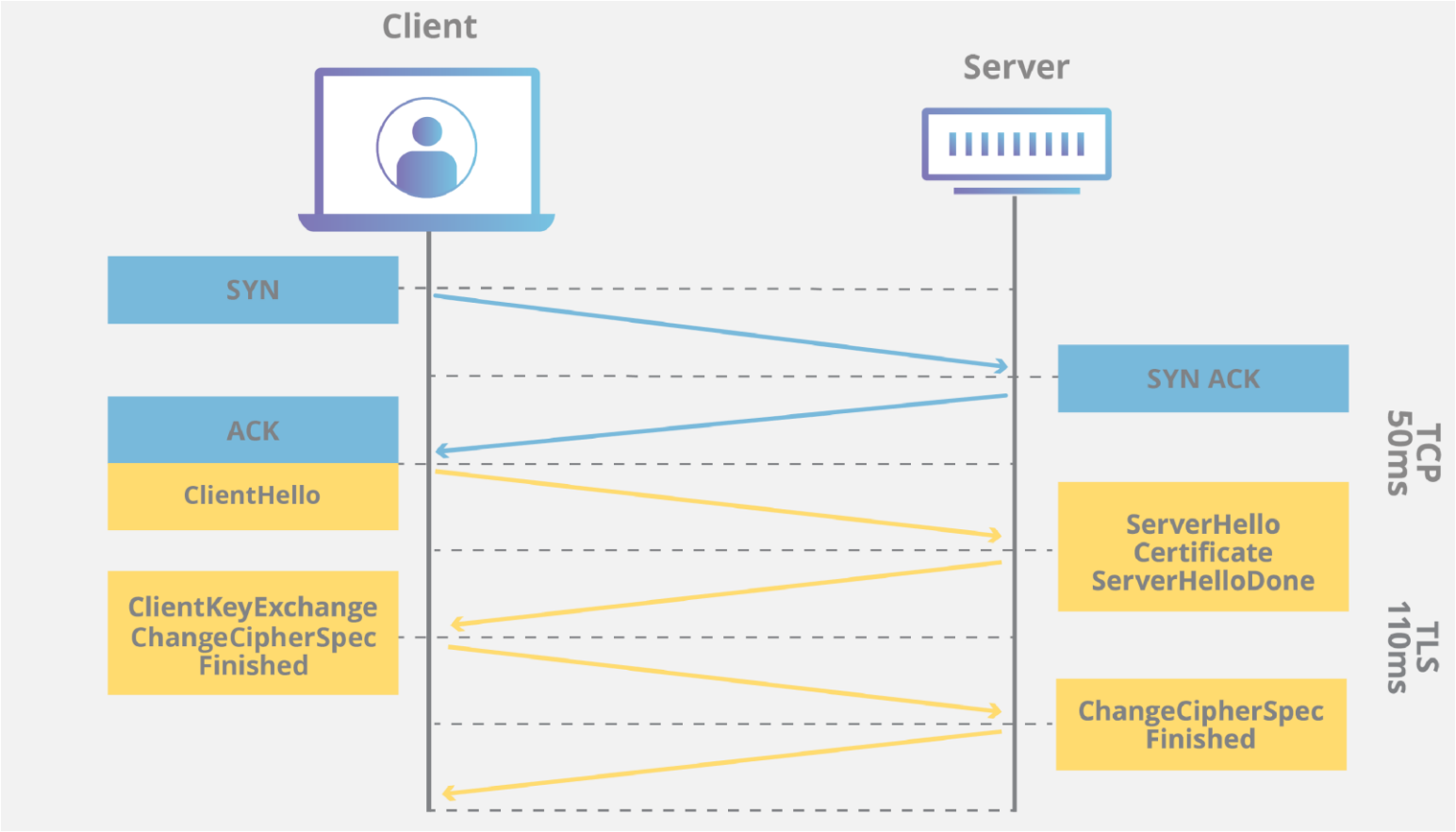
Orange Block: Establish Cryptographic Setup

ClientHello (Client → Server)
- Client proposes TLS version, supported cipher suites, and other advanced features.

ServerHello + Certificate + ServerHelloDone (Server → Client)
- Server selects the TLS version and cipher suite based on compatibility.
- Sends its **certificate** (signed by a trusted Certificate Authority (CA)) to prove its identity.
- `ServerHelloDone` marks the end of the server's part of the handshake.

ClientKeyExchange + ChangeCipherSpec + Finished (Client → Server)
- Client sends key material (often encrypted with server's public key).
- Indicates it is ready to switch to encrypted communication (`ChangeCipherSpec`).
- Sends a hashed summary of the handshake.

ChangeCipherSpec + Finished (Srvr → Client)
- Server also switches to encrypted mode.
- Sends its own `Finished` message to confirm everything is synced.



https://www.cloudflare.com/en-gb/learning/ssl/what-happens-in-a-tls-handshake/

UNIVERSITY OF TWENTE. SDN LABS

Goals & Challenges

There is a **research gap** in understanding how IoT devices implement TLS, specifically whether they:

- Use **secure TLS versions and ciphersuites**,
- **Correctly validate certificates** with trusted root CAs,
- **Adopt new TLS versions** over time.

Studying this is challenging because:

- Devices are often **closed-source**, so **blackbox testing** is needed.
- **TLS traffic is hard to trigger**, as communication depends on device behavior.
- Existing data sources (e.g., ISP/IXP logs) **lack device-level, long-term visibility**.

Research Questions

✅ **RQ1: Do devices securely establish TLS connections?**

- Do IoT devices use secure TLS versions and modern ciphersuites?
- Are they resilient to in-network adversaries?
- Do they adopt secure configurations over time (e.g., TLS 1.3)?

Research Questions

✅ **RQ1: Do devices securely establish TLS connections?**

- Do IoT devices use secure TLS versions and modern ciphersuites?
- Are they resilient to in-network adversaries?
- Do they adopt secure configurations over time (e.g., TLS 1.3)?

🛡️ **RQ2: Do devices properly validate TLS certificates?**

- Do devices accept invalid certificates?
- Do their root stores (i.e., list of trusted CAs on a device) contain **stale or distrusted CAs**?
- Do they properly validate the certificate chain, hostname, and X.509 extensions (as per RFC 2818 and 5280)?

Research Questions

RQ1: Do devices securely establish TLS connections?

- Do IoT devices use secure TLS versions and modern ciphersuites?
- Are they resilient to in-network adversaries?
- Do they adopt secure configurations over time (e.g., TLS 1.3)?

RQ2: Do devices properly validate TLS certificates?

- Do devices accept invalid certificates?
- Do their root stores (i.e., list of trusted CAs on a device) contain **stale or distrusted CAs**?
- Do they properly validate the certificate chain, hostname, and X.509 extensions (as per RFC 2818 and 5280)?

RQ3: What is the diversity of TLS behavior across and within devices?

- How consistent is TLS behavior across different devices?
- Do individual devices show **varying TLS behavior** (e.g., using multiple TLS libraries or components)?

Methodology: Testbed

Cameras (n = 7)	Smart Hubs (n = 7)	Home Automation (n = 7)	TV (n = 5)	Audio (n = 7)	Appliances (n = 7)
Blink Camera*	Blink Hub	Smartlife Bulb	Fire TV	Google Home Mini	GE Microwave
Amazon Cloudcam*	Smartthings Hub	Smartlife Remote	Samsung TV*	Amazon Echo Plus	Samsung Washer*
Zmodo Doorbell	Philips Hub	Meross Dooropener	LG TV	Amazon Echo Dot	Samsung Dryer
Yi Camera	Wink Hub 2	TP-Link Bulb	Roku TV	Amazon Echo Dot 3	Samsung Fridge
D-Link Camera	Sengled Hub*	Nest Thermostat	Apple TV	Amazon Echo Spot	Smarter iKettle
Amcrest Camera	Switchbot Hub	TP-Link Plug		Harman Invoke	Behmor Brewer
Ring Doorbell*	Insteon Hub*	Wemo Plug		Apple HomePod	LG Dishwasher*

The study analyzes **40 TLS-enabled IoT devices** across 6 categories (e.g., cameras, smart hubs, TVs) in a **smart home-like testbed**.

- Devices are placed in an isolated studio-like space and interacted with via apps on smartphones.
- Network traffic is captured at the gateway (at the point where the IoT devices connect to the Internet, e.g., a router or firewall).

Devices follow a realistic update model:

- **Auto-updating devices** follow the manufacturer's schedule.
- **Manually updated devices** are updated only when prompted by their apps.
  This approach reflects typical user behavior and accounts for the difficulty of automating regular updates.

UNIVERSITY OF TWENTE.

SIDN LABS

Methodology: Experiments

The study uses both **passive** and **active experiments**:

UNIVERSITY OF TWENTE.
SDN LABS

Methodology: Experiments

The study uses both **passive** and **active experiments**:

Passive experiments:

- Record normal device traffic without interference.
- Include usage by ≈40 participants over ≈2 years (Jan 2018 – Mar 2020).
- Show real-world behavior both **with and without user interaction**.

Methodology: Experiments

The study uses both **passive** and **active experiments**:

Passive experiments:

- Record normal device traffic without interference.
- Include usage by ≈40 participants over ≈2 years (Jan 2018 – Mar 2020).
- Show real-world behavior both **with and without user interaction**.

Active experiments:

- Use *mitmproxy* to **intercept and analyze TLS traffic**.
  - *mitmproxy* is a tool that lets researchers see inside encrypted traffic by pretending to be the server a device is talking to.
  - In this study, it was used during active experiments to impersonate servers and observe whether IoT devices accept invalid TLS certificates after reboot.

Methodology: Experiments

The study uses both **passive** and **active experiments**:

Passive experiments:

- Record normal device traffic without interference.
- Include usage by ≈40 participants over ≈2 years (Jan 2018 – Mar 2020).
- Show real-world behavior both **with and without user interaction**.

Active experiments:

- Use *mitmproxy* to **intercept and analyze TLS traffic**.
  - *mitmproxy* is a tool that lets researchers see inside encrypted traffic by pretending to be the server a device is talking to.
  - In this study, it was used during active experiments to impersonate servers and observe whether IoT devices accept invalid TLS certificates after reboot.
- Devices are **rebooted using smart plugs** to trigger TLS connections.

Methodology: Experiments

The study uses both **passive** and **active experiments**:

Passive experiments:

- Record normal device traffic without interference.
- Include usage by ≈40 participants over ≈2 years (Jan 2018 – Mar 2020).
- Show real-world behavior both **with and without user interaction**.

Active experiments:

- Use *mitmproxy* to **intercept and analyze TLS traffic**.
  - *mitmproxy* is a tool that lets researchers see inside encrypted traffic by pretending to be the server a device is talking to.
  - In this study, it was used during active experiments to impersonate servers and observe whether IoT devices accept invalid TLS certificates after reboot.
- Devices are **rebooted using smart plugs** to trigger TLS connections.
- Run in March 2021 without user involvement.

Results: TLS Connection Security (1/3)

Figure 1: TLS version support for IoT devices. Devices often use multiple versions (rows 2-12), can encounter lack of server support (rows 2-8) and rarely adopt better TLS versions over time (rows 7-10). 28 devices use TLS 1.2 for the vast majority of their advertised and established connections, and are not shown in this figure.

UNIVERSITY OF TWENTE.

SIDN LABS

Results: TLS Connection Security (1/3)

- On the positive side, the IoT devices in this study often rely on TLS1.2 or above. 28/40 use TLS1.2 and are not shown in the figure. But…

Figure 1: TLS version support for IoT devices. Devices often use multiple versions (rows 2-12), can encounter lack of server support (rows 2-8) and rarely adopt better TLS versions over time (rows 7-10). 28 devices use TLS 1.2 for the vast majority of their advertised and established connections, and are not shown in this figure.

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Connection Security (1/3)

- On the positive side, the IoT devices in this study often rely on TLS1.2 or above. 28/40 use TLS1.2 and are not shown in the figure. But…
- Devices often advertise newer TLS versions but establish connections with older versions due to limited server support (see **LG Dishwasher**, **Samsung Dryer**, and **Apple HomePod)**.

Figure 1: TLS version support for IoT devices. Devices often use multiple versions (rows 2-12), can encounter lack of server support (rows 2-8) and rarely adopt better TLS versions over time (rows 7-10). 28 devices use TLS 1.2 for the vast majority of their advertised and established connections, and are not shown in this figure.

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Connection Security (1/3)

- On the positive side, the IoT devices in this study often rely on TLS1.2 or above. 28/40 use TLS1.2 and are not shown in the figure. But…
- Devices often advertise newer TLS versions but establish connections with older versions due to limited server support (see **LG Dishwasher**, **Samsung Dryer**, and **Apple HomePod)**.
- Devices rarely upgrade to newer versions (since there are long flat patters across the timeline except for a few cases such as **Google Home Mini**, and **Apple TV)**

Figure 1: TLS version support for IoT devices. Devices often use multiple versions (rows 2-12), can encounter lack of server support (rows 2-8) and rarely adopt better TLS versions over time (rows 7-10). 28 devices use TLS 1.2 for the vast majority of their advertised and established connections, and are not shown in this figure.

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Connection Security (2/3)

Table 6: IoT devices that support older TLS versions.

Device	TLS 1.0 Available?	TLS 1.1 Available?
Zmodo Doorbell	✓	✓
Wink Hub 2	✓	✓
Yi Camera	✓	✓
Philips Hub	✓	✓
Smarter Brewer	✓	✓
TP-Link Bulb	✓	✓
Roku TV	✓	✓
Meross Dooropener	✓	✓
LG TV	✓	✓
Google Home Mini	✓	✓
Amazon Fire TV	✓	✓
Amazon Echo Spot	✓	✓
Amazon Echo Plus	✓	✓
Amazon Echo Dot	✓	✓
Amcrest Camera	✓	✓
Samsung Fridge	✗	✓
Samsung Dryer	✗	✓
Wemo Plug	✓	✗

Devices don’t retire older versions even when newer ones are supported, increasing attack surface.

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Connection Security (2/3)

Table 6: IoT devices that support older TLS versions.

Device	TLS 1.0 Available?	TLS 1.1 Available?
Zmodo Doorbell	✓	✓
Wink Hub 2	✓	✓
Yi Camera	✓	✓
Philips Hub	✓	✓
Smarter Brewer	✓	✓
TP-Link Bulb	✓	✓
Roku TV	✓	✓
Meross Dooropener	✓	✓
LG TV	✓	✓
Google Home Mini	✓	✓
Amazon Fire TV	✓	✓
Amazon Echo Spot	✓	✓
Amazon Echo Plus	✓	✓
Amazon Echo Dot	✓	✓
Amcrest Camera	✓	✓
Samsung Fridge	✗	✓
Samsung Dryer	✗	✓
Wemo Plug	✓	✗

Devices don't retire older versions even when newer ones are supported, increasing attack surface. WHY?

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Connection Security (3/3)

Table 5: IoT devices that *downgrade* security upon connection failures (✓ indicates downgrade).

Device	Failed Handshake	Incomplete Handshake	Behavior	Downgraded / Total Destinations
Amazon Echo Dot	✗	✓	Falls back to using SSL 3.0	7 / 9
Amazon Echo Plus	✗	✓	Falls back to using SSL 3.0	6 / 7
Amazon Echo Spot	✗	✓	Falls back to using SSL 3.0	11 / 15
Amazon Fire TV	✗	✓	Falls back to using SSL 3.0	13 / 21
Apple Homepod	✗	✓	Falls back to using TLS 1.0	7 / 9
Google Home Mini	✗	✓	Falls back to supporting a weaker ciphersuite and signature algorithm (TLS_RSA_WITH_3DES_EDE_CBC_SHA and RSA_PKCS1_SHA1)	5 / 5
Roku TV	✓	✓	Falls back from offering 73 ciphersuites to just 1 (TLS_RSA_WITH_RC4_128_SHA)	8 / 15

7 IoT devices weakened their TLS configurations after incomplete handshakes —> a behavior exploitable by attackers.

UNIVERSITY OF TWENTE.
SDN LABS

Results: TLS Connection Security (3/3)

Table 5: IoT devices that *downgrade* security upon connection failures (✓ indicates downgrade).

Device	Failed Handshake	Incomplete Handshake	Behavior	Downgraded / Total Destinations
Amazon Echo Dot	✗	✓	Falls back to using SSL 3.0	7 / 9
Amazon Echo Plus	✗	✓	Falls back to using SSL 3.0	6 / 7
Amazon Echo Spot	✗	✓	Falls back to using SSL 3.0	11 / 15
Amazon Fire TV	✗	✓	Falls back to using SSL 3.0	13 / 21
Apple Homepod	✗	✓	Falls back to using TLS 1.0	7 / 9
Google Home Mini	✗	✓	Falls back to supporting a weaker ciphersuite and signature algorithm (TLS_RSA_WITH_3DES_EDE_CBC_SHA and RSA_PKCS1_SHA1)	5 / 5
Roku TV	✓	✓	Falls back from offering 73 ciphersuites to just 1 (TLS_RSA_WITH_RC4_128_SHA)	8 / 15

7 IoT devices weakened their TLS configurations after incomplete handshakes —> a behavior exploitable by attackers.

- **Amazon Echo and Fire TV devices:**
  - Fell back to **SSL 3.0**, a deprecated and insecure protocol.
- **Apple HomePod:**
  - Downgraded to **TLS 1.0**, also deprecated and vulnerable.
- **Google Home Mini:**
  - Switched to weak cipher: 3DES + SHA1.

UNIVERSITY OF TWENTE. SDN LABS

Results: Certificate Validation

Table 7: IoT devices vulnerable to TLS *interception* attacks. (✓ indicates vulnerability).

Device	No-Validation	InvalidBasic-Constraints	Wrong-Hostname	Vulnerable/Total Destinations
Zmodo Doorbell	✓	✓	✓	6 / 6
Amcrest Camera	✓	✓	✓	2 / 2
Smarter Brewer	✓	✓	✓	1 / 1
Yi Camera	✓	✓	✓	1 / 1
Wink Hub 2	✓	✓	✓	1 / 2
LG TV	✓	✓	✓	1 / 2
Smartthings Hub	✓	✓	✓	1 / 3
Amazon Echo Plus	✗	✗	✓	1 / 8
Amazon Echo Dot	✗	✗	✓	1 / 9
Amazon Echo Spot	✗	✗	✓	1 / 17
Amazon Fire TV	✗	✗	✓	1 / 21

UNIVERSITY OF TWENTE.

SDN LABS

Results: Certificate Validation

🔓 TLS Certificate Validation Failures (Table 7)

- 11 IoT devices were vulnerable to TLS interception attacks (because they do not properly validate server certificates)

Table 7: IoT devices vulnerable to TLS *interception* attacks. (✓ indicates vulnerability).

Device	No-Validation	InvalidBasic-Constraints	Wrong-Hostname	Vulnerable/Total Destinations
Zmodo Doorbell	✓	✓	✓	6 / 6
Amcrest Camera	✓	✓	✓	2 / 2
Smarter Brewer	✓	✓	✓	1 / 1
Yi Camera	✓	✓	✓	1 / 1
Wink Hub 2	✓	✓	✓	1 / 2
LG TV	✓	✓	✓	1 / 2
Smartthings Hub	✓	✓	✓	1 / 3
Amazon Echo Plus	✗	✗	✓	1 / 8
Amazon Echo Dot	✗	✗	✓	1 / 9
Amazon Echo Spot	✗	✗	✓	1 / 17
Amazon Fire TV	✗	✗	✓	1 / 21

UNIVERSITY OF TWENTE. SDN LABS

Results: Certificate Validation

🔓 TLS Certificate Validation Failures (Table 7)

- 11 IoT devices were vulnerable to TLS interception attacks (because they do not properly validate server certificates)

🛑 Types of Validation Failures:

- **NoValidation**: Devices accepted self-signed certs (7 devices)
- **InvalidBasicConstraints**: Devices accepted improperly signed certificates (7 devices)
- **WrongHostname**: Devices didn't verify if cert matched the server name (11 devices)

Table 7: IoT devices vulnerable to TLS *interception* attacks. (✓ indicates vulnerability).

Device	No-Validation	InvalidBasic-Constraints	Wrong-Hostname	Vulnerable/Total Destinations
Zmodo Doorbell	✓	✓	✓	6 / 6
Amcrest Camera	✓	✓	✓	2 / 2
Smarter Brewer	✓	✓	✓	1 / 1
Yi Camera	✓	✓	✓	1 / 1
Wink Hub 2	✓	✓	✓	1 / 2
LG TV	✓	✓	✓	1 / 2
Smartthings Hub	✓	✓	✓	1 / 3
Amazon Echo Plus	✗	✗	✓	1 / 8
Amazon Echo Dot	✗	✗	✓	1 / 9
Amazon Echo Spot	✗	✗	✓	1 / 17
Amazon Fire TV	✗	✗	✓	1 / 21

UNIVERSITY OF TWENTE. SIDN LABS

Results: Certificate Validation

🔓 TLS Certificate Validation Failures (Table 7)

- 11 IoT devices were vulnerable to TLS interception attacks (because they do not properly validate server certificates)

🛑 Types of Validation Failures:

- **NoValidation**: Devices accepted self-signed certs (7 devices)
- **InvalidBasicConstraints**: Devices accepted improperly signed certificates (7 devices)
- **WrongHostname**: Devices didn't verify if cert matched the server name (11 devices)

🔓 Severe Cases:

- **7 devices failed all 3 checks** (e.g., Zmodo Doorbell, LG TV, Amcrest Camera)
- **Amazon Echo & Fire TV** devices failed hostname validation only → Still enough to **decrypt traffic with a valid cert for a different domain**

Table 7: IoT devices vulnerable to TLS *interception* attacks. (✓ indicates vulnerability).

Device	No-Validation	InvalidBasic-Constraints	Wrong-Hostname	Vulnerable/Total Destinations
Zmodo Doorbell	✓	✓	✓	6 / 6
Amcrest Camera	✓	✓	✓	2 / 2
Smarter Brewer	✓	✓	✓	1 / 1
Yi Camera	✓	✓	✓	1 / 1
Wink Hub 2	✓	✓	✓	1 / 2
LG TV	✓	✓	✓	1 / 2
Smartthings Hub	✓	✓	✓	1 / 3
Amazon Echo Plus	✗	✗	✓	1 / 8
Amazon Echo Dot	✗	✗	✓	1 / 9
Amazon Echo Spot	✗	✗	✓	1 / 17
Amazon Fire TV	✗	✗	✓	1 / 21

UNIVERSITY OF TWENTE. SDN LABS

Results: TLS Behavior Diversity

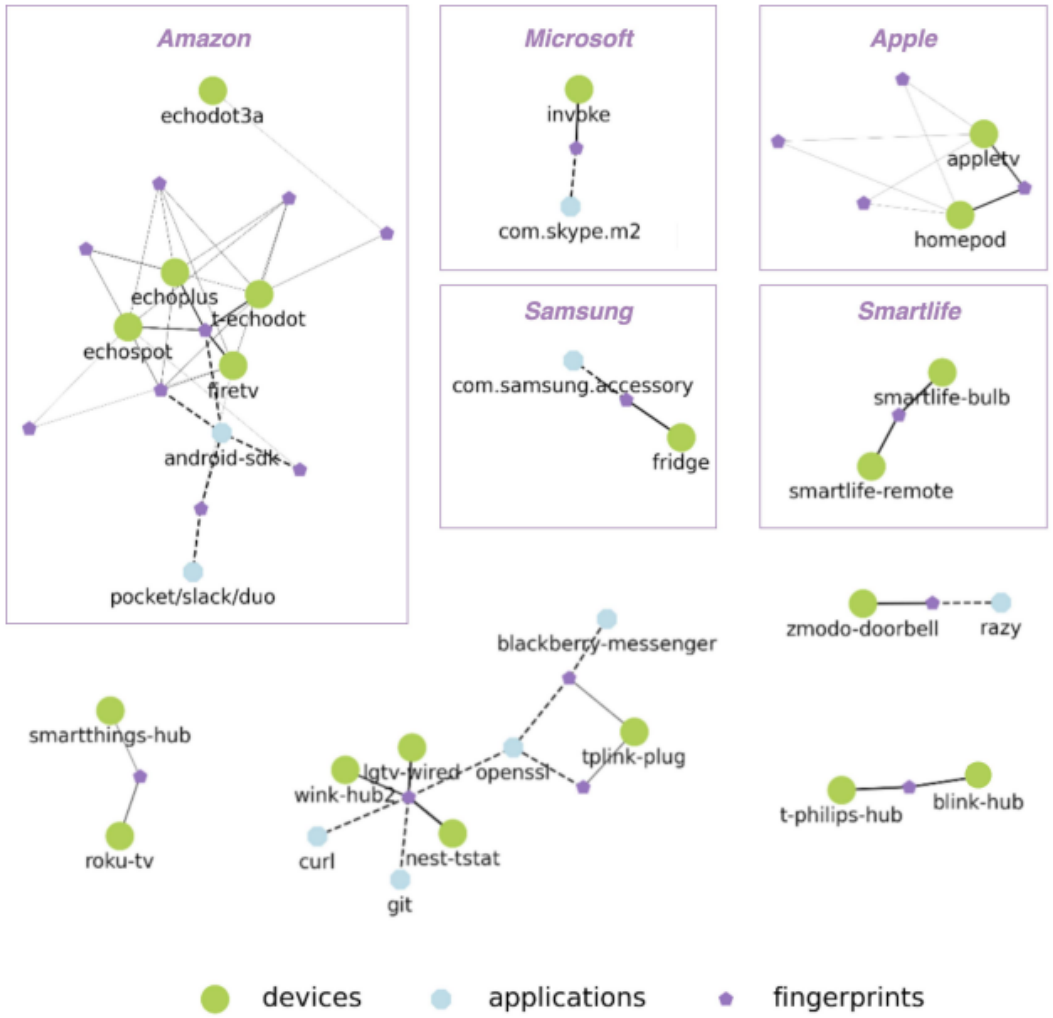


Figure 5: IoT devices that likely share TLS libraries with other devices and applications.

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Behavior Diversity

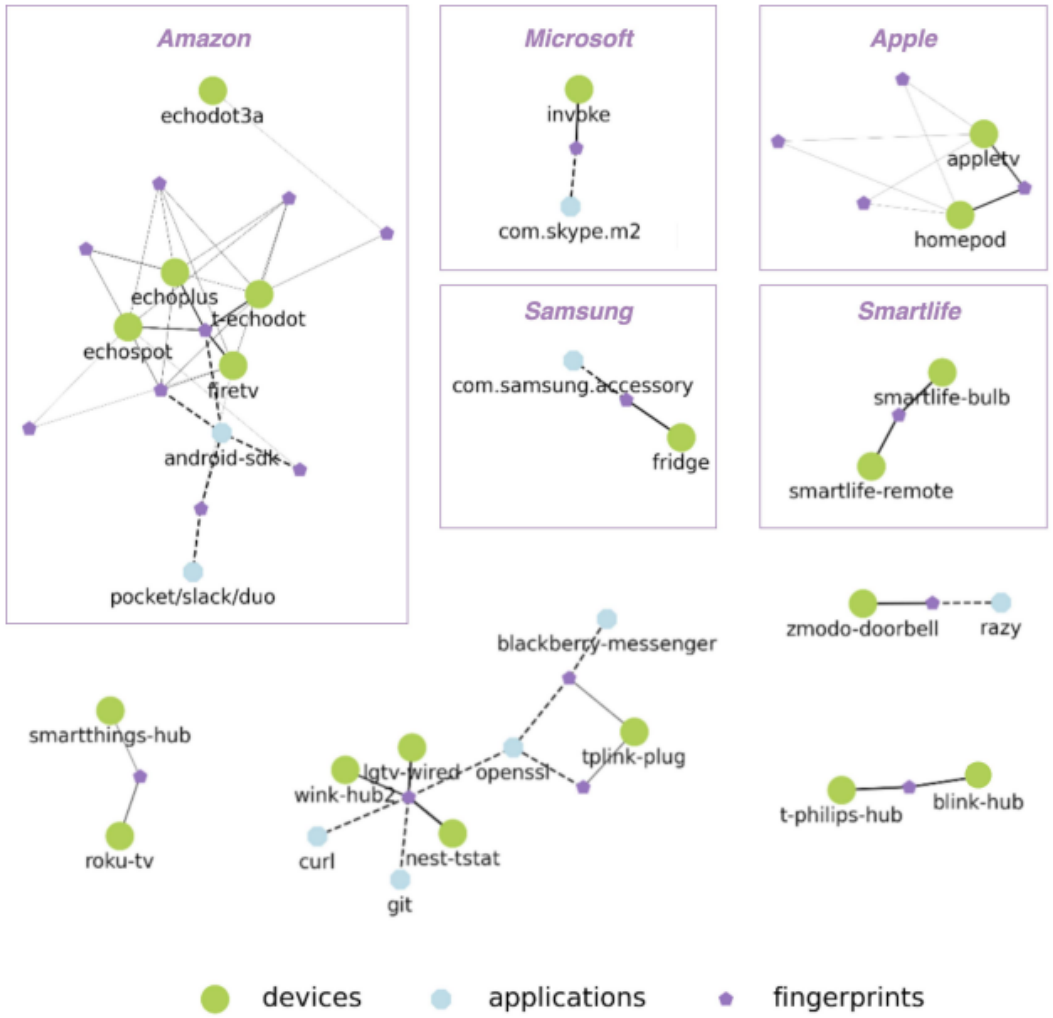


Figure 5: IoT devices that likely share TLS libraries with other devices and applications.

- Many devices use **multiple TLS libraries** internally:
  → 14 of 32 devices showed more than one TLS fingerprint (unique id from TLS session handshake characteristics)
  → Leads to inconsistent validation, harder updates, and potential weak points

UNIVERSITY OF TWENTE.

SDN LABS

Results: TLS Behavior Diversity

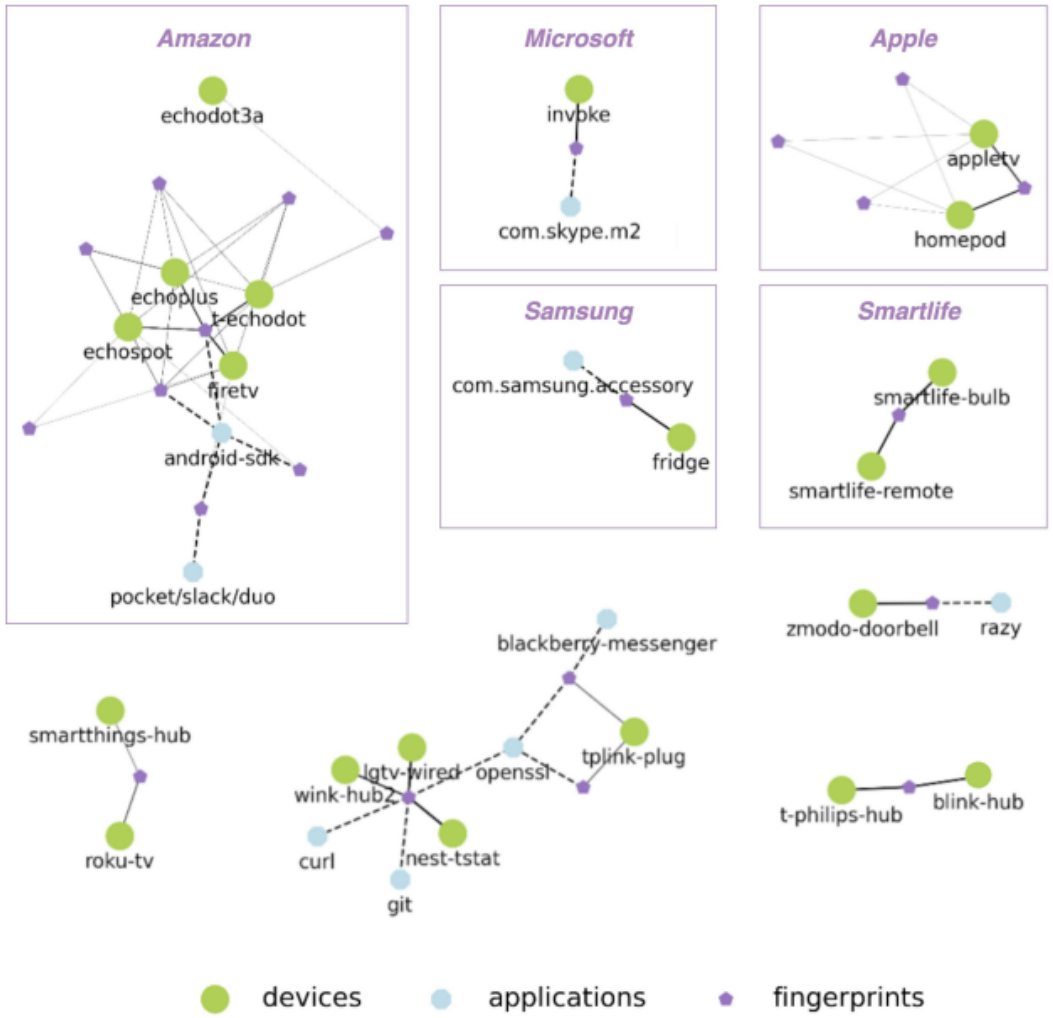


Figure 5: IoT devices that likely share TLS libraries with other devices and applications.

- Many devices use **multiple TLS libraries** internally:
  → 14 of 32 devices showed more than one TLS fingerprint (unique id from TLS session handshake characteristics)
  → Leads to inconsistent validation, harder updates, and potential weak points
- Devices often **share TLS libraries** (e.g., OpenSSL, Android SDK):
  → 19 devices shared fingerprints with known apps/libraries
  → A single vulnerability can impact many devices at once

UNIVERSITY OF TWENTE. SDN LABS

Results: TLS Behavior Diversity

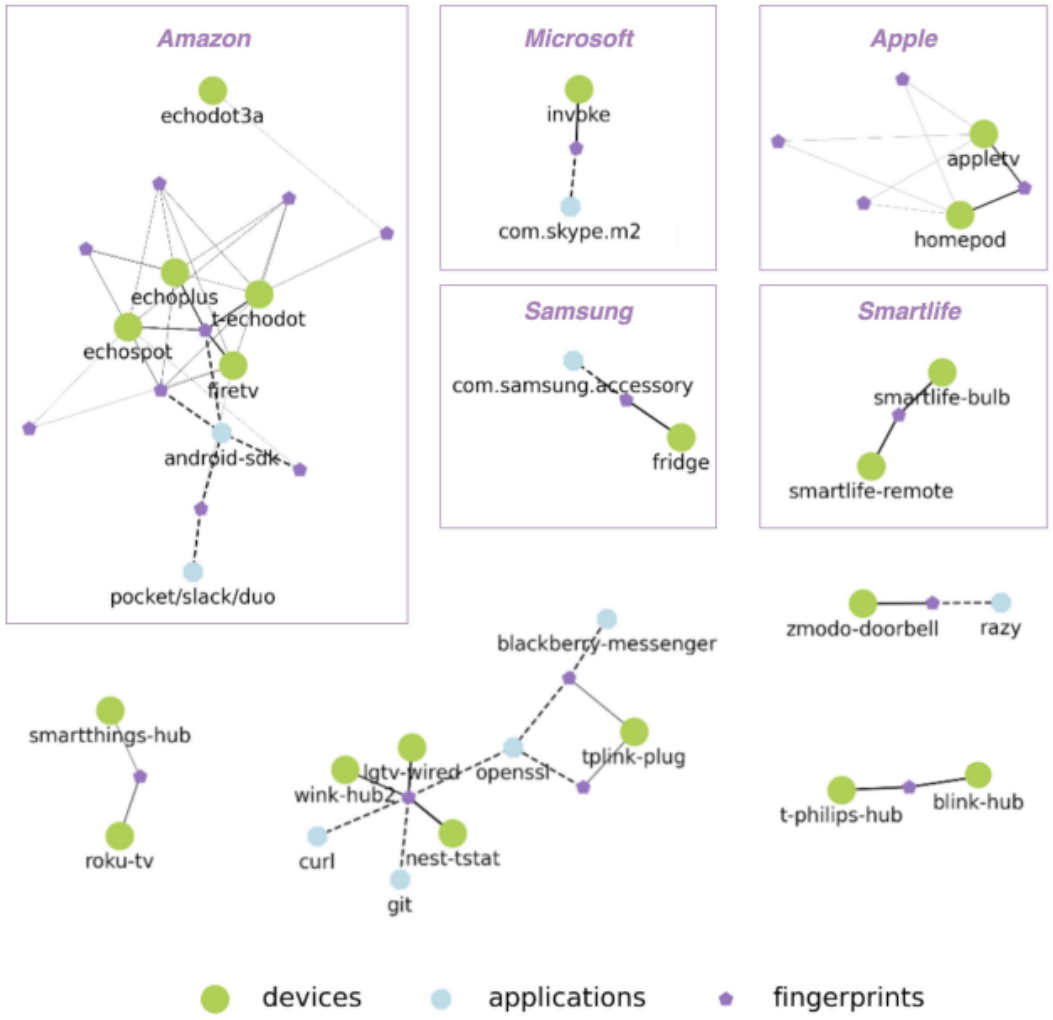


Figure 5: IoT devices that likely share TLS libraries with other devices and applications.

- Many devices use **multiple TLS libraries** internally:
  → 14 of 32 devices showed more than one TLS fingerprint (unique id from TLS session handshake characteristics)
  → Leads to inconsistent validation, harder updates, and potential weak points
- Devices often **share TLS libraries** (e.g., OpenSSL, Android SDK):
  → 19 devices shared fingerprints with known apps/libraries
  → A single vulnerability can impact many devices at once
- Manufacturers **reuse the same TLS stacks** across devices:
  → e.g., all Amazon devices cluster together
  → This helps attackers scale exploits if one stack is insecure

UNIVERSITY OF TWENTE. SIDN LABS

Recommendations

- **Use only modern TLS versions (1.2 or 1.3)** — disable old ones like SSL 3.0 or TLS 1.0.
- **Keep TLS libraries and root certificates up to date** — remove old or untrusted certificates.
- **Avoid downgrade behavior** — devices should reject insecure connections, not retry with weaker settings.
- **Perform proper certificate validation** — always check if certificates are valid, trusted, and not expired.

Limitations

- Only a small set of devices was tested
  → Results may not generalize to the entire IoT ecosystem.

- Attack scope was limited to simple TLS interception
  → More advanced attacks (e.g., POODLE, SWEET32) were not evaluated.

Disclosure

Vulnerabilities were responsibly disclosed to vendors
→ Only one vendor confirmed fixing the issue; others downplayed the risk.

Reproducibility

"To ensure reproducibility and enable new research, we have made all of our longitudinal TLS handshake data, controlled experimentation data and analysis software publicly available at: https://github.com/NEU-SNS/IoTLS."

Summary

1) Studied TLS security in 40 consumer IoT devices over 2 years using both passive and active measurements.

UNIVERSITY OF TWENTE.
SIDN LABS

Summary

1) Studied TLS security in 40 consumer IoT devices over 2 years using both passive and active measurements.

2) Evaluated how devices:
   a. Use secure TLS versions and ciphers
   b. Perform certificate validation
   c. Maintain trusted root certificate stores

Summary

1) Studied TLS security in 40 consumer IoT devices over 2 years using both passive and active measurements.

2) Evaluated how devices:
   a. Use secure TLS versions and ciphers
   b. Perform certificate validation
   c. Maintain trusted root certificate stores

3) Found that while many devices use TLS 1.2, a big number still:
   a. Supports outdated versions or weak ciphers
   b. Fails to validate certificates properly
   c. Retains deprecated or distrusted root certificates

Summary

1) Studied TLS security in 40 consumer IoT devices over 2 years using both passive and active measurements.

2) Evaluated how devices:
   a. Use secure TLS versions and ciphers
   b. Perform certificate validation
   c. Maintain trusted root certificate stores

3) Found that while many devices use TLS 1.2, a big number still:
   a. Supports outdated versions or weak ciphers
   b. Fails to validate certificates properly
   c. Retains deprecated or distrusted root certificates

4) Revealed inconsistent behavior due to multiple TLS libraries per device

Summary

1) Studied TLS security in 40 consumer IoT devices over 2 years using both passive and active measurements.

2) Evaluated how devices:
   a. Use secure TLS versions and ciphers
   b. Perform certificate validation
   c. Maintain trusted root certificate stores

3) Found that while many devices use TLS 1.2, a big number still:
   a. Supports outdated versions or weak ciphers
   b. Fails to validate certificates properly
   c. Retains deprecated or distrusted root certificates

4) Revealed inconsistent behavior due to multiple TLS libraries per device

UNIVERSITY OF TWENTE. SIDN LABS

Q&A Session Lab Assignment

UNIVERSITY OF TWENTE.

SDN LABS

You can find all the **information** about the **lab assignment** and the **course** on the website

https://courses.sidnlabs.nl/ssi-2025/

Understanding the Assignment

- In this project, you will measure, and analyze the network behavior of IoT devices and capture this behavior in a device profile
- Your task is to explore how these devices work behind the scenes:
  - What **services on the Internet they use**, like cloud platforms, remote servers
  - **How** they **connect** to the services— including the protocols they use (e.g., HTTP, MQTT, DNS) and the direction of communication (inbound or outbound)
  - What **kinds of communication they initiate**, such as sending telemetry data, requesting updates, or how they react to external triggers
  - **How they handle security**, including whether communications are encrypted, if credentials are exposed, and how they authenticate with remote services

What to do

- Choose **two** IoT devices (e.g., smart lights, doorbells, plugs)
- Use tools like **Wireshark** or **TCPdump** to capture network traffic
- **Document** how your device:
  - Initiates connections (outbound requests)
  - Respond to events or external triggers
  - Resolves domain names and uses services
  - … Other
- Convert your findings into a **device behavior profile** using the **Manufacturer Usage Description (MUD)** format.

Deliverable and Expectations

- Deliver your measurement results for each IoT device:
  - **PCAP file**: A capture of the network traffic your group recorded during the experiments you conducted
  - **README file**: A README file lists the IP and MAC addresses of the IoT device and gateway you used and an explanation of where in the PCAP you interacted with the device in what way
  - **MUD Profile**: The device's MUD profile
- Sumit your slides **before** your **presentation**
- **Deadline: Wed June 25, 9AM CEST**

Group Presentation

- At most 25-minute group presentation
  - **Followed by 15-minute Q&A with two teachers**
  - Everyone must attend in person. **No exceptions**

- Why IoT security is a **challenge**
- Your methodology explaining the measurement setup, tools, and process
- Key findings and insights from your two IoT devices
- Analysis of your generated MUD specifications
- Proposed extensions or new uses of MUD for improving IoT security
- Use of visuals: text, graphs, and tables

Q&A Session Lab Assignment

UNIVERSITY OF TWENTE.

SDN LABS

How far are with your project?

A. **Selecting devices** – We are figuring out which IoT devices we will analyze
B. **Planning** – We have chosen our devices and are setting up tools
C. **Capturing** – We have started capturing traffic
D. **Analyzing** – We are analyzing traffic and identifying patterns

Schedule

Date	Time	Room A	Room B
27-Jun	08:45-9:30	Group 1	Group 2
27-Jun	9:30-10:15	Group 4	
27-Jun	10:15-11:00		Group 14
27-Jun	11:00-11:45	Group 16	Group 17
30-Jun	08:45-9:30	Group 3	Group 6
30-Jun	9:30-10:15	Group 8	Group 9
30-Jun	10:15-11:00	Group 11	Group 12
30-Jun	11:00-11:45	Group 13	Group 15
30-Jun	11:45-12:30	Group 7	Group 5

UNIVERSITY OF TWENTE.

SDN LABS