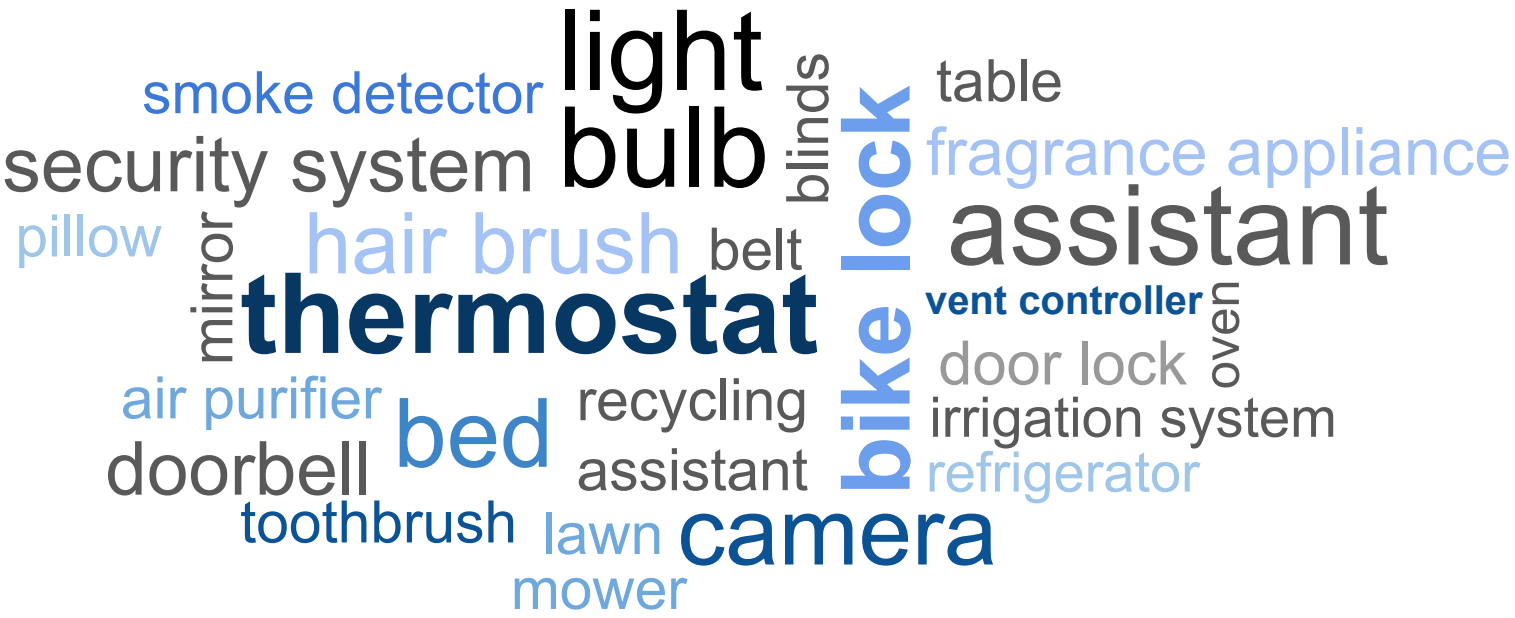
Trust but Verify: Auditing the Secure Internet of Things

Judson Wilson*, Riad S. Wahby, Henry Corrigan-Gibbs, Dan Boneh, Philip Levis, Keith Winstein

Stanford University

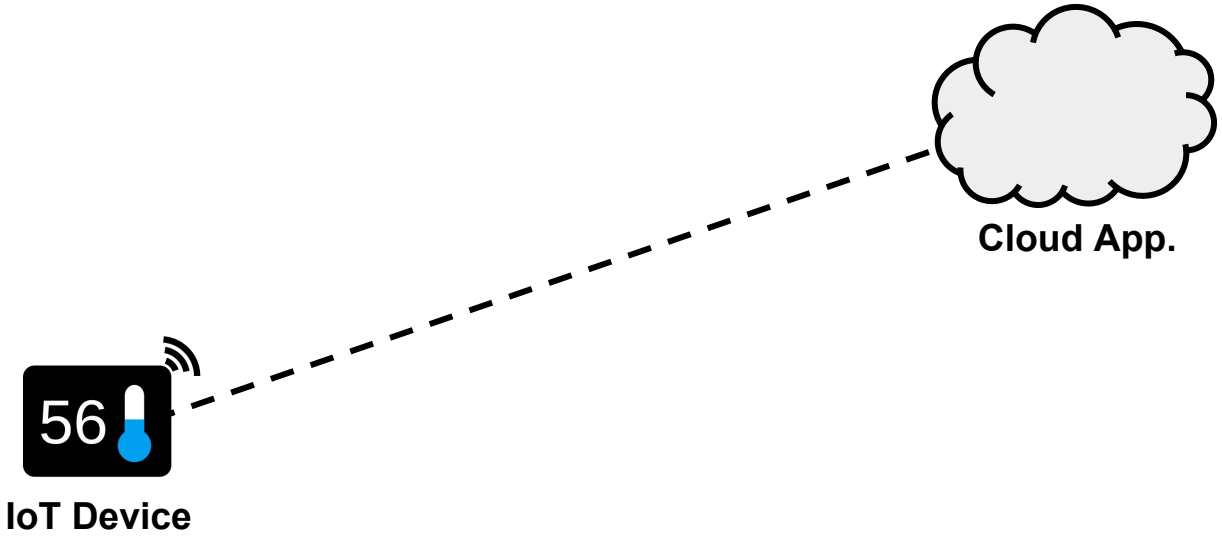
*Currently working at Google. The views expressed are personal opinions of the authors, not of Google.

Today, we have millions of smart devices in our homes:



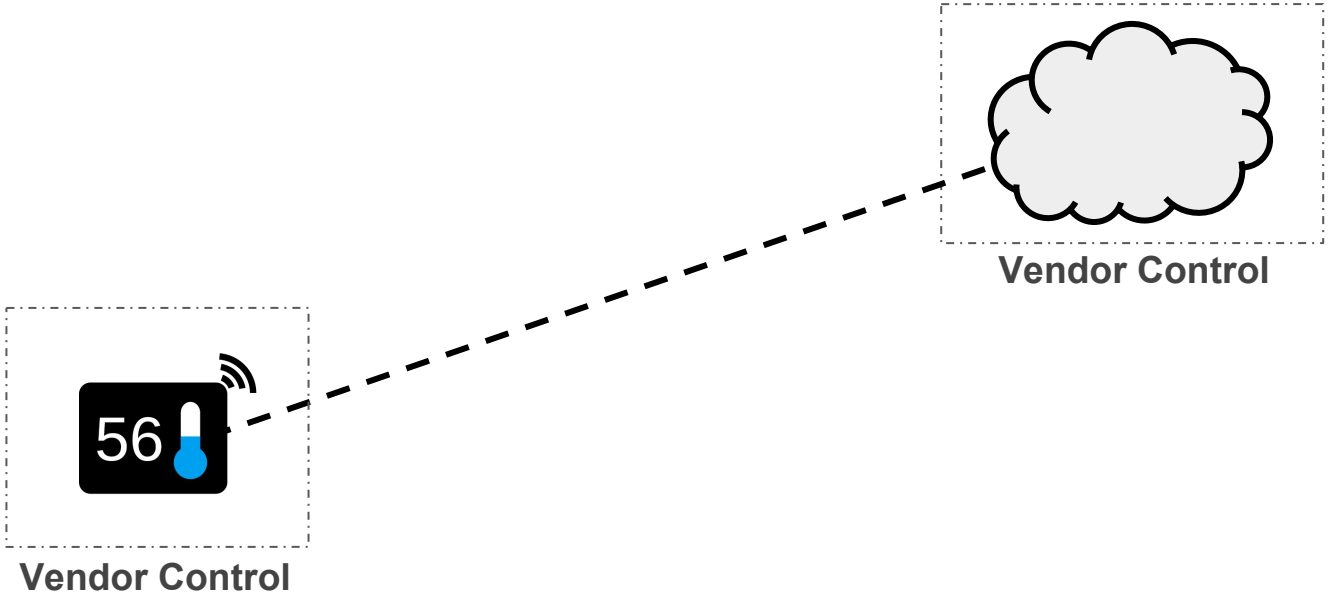
Do you know what your devices are saying about you?

IoT devices typically talk to a service in the cloud*:



*Grover, Feamster. “The Internet of Unpatched Things,” PrivacyCon ’16.

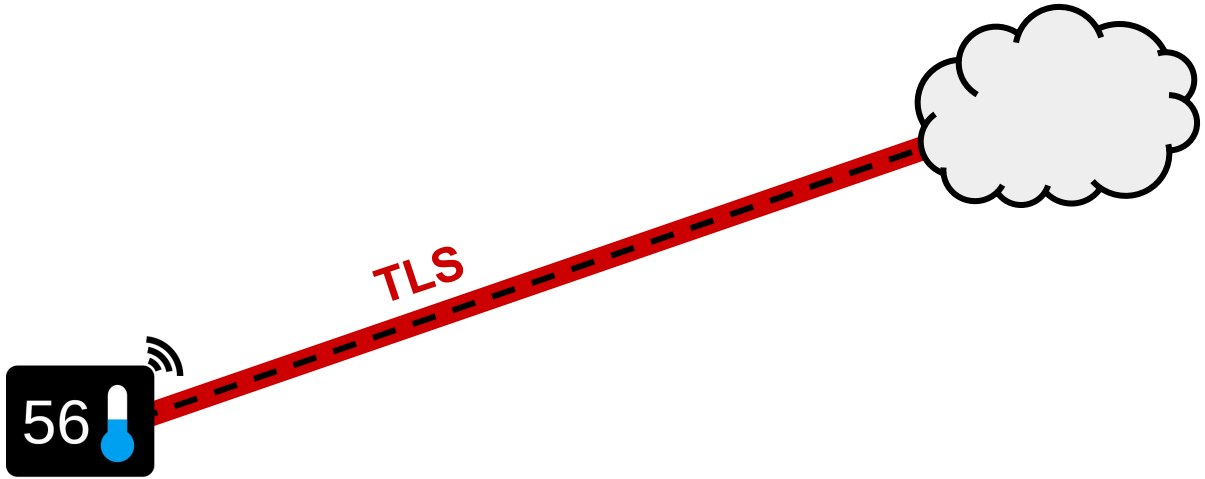
IoT devices typically talk to a service in the cloud*:



and a vendor controls the software in both the device and the cloud service.

*Grover, Feamster. “The Internet of Unpatched Things,” PrivacyCon ’16.

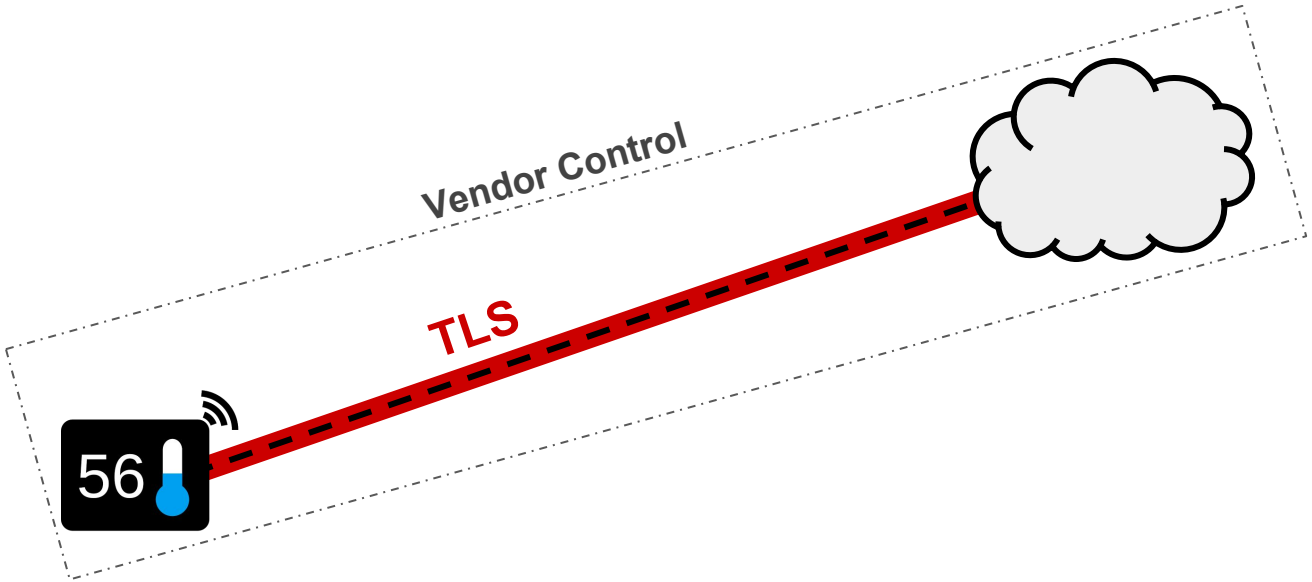
Recommended: Secure Communications*



- TLS encrypts traffic to prevent attackers from observing traffic plaintext.

*OWASP. “Manufacturer IoT Security Guidance.” https://www.owasp.org/index.php/IoT_Security_Guidance

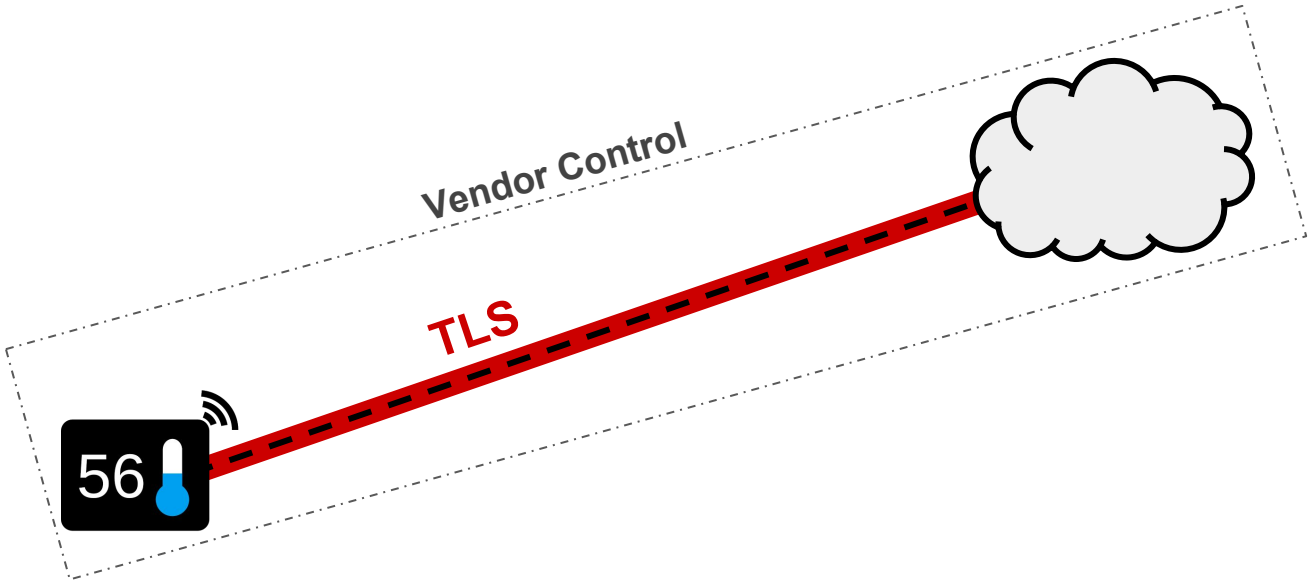
Recommended: Secure Communications*



- TLS encrypts traffic to prevent attackers from observing traffic plaintext.

*OWASP. “Manufacturer IoT Security Guidance.” https://www.owasp.org/index.php/IoT_Security_Guidance

Recommended: Secure Communications*



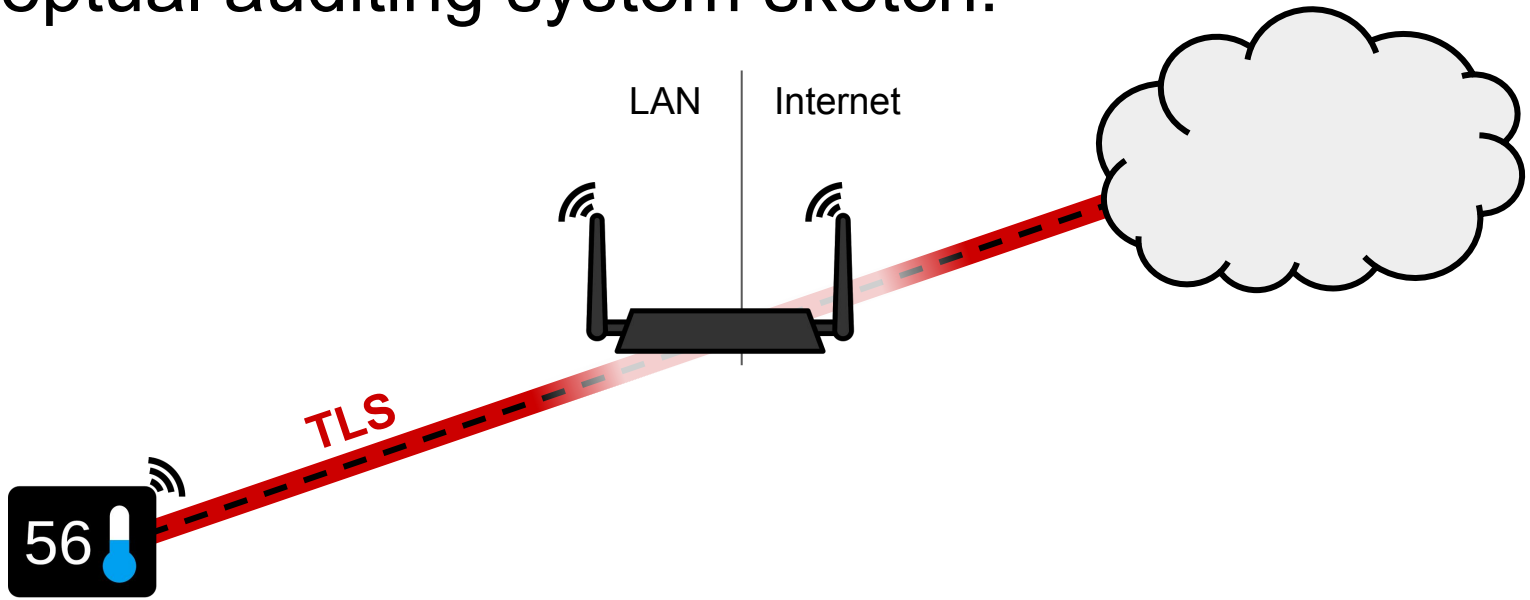
- TLS encrypts traffic to prevent attackers from observing traffic plaintext.
- **TLS also prevents device owners from seeing what private data leaves their home.**

*OWASP. “Manufacturer IoT Security Guidance.” https://www.owasp.org/index.php/IoT_Security_Guidance

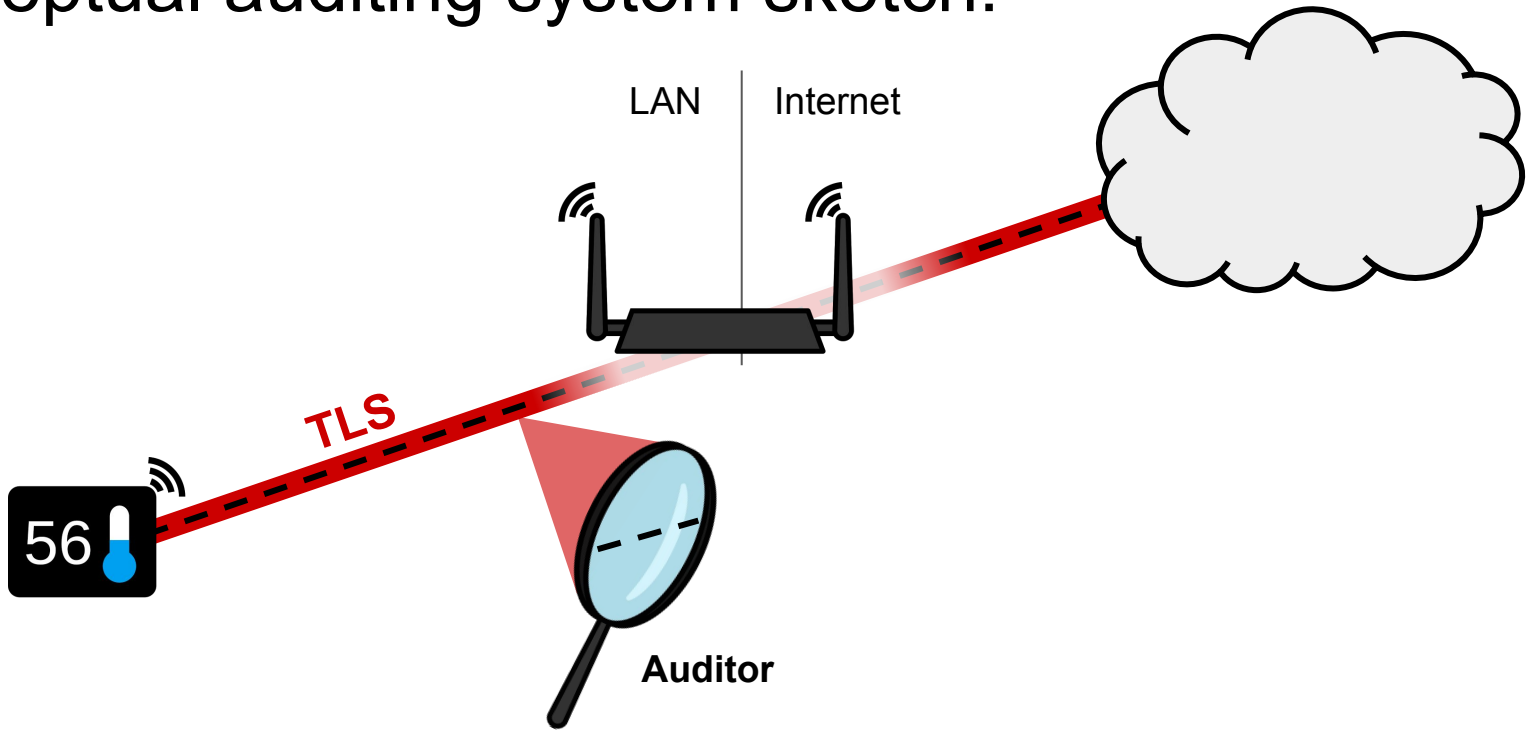
How might vendors with good intentions allow device owners to audit their own devices' communications, while maintaining security?

The Goal

Conceptual auditing system sketch:



Conceptual auditing system sketch:



- “Auditors” on LAN collect ciphertext packets, somehow decrypt them.

Auditing system requirements:

Security:

1. **Past auditability:** ensure auditors can decrypt past traffic, (or report FAIL).

Auditing system requirements:

Security:

1. **Past auditability:** ensure auditors can decrypt past traffic, (or report FAIL).
2. **Audit robustness:** ensure auditors can verify correctness of their audits.

Auditing system requirements:

Security:

1. **Past auditability:** ensure auditors can decrypt past traffic, (or report FAIL).
2. **Audit robustness:** ensure auditors can verify correctness of their audits.
3. **Present moment integrity:** ensure device/server end-to-end integrity,
   - Prevent tampering with data, billing information, etc.
   - Prevent cloud API and device hacking, repurposing subsidized devices, etc.

Auditing system requirements:

Security:

1. **Past auditability:** ensure auditors can decrypt past traffic, (or report FAIL).
2. **Audit robustness:** ensure auditors can verify correctness of their audits.
3. **Present moment integrity:** ensure device/server end-to-end integrity,
   - Prevent tampering with data, billing information, etc.
   - Prevent cloud API and device hacking, repurposing subsidized devices, etc.

Deployment:

4. **TLS compatibility:** maintain standard TLS wire format and server implementation.
   - Ensure compatibility with TLS termination proxies, accelerators, load balancers, cloud services, etc.

Overview

- ~~Introduction~~
- ~~Problem Description and Requirements~~
- Technical Summary
- Threat Model
- Straw Man Solutions
- Proposed Solution: TLS-RaR
- Evaluation
- Conclusion

Technical Summary

We introduce and evaluate TLS-RaR, a protocol which ensures:

- Robust, delayed auditing of secure, TLS communication

Technical Summary

We introduce and evaluate TLS-RaR, a protocol which ensures:

- Robust, delayed auditing of secure, TLS communication

TLS-RaR targets adoptability by:

- Preserving wire-format and server side TLS implementation
- Ensuring tamper-proof communication

Technical Summary

We introduce and evaluate TLS-RaR, a protocol which ensures:

- Robust, delayed auditing of secure, TLS communication

TLS-RaR targets adoptability by:

- Preserving wire-format and server side TLS implementation
- Ensuring tamper-proof communication

TLS-RaR works by:

- Using built in mechanisms to rotate keys
- New concepts: “Authenticated Key-Retirement ACK” and “Sealed-History Key Release” enable decryption of traffic before the most recently ACKed rotation.

Threat Model

Threat Model Summary

- Standard TLS threats
  - (except trusted auditors can see plaintext)
- Auditors or endpoints may try to sidestep the security requirements, i.e.
  - either may attempt to fool an auditor into reporting incorrect output, or
  - auditors may attempt to tamper with traffic.

Threat Model Summary

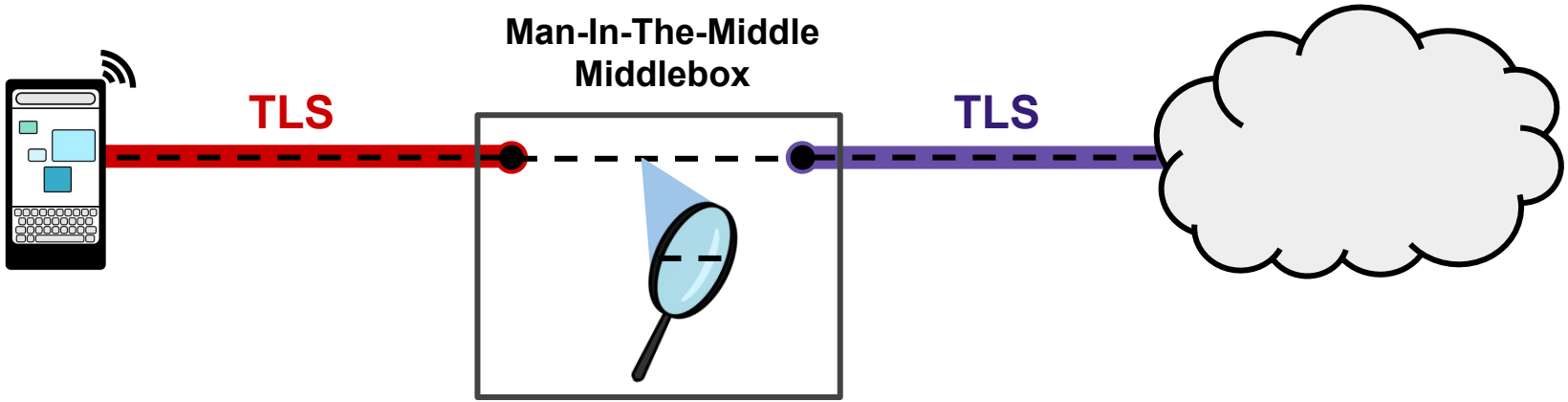
- Standard TLS threats
  - (except trusted auditors can see plaintext)
- Auditors or endpoints may try to sidestep the security requirements, i.e.
  - either may attempt to fool an auditor into reporting incorrect output, or
  - auditors may attempt to tamper with traffic.

Out of scope:

- Denial of Service
- Covert channel attacks: hiding information in other layers to evade audit
  - e.g. steganography,double encryption, packet timing.
  - (Problem exists even when auditing plaintext communication.)

Straw Man Solutions

Straw Man (in-the-Middle) Solution:

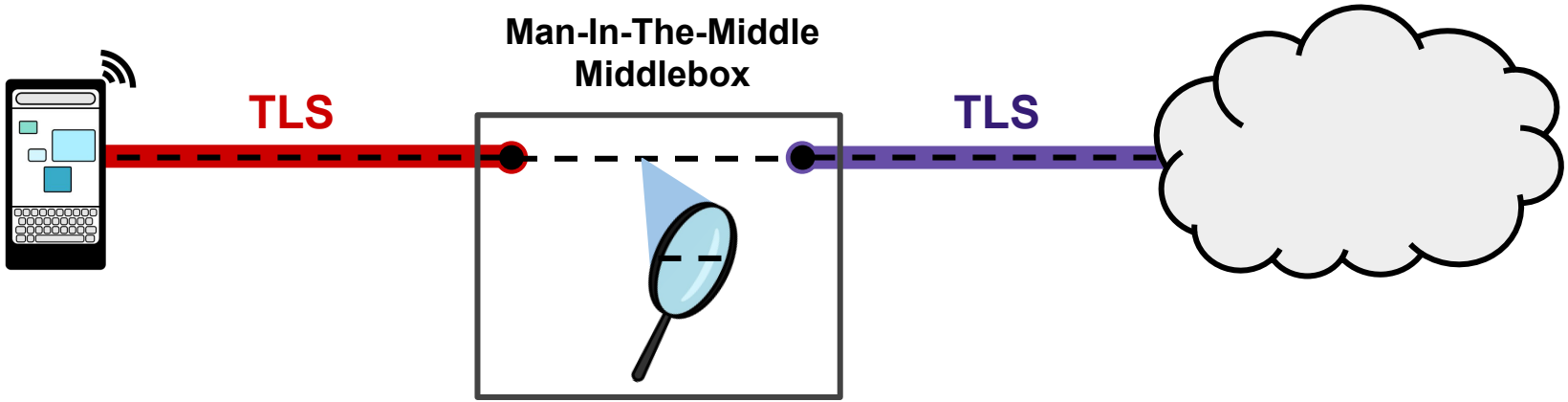


Traditional devices support trusting a TLS man-in-the-middle* by installing a root certificate to bypass authentication. <u>IoT devices typically DO NOT!</u>

Advantages: Simple, effective.

*Huang et al. Analyzing forged SSL certicates in the wild. SP '14.

Straw Man (in-the-Middle) Solution:



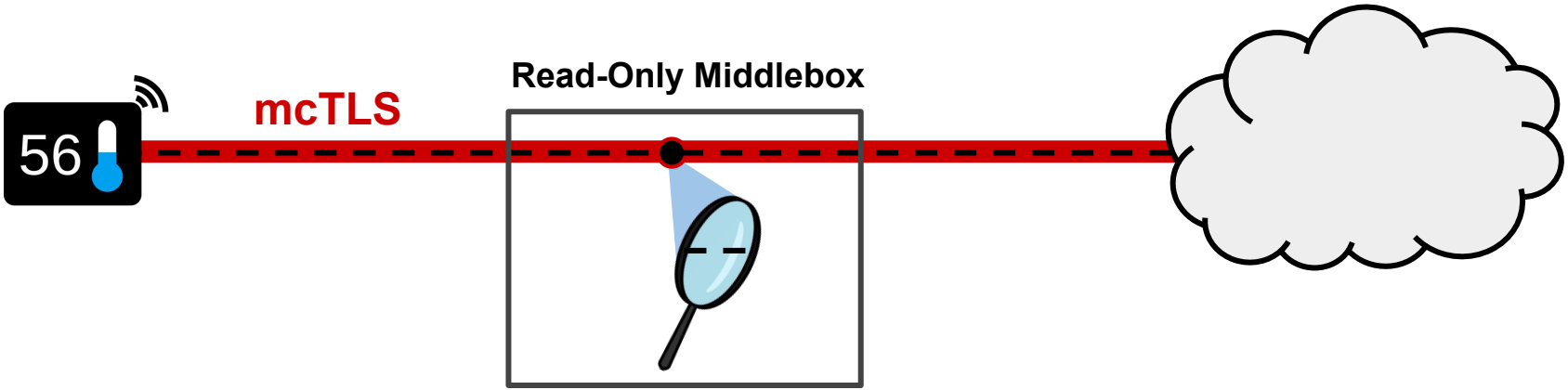
Traditional devices support trusting a TLS man-in-the-middle* by installing a root certificate to bypass authentication. <u>IoT devices typically DO NOT!</u>

Advantages: Simple, effective.

Problem: No end-to-end integrity! Broken authentication!

*Huang et al. Analyzing forged SSL certicates in the wild. SP '14.

Straw Man Solution: different protocols

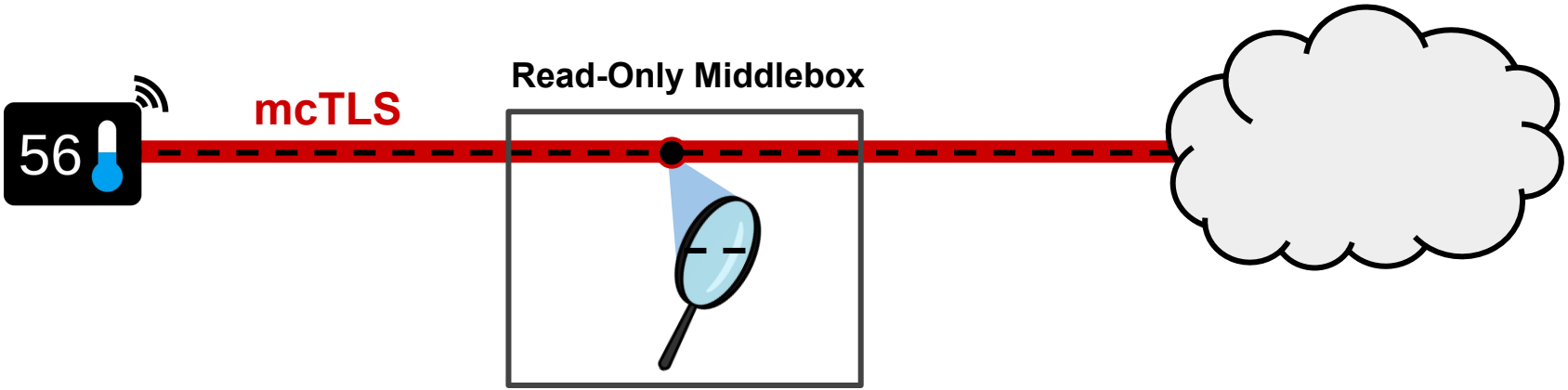


Other protocols, such as mcTLS [1], satisfy different requirements.

Advantages: Already exist. Can audit data before relaying it. (Others…)

*Naylor et al. Multi-context TLS (mcTLS): Enabling secure in-network functionality in TLS. SIGCOMM '15.

Straw Man Solution: different protocols



Other protocols, such as mcTLS [1], satisfy different requirements.

Advantages: Already exist. Can audit data before relaying it. (Others…)

Problems: Different wire format. Not compatible with existing data centers.

*Naylor et al. Multi-context TLS (mcTLS): Enabling secure in-network functionality in TLS. SIGCOMM '15.

Proposed Solution:
TLS-Rotate and Release (TLS-RaR)

TLS-Rotate and Release (TLS-RaR)

TLDR:

When it is time to audit past traffic,

1) The IoT device rotates traffic keys.

Once the device verifies rotation is complete,*

2) the IoT device securely* releases previous traffic keys to auditors.

* Performing these steps securely is NOT trivial.

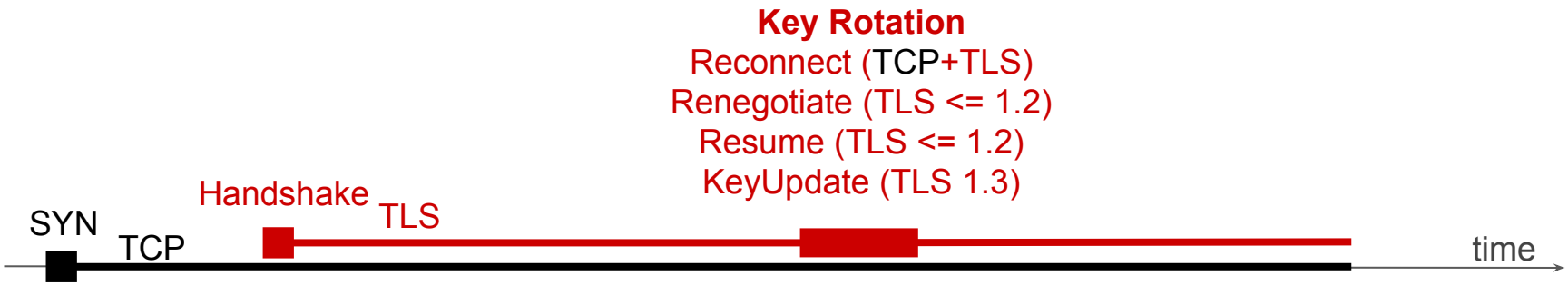
Let’s look at this...

Life of a typical TLS Connection

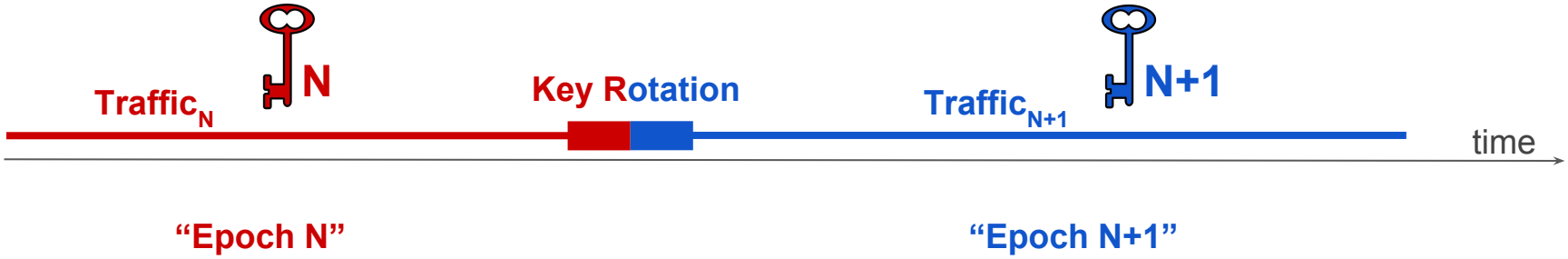
Life of a typical TLS Connection

Life of a typical TLS Connection

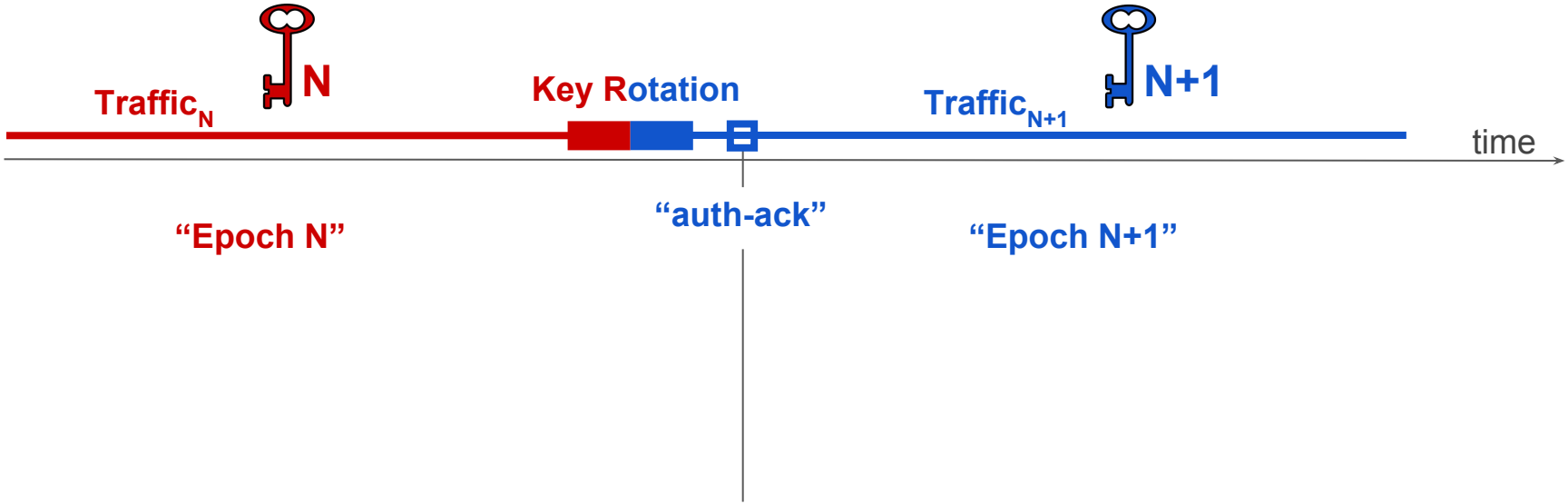
TLS with Rotate and Release



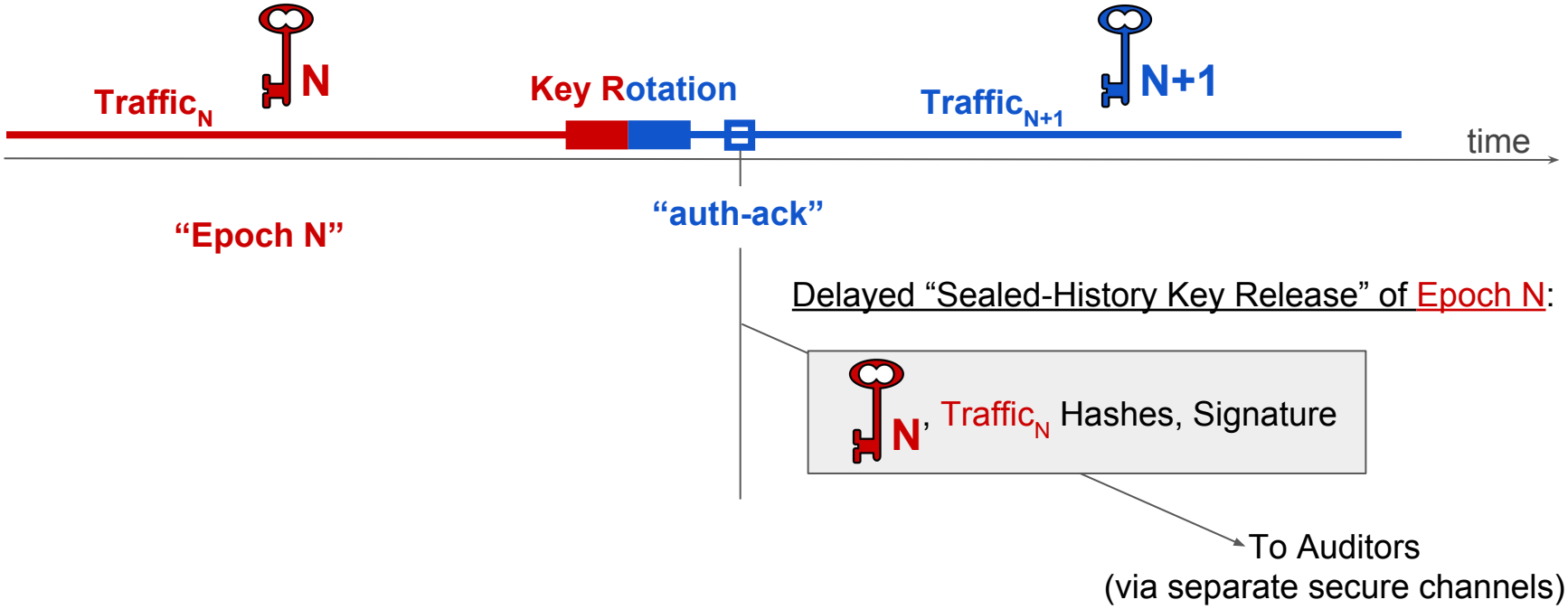
TLS with Rotate and Release



TLS with Rotate and Release



TLS with Rotate and Release



Evaluation

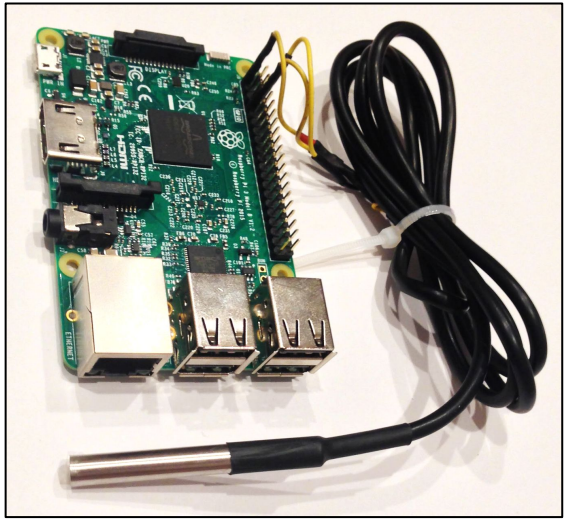
Evaluation Summary *- See Paper for Details*

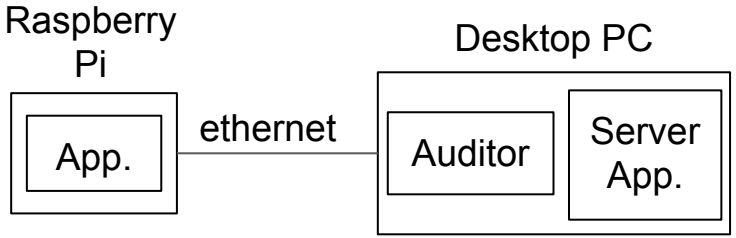
To evaluate TLS-RaR, we:

1. Implemented TLS-RaR In our own ARM-based IoT device, as well as an auditor and server software stack.
2. Evaluated the relative performance costs of TLS-RaR on the IoT device.
3. Probed the Alexa Top 1,000,00 Sites* to assess server compatibility.
4. Observed the traffic patterns of several off-the-shelf devices to reason about feasibility.

*Top 1,000,000 sites (updated daily). Alexa Internet Inc. http://s3:amazonaws:com/alexa-static/top-1m:csv:zip. Accessed: 2016-01-18.

Evaluation Platform





Temperature Sensing Evaluation-Device

- Raspberry Pi 3 B (ARM, 4 cores)
- OpenSSL
  - Added: key exporting callback
- Custom C Application

Auditor (Desktop PC)

- tshark (modified to import released keys)
- Python scripts for verification

Web Server (Desktop PC)

- Python Twisted Reactor

Evaluation-Device Performance Measurements

Ongoing Encryption Overhead:

	(CPU seconds / GiB)
Baseline TLS (encrypt/send)	9.8
TLS-RaR (encrypt/hash/send)	11.9
Mean CPU overhead per byte:	22%

Takeaway:

Hashing cost (for sealed-history key release) is a significant but modest portion of ongoing TLS CPU cost.

Evaluation-Device Performance Measurements

Ongoing Encryption Overhead:

	(CPU seconds / GiB)
Baseline TLS (encrypt/send)	9.8
TLS-RaR (encrypt/hash/send)	11.9
Mean CPU overhead per byte:	22%

Per-Rotation Overhead:

	(CPU milliseconds)
Rotate by Renegotiation	46
Rotate by Resume + heartbeat	1.2
Release	0.7

Takeaway:

Rotation + release using Resume consumes as much CPU as encrypting, hashing and sending approximately 160kB of plaintext. KeyUpdate should be similar. Renegotiate is approximately 25 times more expensive.

Server Compatibility Survey

We surveyed servers in the Alexa Top 1,000,000 sites that support long-lived HTTPS connections to see which features they support:

	Fraction of Servers
Rotation by Reconnect	54.2%
Rotation by Renegotiate	12.2%
Rotation by Resume	0.5%
TLS 1.3 KeyUpdate	*0% at time of survey*

We *believe* servers using TLS 1.3 in the future will widely support KeyUpdate. It is a standard part of the RFC draft 19 standard that is simple, light weight, and enhances security.

*Top 1,000,000 sites (updated daily). Alexa Internet Inc. http://s3:amazonaws:com/alexa-static/top-1m:csv:zip. Accessed: 2016-01-18.

Conclusions

- Auditing IoT communication is important, but security prevents it.
- TLS-RaR allows read-only auditing of secured communication, and has these useful properties:
  - Auditors see the exact plaintext encrypted by TLS (or report failure).
  - The format of TLS on the wire is not changed.
  - No TLS-layer changes are required for *some* servers.
    (Likely improved compatibility once TLS 1.3 rolls out.)
  - Only minimal changes to OpenSSL are required on the device.

Contact: Judson Wilson <judsonw@cs.stanford.edu>

Backup Slides

Potential vendor & consumer concerns:

- Maintain privacy.
  - Don’t introduce mechanisms for unauthorized snooping.

- Prevent communications tampering.
  - No device control takeover (e.g. unlocking your front door).
  - No cloud API hacking.
  - No falsifying data (e.g. billing data, software updates)
  - No unintended use of subsidized devices.
  - …

- Don’t change lower layers of cloud service.
  - Maintain compatibility with TLS accelerator boxes, load balancers and reverse proxies.
  - Different layers on different physical devices - maintain separation of concerns.
  - Much of this may be provided by cloud provider, out of vendor’s control.

Vital: Authenticated Acknowledgement (Auth-Ack)

Before releasing keys to auditors, IoT devices MUST wait for an <u>authenticated</u> message from the server acknowledging it has rotated keys, otherwise:

1) Malicious auditor can spoof message indicating rotation is complete.
2) Malicious auditor receives keys which it can then use to spoof TLS records to the server, breaking integrity requirements.

TCP FIN, RST, ACK, etc. are NOT authenticated. Must use TLS layer (and above).

Auth-Ack Mechanisms

Rotation Method	Auth-Ack methods
Reconnect	CLOSE_NOTIFY*
Renegotiation	Built-in
Resume	Heartbeat or app. request/response (HTTP OPTIONS)
KeyUpdate	Heartbeat or app. request/response (HTTP OPTIONS)

* Implementation dependent.

Off-the-Shelf Devices Communication Survey

Captured and inspected traffic of several off-the-shelf devices. Observed:

- (Unsecured) HTTP, HTTPS, and other security protocols.
- Connections to cloud services.
- Various traffic patterns.
- No obvious signs that TLS-RaR would be unreasonable.

See the paper for details.