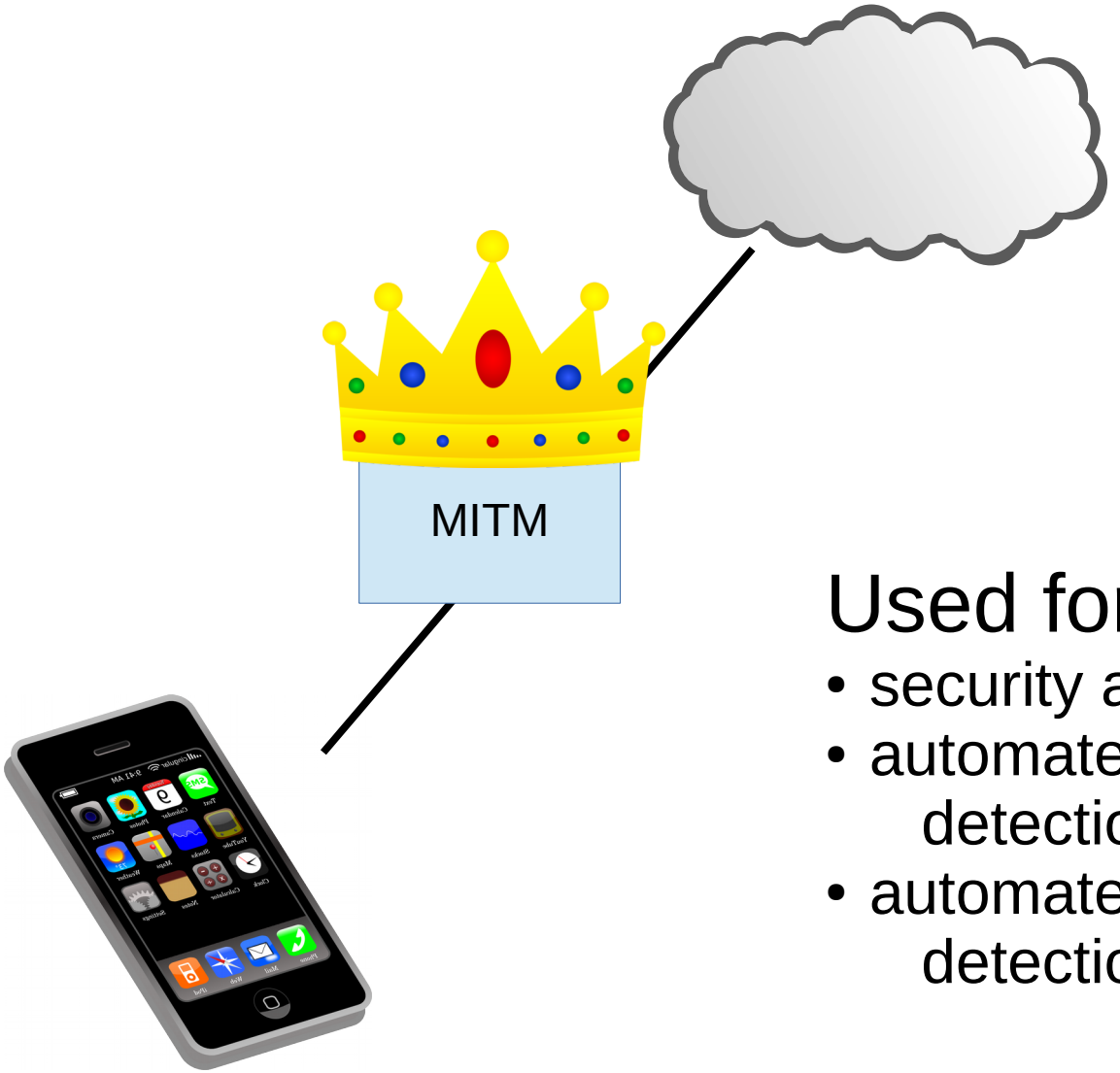
Auditing IoT Communications with TLS-RaR

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

Stanford University

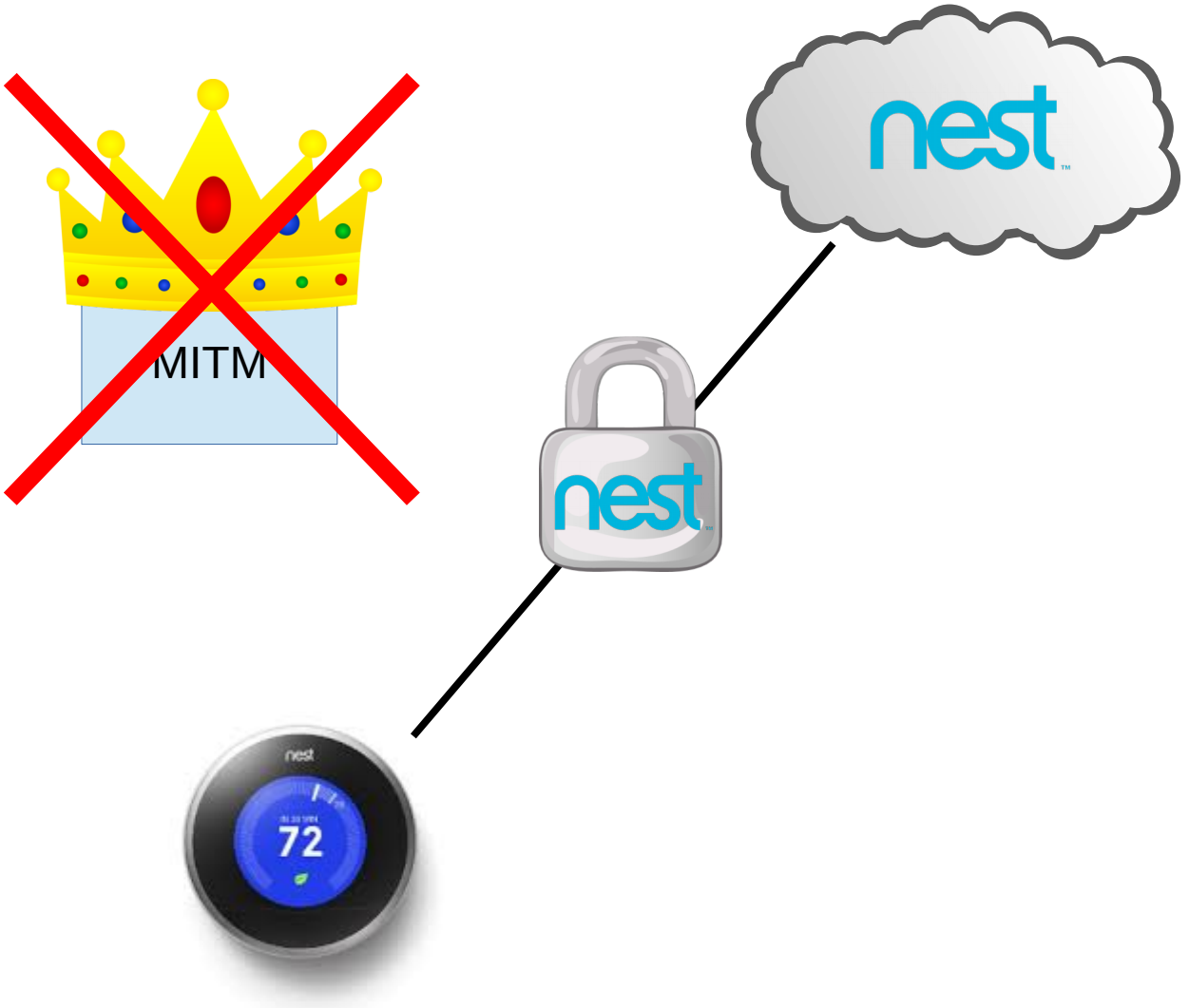
Auditing Standard Devices



Used for:

- security audit
- automated exfiltration detection
- automated intrusion detection

IoT is Different



Troubling Facts

1) We have no way to audit IoT communications, so we must trust companies to do what they claim.

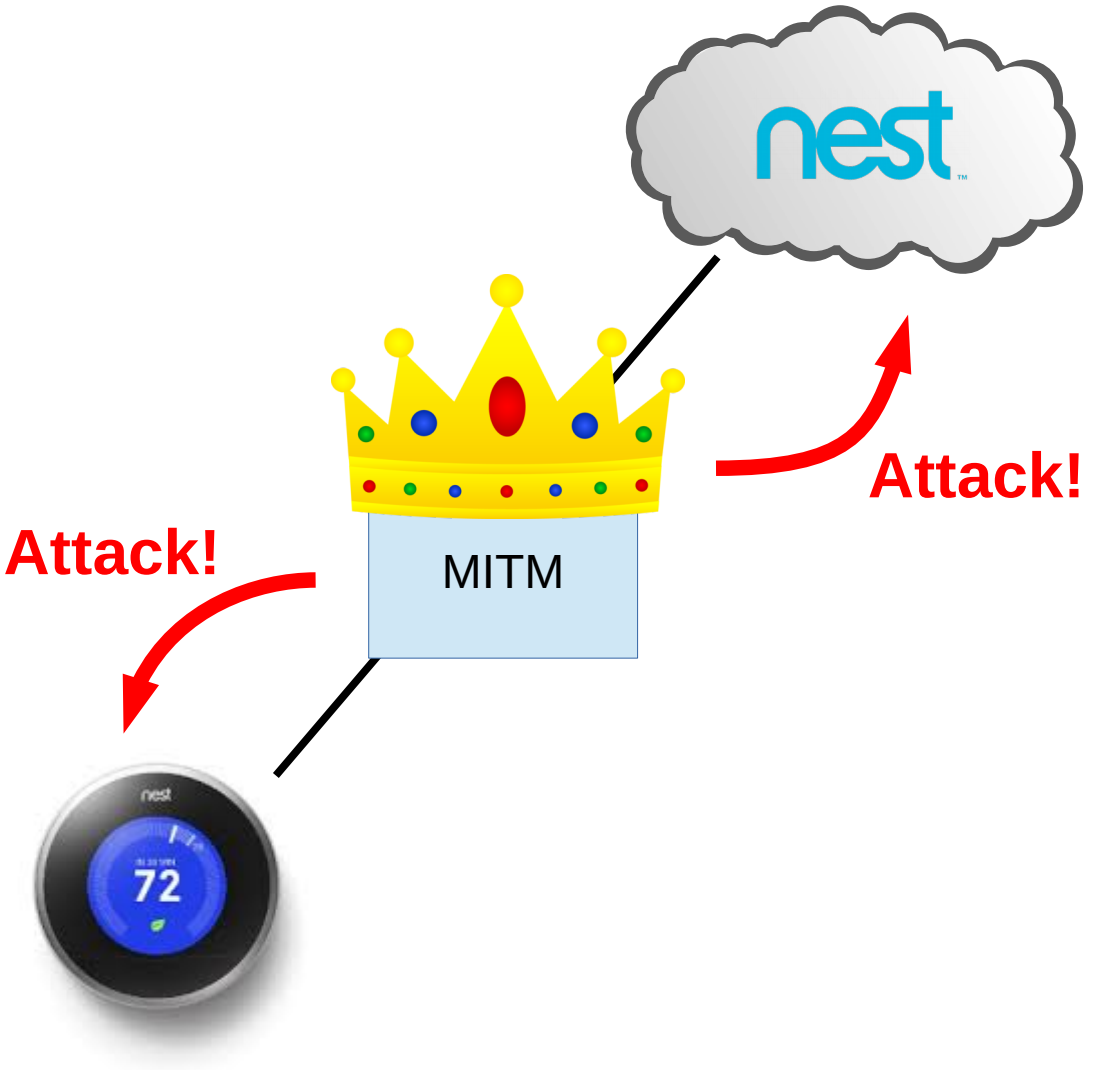
Troubling Facts

1) We have no way to audit IoT communications, so we must trust companies to do what they claim.

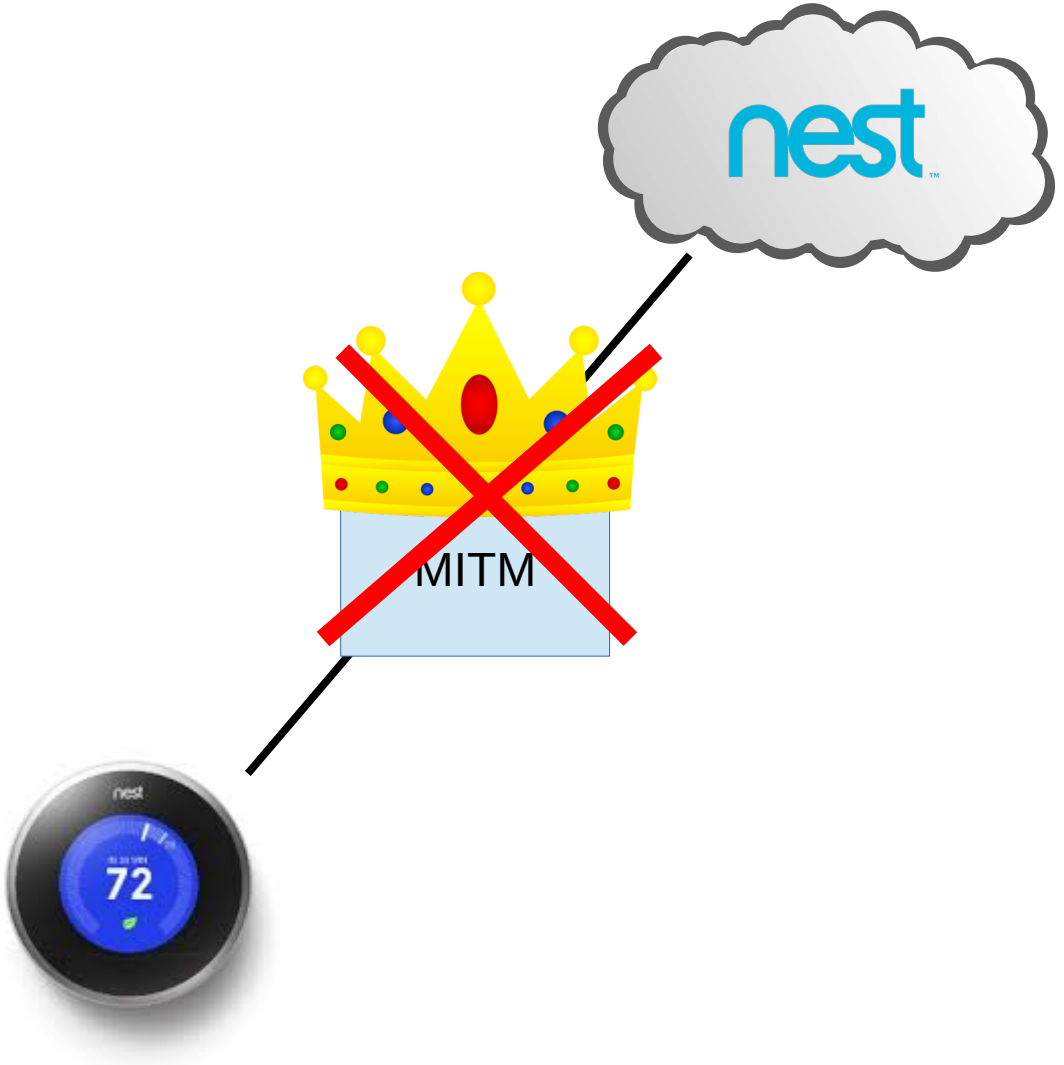
2) Respected Companies Have Misrepresented Their Actions

- Google
  - *“Google's iPhone Tracking, Web Giant, Others Bypassed Apple Browser Settings for Guarding Privacy.” WSJ, Feb. 17, 2012.*
- Volkswagen
  - *“Volkswagen Admits to Cheating on U.S. Emissions Tests,” Bloomberg, Sept. 18, 2015*
- ...

MITM does have problems.



MITM does have problems.



Overview

- Setting
- **Technical Problem**
- Our Scheme: TLS RaR
  - Main Idea
  - Corner Cases
    - Secure Key Release
    - Clean Shutdown
- Evaluation
- Related Work
- Conclusions

Different Parties, Different Concerns

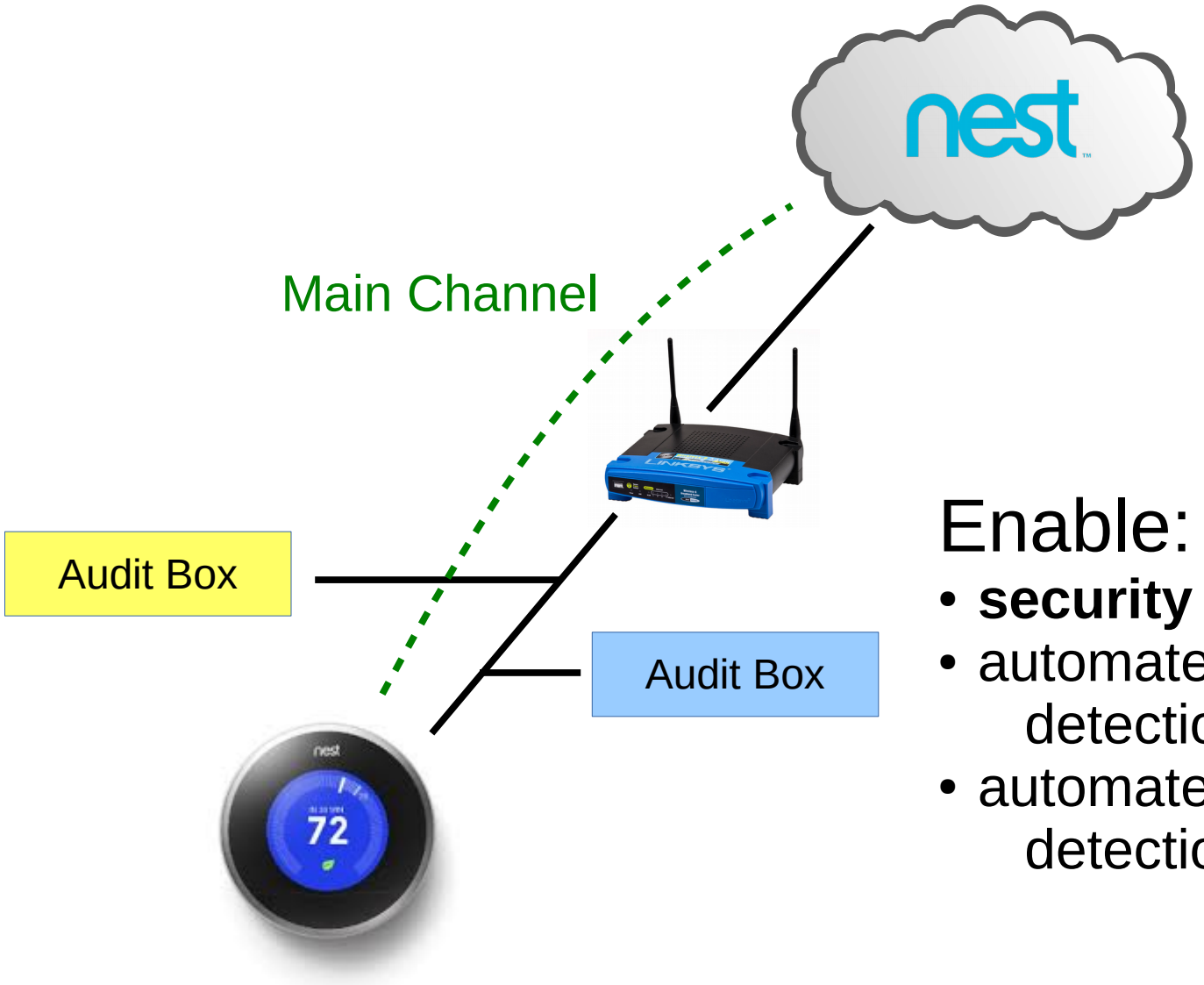
<u>Potential</u> concerns of IoT device company:

- Prevent tampering, back doors
- Prevent usage of device on other services
- Solution that is easy to incorporate.
- Protecting customer data

Customer's concerns:

- Desire an accurate audit, as good as MITM
- Preserve privacy

Compromise: Replace MITM with passive, read only auditors.



Enable:

- **security audit**
- automated exfiltration detection
- automated intrusion detection

The Technical Problem:

Create a method for passive, read only auditing of TLS-protected communication, to replace the man in the middle method.

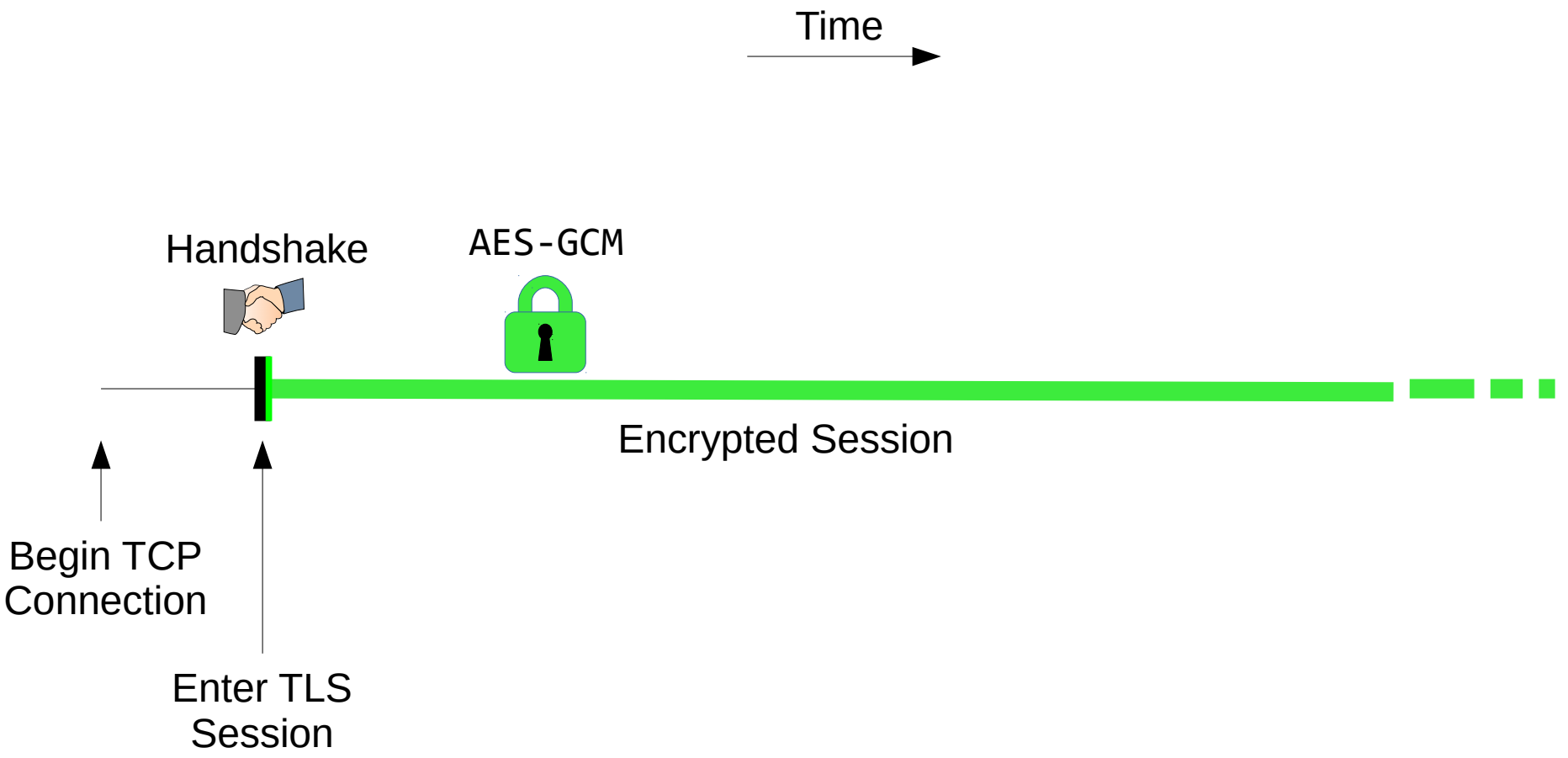
In other words: Remove the TLS barrier from a communications audit.

Overview

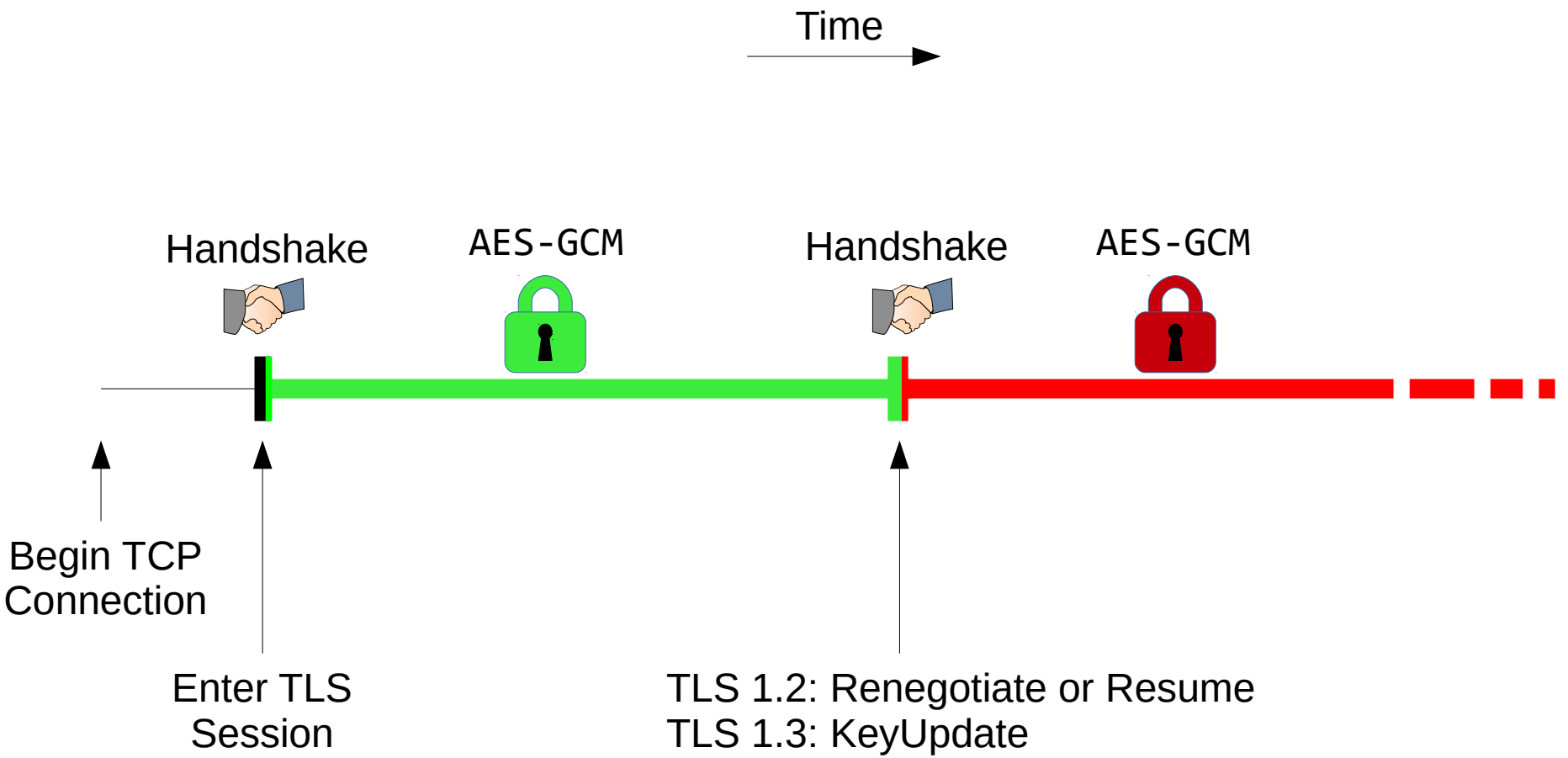
- Setting
- Technical Problem
- **Our Scheme: TLS RaR**
  - **Main Idea**
  - Corner Cases
    - Secure Key Release
    - Clean Shutdown
- Evaluation
- Conclusions

TLS-RaR: Rotate and Release

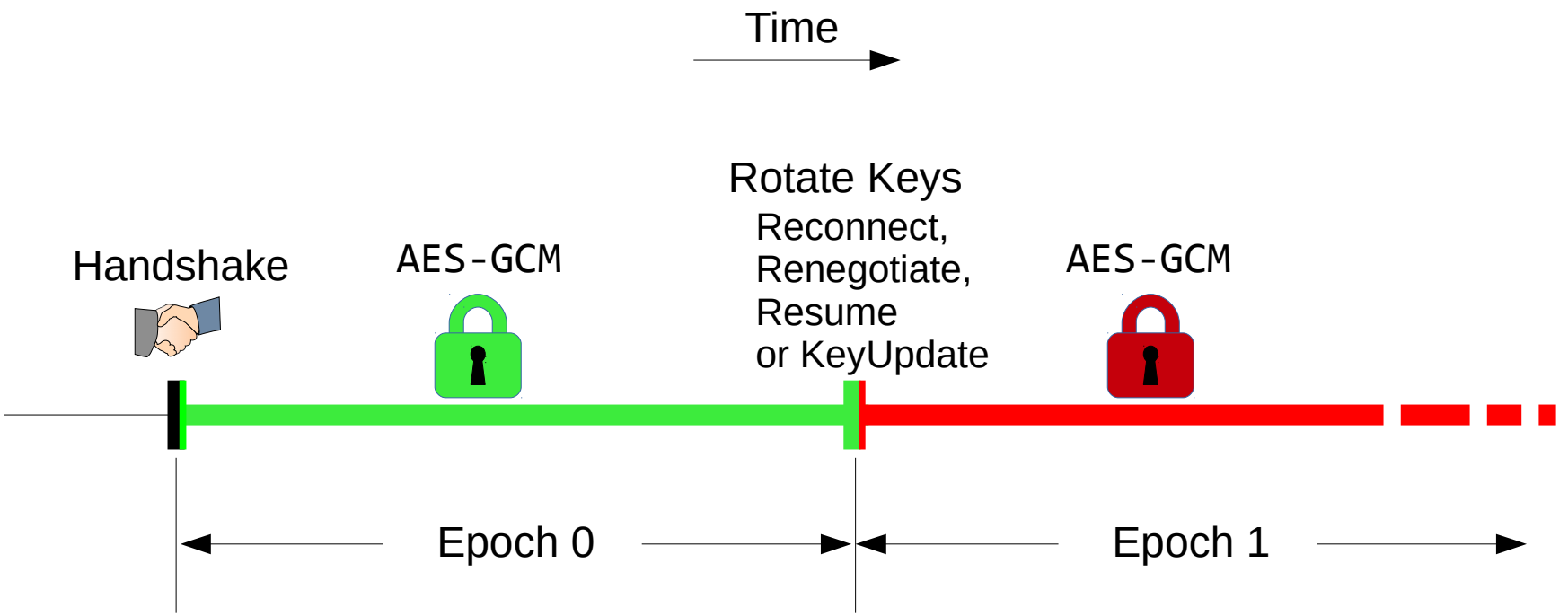
Device to Cloud TLS



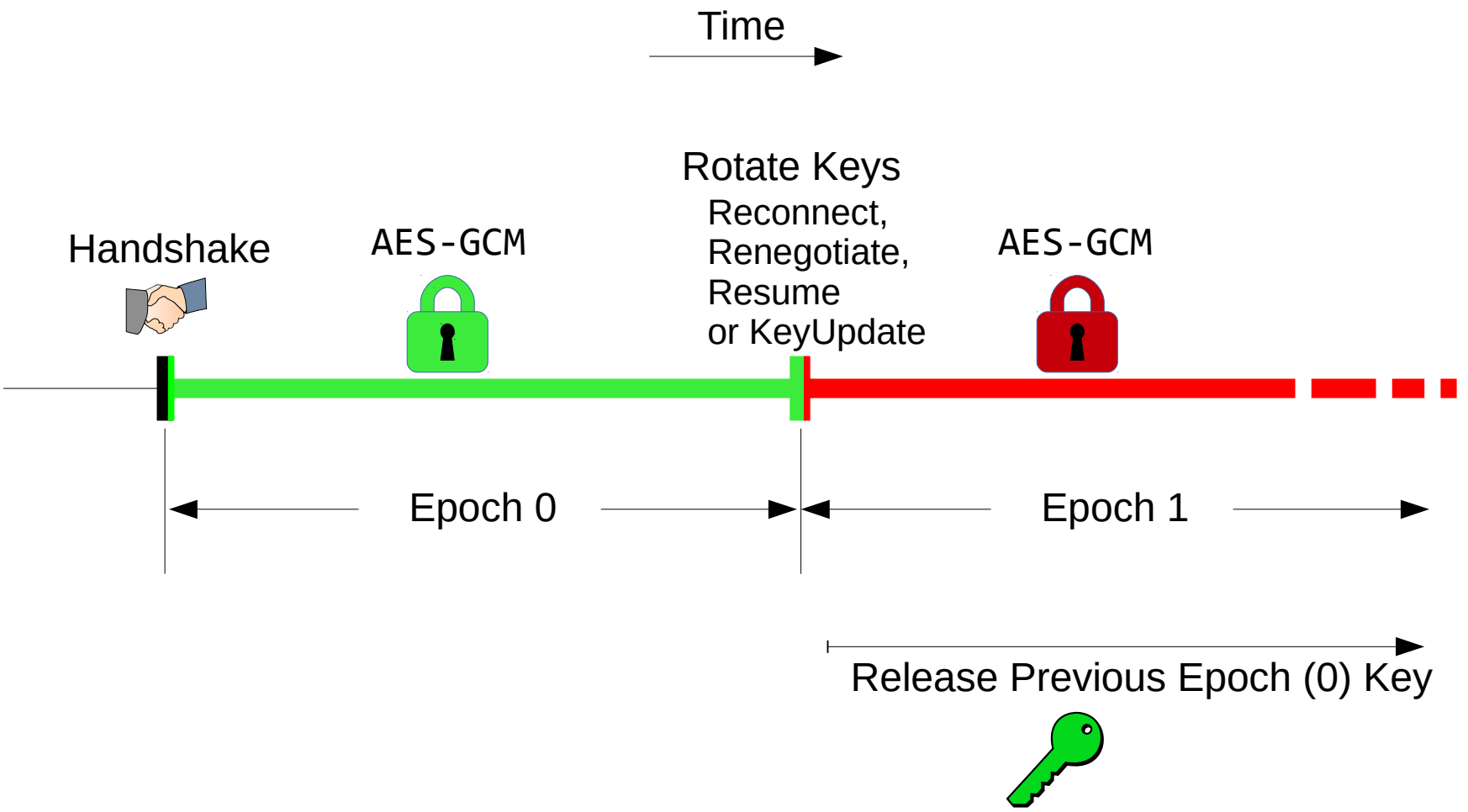
Device to Cloud TLS



Device to Cloud TLS
With a Twist



Device to Cloud TLS
With a Twist



Nice Properties

- Audit box's decryption yields the same stream of data as endpoints' `SSL_read()` calls, but delayed
  - ➔ Audit matches what was received

Nice Properties

- Audit box's decryption yields the same stream of data as endpoints' `SSL_read()` calls, but delayed
  - ➔ Audit matches what was received
- Format of TLS on the wire is not changed
  - ➔ Easy to reason about security of the protocol
  - ➔ Easy to adopt

Nice Properties

- Audit box's decryption yields the same stream of data as endpoints' `SSL_read()` calls, but delayed
  - ➔ Audit matches what was received
- Format of TLS on the wire is not changed
  - ➔ Easy to reason about security of the protocol
  - ➔ Easy to adopt
- For some existing servers no change is necessary
  - ➔ Really easy to adopt

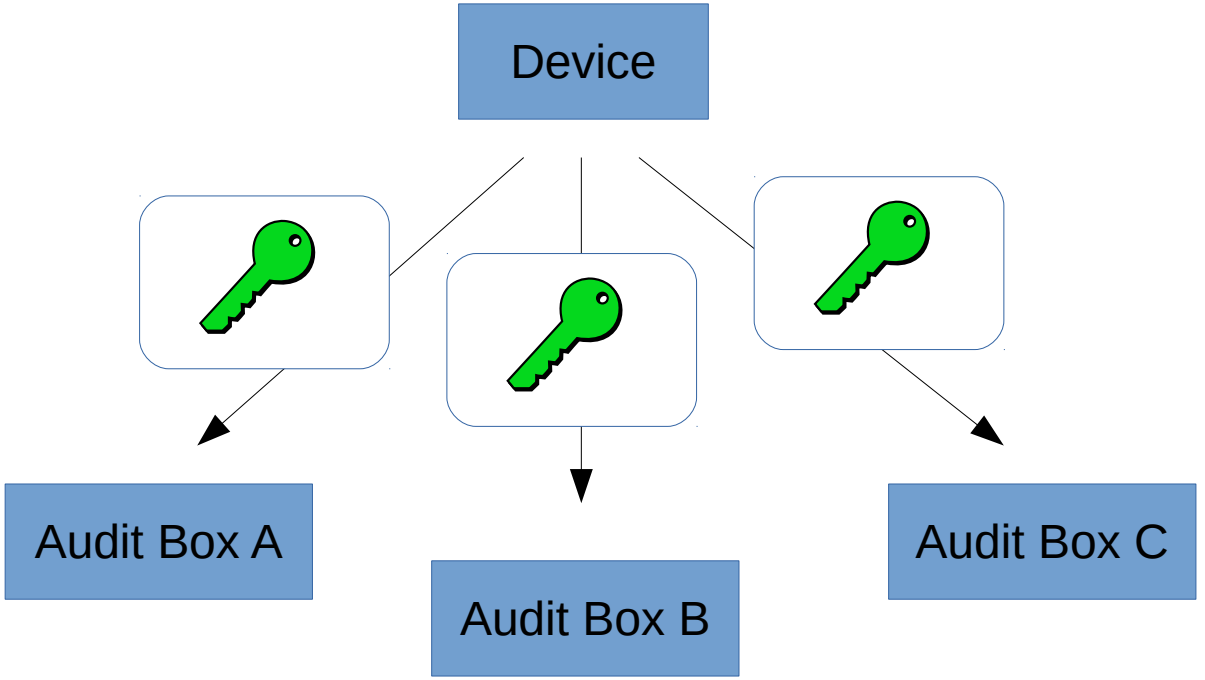
Nice Properties

- Audit box's decryption yields the same stream of data as endpoints' `SSL_read()` calls, but delayed
  - ➔ Audit matches what was received
- Format of TLS on the wire is not changed
  - ➔ Easy to reason about security of the protocol
  - ➔ Easy to adopt
- For some existing servers no change is necessary
  - ➔ Really easy to adopt
- Minimal change to OpenSSL on the device
  - ➔ Easy to reason about security of the implementation
  - ➔ Easy to adopt

Overview

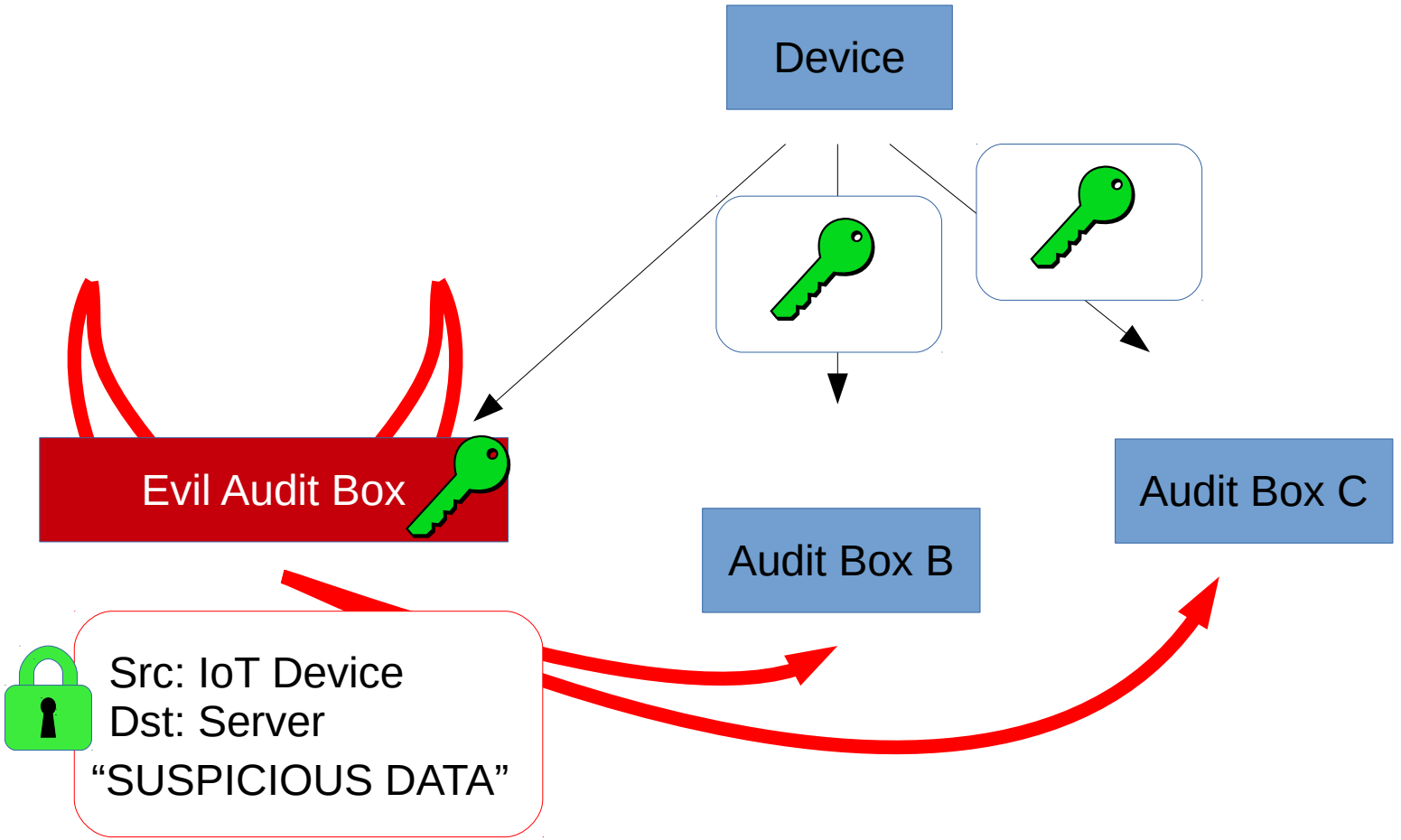
- Setting
- Technical Problem
- Our Scheme: TLS RaR
  - Main Idea
  - Corner Cases
    - **Secure Key Release**
    - Clean Shutdown
- Evaluation
- Conclusions

Key Release Procedure: Straw Man

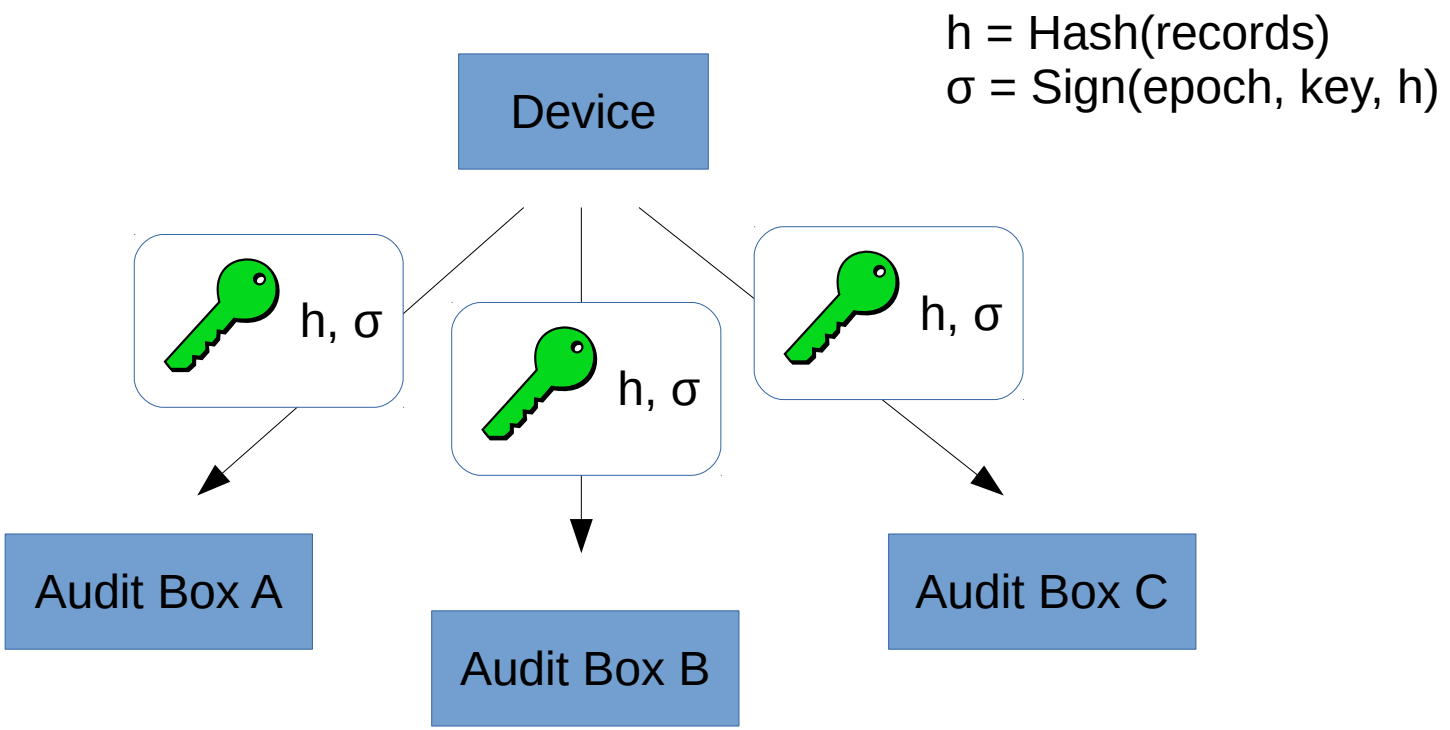


Device simply distributes key to Audit Boxes.

Key Release Procedure: Straw Man



Sealed-History Key Release



Cryptographic Hashes and Signatures ensure integrity to the auditors.

Overview

- Setting
- Technical Problem
- Our Scheme: TLS RaR
  - Main Idea
  - Corner Cases
    - Secure Key Release
    - **Clean Shutdown**
- Evaluation
- Conclusions

Connection Shutdown: Straw Men

1) Device naively releases key after disconnecting

Connection Shutdown: Straw Men

1) Device naively releases key after disconnecting

Attack: Auditors use key to append data to IoT device-to-server stream.

Connection Shutdown: Straw Men

1) Device naively releases key after disconnecting

Attack: Auditors use key to append data to IoT device-to-server stream.

2) Device doesn't release key after disconnecting

Connection Shutdown: Straw Men

1) Device naively releases key after disconnecting

Attack: Auditors use key to append data to IoT device-to-server stream.

2) Device doesn't release key after disconnecting

Problem: Auditor can't decrypt the last epoch.

Clean Connection Shutdown

Clean shutdown: IoT application ensures the last key encrypting data is not useful (e.g. authenticated acknowledgment), then securely releases the key.

TLS's close_notify is probably good enough.

Clean Connection Shutdown

Clean shutdown: IoT application ensures the last key encrypting data is not useful (e.g. authenticated acknowledgment), then securely releases the key.

TLS's close_notify is probably good enough.

Unclean shutdown results in **unauditable final epoch**.

Clean Connection Shutdown

Clean shutdown: IoT application ensures the last key encrypting data is not useful (e.g. authenticated acknowledgment), then securely releases the key.

TLS's close_notify is probably good enough.

Unclean shutdown results in **unauditable final epoch**.

Note: Unclean shutdown can be caused by hardware/network failure or actions by IoT device, cloud server, and unauthenticated third parties.

Overview

- Setting
- Technical Problem
- Our Scheme: TLS RaR
  - Main Idea
  - Corner Cases
    - Secure Key Release
    - Clean Shutdown
- **Evaluation**
- Conclusions

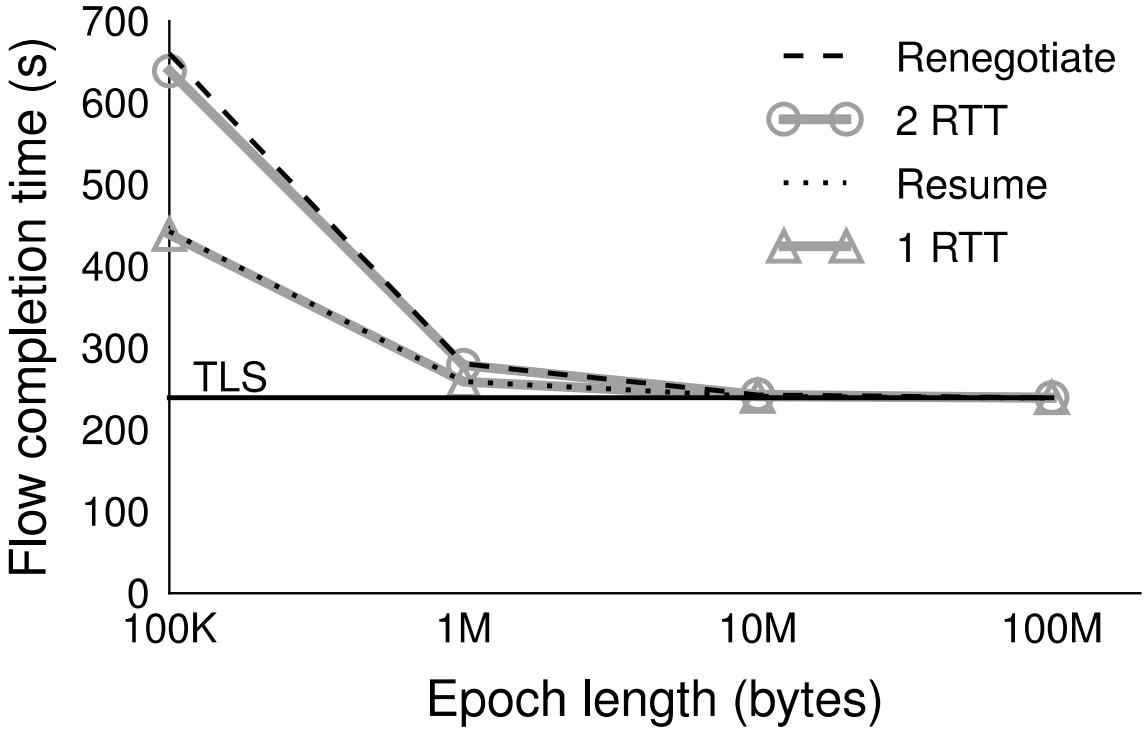
Alexa Top 1,000,000 Compatibility Survey

	Fraction of Servers*
Rotation by Reconnect	54.2%
Rotation by Renegotiation	12.2%
Rotation by Resume (requires heartbeat)	0.5%

*Includes only the ≈400,000 servers that support HTTPS and keep-alive, January, 2016.

Performance Impact

Completion time for 1000 simulated sequential downloads of a 100kB resource, over a 24 Mbps link with 100 ms latency:



Takeaway: In the *worst case* scenario (unlikely in IoT), epoch lengths can be chosen for minimal impact.

Overview

- Setting
- Technical Problem
- Our Scheme: TLS RaR
  - Main Idea
  - Corner Cases
    - Secure Key Release
    - Clean Shutdown
- Evaluation
- **Conclusions**

Conclusions

- Auditing the IoT is important, but not presently possible.

Conclusions

- Auditing the IoT is important, but not presently possible.
- Allowing a read only audit is a potential compromise.

Conclusions

- Auditing the IoT is important, but not presently possible.
- Allowing a read only audit is a potential compromise.
- TLS RaR is a technical solution with these nice properties:
  - `SSL_read()` returns same data for all trusted viewers
  - format of TLS on the wire is not changed
  - no changes for some servers
  - minimal change to OpenSSL on the device

Conclusions

- Auditing the IoT is important, but not presently possible.
- Allowing a read only audit is a potential compromise.
- TLS RaR is a technical solution with these nice properties:
  - `SSL_read()` returns same data for all trusted viewers
  - format of TLS on the wire is not changed
  - no changes for some servers
  - minimal change to OpenSSL on the device

Questions?

Judson Wilson
judsonw@stanford.edu

Backup Slides

Security Properties

- Present-Moment Integrity
  - ➔ Main channel's end-to-end integrity is preserved
- Present-Moment Secrecy
  - ➔ Auditors can't decrypt traffic until after a key release.

Security Properties

- Present-Moment Integrity
  - ➔ Main channel's end-to-end integrity is preserved
- Present-Moment Secrecy
  - ➔ Auditors can't decrypt traffic until after a key release.
- Past Auditability
  - ➔ Auditors can decrypt previously observed records for which they have the key, or return “fail.”

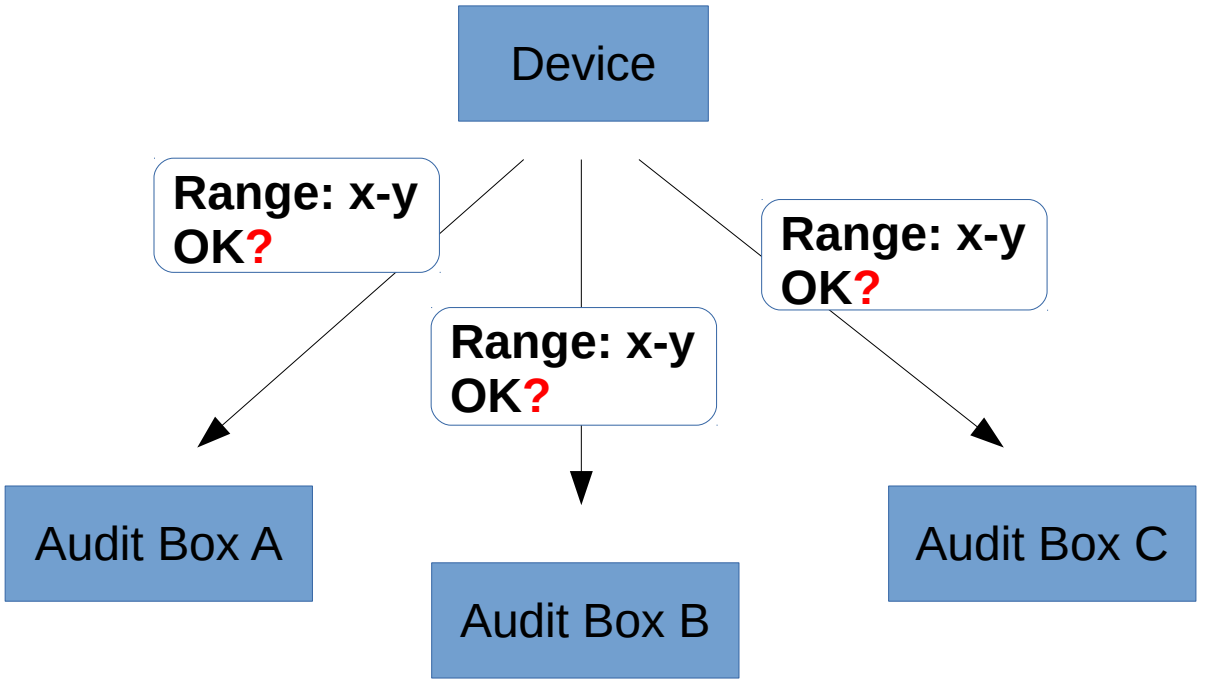
Security Properties

- Present-Moment Integrity
  - ➔ Main channel's end-to-end integrity is preserved
- Present-Moment Secrecy
  - ➔ Auditors can't decrypt traffic until after a key release.
- Past Auditability
  - ➔ Auditors can decrypt previously observed records for which they have the key, or return “fail.”
- Audit Robustness
  - ➔ Auditors cannot be convinced that a forgery (possibly from another auditor) came from one of the endpoints.

Unmitigated Threats

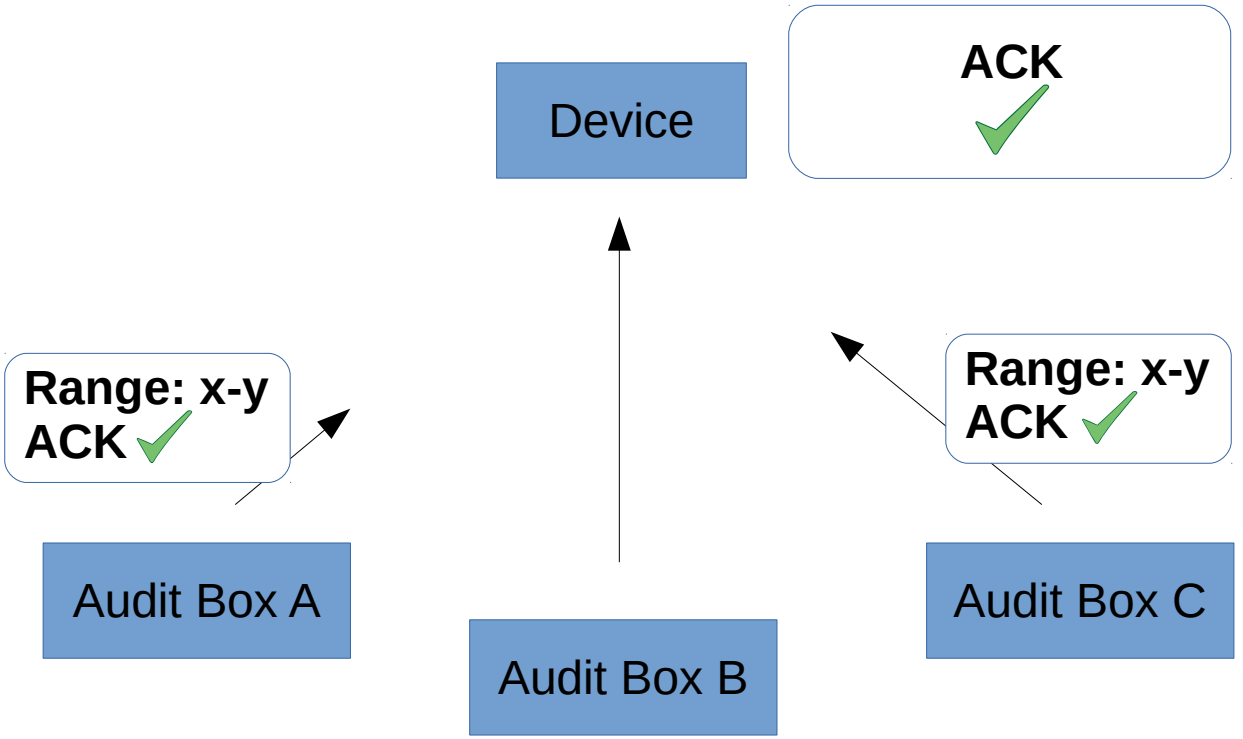
- Covert Channels
  - ➔ TLS-RaR cannot prevent secret communication between endpoints.
- Denial of Service
- Incompatible Application-Layer Authentication
  - ➔ Auditors may see and replay all plaintext, including cookies, passwords, and tickets.
  - ➔ Remedy: Use TLS client certificates

Two-Phase Key Release Procedure



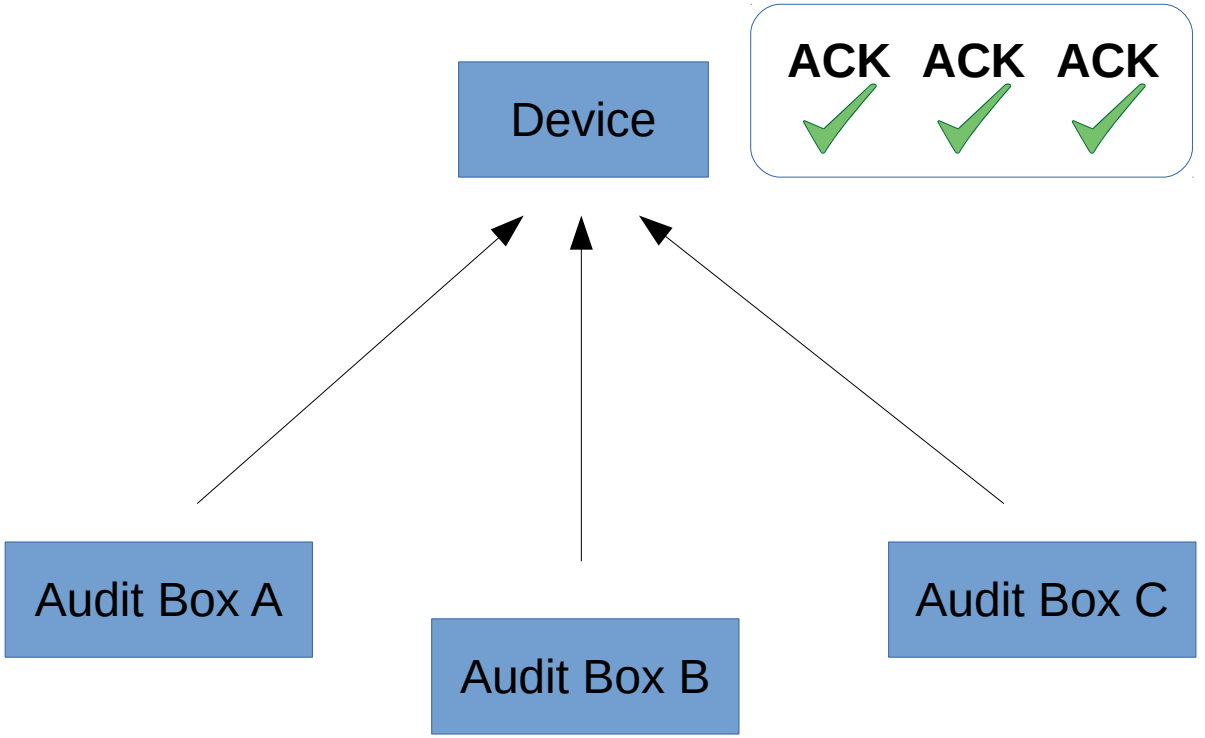
1) Device requests auditors' permission to release key protecting TCP data sequence numbers x through y.

Two-Phase Key Release Procedure



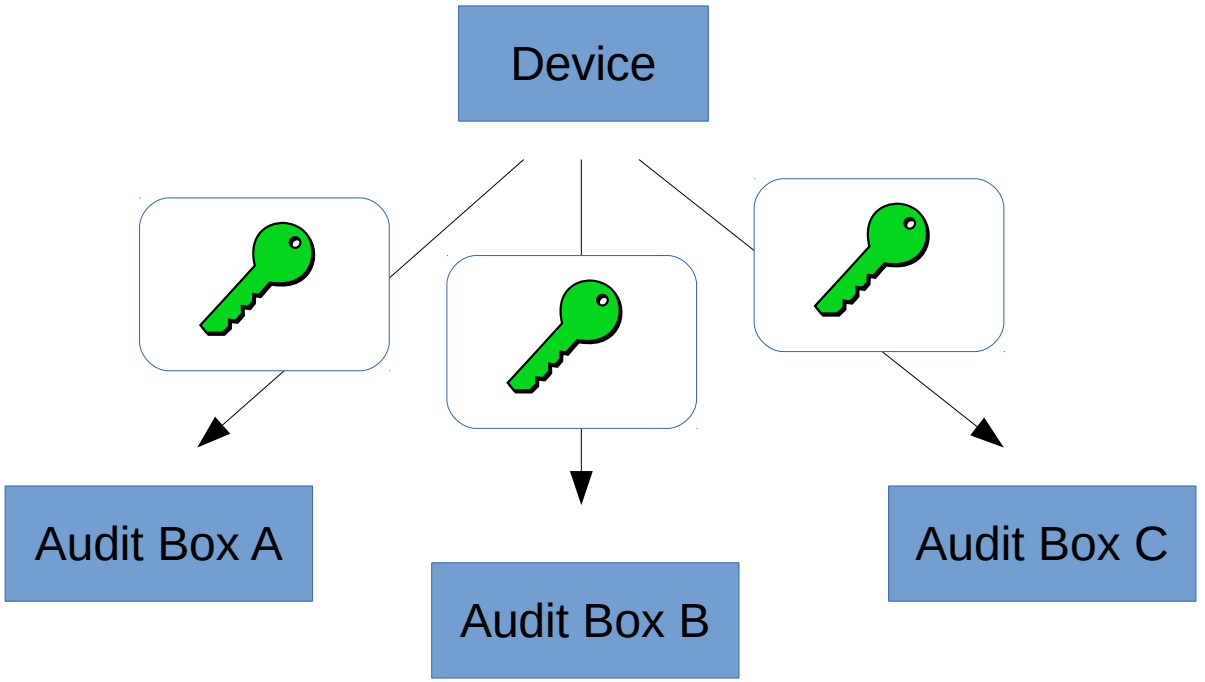
2) Audit Boxes acknowledge after they are ready for **other** Audit Boxes to receive the key.

Two-Phase Key Release Procedure



3) Device waits for all Audit Boxes to respond.
   (May time out and notify Audit Boxes.)

Two-Phase Key Release Procedure



4) Device distributes key to Audit Boxes.

Overview

- Setting
- Technical Problem
- Our Scheme: TLS RaR
  - Main Idea
  - Corner Cases
    - Secure Key Release
    - Clean Shutdown
- Evaluation
- **Related Work**
- Conclusions

Related Work

mcTLS [Naylor et. al. 2015]

- targets several problems, including read only middleboxes which could be auditors
- different approach: uses multiple MACs

Related Work

mcTLS [Naylor et. al. 2015]

- targets several problems, including read only middleboxes which could be auditors
- different approach: uses multiple MACs

BlindBox [Sherry et. al. 2015]

- solves the opposite problem, a “blind” inspection
- different trust model

Rotation Policy

Device attempts to limit Epoch length by:

- **Age** of data under new key: **60 seconds**
- **Amount** of data protected by new key: **10 MB**

Auditor logs when:

- **Age** of data under new key: **5 minutes**
- **Amount** of data protected by new key: **50 MB**

TLS 1.2 Key Generation (RFC 5246)

Inputs:

`client_random, server_random:` random data (nonces) from client/server

`pre_master_secret:` shared secret from client or Diffie-Hellman key exchange.

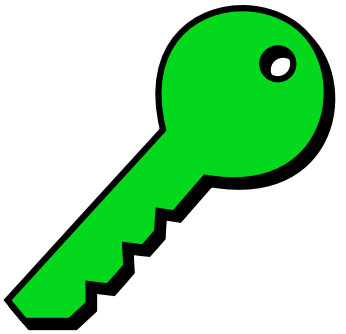
Computations:

```
master_secret = PRF(pre_master_secret, "master secret",
                                       client_random + server_random)

key_block = PRF(master_secret,
                "key expansion",
                server_random +
                client_random);
```

Outptuts:

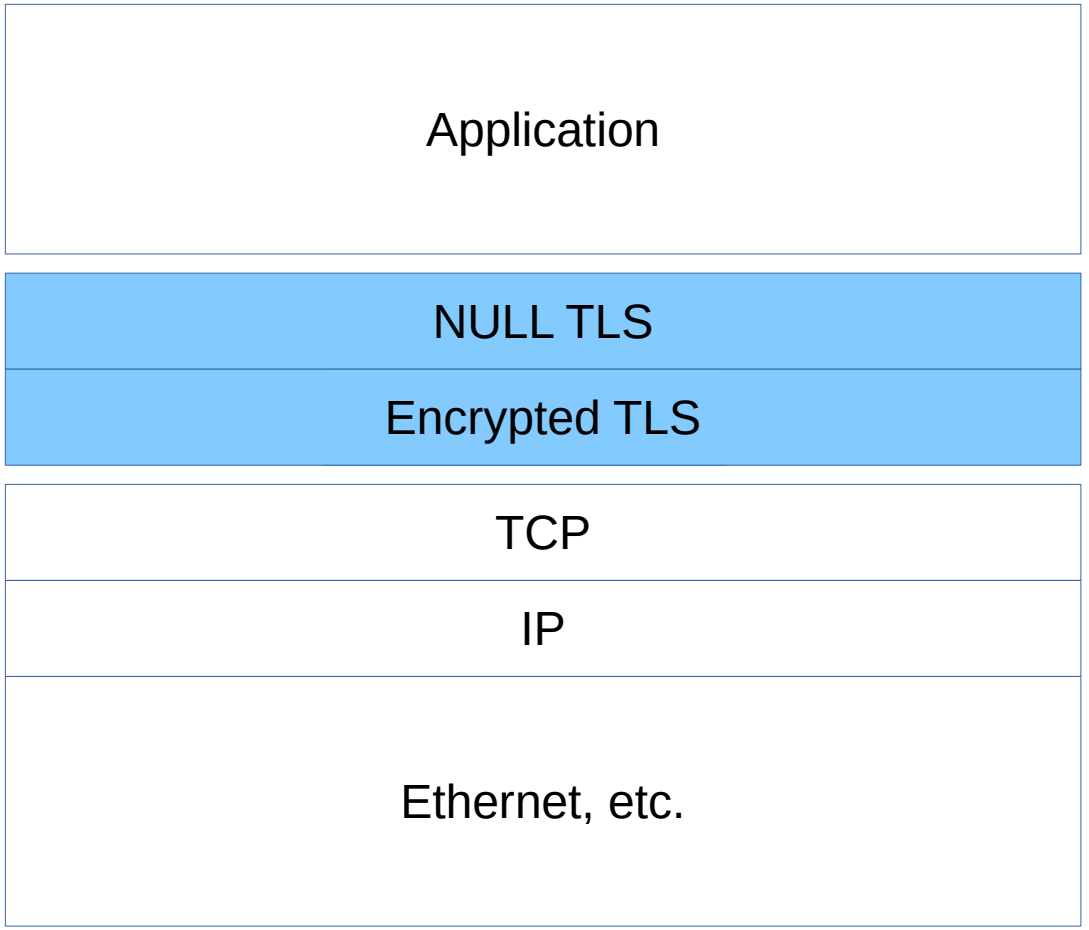
```
key_block {
        client_write_MAC_secret[hash_size]
        server_write_MAC_secret[hash_size]
        client_write_key[key_material_length]
        server_write_key[key_material_length]
        client_write_IV[IV_size]
        server_write_IV[IV_size]
}
```



A Different Idea:

TLS-in-TLS Tunneling

TLS-in-TLS



NULL TLS
- Unencrypted
- End to end integrity and Authenticity.

Encrypted TLS
- Auditor has full access, either by proxy or client releasing the keys.

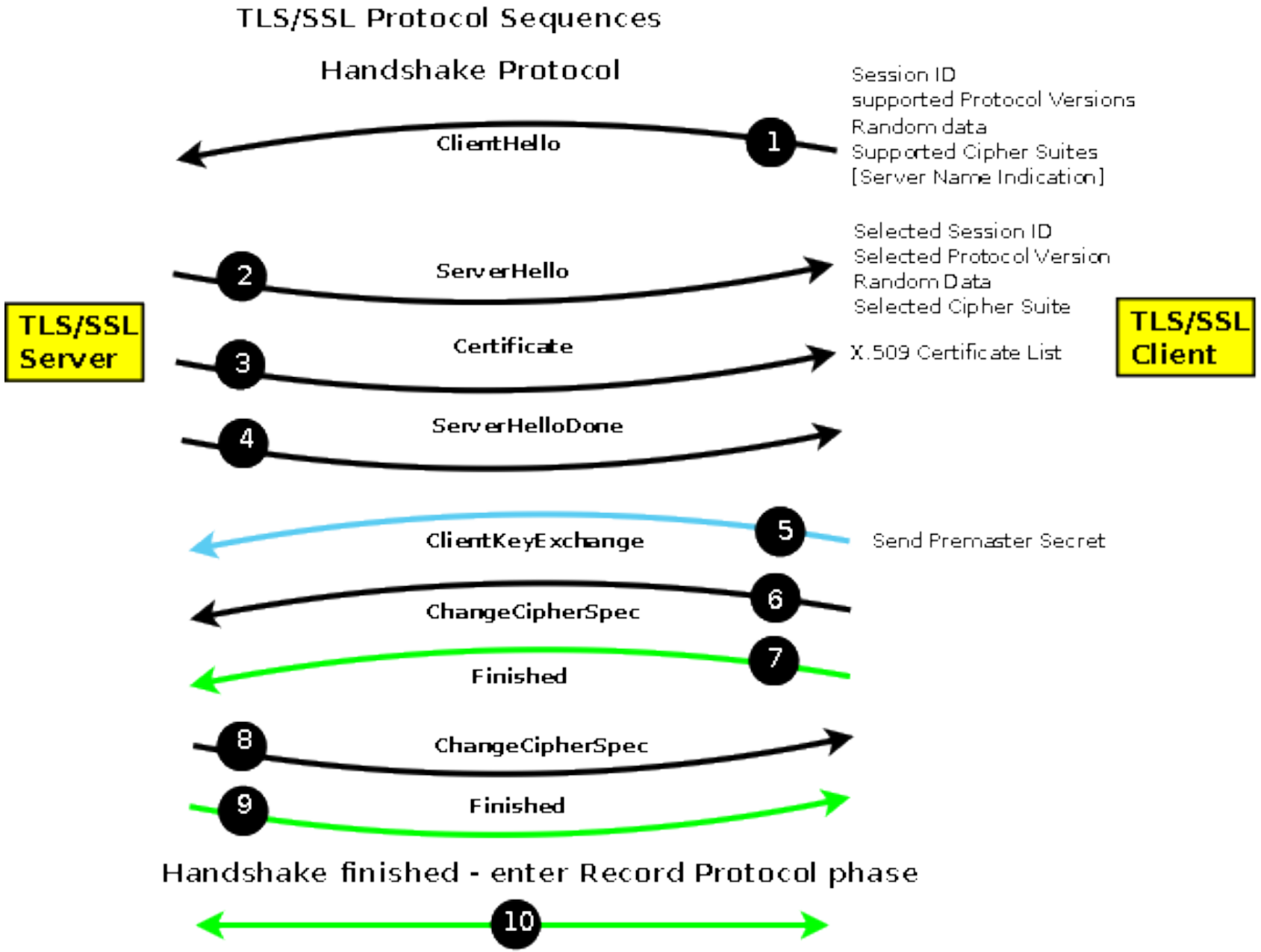
TLS-in-TLS

Benefits:

- TLS is still intact.
- No renegotiation.
- Auditor has instantaneous access to data.

Problems:

- Incompatible with existing web servers, load balancers, SSL terminators.
- Creates a new layer of cryptography where auditor is not trusted.
- NULL ciphers are unsupported, meant as a debug tool.
- Potential attacks between auditors.



Survival guides - TLS/SSL and SSL (X.509) Certificates
http://www.zytrax.com/tech/survival/ssl.html

Same Nice Properties?

BlindBox	mcTLS	
N/A	No	– `SSL_read()` returns same data for all parties.
?	No	– Format of TLS on the wire is not changed.
No	No	– No changes for some servers.
?	?	– Minimal change to OpenSSL.

Threat Model Exclusions

We **exclude** the following threats from our model:

- Malicious parties, including the **IoT device and cloud server**, may communicate by **covert channels**.
- A **malicious auditor** may **leak private data or keys**.
- **Malicious auditors may collude** to produce the same incorrect audit record.

Threat Model

An attacker may attempt anything from the TLS threat model, such as passive eavesdropping, replay and masquerading.

Threat Model

An attacker may attempt anything from the TLS threat model, such as passive eavesdropping, replay and masquerading.

A malicious audit box may try to:

- **tamper with the communications** that are being audited
- **falsify their audit** record and not be detected
- masquerade as the device or cloud server to **modify the decrypted cleartext in other auditors' records**

Threat Model

An attacker may attempt anything from the TLS threat model, such as passive eavesdropping, replay and masquerading.

A malicious audit box may try to:

- **tamper with the communications** that are being audited
- **falsify their audit** record and not be detected
- masquerade as the device or cloud server to **modify the decrypted cleartext in other auditors' records**

A malicious IoT device and/or cloud server may try to make an audit record different from the data sent (ignoring covert channels).