Privacy Enhancing Technologies (BSc)

Lecture VI: TLS

Univ.-Prof. Dr. Dominique Schröder

November 18, 2024

Overview

KW	Type	Lectures
42	**Lecture**	Internet and Privacy in the Web
43	**Lecture**	Fingerprinting & Censorship
44	**Exercise**	Introduction to Rust & Challenge Start
45	**Lecture**	Cryptographic Preliminaries
46	**Lecture**	Public Key Cryptography and Security Reductions
47	**Exercise**	Proofs and Theoretical Deepening
48	**Lecture**	**TLS**
49	**Exam**	**Midterm Exam**
50	**Exercise**	Challenge Hands-On
51	**Lecture**	Messaging I
3	**Lecture**	Messaging II (Forward Security, Post Compromise Security)
4	**Lecture**	Onion
5	**Lecture**	Q & A
6	**Exam**	**Final Exam**
9	**Exam**	**Reexam** (Both for Midterm and Final)

This Lecture's Agenda

- What is TLS and why is it crucial for internet security?
- How does TLS secure communications?
- What are the main components of the TLS handshake?
- How do common attacks exploit TLS, and how can they be mitigated?
- TLS in an Post-Quantum World

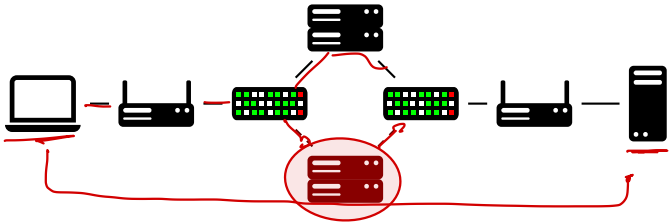
What is TLS?

Definition and Purpose

Definition (TLS; RFC 8446)

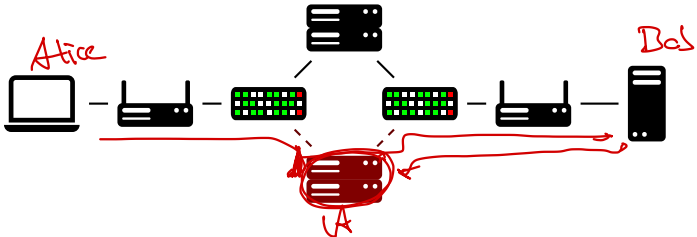
Transport Layer Security (TLS) is a protocol that provides secure communications over a computer network by encrypting data to ensure confidentiality, integrity, and authentication.

Why Transport Layer Security?



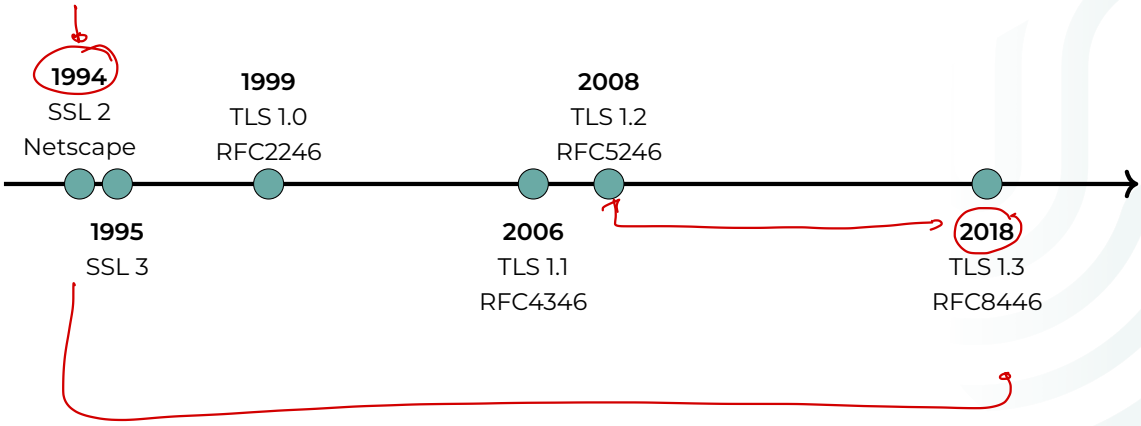
- **Confidentiality:** Prevents unauthorized parties from eavesdropping on sensitive information during transmission.
- **Integrity:** Ensures that data is not tampered with or altered during transit.
- **Authentication:** Verifies the identity of the communicating parties to prevent impersonation or man-in-the-middle attacks.

Why Transport Layer Security?



- **Confidentiality:** Prevents unauthorized parties from eavesdropping on sensitive information during transmission.
- **Integrity:** Ensures that data is not tampered with or altered during transit.
- **Authentication:** Verifies the identity of the communicating parties to prevent impersonation or man-in-the-middle attacks.

Brief History of SSL/TLS



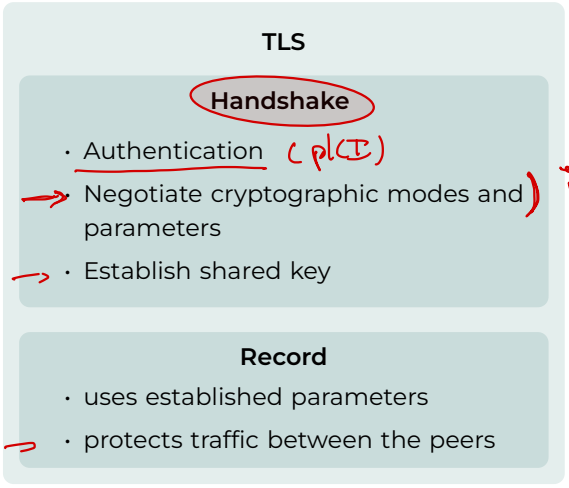
Major differences between TLS 1.2 and TLS 1.3

- Zero round-trip time (0-RTT) mode was added, saving a round trip at connection setup for some application data, at the cost of certain security properties
- Static RSA and Diffie-Hellman cipher suites have been removed; all public-key based key exchange mechanisms now provide forward secrecy → formal security model
- All handshake messages after the ServerHello are now encrypted
- The key derivation functions have been redesigned
- Session resumption with and without server-side state as well as the PSK-based cipher suites of earlier TLS versions have been replaced by a single new PSK exchange

→ old attacks → new version
→ stronger security
→ more efficient → trade off.

What is TLS?

An Overview



TLS Handshake

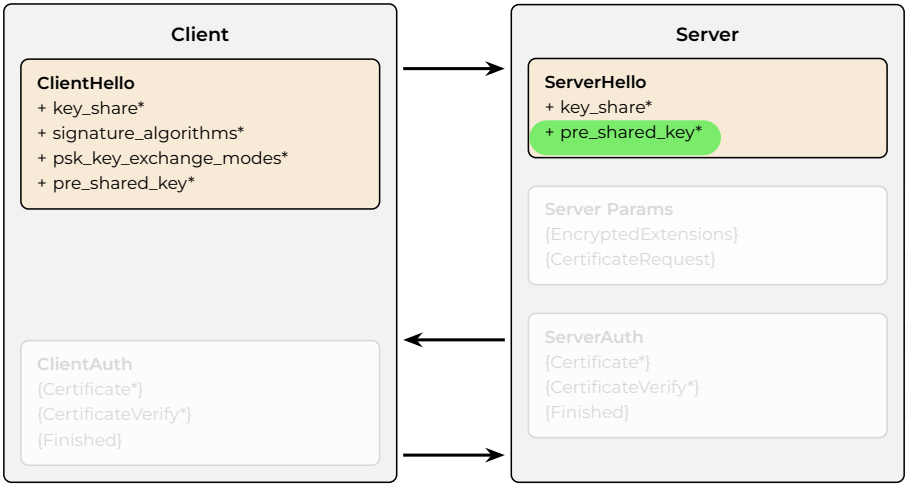
Handshake Protocol

An Overview

- **Purpose:** Authenticates communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material.
- **Security Design:** Designed to resist tampering, preventing an active attacker from forcing peers to use different parameters than intended.
- **Key Features:**
  - Mutual authentication (optional).
  - Establishment of a shared secret for secure communication.
  - Negotiation of cryptographic algorithms and parameters.

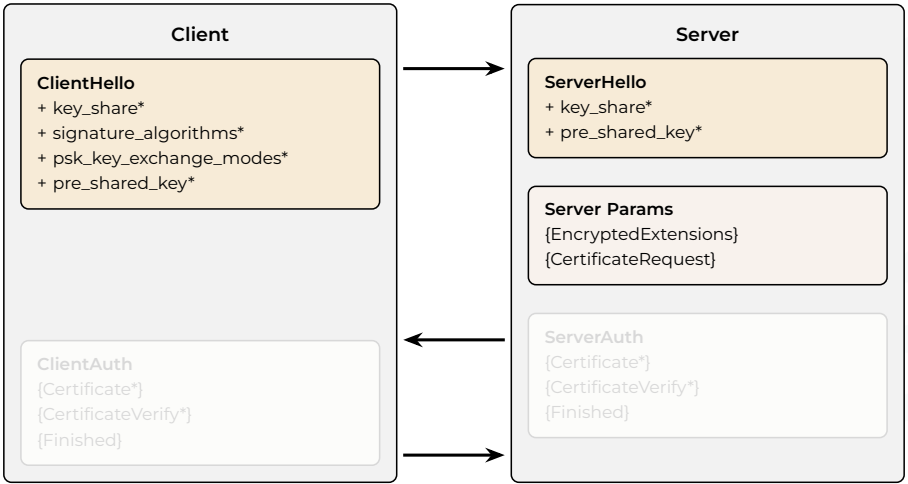
TLS Handshake

How a Secure Connection is Established



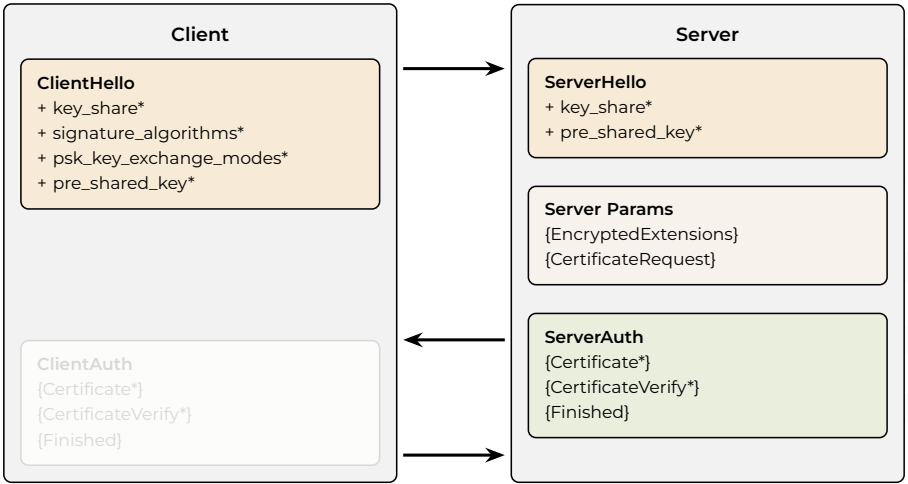
TLS Handshake

How a Secure Connection is Established



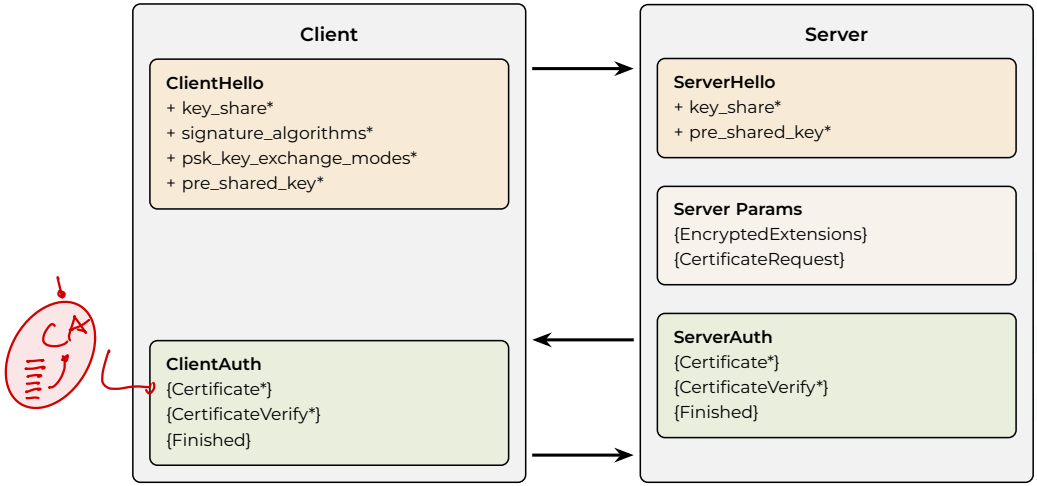
TLS Handshake

How a Secure Connection is Established



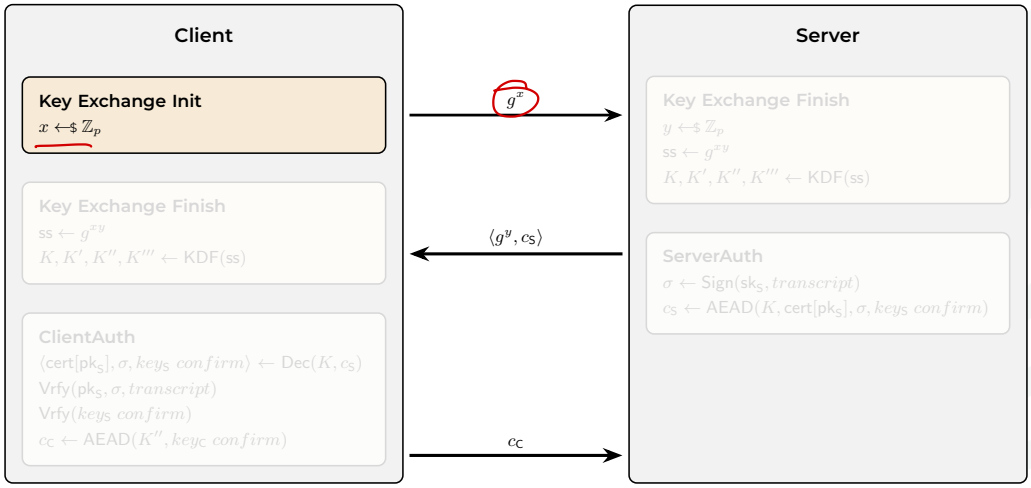
TLS Handshake

How a Secure Connection is Established



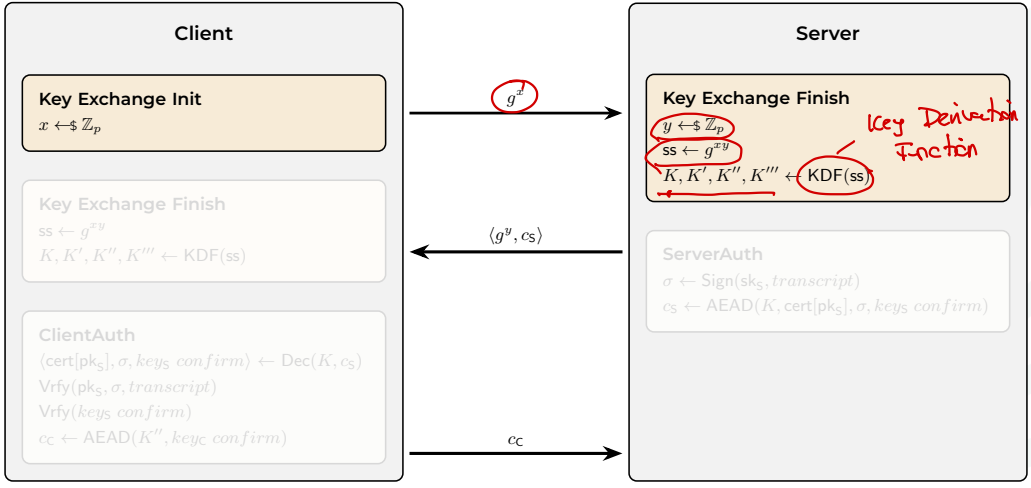
TLS Handshake

Cryptographic Core (EL Gamal)



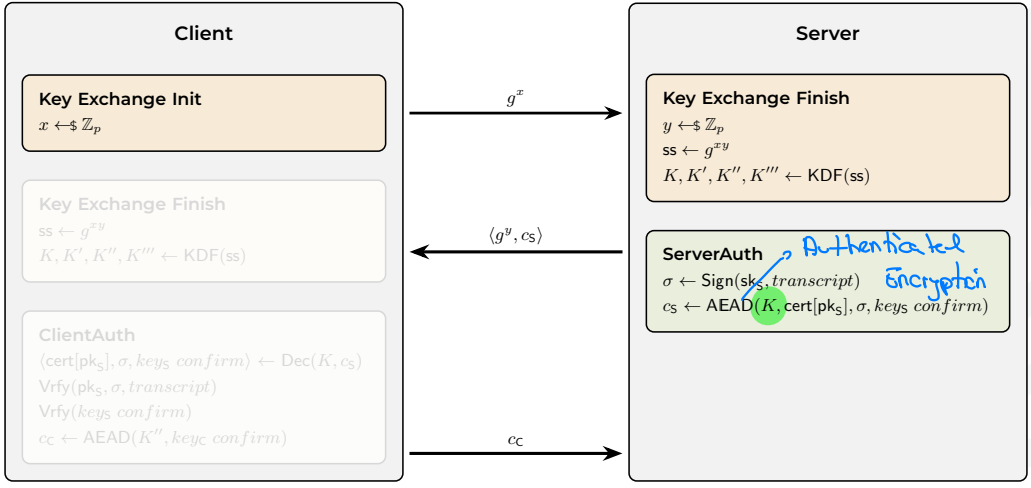
TLS Handshake

Cryptographic Core



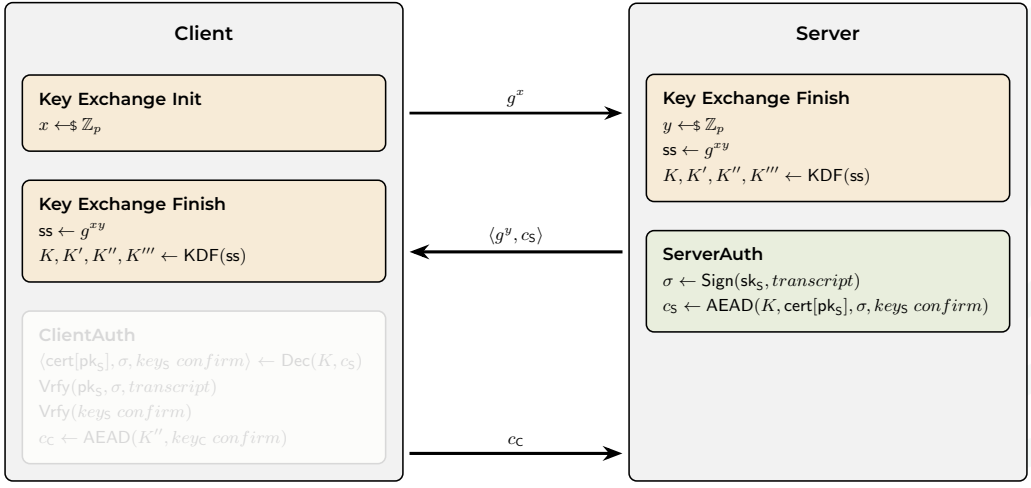
TLS Handshake

Cryptographic Core



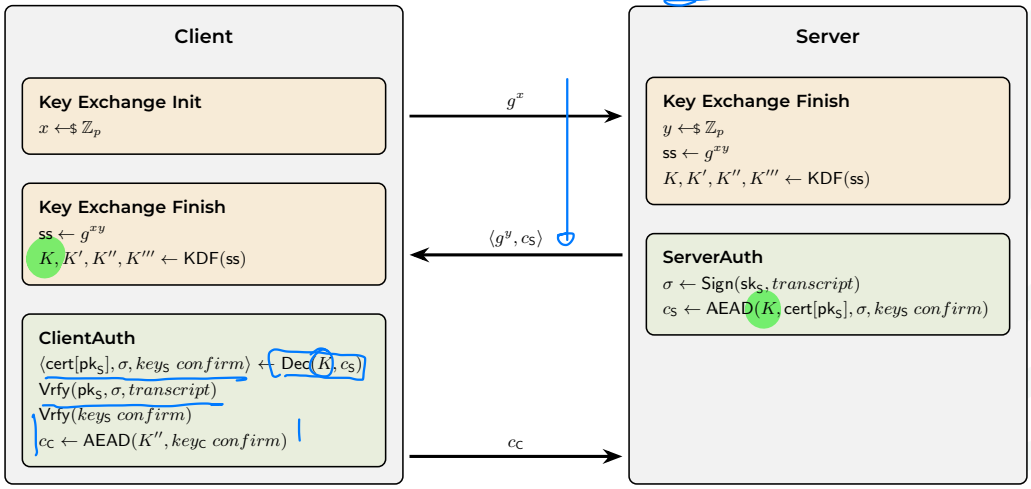
TLS Handshake

Cryptographic Core



TLS Handshake

Cryptographic Core



TLS Record

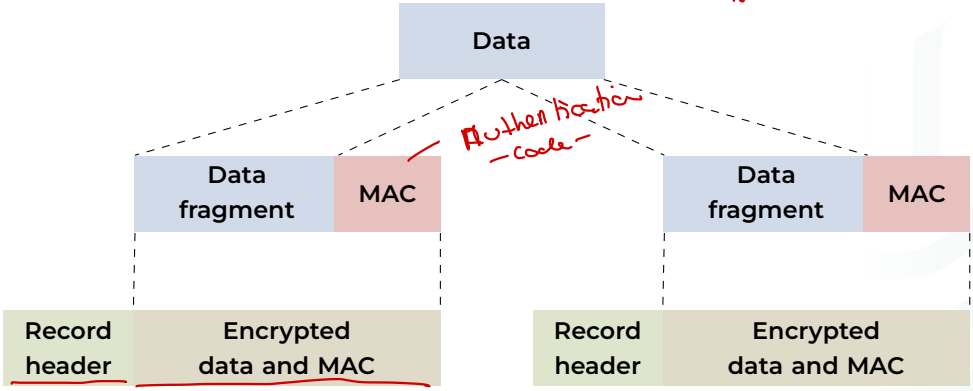
Record Protocol

An Overview

- **Purpose:** Uses parameters established by the handshake protocol to protect data exchanged between peers.
- **Functionality:** Divides traffic into a series of records, each independently protected using traffic keys.
- **Protection Mechanisms:**
  - Ensures data confidentiality and integrity.
  - Applies encryption and authentication to transmitted records.

TLS Record Protocol

Functionality



- Record header: content type; version; length
- MAC: includes the sequence number
- Fragment: each SSL fragment is 2^{14} bytes (16 Kbytes)

TLS Record

Forward Secrecy? (Nice :))

- The symmetric keys for a connection remain unchanged until the end of that connection.
- A connection lives as long as the client and server allow it; in a Web context, the server will normally close connections that have gone inactive for more than one or two minutes.
- TLS 1.3 achieves perfect forward secrecy since no long-term keys are used for key-exchange.

↳ regular key update.

— ✖ Signal —

Why Not Encrypt Data in a Constant Stream Over TCP?

- **Where to put the MAC (Message Authentication Code)?**
  - If placed at the end, there is no message integrity until all data is processed (e.g., in instant messaging scenarios).
  - How can we perform integrity checks on all bytes sent before displaying them?
- **Solution:** Break the stream into a series of records.
  - Each record carries its own MAC.
  - The receiver can act on each record as it arrives, ensuring timely processing and security.

TLS in Practice

The TLS 1.3 Cryptographic Building Blocks

Abstract Primitive	Construction	Based on	Security Guarantee
AEAD	AES, -GCM, -CCM; ChaCha20-Poly1305	Blockcipher; Mode of Operation; Streamcipher	Message-Indistinguishability, Integrity
Signature Algorithm	RSA, -PKCS, -RSS; EdDSA (Standard)		Authentication, Integrity
Key Derivation Function	HKDF (protocol based on hash functions)	HMAC (Hash function)	Key Indistinguishability

DSA → standard BUT NO security proof is known

TLS in Practice

Example Use Case

- **HTTPS:** TLS secures web traffic over HTTPS, providing encrypted communication between browsers and web servers.
- **Use Case:** Accessing a banking website securely.
- **Steps:**
  - Browser connects to the bank's website.
  - TLS handshake occurs, and the server's identity is verified.
  - All subsequent communication is encrypted.

TLS in Practice

Common Attacks on TLS

- **Man-in-the-Middle (MitM) Attacks:** Attackers intercept and alter communications.
- **Downgrade Attacks:** Forcing connections to use older, less secure protocols (e.g., SSL 3.0).
- **Heartbleed Bug:** A vulnerability in the OpenSSL library allowing data leakage.
- **TLS Stripping:** Downgrading HTTPS to HTTP.

Countermeasures and Best Practices

Mitigating TLS Attacks

- **Use Modern Versions:** Enforce TLS 1.2 or 1.3 for stronger security.
- **Certificate Pinning:** Ensure a known certificate is used by a server.
- **Strict Transport Security (HSTS):** Enforce HTTPS connections.
- **Keep Libraries Updated:** Regularly update TLS libraries to mitigate known vulnerabilities.

Heartbleed

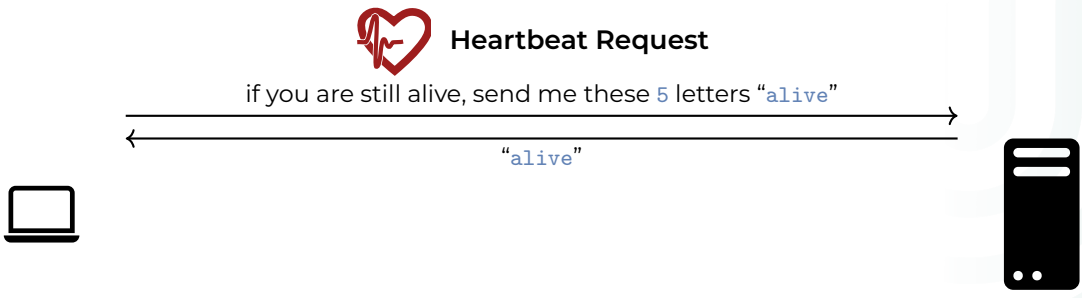
The Heartbleed Bug

Overview and Background

- **What is Heartbleed?**
  - A critical security vulnerability in the OpenSSL library, discovered in April 2014.
  - Affected versions of OpenSSL from 1.0.1 to 1.0.1f.
- **Impact and Importance**
  - Allowed attackers to read up to 64 KB of memory from a connected server or client.
  - Could expose sensitive data, such as private keys, passwords, session tokens, and more.
  - Estimated to affect around 17
- **Naming and Popularity**
  - Named "Heartbleed" due to its connection with the TLS heartbeat extension.
  - Became widely publicized, raising public awareness about the importance of internet security.

The Heartbleed Bug

Technical Overview



The Heartbleed Bug

Technical Overview

The Heartbleed Bug

Technical Details

- **Vulnerability Mechanism**
  - Exploited a flaw in the implementation of the **TLS heartbeat extension** (RFC 6520).
  - Heartbeat requests are used to keep connections alive by sending a small packet containing a payload and its length.
  - Vulnerable versions of OpenSSL failed to properly validate the payload length, allowing an attacker to request a larger response than the payload size.
- **Exploit Example**
  - An attacker sends a request with a small payload (e.g., 1 byte) but claims a large length (e.g., 64 KB).
  - The server responds by sending back up to 64 KB of memory, potentially containing sensitive data.
- **Mitigation and Patching**
  - Update OpenSSL to version 1.0.1g or later.
  - Regenerate private keys and reissue SSL certificates.

TLS in a Post Quantum World

TLS In A Post Quantum World

Quantum Computers

- **Quantum Computers:** Leverage principles of quantum mechanics, such as superposition and entanglement, to process information in fundamentally new ways.
- **Qubits:** Unlike classical bits, qubits can represent both 0 and 1 simultaneously, enabling parallel computation on an unprecedented scale.
- **Potential Impact:** Quantum algorithms, like Shor's and Grover's, could solve certain problems exponentially faster than classical algorithms, posing significant challenges to current cryptographic systems.

→ key-size doubb

$g^x = x$

TLS In A Post Quantum World

Shor's Algorithm

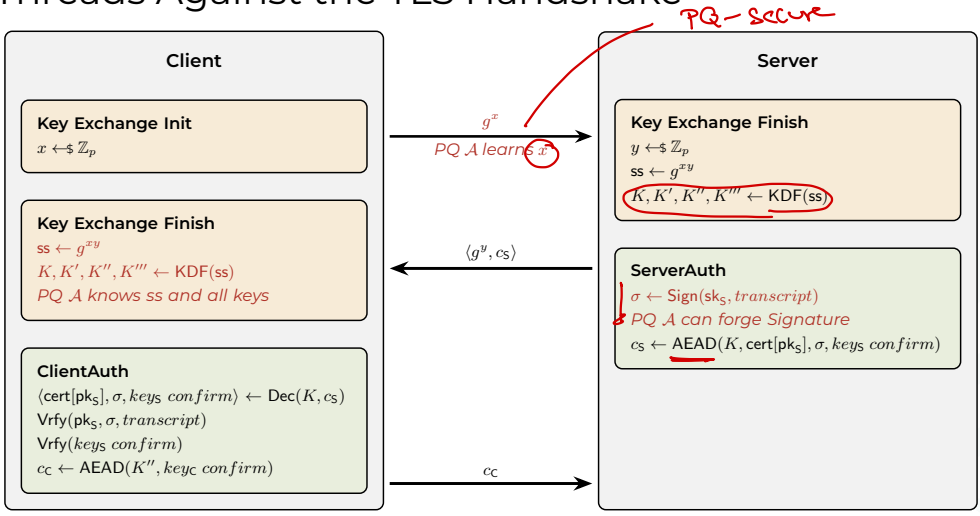
- Exponentially **efficient at factoring large integers and computing discrete logarithms**.
- Hard problems for classical computers like RSA, DDH, CDH, and DLog are solvable in polynomial time using Shor's algorithm on a quantum computer.
- Threatens the security of traditional public-key cryptosystems, such as RSA and elliptic curve cryptography (ECC), which underpin TLS key exchange and authentication.

TLS In A Post Quantum World

Grover's Algorithm

- Provides a **quadratic speedup for unstructured search problems**, affecting symmetric cryptographic schemes like AES and hash functions.
- Reduces the effective key space of symmetric ciphers by half, necessitating larger key sizes to maintain equivalent security (e.g., AES-512 for post-quantum security).
- Impacts the integrity guarantees of TLS by challenging hash-based message authentication codes (HMACs).

PQ Threads Against the TLS Handshake



Mitigating PQ Adversaries

KEMTLS

- **Post-Quantum Key Exchange:** Replaces RSA/ECC with quantum-resistant key encapsulation mechanisms (KEMs) like Kyber or NTRU.
- **Simplified Authentication:** Combines KEMs with symmetric cryptography, avoiding vulnerabilities of quantum-broken digital signatures.
- **Resilience to Grover's Algorithm:** Employs symmetric keys with larger sizes (e.g., AES-256) to ensure security.

Lattice, Hash function, Code-based, Isogenies (6)

"Rainbow" ~recent

( )( ) +ε

① representation equiv.
② ppt

Proving TLS Secure

Why is Proving TLS 1.3 Secure Hard? (1)

Challenges in Formal Security Proofs

- **Complexity of the Protocol:**
  - TLS 1.3 has multiple components (handshake, record protocols, key derivation, etc.).
  - These components are tightly intertwined, making the protocol challenging to model formally.
- **Real-World Implementation Variations:**
  - Implementations may vary in subtle ways, introducing unforeseen vulnerabilities.
  - Ensuring that the theoretical security model aligns with practical implementations is difficult.
- **Security Goals are Multi-Faceted:**
  - TLS 1.3 aims to provide multiple properties: authentication, confidentiality, forward secrecy, resistance to tampering, etc.
  - Proving all these properties simultaneously in a single framework is complex.

Why is Proving TLS 1.3 Secure Hard? (2)

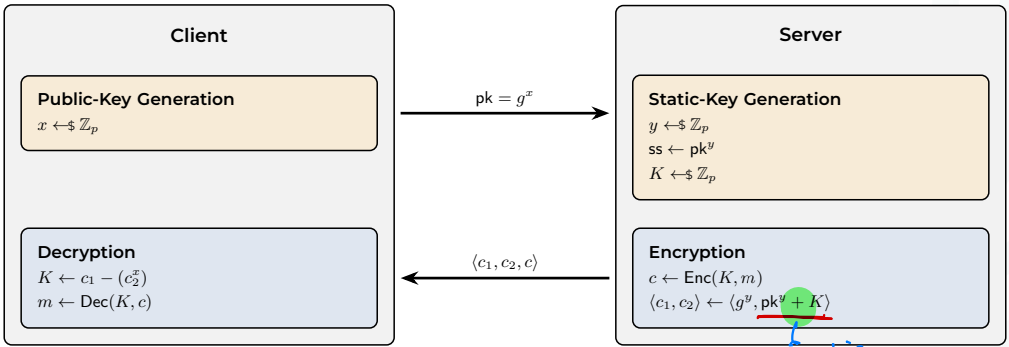
Challenges in Formal Security Proofs

- **Active Adversary Models:**
  - TLS 1.3 is designed to withstand active attackers (e.g., man-in-the-middle), requiring stronger proofs.
  - Modeling all possible attack vectors is non-trivial.
- **Backward Compatibility:**
  - TLS 1.3 incorporates features to ensure compatibility with legacy systems, adding complexity.
- **Cryptographic Assumptions:**
  - Proofs rely on multiple cryptographic assumptions (e.g., secure AEAD, secure key exchange).
  - Proving these assumptions hold under all conditions is challenging.

Proving TLS Secure

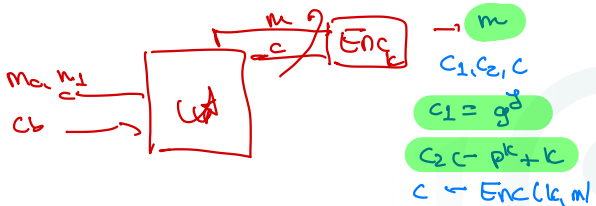
Hybid Encryption As A TLS Simplification

DH-Key Exchange (EC Gamal)



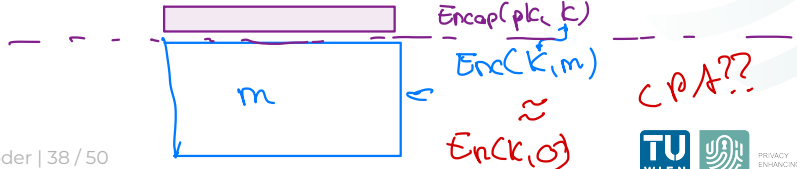
Hybid Encryption

A Construction



HybridKGen(λ)	HybridEncrypt(pk, m)	HybridDecrypt(sk, $\langle c_1, c_2, c \rangle$)
1 : sk $\leftarrow\$$ $\mathbb{Z}_p$	1 : $K \leftarrow\$ \mathbb{Z}_p$	1 : $K \leftarrow c_2 - c_1^{\mathsf{sk}}$
2 : pk $\leftarrow g^{\mathsf{sk}}$	2 : $c \leftarrow$ AESEncrypt(K, m)	2 : $m \leftarrow$ AESDecrypt(K, c)
3 : **return** (sk, pk)	3 : $y \leftarrow\$ \mathbb{Z}_p$	3 : **return** m
	4 : $\langle c_1, c_2 \rangle \leftarrow \langle g^y, \mathsf{pk}^y + K \rangle$	
	5 : **return** $\langle c_1, c_2, c \rangle$	

≈ ElGamal encryption of K



Hybrid Encryption

Proof Outline (1)

We Prove the security of hybrid encryption in two steps:

- First we modify the hybrid encryption ciphertexts to look like

$$\langle c_1, c_2, c \rangle = \langle g^y, \mathsf{pk}^y, \mathsf{AESEncrypt}(K, m) \rangle$$

  for a random key K
- This seperates the ciphertexts and makes Decryption impossible.
- This gap can be only discovered by an adversary breaking the CPA security of El Gamal.
- We show using a reduction from the gap between the ciphertexts to the CPA security of El Gamal that the difference between the original ciphertext and this ciphertext is negligible.

Hybrid Encryption

Proof Outline (2)

- Second, we construct a reduction from the modified ciphertext to the CPA security of AES.
- To do this, we again change the ciphertext using the Encryption oracle of the CPA security game.
- The final ciphertext looks like this:

$$\langle c_1, c_2, c \rangle = \langle g^y, \mathsf{pk}^y, \mathsf{Enc}\mathcal{O}(m) \rangle$$

- the reduction does not know the effective encryption key

Taking both steps together, we show that El Gamal's CPA security and AES's CPA security bounds the probability of breaking the CPA security of hybrid encryption. Since we assume both El Gamal and AES to be CPA secure, Hybrid Encryption is also secure.

Hybrid Encryption

First Reduction: Intuition

Game 0

$\text{Enc}(pk, m)$

$k \leftarrow \{0,1\}^\lambda$

$c_0 \leftarrow \text{AES}(k, m)$

$c_1 \leftarrow \text{Enc}(pk, k)$

Game 1

$\text{Enc}(pk, m)$

$k_0, k_1 \leftarrow \{0,1\}^\lambda$

$c_0 \leftarrow \text{AES}(k_0, m)$

$c_1 \leftarrow \text{Enc}(pk, k_1)$

Claim:

$$\Pr[\text{CPA}^{\text{Game 0}}_{pk,\mathcal{A}} = 1] \approx \Pr[\text{CPA}^{\text{Game 1}}_{pk,\mathcal{A}} = 1]$$

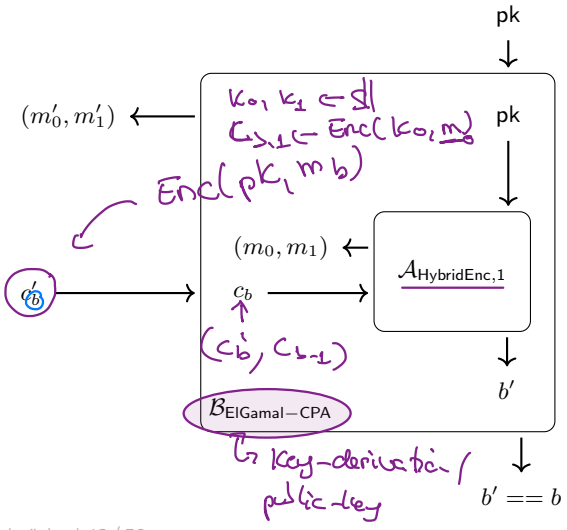
Proof: Contradiction: Game 0 $\not\approx$ Game 1

$\Rightarrow$ non-negl

$\Rightarrow \mathcal{A}_{\text{hybrid-enc}}$

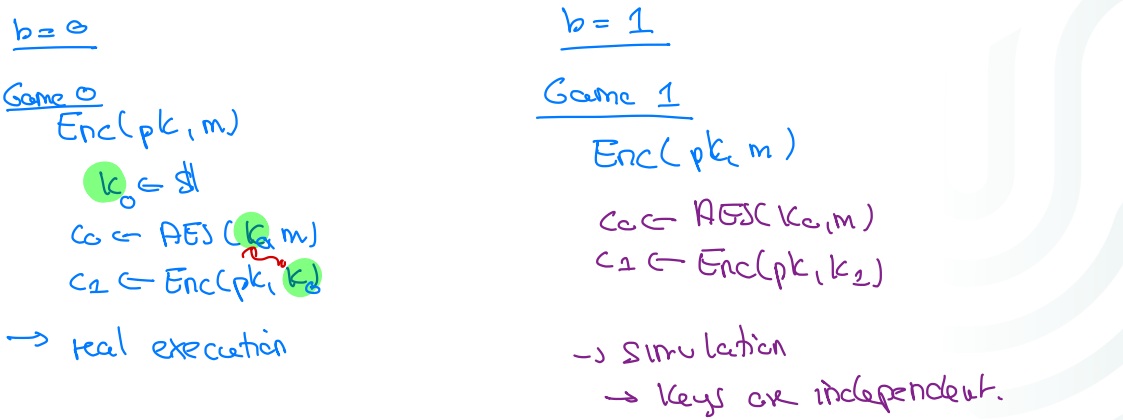
Hybrid Encryption

The First Reduction



Hybrid Encryption

Probabilities of First Reduction

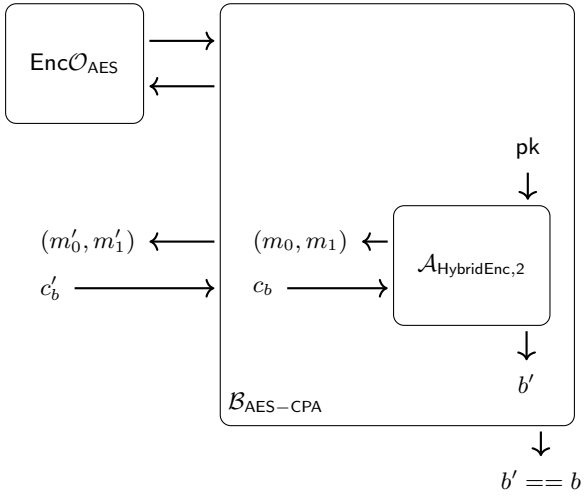


Hybrid Encryption

Second Reduction: Intuition

Hybrid Encryption

The Second Reduction



Hybrid Encryption

Probabilities of Second Reduction

Summary and Remarks

TLS Proof

The explanation provided here captures the intuition behind a small part of the TLS protocol. Proving the security of the complete, full TLS protocol is a highly technical and complex process involving rigorous cryptographic analysis. This level of detail is beyond the scope of this presentation. For a deeper understanding, we encourage you to refer to the formal security proofs and protocol specifications.

Summary and Remarks

AES and CPA Security

AES on its own is not CPA-secure. It achieves CPA security only when used in an appropriate mode of operation, such as CBC with random IVs, CTR mode, or authenticated encryption modes like GCM. For more details, refer to the lecture on cryptography.

Feedback

Your *anonymous feedback* is invaluable to us, and we are truly grateful for your time and insights.

Thank you for helping us improve!

https://tuwel.tuwien.ac.at/mod/feedback/view.php?id=2459252

PRIVACY
ENHANCING
TECHNOLOGIES