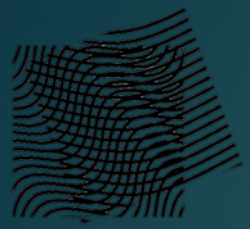


Security Features of TLS 1.3

Cyprien de SAINT GUILHEM

IMEC-COSIC, KU LEUVEN

cdsg<at>esat<dot>kuleuven.be

SECAPPDEV 2019

KU LEUVEN

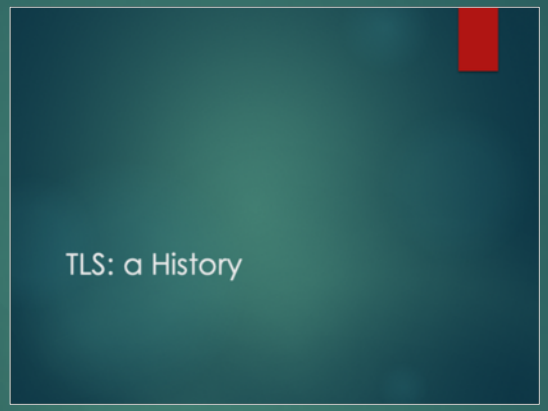
Thanks to Bart Preneel for slide inspiration.

Outline

- Transport Layer Security Protocol – where, why and how?
- The road to TLS 1.2 and after.
- What's new in TLS 1.3.

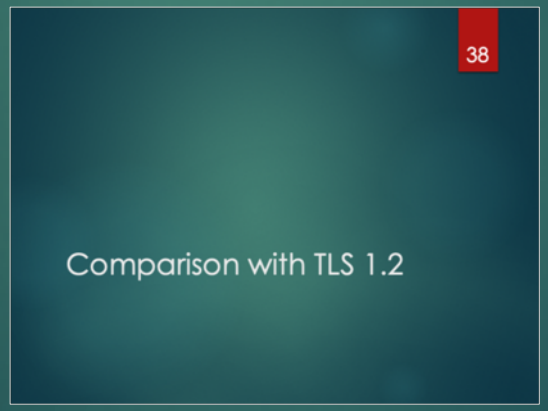
Overview


Transport Layer Security
WHERE? WHY? HOW?


TLS: a History


Transport Layer Security
v1.2
IS IT THE END?

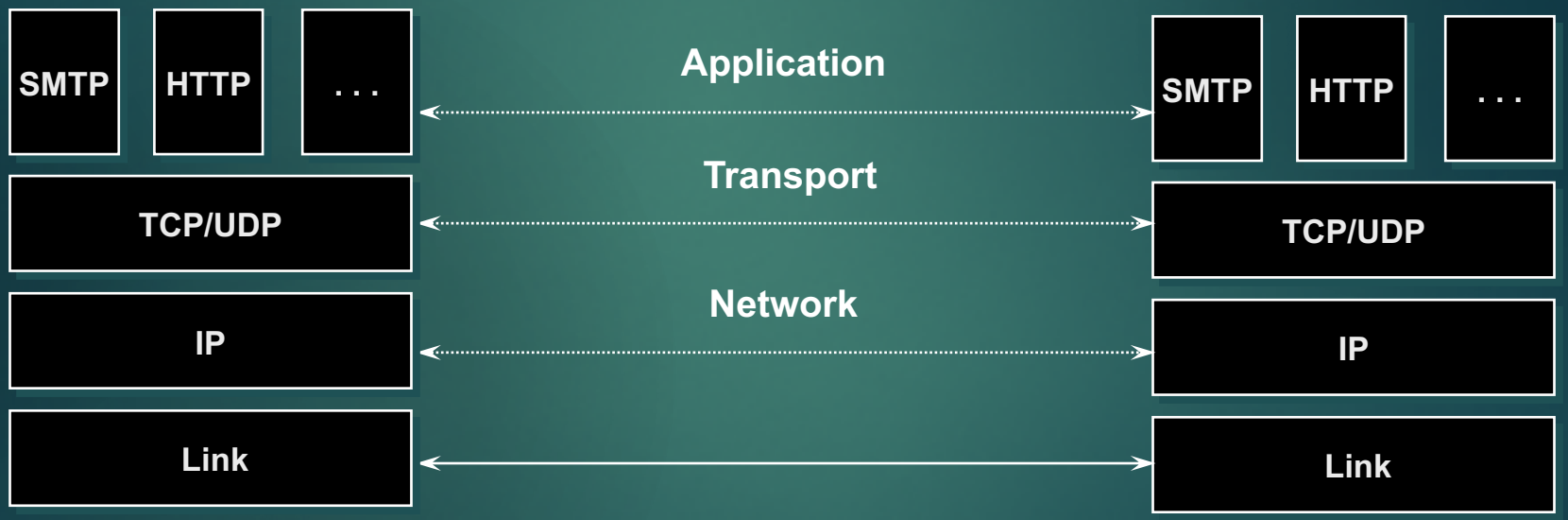

Transport Layer Security
v1.3
SECURITY FEATURES


38
Comparison with TLS 1.2

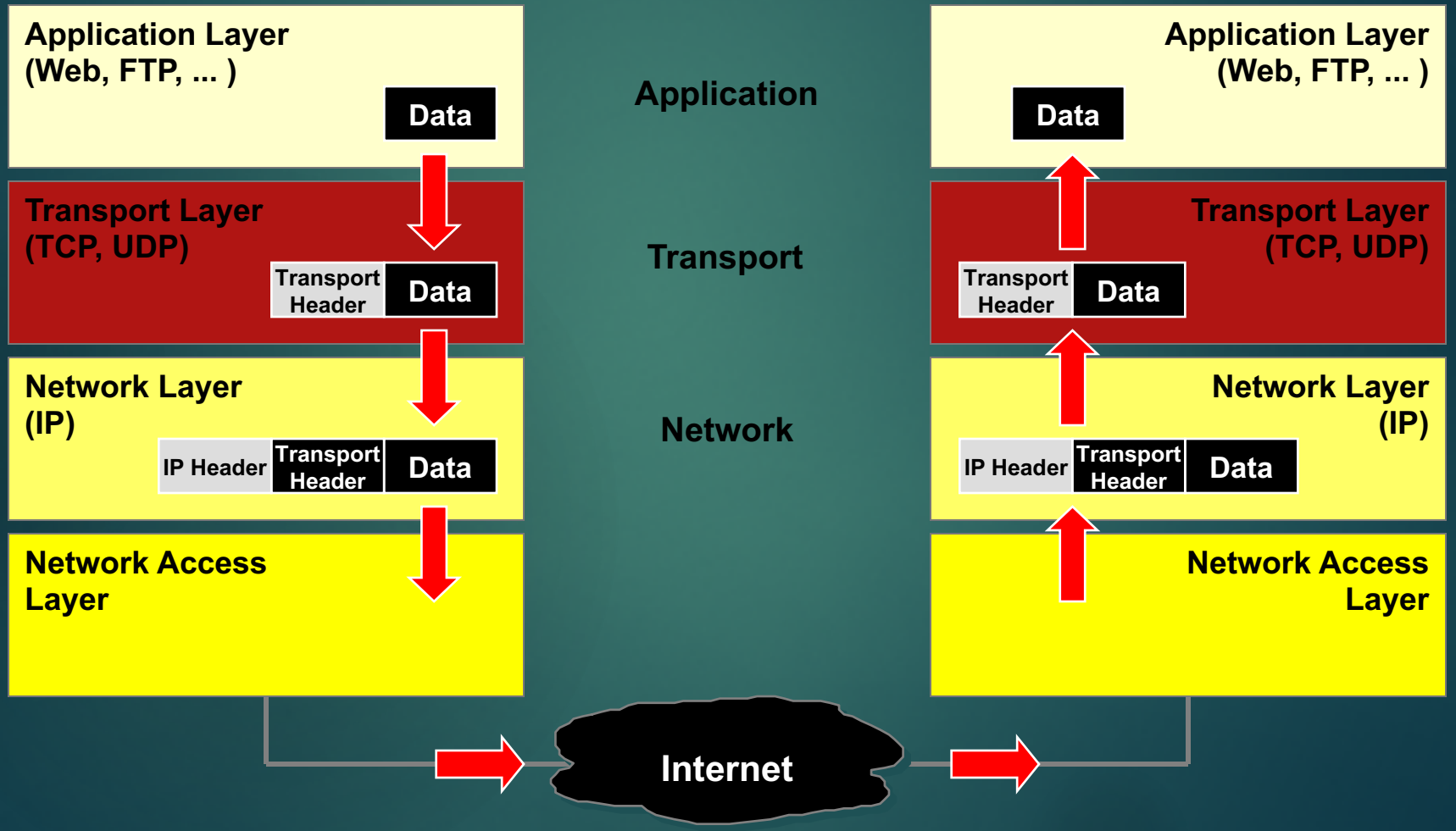
Transport Layer Security

WHERE? WHY? HOW?

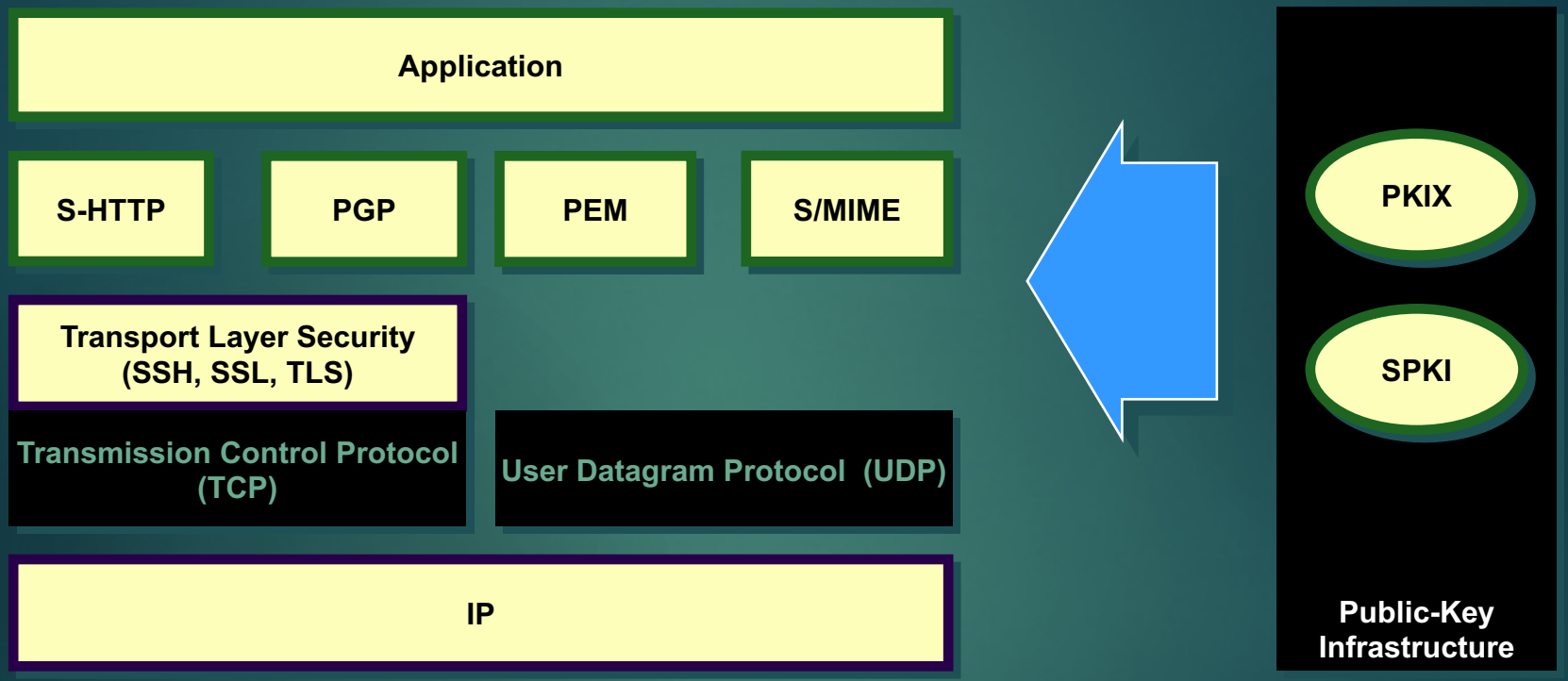
Internet Protocols



Data Encapsulation



TLS, where are you?

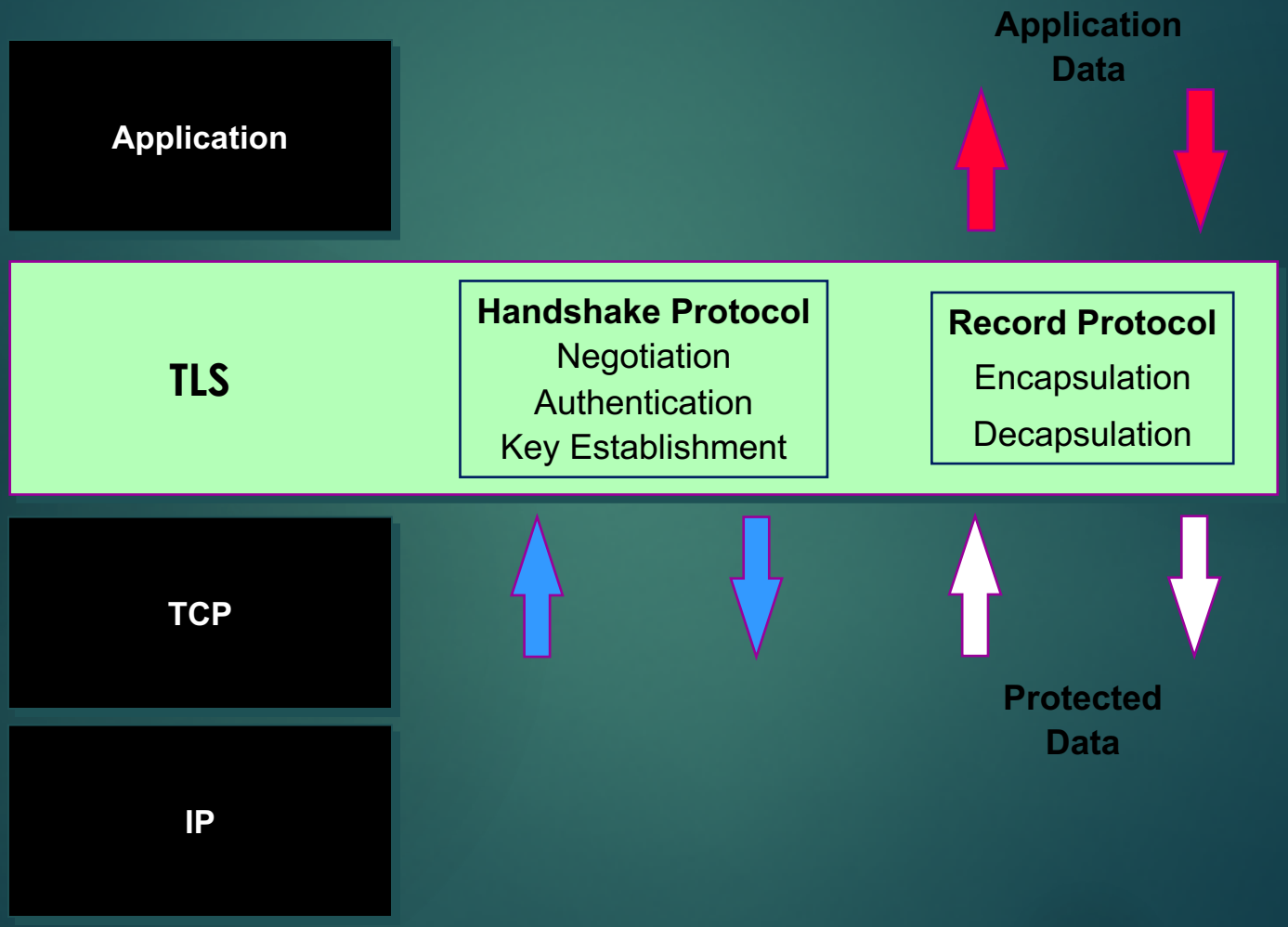


- the protocol can only protect the payload and/or header information available at its layer

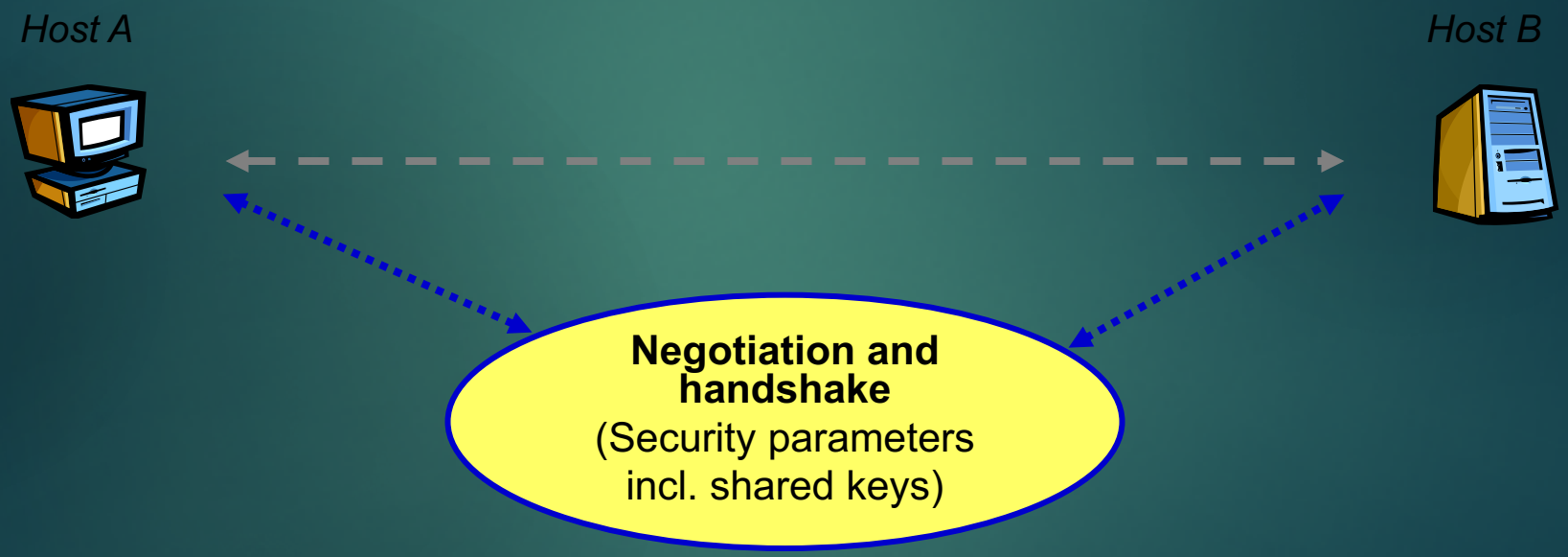
TLS – goals:

- Authentication:
  - Servers always authenticate to clients.
  - Clients can authenticate to servers.
- Confidentiality:
  - Data cannot be seen in transit.
- Integrity:
  - Data cannot be modified in transit.

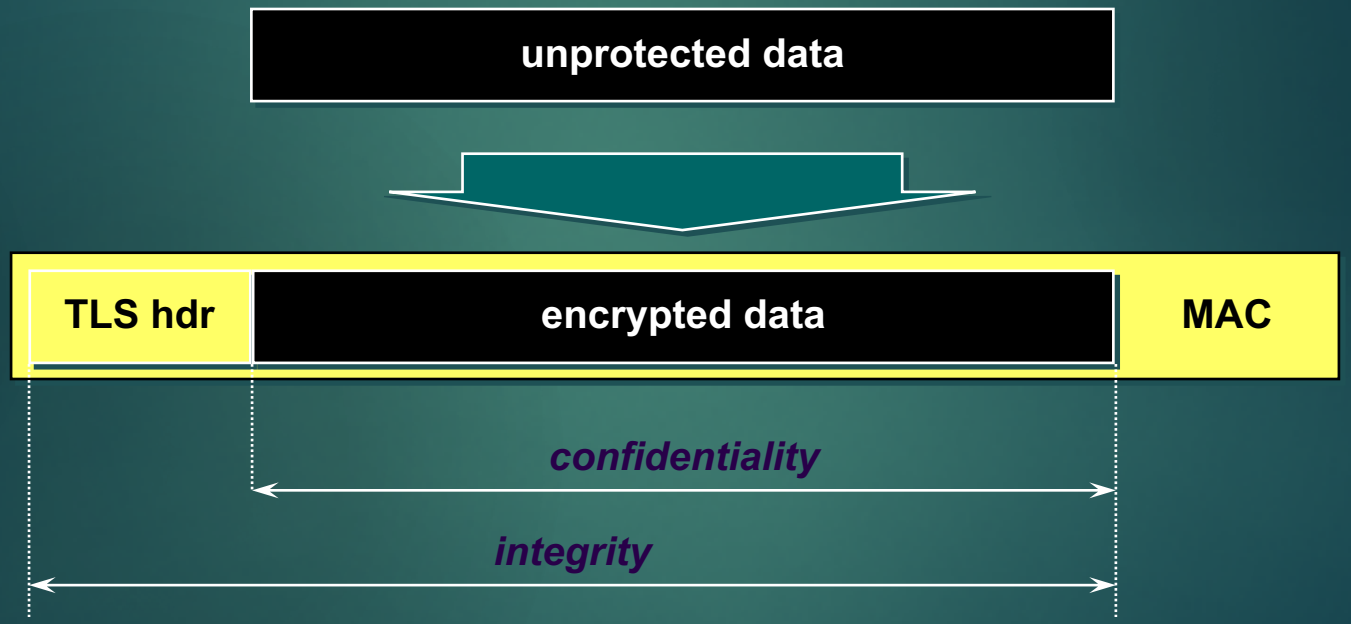
TLS – function:



TLS – function: Session establishment



TLS – function: Data encapsulation



Algorithm Selection

"Suite negotiation"

- Different fixed suites.
  - Encodes algorithms and parameters.
- Simple and compact.
- Management of risk (somewhat).
- Less flexible.
- Potentially exponential number of suites (314 for TLS 1.2).

TLS and cryptography

e.g. `TLS_ECDHE_RSA_WITH_AES_GCM_SHA256`

- Cryptographic techniques:
  - Key establishment (secrecy): `ECDHE`
  - Entity authentication: `RSA`
  - Symmetric encryption (data privacy): `AES_GCM`
  - MACs: Message authentication mechanisms (data integrity): `SHA256` (with `HMAC`)

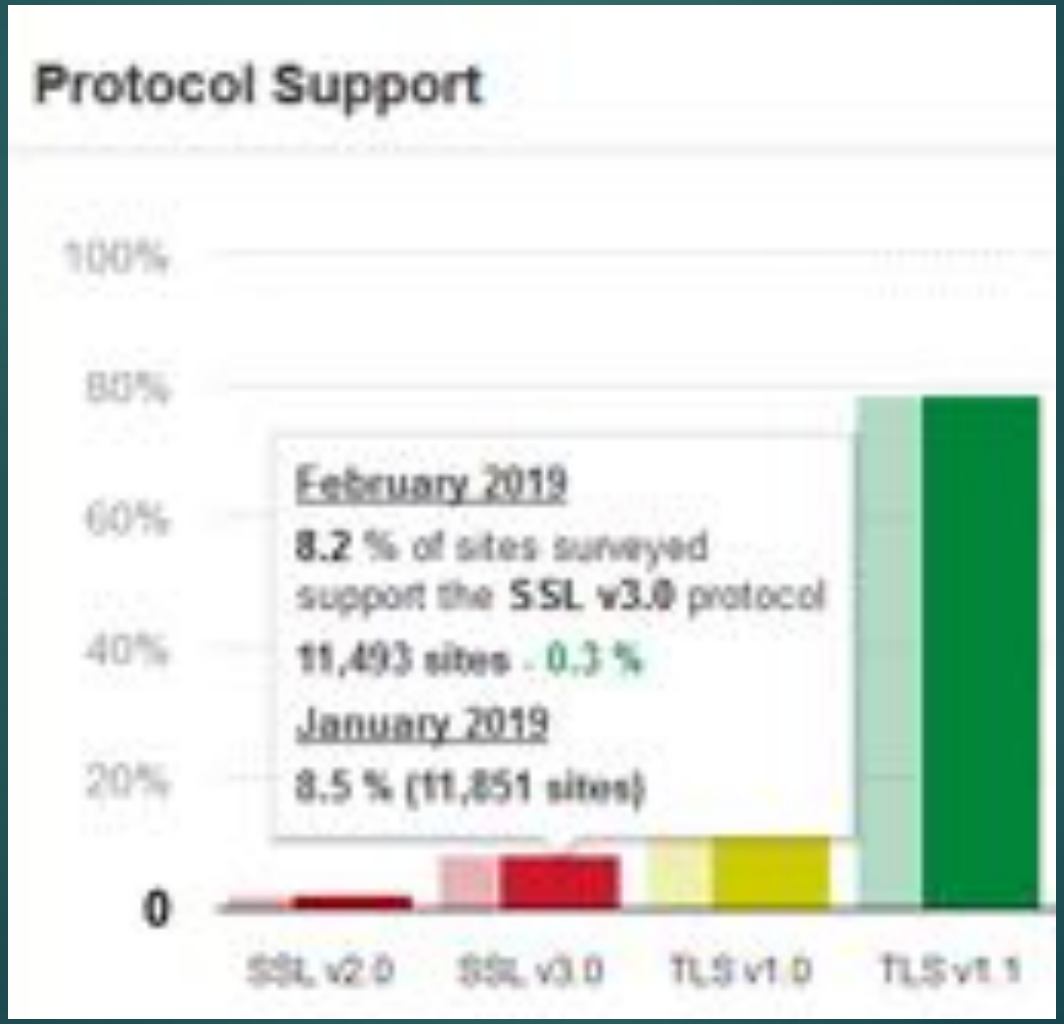
TLS: a History

TLS – a History

SSL: Secure Sockets Layer (Netscape)

- SSL 2.0:
  - Released in 1995.
  - Security flaws.
  - **Deprecated** in 2011, RFC 6176.
- SSL 3.0:
  - Released in 1996, RFC 6101 (historic).
  - **Not** interoperable with TLS.
  - In 2014, POODLE attack on block ciphers.
  - **Deprecated** in 2015, RFC 7568.

TLS – a History



Capture from https://www.ssllabs.com/ssl-pulse/ - taken Feb 19 2019.

TLS – a History

TLS: Transport Layer Security (IETF)

- TLS 1.0:
  - Defined in Jan 1999, RFC 2246.
  - SSL 3.0 with minor changes – **not interoperable**.
  - Includes downgrade mechanism to SSL 3.0.
  - Default: DSA/3DES

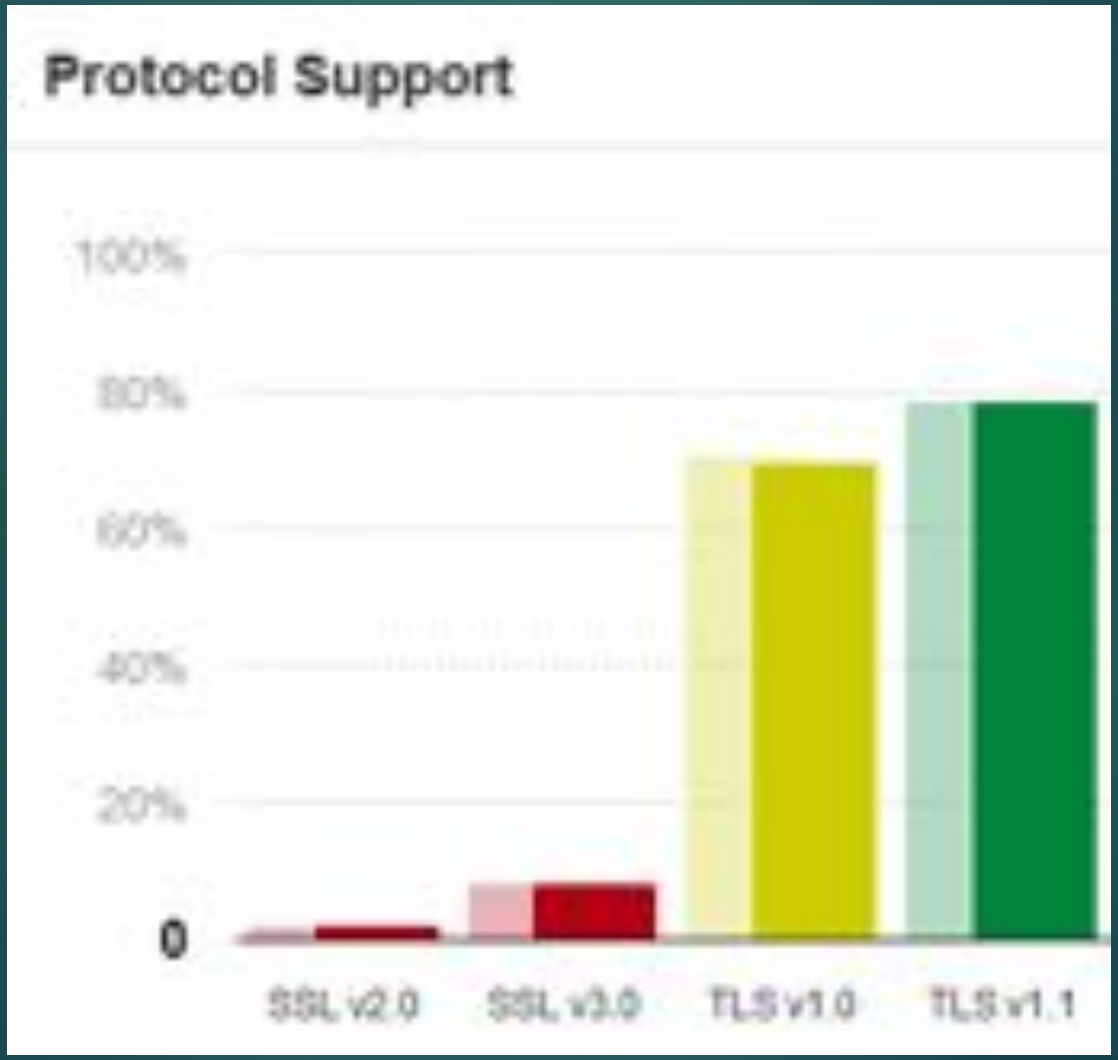
TLS – a History

TLS: Transport Layer Security (IETF)

- TLS 1.1:
  - Defined in Apr 2006, RFC 4346.
  - TLS 1.0 with fixes:
    - Padding oracles removed.
    - Explicit Initialisation Vector to prevent CBC attacks.
  - Default: RSA/3DES

October 2018: Apple, Google, Microsoft and Mozilla will deprecate TLS 1.0 **and** 1.1 by March 2020

TLS – a History



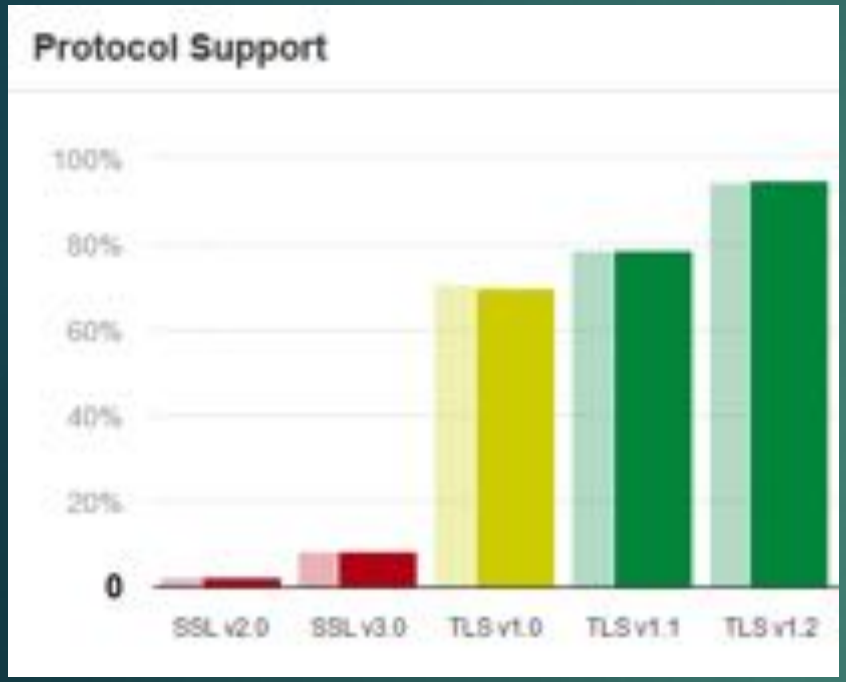
Capture from https://www.ssllabs.com/ssl-pulse/ - taken Feb 19 2019.

TLS – a History

TLS: Transport Layer Security (IETF)

- TLS 1.2:
  - Defined in Aug 2008, RFC 5246.
  - MD5 and SHA-1 → SHA256 (mostly).
  - Add AES cipher-suites (but still supports RC4!).
  - Add AEAD: GCM and CCM with AES.
  - Many options for suite-specified PRFs and Hashes.
  - Currently 314 cipher-suites!
- A lot of flexibility → many ways to get it wrong.

Current deployment



94.7% of websites support TLS 1.2.

Only 65.8% securely configured.

Capture from https://www.ssllabs.com/ssl-pulse/ - taken Feb 19 2019.

Transport Layer Security v1.2

IS IT THE END?

TLS Overview [Stebila'14]

RSA, DSA, ECDSA

DH, EC-DH

HMAC

MD5, SHA-1, SHA-2

DES, 3DES, RC4, AES

Data structures

Key derivation

Encryption modes and IVs

Padding

Compression

Alerts and errors

Certification/ revocation

(re)Negotiation

Session Resumption

Key reuse

OpenSSL

GnuTLS

SChannel

Java JSSE0

Web browsers

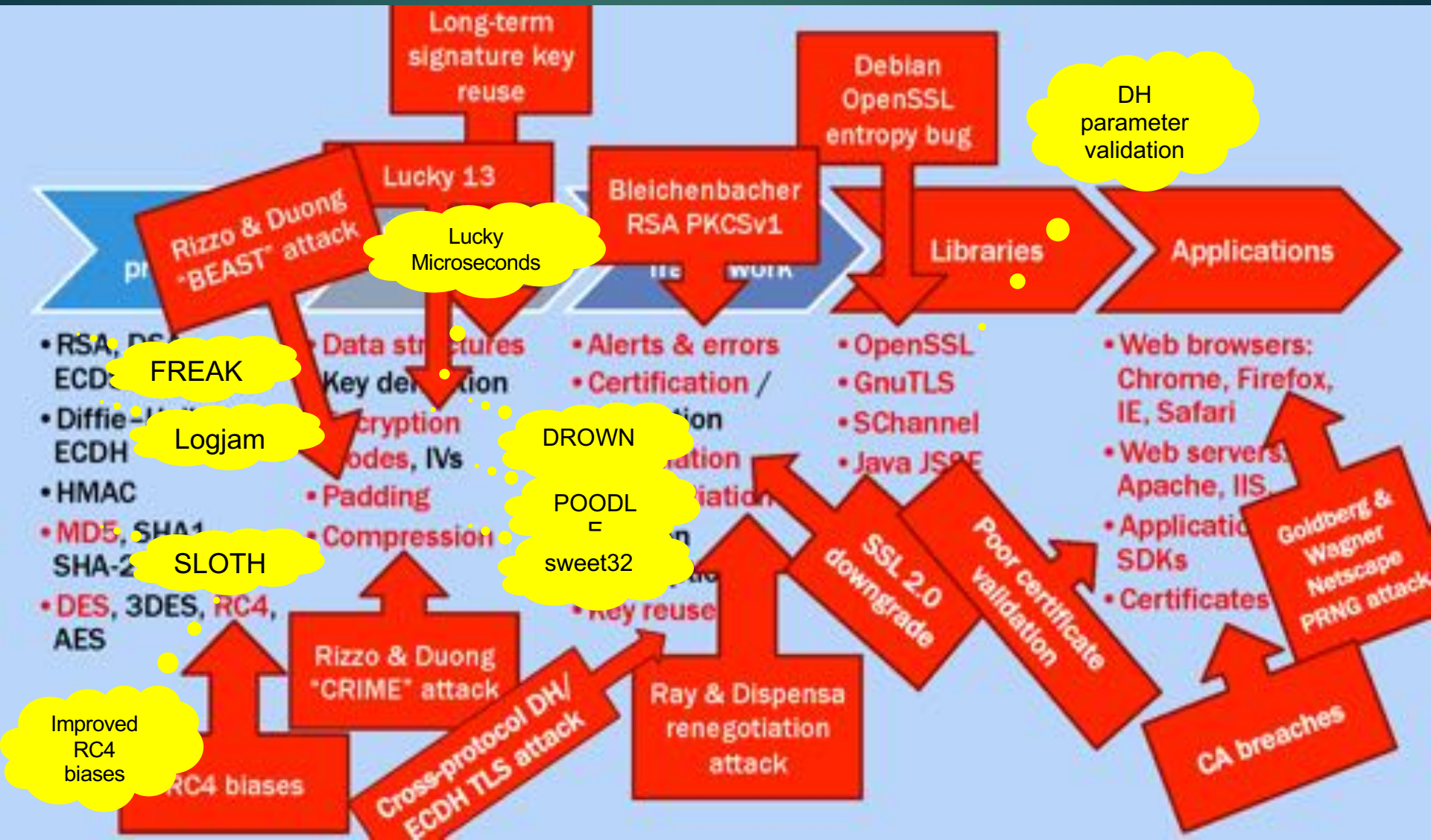
Web servers

Application SDKs

Certificates

TLS Overview [Stebila'14]

Updated November 2016



TLS Renegotiation Attack

[Marsh Ray Nov.09]

- Suite can be renegotiated.
  - Looks like negotiation.
- Person-In-The-Middle can inject (plaintext) traffic.
- Fix: TLS renegotiation indication extension.
  - Feb 2010, RFC 5746.
  - 84% deployment in Jan.'14.

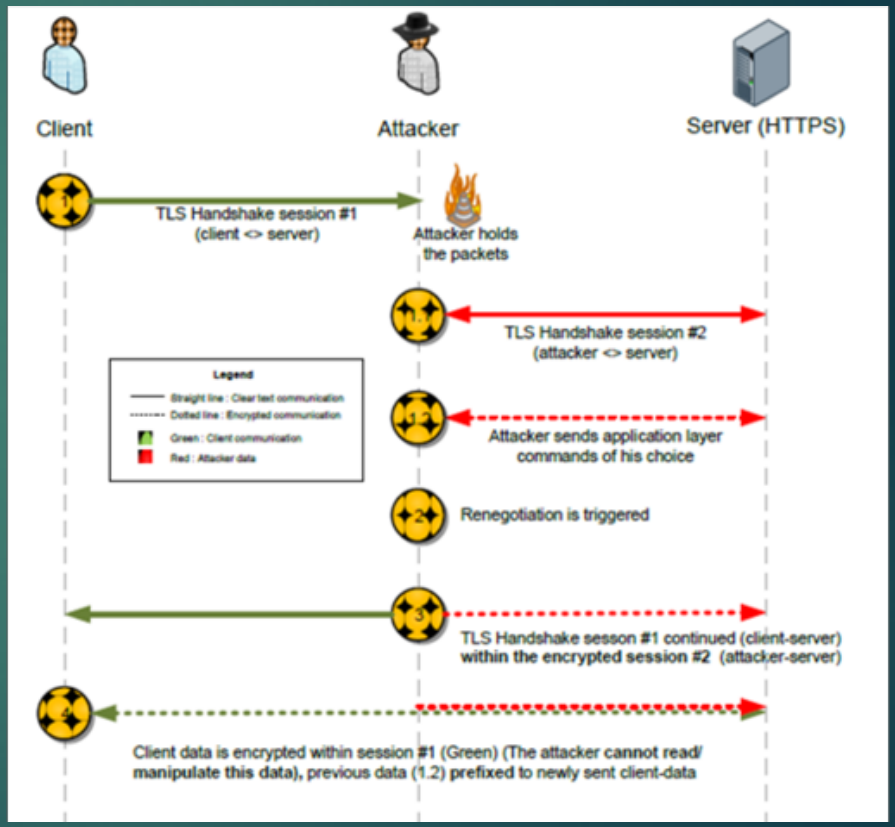


Figure: L. O'Connor

Transport Layer Security v1.3

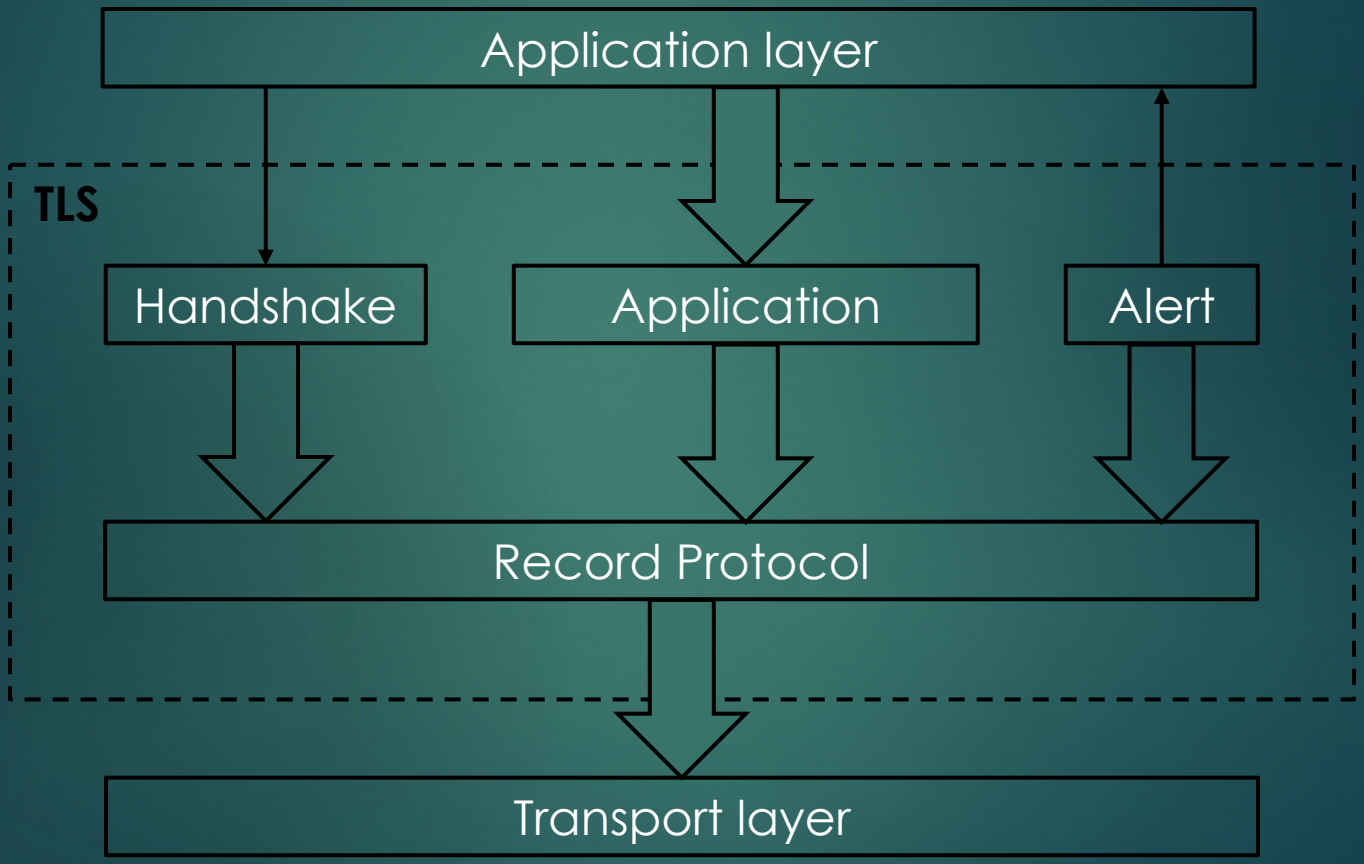
SECURITY FEATURES

TLS 1.3 (Aug 2018, RFC 8446)

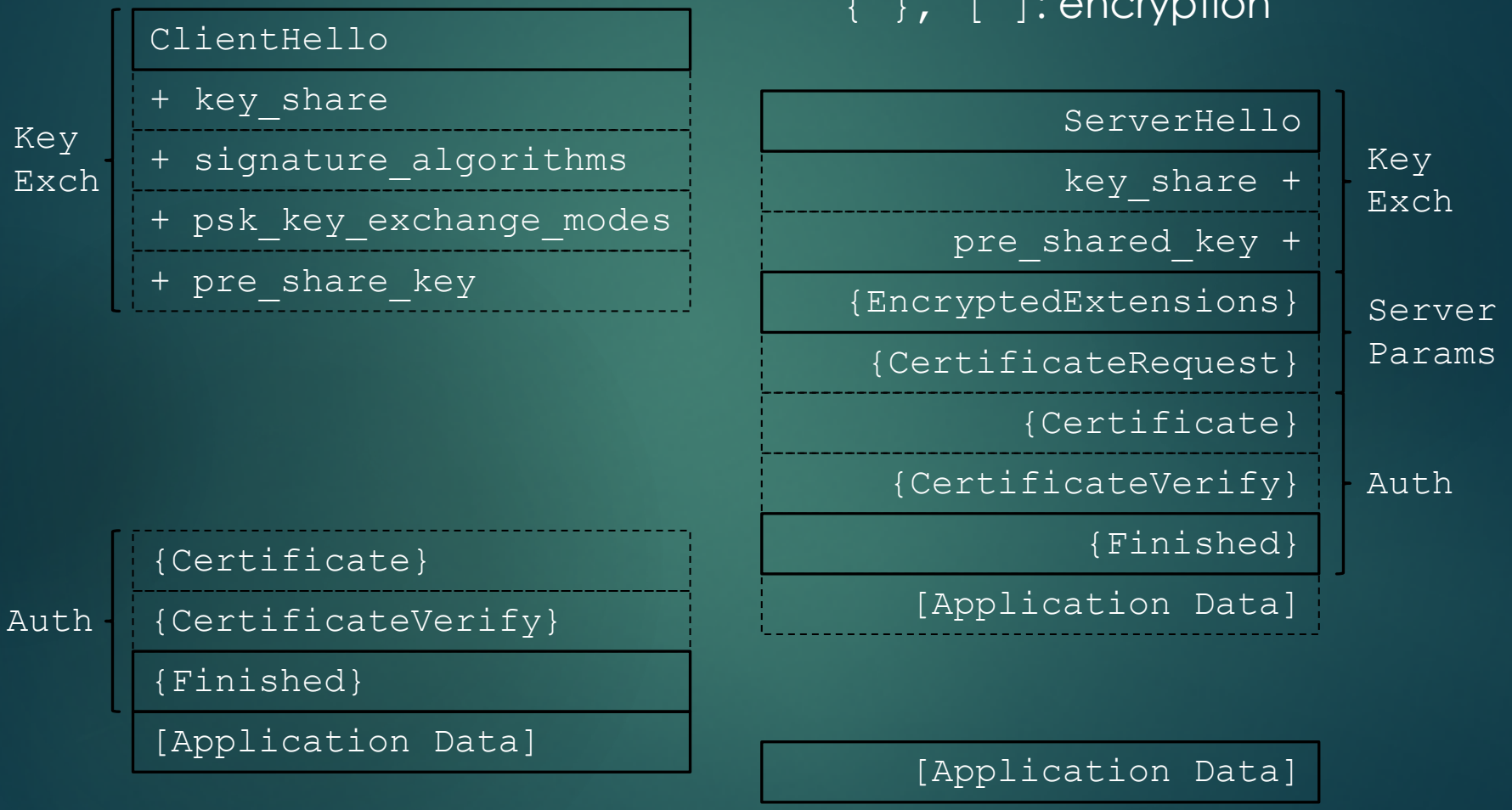
Major changes:

- **Only** AEAD symmetric algorithms allowed for encryption.
- **Changed** negotiation.
  - Separated: auth. & KE / record protection & hash function.
- **Deprecated** *static* RSA and DH key exchange (KE).
  - **Forward secrecy** for all *PK-based* KE.
- **Encrypted** handshake after `ServerHello` (incl. identities).
- **Redesign** of key derivation (based on `HKDF`).
  - Better formal analysis.
- **1-RTT handshake** and **0-RTT** option.

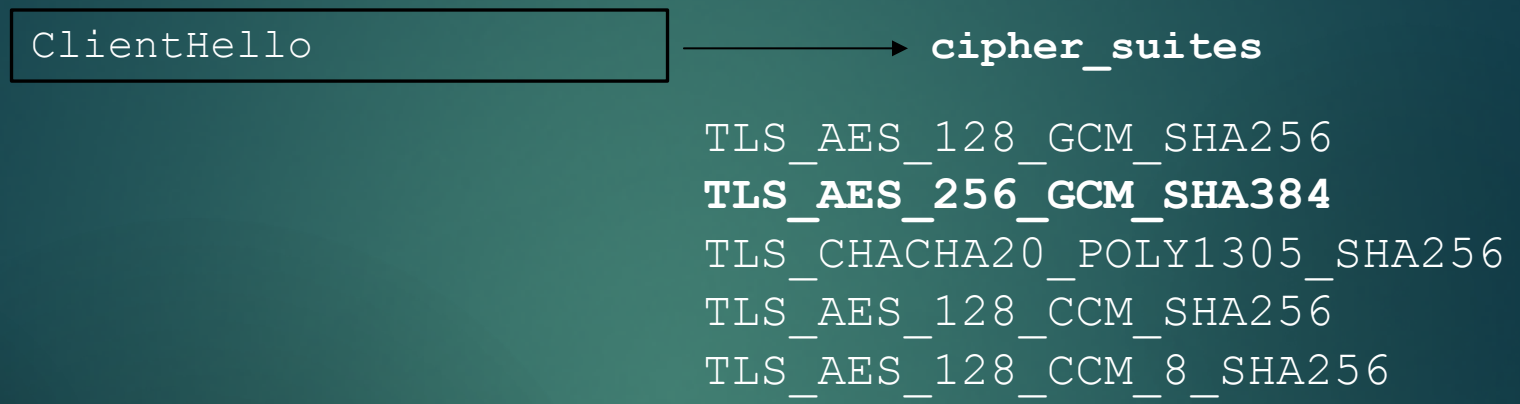
Protocol Structure



Handshake Protocol - AKE

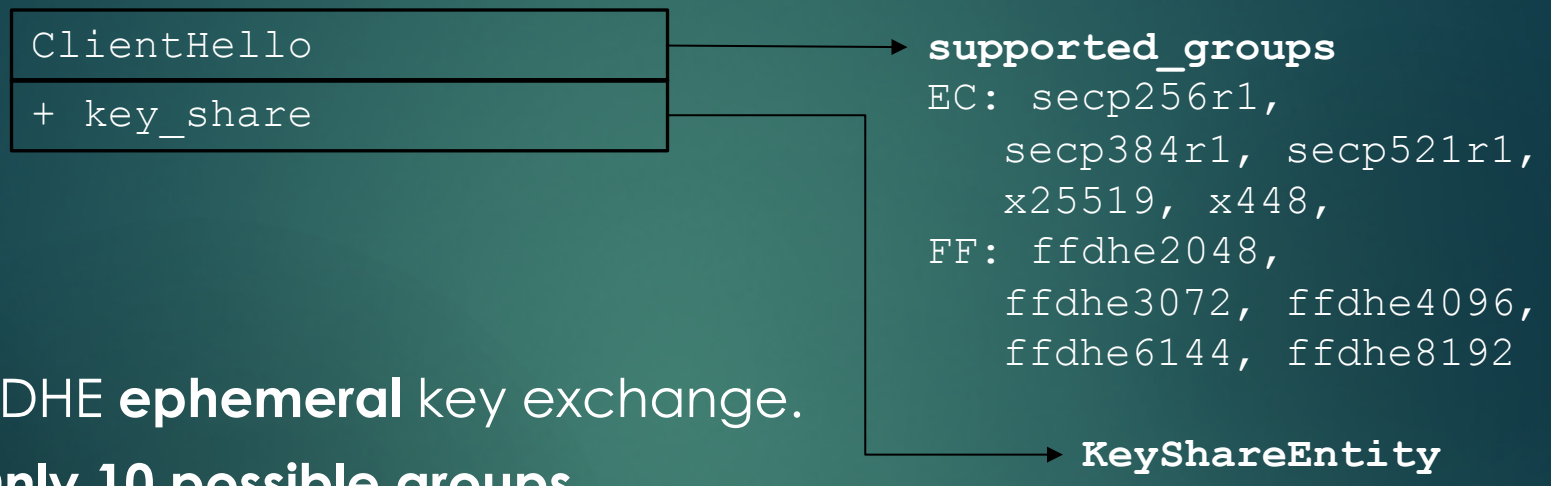


Cipher pairs negotiation



- **Only 5 possible pairs.** (At the time of presenting.)
  - **No weak crypto allowed** (unlike TLS 1.2)
- Provides **AEAD:**
  - authenticated encryption with associated data.
  - Application data **confidentiality and integrity**.
- Hash functions `SHA256, SHA384` used in `HKDF`.
- Cannot re-negotiate.

Ephemeral KE - Client

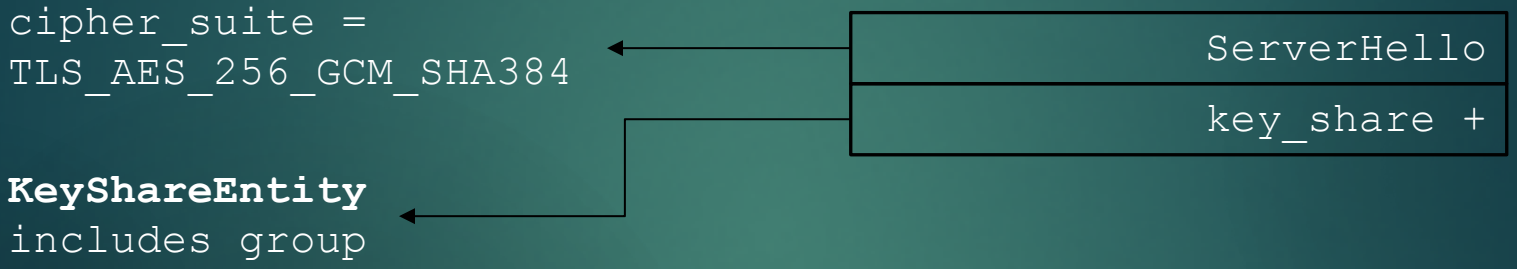


(EC)DHE **ephemeral** key exchange.

- **Only 10 possible groups.**
  - Over finite field or elliptic curves.
  - 10 x 5 = **50** possible suites. (TLS 1.2 has **314**).
- **Mandatory**.
- Provides **forward secrecy**.
  - If authentication keys are compromised later, then session remains secret if session-key was erased.

Ephemeral KE - Server

- Chooses **one** cipher suite and **one** KE group.



- Strict conditions for acceptance and downgrade.
  - Possible `HelloRetryRequest`.
  - Many `abort` scenarios.
  - Prevents many nasty situations.

Server Authentication (one-way authentication)

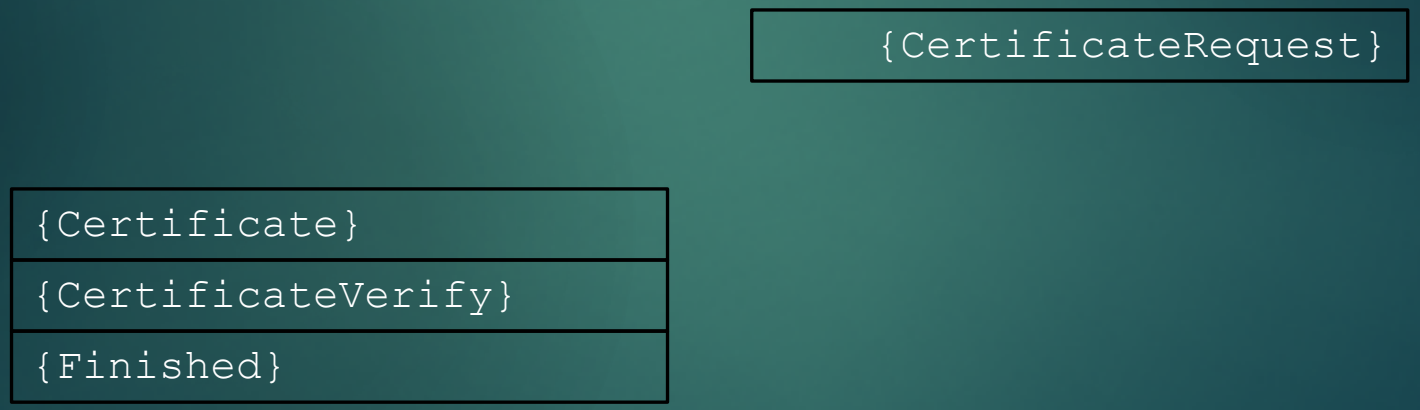
- Server identity is **encrypted**.
- `CertificateVerify = sign(hash(handshake||certificate)`
  - **Authentication.**
  - **Handshake integrity.**

{Certificate}
{CertificateVerify}
{Finished}

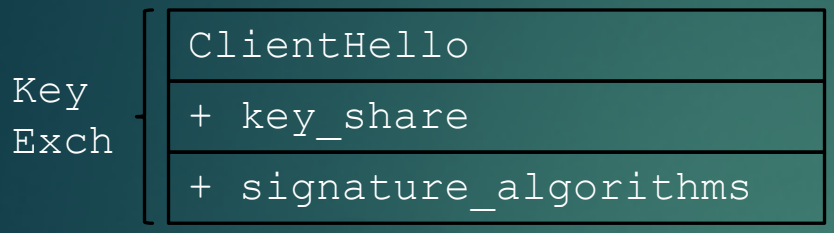
- `Finished` contains HMAC over `hash(handshake||certificate||CertificateVerify)`
  - **Key confirmation**.

Client Authentication (mutual authentication)

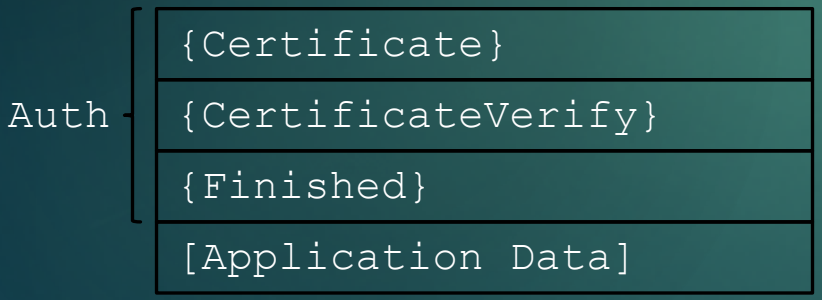
- **Optional**, triggered by `CertificateRequest`.
  - `signature_algorithms` sent as extension.
- `CertificateVerify` and `Finished` provide further **integrity** and **key confirmation**.

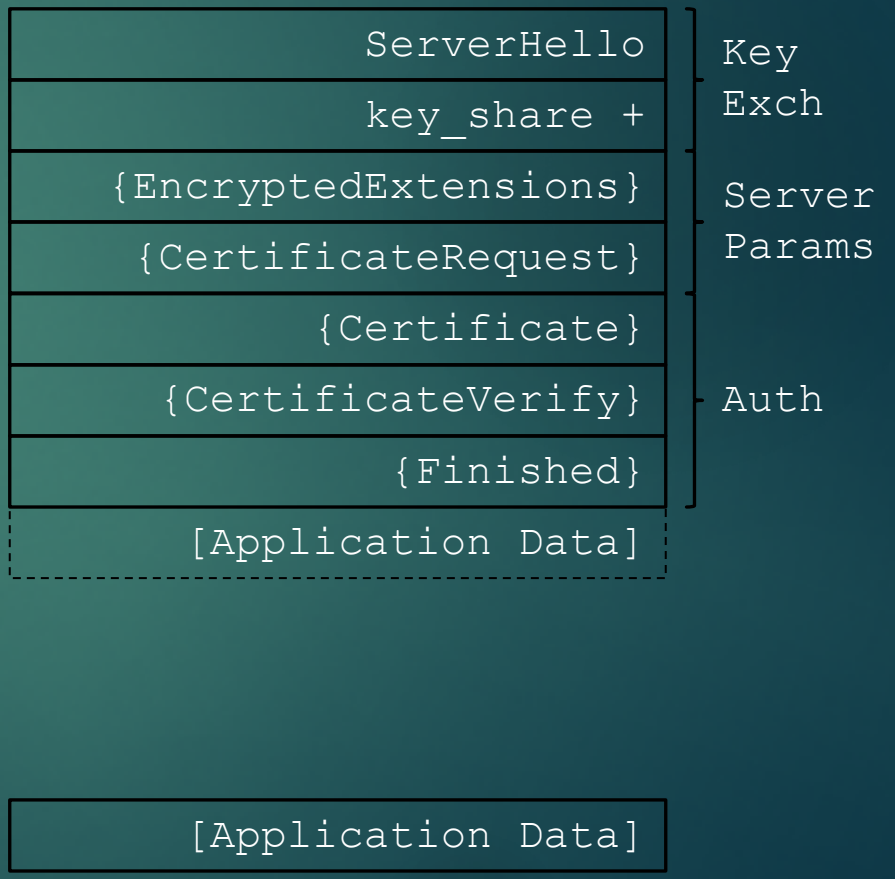


Handshake Protocol – AKE

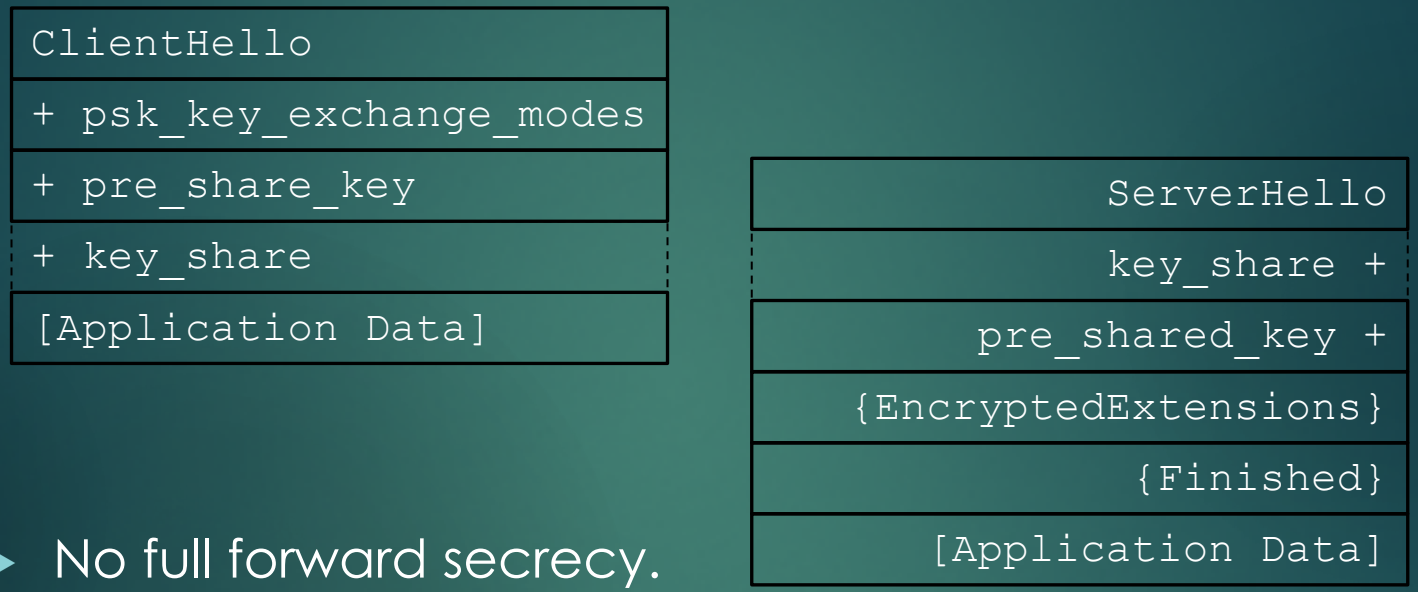


- Forward secrecy
- Key compromise impersonation (KCI) resistance
- 1-RTT, better than TLS 1.2.





0-RTT Handshake



- No full forward secrecy.
- Server cannot guarantee uniqueness without keeping huge log.

TLS: security guarantees

TLS 1.3 provides:

- Data confidentiality with forward secrecy.
- Identity privacy.
- Entity authentication.
- Downgrade protection.
- Data integrity.
- 1-RTT latency (and 0-RTT with security trade-off).

Comparison with TLS 1.2

Comparison with TLS 1.2

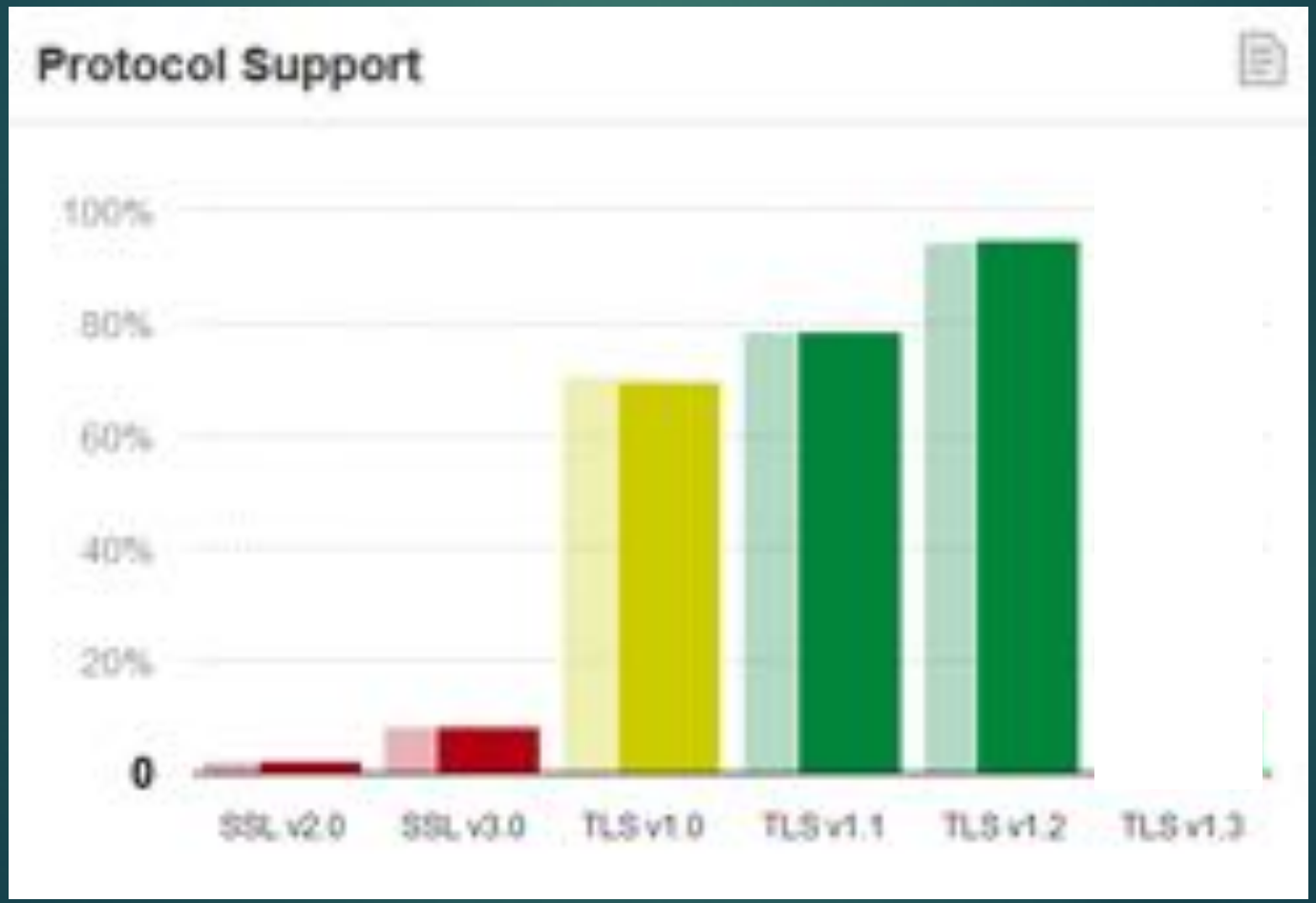
TLS 1.3 …

- **Mandates** strong crypto
- **Mandates** strong suites
- **Mandates** ephemeral KE
  - FS by default
- **Mandates** server auth.
- **1-RTT** by default
  - Possible **0-RTT**
- **Prevents** downgrade
- **Prevents** renegotiation

TLS 1.2 …

- Allows **weak** crypto
- Allows **weak** suites
- Allows **static** KE
  - FS by choice
- Allows **anon.** auth.
- **2-RTT** by default
- **Allows** downgrade
- **Allows** renegotiation

Deployment status (after ~6 months)



Capture from https://www.ssllabs.com/ssl-pulse/ - taken Feb 21 2019.

Choose **security**, choose **TLS 1.3.**

List of librairies:
https://github.com/tlswg/tls13-spec/wiki/Implementations
OpenSSL, GnuTLS, etc.

any questions?