Building Secure Channels

Kenny Paterson

Information Security Group

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Overview

- Why do we need secure channels?
- What properties should they have?
- Literature on secure channels
- Extended Example: TLS

Why do we need secure channels?

- Secure communications is the most common real-world application of cryptography today.
    - No, it's not MPC for sugar beet auctions!
- Secure channels are extremely widely-deployed in practice:
    - SSL/TLS, DTLS, IPsec, SSH, OpenVPN,…
    - WEP/WPA/WPA2
    - GSM/UMTS/LTE
    - Cryptocat, OTR, SilentCircle,…
    - (QUIC, MinimalT, TCPcrypt)
    - Mostly, but not exclusively in the 2-party case.

What security properties should they have?

- **Confidentiality** – privacy for data
- **Integrity** – detection of data modification
- **Authenticity** – assurance concerning the source of data
- All in the face of active attackers who can modify, delete, inject, re-order network messages.
- These three properties are relatively easy to achieve using Authenticated Encryption (AE) or AEAD.
    - Recall AE notion from Monday.
    - AEAD: some data encrypted and integrity protected, other data (the header) only integrity protected.
    - But AE/AEAD were not available at the time many of today's systems were designed.
- Note that authenticated key establishment (AKE) is out-of-scope for this talk.
    - We assume the keys are already in place.
    - A major assumption, but a different summer school!

What security properties should they have?

Less obvious security properties:

- Anti-replay – detection that messages have been repeated.
- Drop-detection – detection that messages have been deleted by the adversary or dropped by the network.
    - Particularly acute for the last message on a channel.
    - Cookie cutter attack.
- Prevention of re-ordering attacks.
    - Preserving the relative order of messages in *each* direction.
- Prevention of re-ordering attacks against duplex communications.
    - Preserving the relative order of messages in *both* directions.
- Prevention of traffic-analysis.
    - Using traffic padding and length-hiding techniques.

- None of these properties are met by using raw AE/AEAD!

What additional properties should they have?

- Fast, low memory requirements, on-line/parallelisable crypto-operations.
    - Performance is heavily hardware-dependent.
    - May have different algorithms for different platforms.
- IPR-friendly.
    - This issue has slowed down adoption of many otherwise good AE algorithms, e.g. OCB.
- Easy to implement in a side-channel-free manner.
    - Rules out many candidates!

What additional properties should they have?

- Clean and well-defined interface for applications.
- Related questions:
    - Does the channel provide a stream-based functionality or a message-oriented functionality?
    - Does the channel accept messages of arbitrary length and perform its own fragmentation and reassembly, or is there a maximum message length?
    - How is error handling performed? Is a single error fatal, leading to tear-down of channel, or is the channel tolerant of errors?
    - Does the secure channel itself handle retransmissions? Or is this left to the application? Or is it guaranteed by the underlying network transport?
    - Does the channel offer data compression?
- These are design choices that can have a substantial impact on security.
- They are not well-reflected in security definitions for AE/AEAD.

How do we build secure channels?

- Basic messages so far:
    - We can start with AE/AEAD, but must recognise that it's only a starting point.
    - There are many other issues that arise in designing a practical secure channel protocol.
    - And many design choices to be made.
    - Which themselves can have security consequences.
    - Don't try this at home!

What does the literature tell us?

- Shoup (http://shoup.net/papers/skey.pdf, 1999):
  - 2 pages on secure sessions in a 50 page+ paper on key exchange.
  - Simulation-based rather than game-based indistinguishability notions.
  - "It should be simple to fill in the details…"
- Canetti (eprint 2000/067):
  - The Universal Composability framework.
  - Heavy use of *ideal* secure channels.
  - *Impractical* construction of secure channels via one-time use of public keys and ideal authenticated channels.
  - Needs non-committing encryption to achieve UC against adaptive corruptions.
- Canetti-Krawczyk (eprint 2001/040):
  - Basic definition for secure channels using game-based, indistinguishability notion.
  - Construction via "EtM".

What does the literature tell us?

- Canetti-Krawczyk (eprint 2002/059):
  - UC notion for secure channels, realization using EtM.
- Bellare-Kohno-Namprempre (CCS'02):
  - Game-based stateful security notions for AE.
  - Capturing reordering and dropping attacks in addition to the usual CIA attacks.
- Kohno-Palacio-Black (eprint 2003/177):
  - Explicit consideration of reordering, replay, packet drop issues in game-based setting.
  - Different models allowing/denying different combinations of features.

What does the literature tell us?

- Maurer-Tackmann (CCS'10)
  - Secure channels in the "constructive cryptography" framework.
- P.-Ristenpart-Shrimpton (Asiacrypt'11)
  - LH-AEAD notion.
  - Incorporating basic length-hiding into AEAD notions.
- Jager-Kohlar-Shäge-Schwenk (Crypto'12)
  - ACCE: secure key establishment and channel definition built on LH-AEAD + key exchange.
  - Monolithic and hard to work with, but justified for analysing TLS.
  - Used in Krawczyk-P.-Wee (Crypto'13) to analyse many TLS ciphersuites.

Summary of the literature

- Lots of literature on AE/AEAD.
- Much less on the more complex secure channel primitive.
- Current models are far from capturing all of subtleties of secure channels as they are used in practice.
- There is a great research opportunity here!

Extended example: TLS

SSL = Secure Sockets Layer.

Developed by Netscape in mid 1990s.

SSLv2 now deprecated; SSLv3 still widely supported.

TLS = Transport Layer Security.

IETF-standardised version of SSL.

TLS 1.0 = SSLv3 with minor tweaks, RFC 2246 (1999).

TLS 1.1 = TLS 1.0 + tweaks, RFC 4346 (2006).

TLS 1.2 = TLS 1.1 + more tweaks, RFC 5246 (2008).

TLS 1.3?

Importance of TLS

Originally for secure e-commerce, now used much more widely.

- Retail customer access to online banking facilities.
- User access to gmail, facebook, Yahoo.
- Mobile applications, including banking apps.
- Payment infrastructures.
- User-to-cloud.
- Post Snowden: back-end operations for google, yahoo, …

TLS has become the de facto secure protocol of choice.

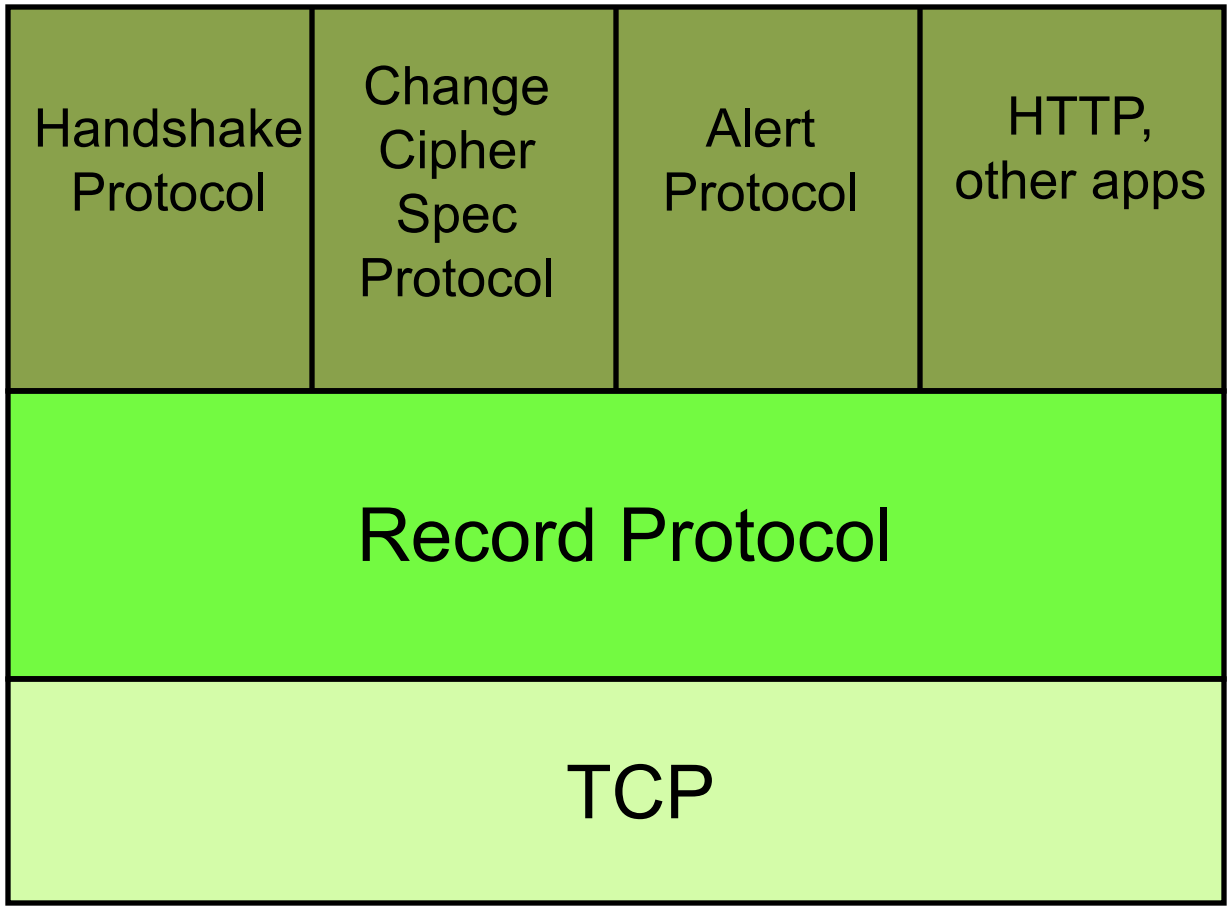
- Used by hundreds of millions of people and devices every day.
- A serious attack could be catastrophic, both in real terms and in terms of perception/confidence.

TLS is newsworthy!

TLS has been in the news…..

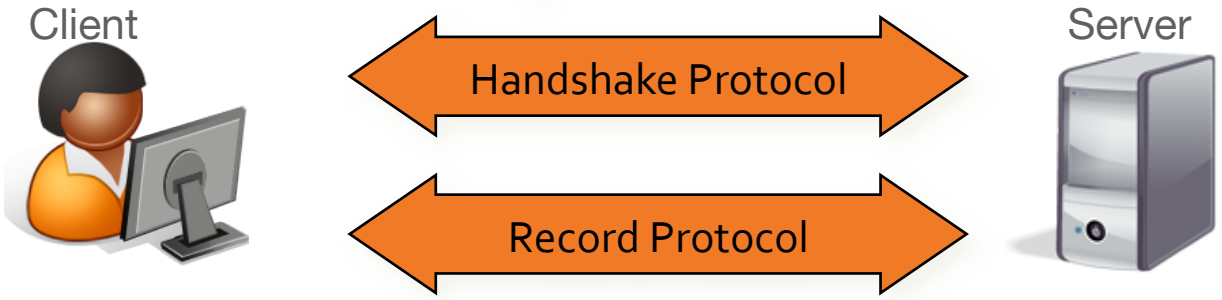
- BEAST, CRIME, Lucky 13, RC4 weaknesses.
- Renegotiation attack (2009), triple Handshake attack (3/2014).
- Poor quality of implementations (particularly in certificate handling).
  - Apple *goto fail.*
  - GnuTLS certificate handling fail.
  - "Why Eve and Mallory Love Android" and "The most dangerous code in the world".
  - And then Heartbleed…

TLS protocol architecture



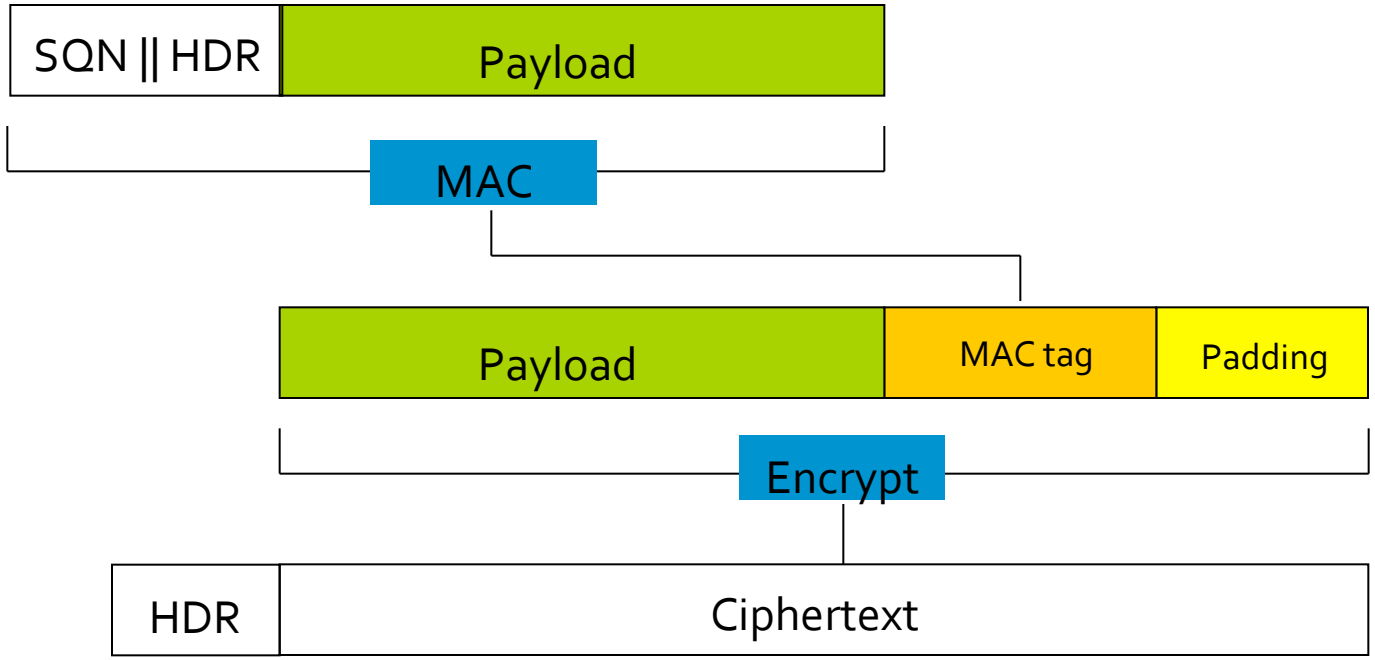
Simplified view of TLS

Used by client and server to
1.Negotiate ciphersuite
2.Authenticate
3.Establish keys used in the Record Protocol



Provides confidentiality and authenticity of application layer data using keys from Handshake Protocol – a secure channel.

TLS Record Protocol: MAC-Encode-Encrypt (MEE)



MAC — HMAC-MD5, HMAC-SHA1, HMAC-SHA256

Encrypt — CBC-AES128, CBC-AES256, CBC-3DES, RC4-128

AE and TLS Record Protocol

Dedicated AE algorithms are supported in TLS 1.2 in addition to MEE.

- Need not conform to MEE template
- AES-GCM specified in RFC 5288.
- AES-CCM specified in RFC 6655.

AE and TLS Record Protocol

- TLS 1.2 support in browsers:
  -  Chrome: since release 30.
  -  Firefox: since release 28.
  -  IE: since IE11.
  -  Safari: since iOS5 and OS X 10.9.

(source: wikipedia, Nov. 2013)

- Only 36% of Alexa top 200k servers support TLS 1.2.

(source: ssl pulse, May. 2014)

- Stronger, modern AE designs are not yet in universal use.
- 6 months ago, the situation was much bleaker.
  - Attacks have driven accelerating pace of adoption of TLS 1.2.

Operation of TLS Record Protocol

Out-bound processing:

- Data from application is received and partitioned into fragments (max size 2^{14} bytes).
- Optional data compression.
    - Default option is no compression.
- Calculate MAC on sequence number, header fields, and data, and append MAC to data.
- Pad (if needed by encryption mode), then encrypt.
- Prepend 5-byte header, containing:
    - Content type (1 byte, indicating content of record, e.g. handshake message, application message, etc),
    - SSL/TLS version (2 bytes),
    - Length of fragment (2 bytes).
- Submit to TCP.

Operation of TLS Record Protocol

In-bound processing reverses these steps:

- Receive Record Protocol message, of length specified in HDR.
- Decrypt.
- Remove padding (CBC-mode).
- Check MAC.
- (Decompress payload.)
- Pass payload to upper layer (no defragmentation).

TLS Record Protocol design decisions

- Stream-oriented.
    - Application layer is responsible for demarcating message boundaries if desired.
    - Fragmentation done by Record Protocol when sending, but defragmentation not done when receiving.
- Most errors are fatal.
    - TLS runs over TCP, which is assumed to provide reliable transport.
    - Hence any error arising during in-bound processing should be treated as an attack.
    - Session terminated with error message, keys thrown away.
    - So DoS attacks are trivial to mount.
    - No retransmission of lost messages by TLS itself.

TLS Record Protocol design decisions

- Implicit sequence numbers.
  - 8-byte SQN included in MAC calculation, but not sent on the wire as part of Record Protocol messages.
  - Sender and receiver are assumed to maintain local copies of SQN, incrementing for each message sent/received.
  - Any replay, re-ordering or dropping of messages should be detected through MAC verification failure at receiver.
  - MAC verification failure is fatal error.
- No attempt to hide message/fragment lengths.
  - Leads to fingerprinting attacks (e.g. Pironti-Strub-Bhargavan, INRIA research report 8067, 2012).
  - Can be partially addressed by use of variable length padding in CBC mode.

TLS Record Protocol design decisions

- Use of compression was known in theory to be dangerous.
  - Kelsey, FSE'04.
- Choice of MEE is *not* fully-supported by theory.
  - MtE known to be *not* generically secure (Bellare-Namprempre, Asiacrypt'01).
  - Krawczyk (Crypto'01) provides support for MtE when CBC-mode is used or when stream cipher is used.
  - But the analysis assumes:
    - Random per message IV, no padding, block-size = MAC tag size for CBC mode.
    - Stream cipher has outputs that are indistinguishable from random.
  - More recent analysis of Namprempre-Rogaway-Shrimpton-(Eurocrypt'14) says MtE provides AE if "E" is "tidy".
    - TLS's choice of "E" is not tidy!

TLS Record Protocol design decisions

- The fact is that suitable theory did not exist at the time TLS was designed.
    - Essentially, we need stateful AEAD security.
- Consensus then was that "MtE" is better than "EtM".
    - "Authenticate what you mean to say, not an encrypted version of it." – the Horton principle.
    - "Maybe our MAC algorithms are weak, so we should protect the MAC value by encrypting it."
- Today, we have better theory, but it's hard to get it deployed.
    - Because it has to displace what's already been massively deployed.
- This has had many interesting consequences for attacks.

Overview of TLS Record Protocol attacks

- **BEAST** (2011)– exploits TLS 1.0's use of predictable IVs.
- **CRIME** (2012) – exploits TLS's support for compression.
- **Padding oracle attack** (2002, 2003) – exploits TLS 1.0's use of distinguishable error messages for padding and MAC failures.
- **Lucky 13** (2013) – padding oracle attacks are still possible, even after application of recommended countermeasures; MEE with CBC is hard to implement without side channels.
- **RC4 attacks** (2013) – RC4 is not such a good stream cipher after all.

Current status

- CBC-mode ciphersuites can be patched against BEAST and Lucky 13, but their reputation has been damaged by a long series of attacks.
    - Relative performance also an issue (AES-CBC + HMAC quite slow).
- RC_4 pretty much dead.
- AES-GCM and AES-CCM are only available for TLS 1.2.

Current status – AES-GCM and AES-CCM

- AES-based ciphersuites are generally slow without AES-NI instruction.
- AES-GCM is tricky to implement securely.
    - One issue is avoiding leakage of hash key via side-channel attack.
    - Also need side-channel resistant implementation of AES.
- AES-GCM is relatively fast
    - Especially with AES-NI and PCLMULQDQ instructions (Intel and AMD).
    - 2.53 to 1.03 cycles per byte, depending on processor.

      http://2013.diac.cr.yp.to/slides/gueron.pdf
    - Roughly twice as fast as AES-CBC + HMAC-SHA-*
    - But OCB would be even faster!
- AES-CCM is relatively slow.
    - Two block cipher calls per block of data, similar to AES-CBC + HMAC-SHA-*.

Current and future developments

- Fresh algorithms are under active consideration in IETF TLS WG.
    - Important for environments where AES is not available in hardware.
    - Some momentum behind Salsa20/ChaCha20 stream ciphers plus Poly1305 MAC.
    - http://tools.ietf.org/html/draft-agl-tls-chacha20poly1305-04

- Reform of MEE to EtM to make CBC-mode easier to implement securely.
    - IETF draft by Gutmann exists and under review.
    - Deployment via TLS extension, unclear how widely adopted it will become.

Concluding remarks

- Secure channels are one of the most basic cryptographic applications.

- We do not have formal models for secure channels that accurately capture all the features expected by implementers.

- TLS as a case study highlights many of the real-world issues.
    - Design in the absence of good theory.
    - Legacy and slow adoption of better crypto.
    - Weak algorithms hard to remove.
    - Exploitation of novel network-based, side-channel attacks.