CS 578: Cyber-security

Part I: Internet protocols and ecosystem

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab

ANNOUNCEMENT

- TA office hours
  - Tu 11 am – 12 pm on Zoom (the link is available on Canvas)
- Call for actions
  - Homework 1 out
  - Term-project team-up
  - In-class presentation sign-up
    - May not have open-slots for yours if you are late
    - No exceptions for this case; you will lose 10%

ANNOUNCEMENT

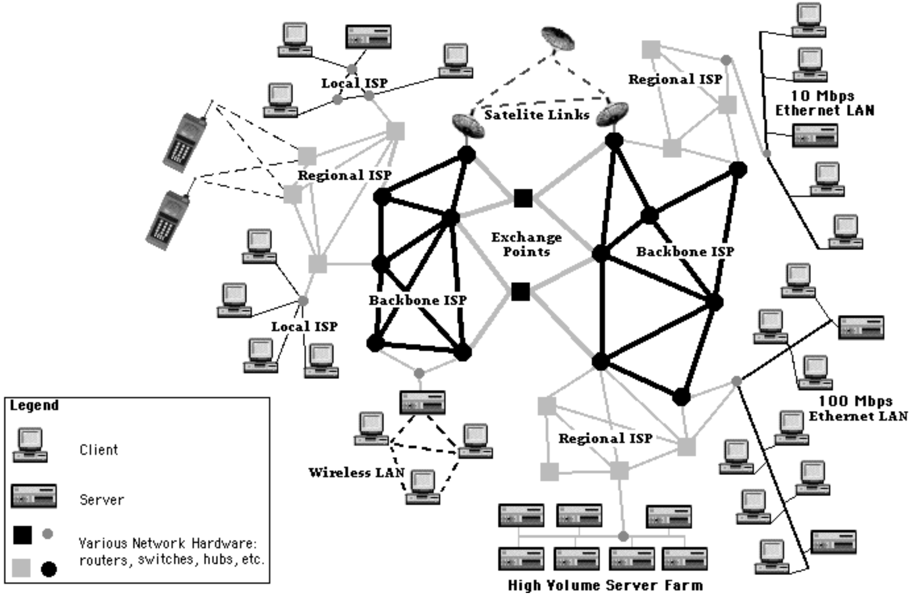
- TA office hours
  - Tu 11 am – 12 pm on Zoom (the link is available on Canvas)
- Call for actions
  - Homework 1 out
  - Term-project team-up
  - In-class presentation sign-up
    - May not have open-slots for yours if you are late
    - No exceptions for this case; you will lose 10%
  - Note on paper critiques
    - It is not a pleasant reading (2.5 hours of focused reading)
    - Avoid generic comments, e.g.,
      - "Good figures"
      - "Awesome evaluation"
      - "The paper is difficult-to-follow"

PRIMER ON THE INTERNET INFRASTRUCTURE

THE INTERNET

- The Net
  - A system of computer networks; a network of networks
  - Uses the Internet protocol suite (TCP/IP) to communicate

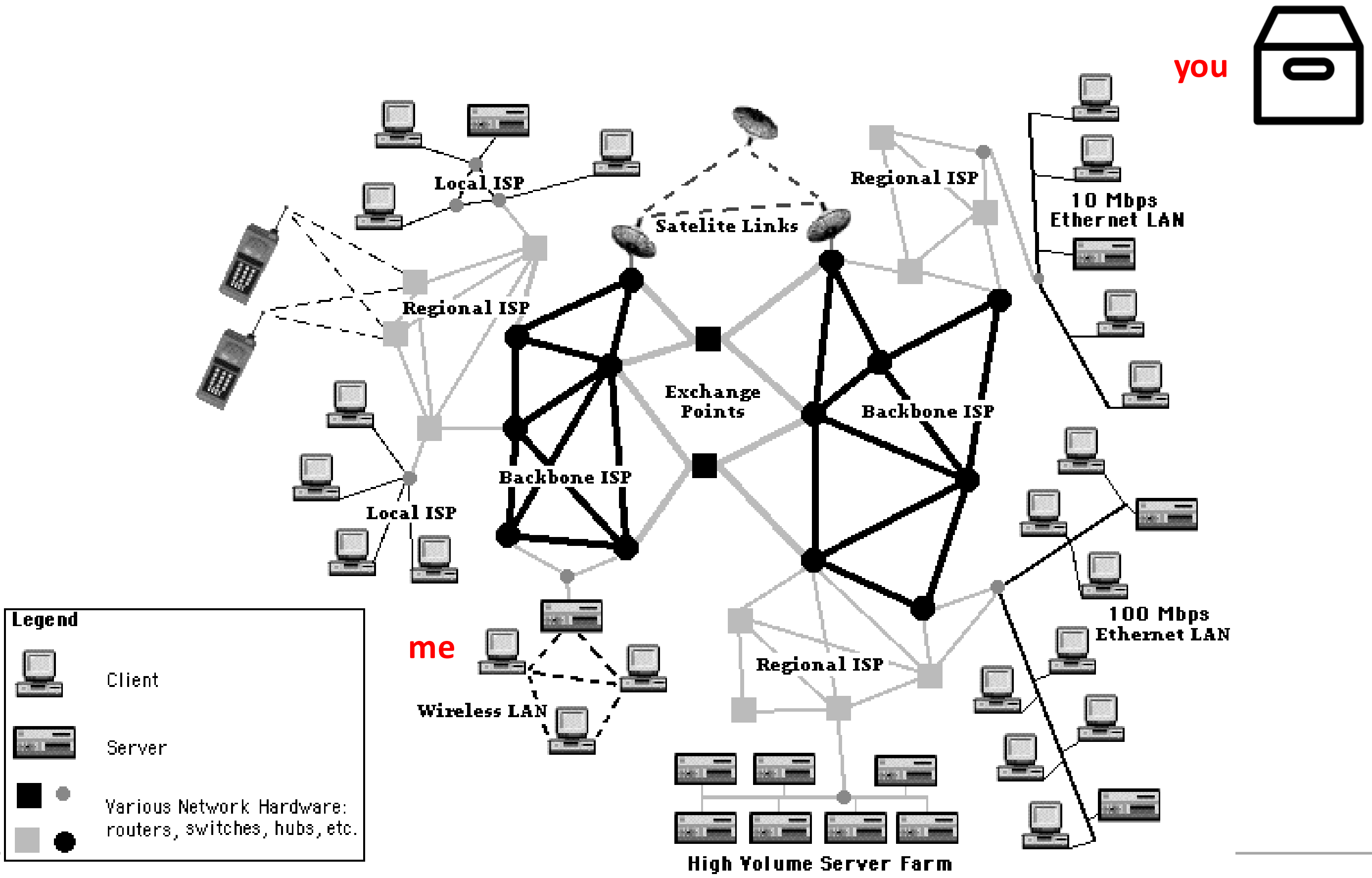
- Design principle
  - Network is complex, $O(N^2)$
  - Manage small network, $O(n^2)$
  - Manage network of networks $O(m^2)$
  - N >>>>> m,n
  - Make it simple!



[1]https://www.cs.utexas.edu/~mitra/csFall2018/cs329/lectures/fig1.gif

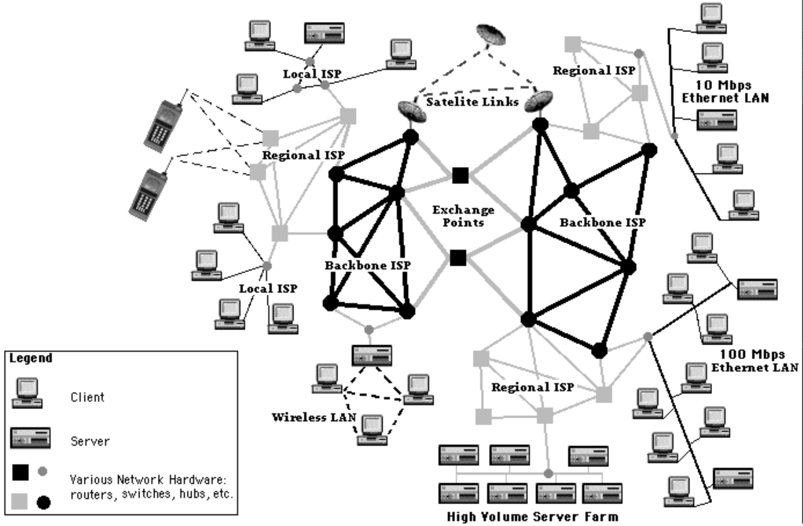
Oregon State University

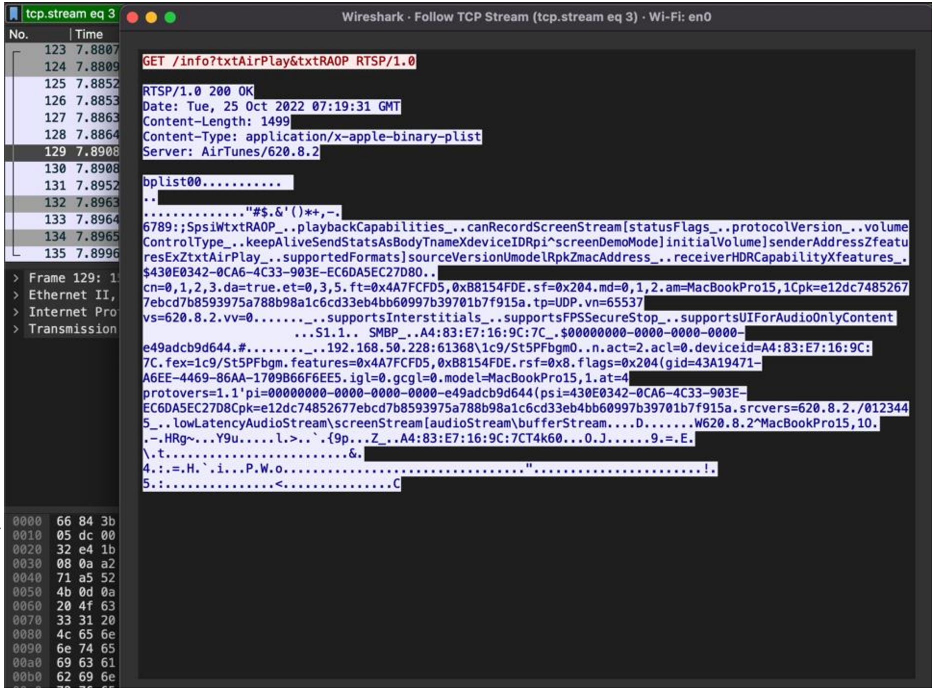
THE INTERNET: PACKET ROUTING



THE INTERNET: (NO) SECURITY

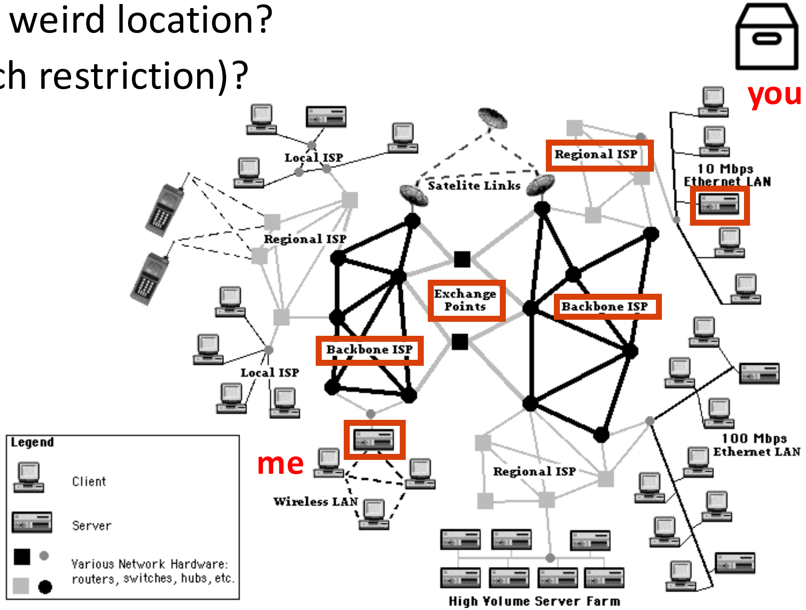
- No security (in TCP communication)
  - Any router in the middle can see any packet content :(





THE INTERNET: (NO) SECURITY

- Routers:
  - Decide where the packet should go as a next step
  - What if
    - the router in the middle sends a packet to weird location?
    - the router(s) are malicious (there is no such restriction)?



We Cannot Establish Trust in Routers

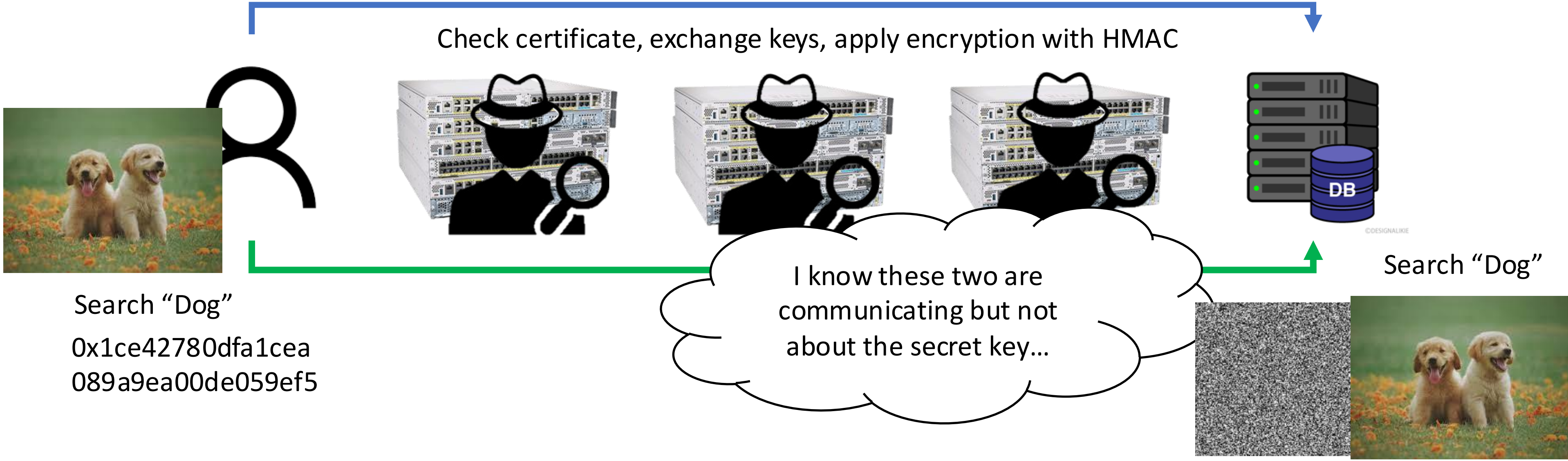
The Internet without security

Search “Dog”

Everybody in the Middle Knows That I Searched ‘dogs’ and They Also Know the Search Result… Ugh…

THE INTERNET WITH A SECURE MECHANISM (SSL/TLS)

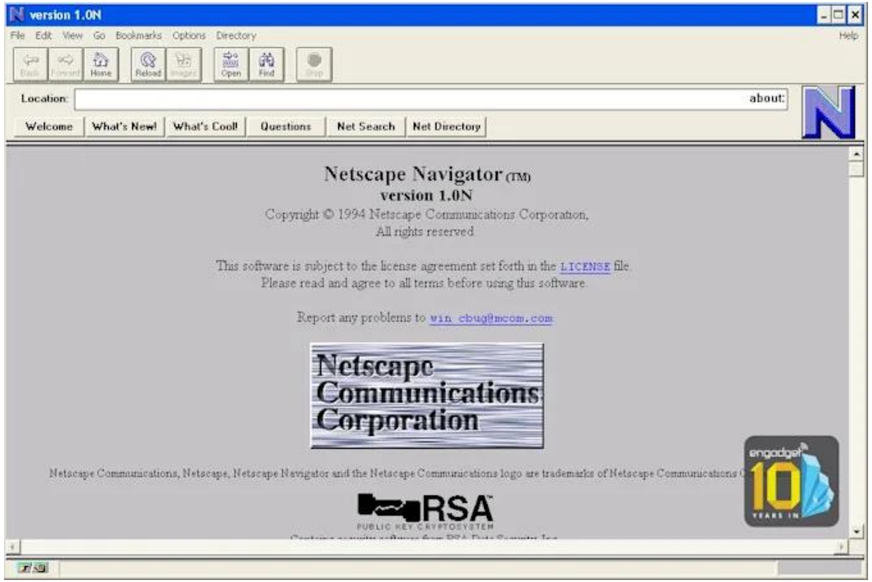
Middle mans never know DH exchange keys!!



The Middlemen Will Only See the Encrypted Contents
They Will Never Know the Secret Key …

SSL/TLS: SECURE SOCKET LAYER AND TRANSPORT LAYER SECURITY

- SSL/TLS
  - Developed by Netscape in 1995
  - Standardized by IETF as TLS
  - https://www.ietf.org/rfc/rfc2246.txt

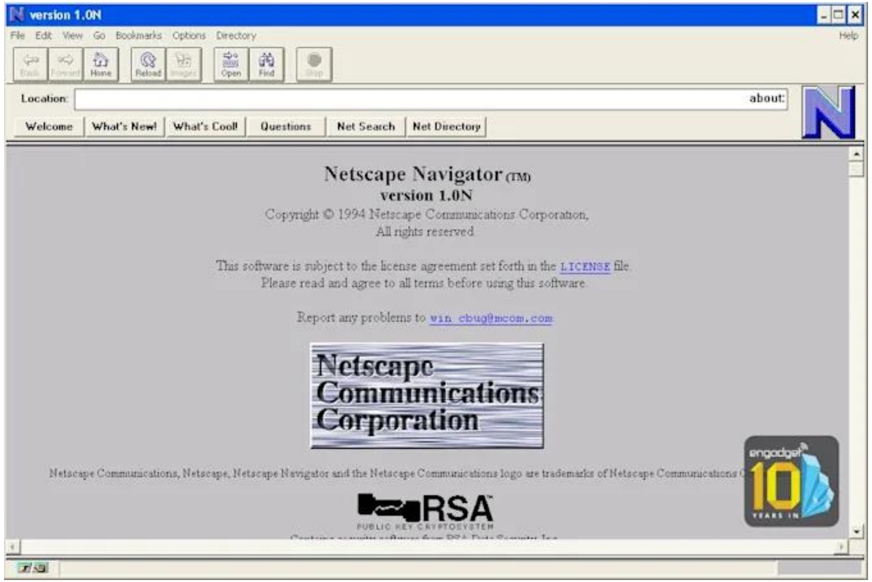


Oregon State University

SSL/TLS: SECURE SOCKET LAYER AND TRANSPORT LAYER SECURITY

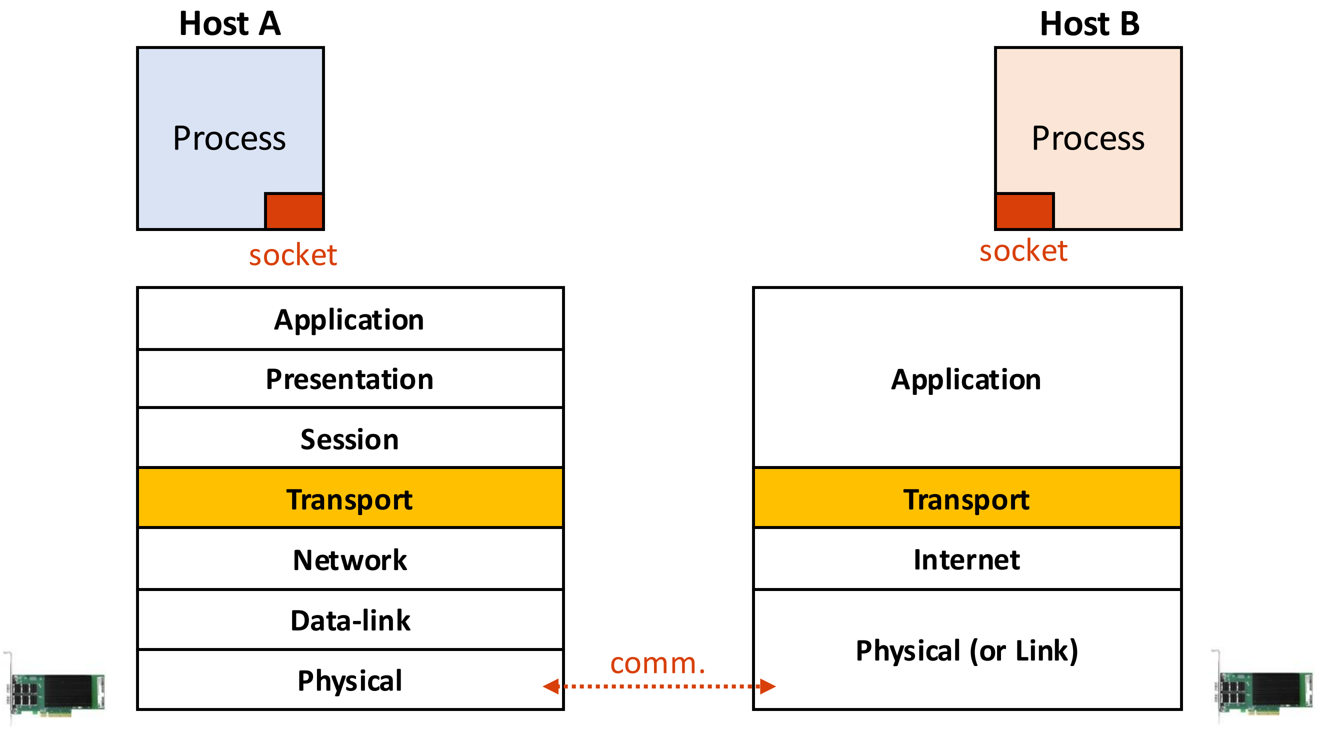
- SSL/TLS
  - Developed by Netscape in 1995
  - Standardized by IETF as TLS
  - https://www.ietf.org/rfc/rfc2246.txt

- "Transport Layer" Security
  - Why?



SSL/TLS: Transport layer security, why?

- Independent from the application running on a host



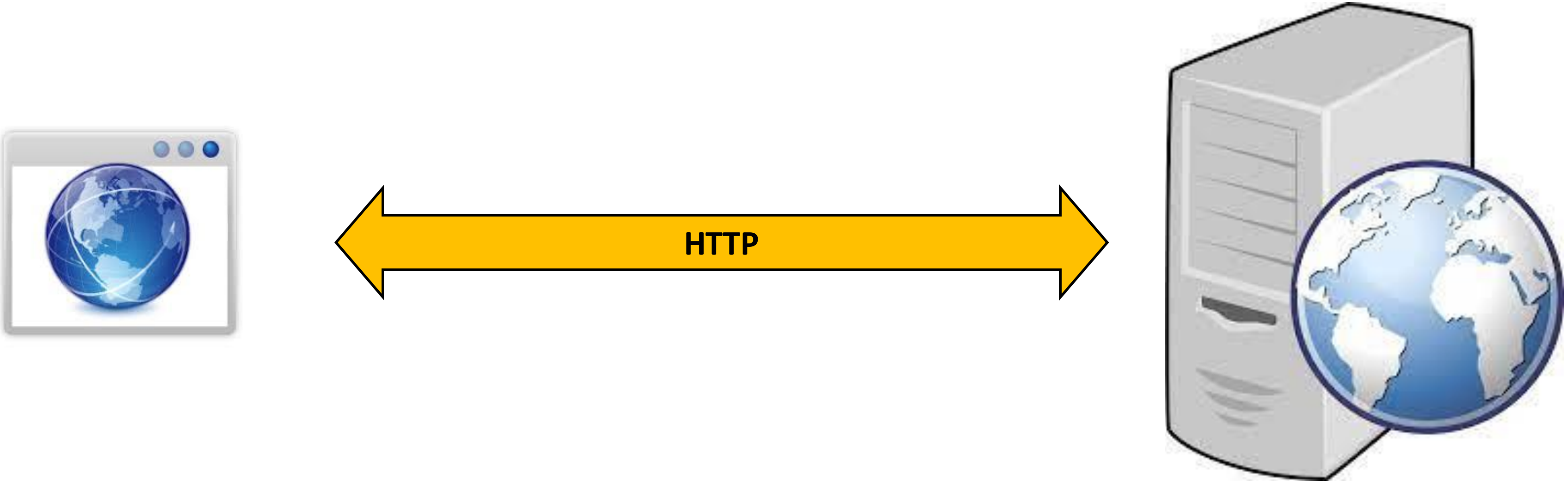
SSL/TLS: BENEFITS

- Enable to
  - Establish secure comm channels btw two ends (hosts) on the Internet
    - Client <-> Server (ex. OSU login)
    - Server <-> Server (ex. Amazon requests a transaction with your credit card)
    - Client <-> Client (ex. chat applications)
  - Verify the server entity
    - Use a digital certificate

- end-to-end secure communication channels
  - Authentication: a digital certificate
  - Key-exchange: Diffie-Hellman key exchange
  - Confidentiality: Block ciphers
  - Integrity: HMAC / MAC

HTTP: AN APPLICATION LAYER PROTOCOL

- Suppose we talk to a webserver



HTTP: An application layer protocol

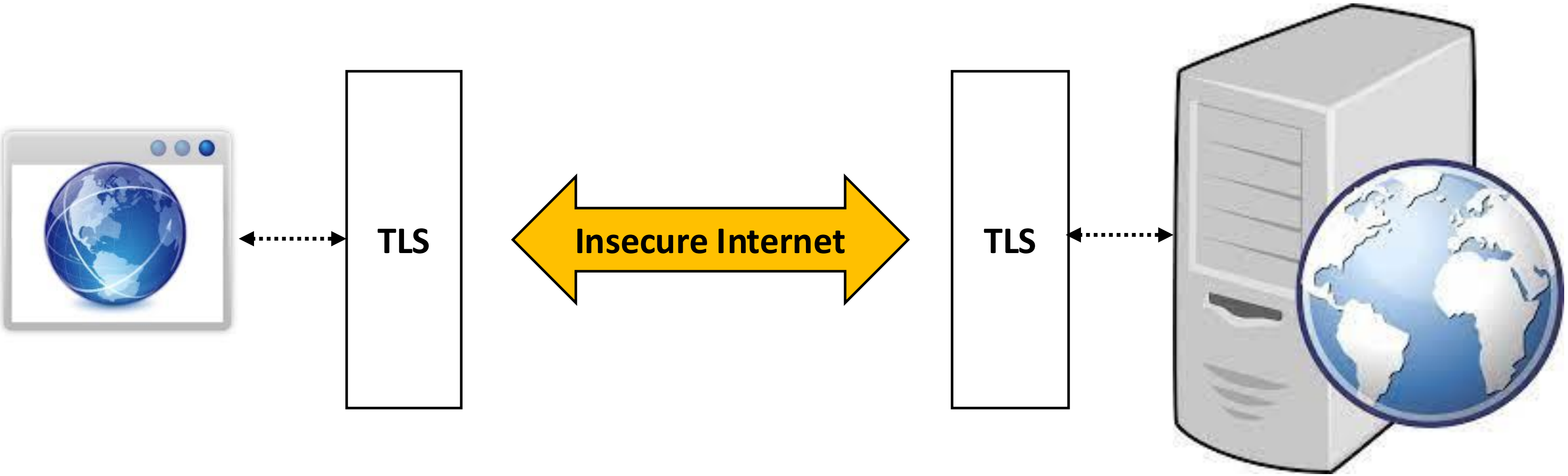
- Suppose we talk to a webserver


```
GET / HTTP/1.0
```

```
HTTP/1.0 200 OK
Date: Tue, 25 Oct 2022 12:53:12 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO
P3P: CP="This is not a P3P policy! S
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
```

HTTPS: An application layer (secure) protocol

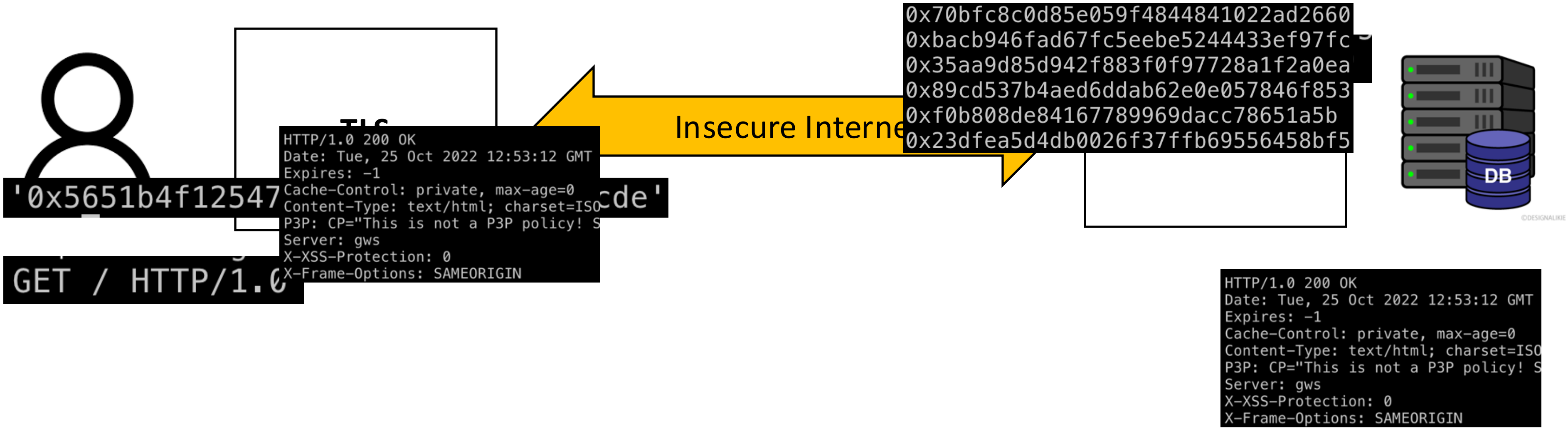
- Suppose we use HTTPs (instead of HTTP)



HTTPS: An application layer (secure) protocol

Run **TLS handshake** to establish a secure channel

HTTPS: AN APPLICATION LAYER (SECURE) PROTOCOL



LET'S SEE HOW HTTP PACKETS LOOK LIKE

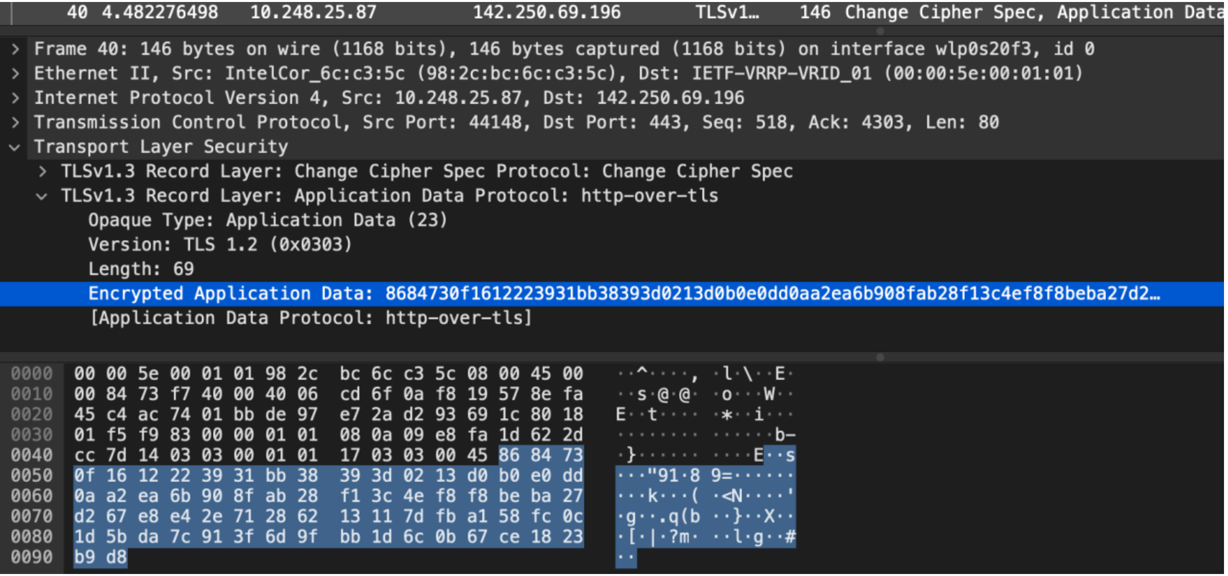
```
4 0.010756057   10.248.25.87        142.250.69.196      HTTP      144  GET / HTTP/1.1
```

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: curl/7.81.0
Accept: */*

HTTP/1.1 200 OK
Date: Tue, 25 Oct 2022 13:25:43 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2022-10-25-13; expires=Thu, 24-Nov-2022 13:25:43 GMT; path=/; domain=.google.com;
Secure
Set-Cookie: AEC=AakniGOAPvX70HdROvGjd5tdhzmMk-ZntDxb9jZGhAdPNSmqmwQc2AumlRI; expires=Sun, 23-Apr-2023
13:25:43 GMT; path=/; domain=.google.com; Secure; HttpOnly; SameSite=lax
Set-Cookie: NID=511=MkIeDpP817QKD-9oufZM9-
MAnHFlDpvagPc6jwK6l-2onKyCQID83aSymrg5ss1SUexUDpaSsNb9MrcxpnaXhezc9engEZrNmX4qgoG7Zodt4Fy-
HP9FQI6DbeY6GLGCma0MBOnUmze5m6Ys-i6jSvc6WFJUkye67iOSgFuG72c; expires=Wed, 26-Apr-2023 13:25:43 GMT;
path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

348f
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta
content="Search the world's information, including webpages, images, videos and more. Google has many
special features to help you find exactly what you're looking for." name="description"><meta
content="noodp" name="robots"><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta
content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><title>Google</
title><script nonce="zhUe3NfmQtn_Ha4HtJSi3A">(function(){window.google={kEI:'1-
NXY8iGIs2T0PEP5fy9CA',kEXPI:'0,18167,1284369,56873,6059,206,4804,2316,383,246,5,5367,1123753,1197698,703
,302561,77529,16114,19398,9286,22431,1361,284,12036,17579,4998,13228,3847,6885,3737,22741,5081,1594,1278
,2742,149,1943,1983,214,4100,109,3405,606,2023,1777,520,14670,605,2622,2845,7,4808,791,28171,1851,2614,1
2710,432,3,1590,1,5444,149,11323,2652,4,1528,2304,7039,22023,5708,7356,16639,16808,1435,5827,2530,4094,1
7,4035,3,3541,1,14263,27894,2,14019,2373,342,4931,6470,9868,1755,5679,1021,2380,22668,6074,4568,6258,234
18,1252,5835,14968,4332,2204,5280,445,2,2,1,10956,15676,8155,7381,2,3,15965,873,6577,3048,10007,9,1921,5
784,3995,19130,12192,4832,17016,122,700,4,1,2,2,2,2,8652,107,821,4337,785,1765,978,3023,2756,3546,2,2017
,14,82,950,1758,168,1014,751,202,1866,125,6416,1,1015,51,2197,488,922,613,1323,346,109,364,466,683,899,2
```

Packet 4. 1 client pkt, 9 server pkts, 1 turn. Click to select.

Entire conversation (16kB)   Show data as ASCII   Stream 0

LET'S SEE HOW HTTPS PACKETS LOOK LIKE



```
00000000  16 03 01 02 00 01 00 01  fc 03 03 cb 6c ea fb 9f   ........ ....l...
00000010  71 f0 1d 41 6a 19 4d 76  10 3b 3a e2 eb e5 1d 63   q..Aj.Mv .;:....c
00000020  92 d2 da d2 d2 46 98 73  16 b6 75 20 f8 43 a8 eb   .....F.s ..u .C..
00000030  05 41 47 7e 53 47 37 ad  39 78 32 5a f7 88 ae c1   .AG~SG7. 9x2Z....
00000040  64 77 d6 51 e6 e4 ac ef  03 26 6a a2 00 3e 13 02   dw.Q.... .&j..>..
00000050  13 03 13 01 c0 2c c0 30  00 9f cc a9 cc a8 cc aa   .....,.0 ........
00000060  c0 2b c0 2f 00 9e c0 24  c0 28 00 6b c0 23 c0 27   .+./...$ .(.k.#.'
00000070  00 67 c0 0a c0 14 00 39  c0 09 c0 13 00 33 00 9d   .g.....9 .....3..
00000080  00 9c 00 3d 00 3c 00 35  00 2f 00 ff 01 00 01 75   ...=.<.5 ./.....u
00000090  00 00 00 13 00 11 00 00  0e 77 77 77 2e 67 6f 6f   ........ .www.goo
000000A0  67 6c 65 2e 63 6f 6d 00  0b 00 04 03 00 01 02 00   gle.com. ........
000000B0  0a 00 16 00 14 00 1d 00  17 00 1e 00 19 00 18 01   ........ ........
000000C0  00 01 01 01 02 01 03 01  04 33 74 00 00 00 10 00   ........ .3t.....
000000D0  0e 00 0c 02 68 32 08 68  74 74 70 2f 31 2e 31 00   ....h2.h ttp/1.1.
000000E0  16 00 00 00 17 00 00 00  31 00 00 00 0d 00 2a 00   ........ 1.....*.
000000F0  28 04 03 05 03 06 03 08  07 08 08 08 09 08 0a 08   (....... ........
00000100  0b 08 04 08 05 08 06 04  01 05 01 06 01 03 03 03   ........ ........
00000110  01 03 02 04 02 05 02 06  02 00 2b 00 05 04 03 04   ........ ..+.....
00000120  03 03 00 2d 00 02 01 01  00 33 00 26 00 24 00 1d   ...-.... .3.&.$..
00000130  00 20 31 6b 2c 95 bb 6c  06 fb 83 c0 b9 82 1d ee   . 1k,..l ........
00000140  5f 85 0c da 5c 31 9d b6  dc 00 72 d5 06 08 90 d4   _...\1.. ..r.....
00000150  85 60 00 15 00 af 00 00  00 00 00 00 00 00 00 00   .`...... ........
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000200  00 00 00 00 00                                     .....
    00000000  16 03 03 00 7a 02 00 00  76 03 03 82 f4 4b ce 9f   ....z... v....K..
    00000010  b3 46 18 0f 31 0b 53 1f  4d a0 e6 17 07 3a 83 f6   .F..1.S. M....:..
    00000020  06 c0 4c a2 eb 2c a3 6f  b3 c2 f8 20 f8 43 a8 eb   ..L..,.o ... .C..
    00000030  05 41 47 7e 53 47 37 ad  39 78 32 5a f7 88 ae c1   .AG~SG7. 9x2Z....
    00000040  64 77 d6 51 e6 e4 ac ef  03 26 6a a2 13 02 00 00   dw.Q.... .&j.....
    00000050  2e 00 33 00 24 00 1d 00  20 85 51 b9 c0 6e b7 59   ..3.$...  .Q..n.Y
    00000060  4e 79 54 6a dc f5 c2 5b  7d 0b 5e 59 a7 50 a4 37   NyTj...[ }.^Y.P.7
    00000070  58 20 c8 6a d6 58 7d 55  31 00 2b 00 02 03 04 14   X .j.X}U 1.+.....
```


Application Data

id 0

01)

8beba27d2…

```
00000000  16 03 01 02 00 01 00 01  fc 03 03 cb 6c ea fb 9f
00000010  71 f0 1d 41 6a 19 4d 76  10 3b 3a e2 eb e5 1d 63
00000020  92 d2 da d2 d2 46 98 73  16 b6 75 20 f8 43 a8 eb
00000030  05 41 47 7e 53 47 37 ad  39 78 32 5a f7 88 ae c1
00000040  64 77 d6 51 e6 e4 ac ef  03 26 6a a2 00 3e 13 02
00000050  13 03 13 01 c0 2c c0 30  00 9f cc a9 cc a8 cc aa
00000060  c0 2b c0 2f 00 9e c0 24  c0 28 00 6b c0 23 c0 27
00000070  00 67 c0 0a c0 14 00 39  c0 09 c0 13 00 33 00 9d
00000080  00 9c 00 3d 00 3c 00 35  00 2f 00 ff 01 00 01 75
00000090  00 00 00 13 00 11 00 00  0e 77 77 77 2e 67 6f 6f
000000A0  67 6c 65 2e 63 6f 6d 00  0b 00 04 03 00 01 02 00
000000B0  0a 00 16 00 14 00 1d 00  17 00 1e 00 19 00 18 01
000000C0  00 01 01 01 02 01 03 01  04 33 74 00 00 00 10 00
000000D0  0e 00 0c 02 68 32 08 68  74 74 70 2f 31 2e 31 00
000000E0  16 00 00 00 17 00 00 00  31 00 00 00 0d 00 2a 00
000000F0  28 04 03 05 03 06 03 08  07 08 08 08 09 08 0a 08
00000100  0b 08 04 08 05 08 06 04  01 05 01 06 01 03 03 03
00000110  01 03 02 04 02 05 02 06  02 00 2b 00 05 04 03 04
00000120  03 03 00 2d 00 02 01 01  00 33 00 26 00 24 00 1d
00000130  00 20 31 6b 2c 95 bb 6c  06 fb 83 c0 b9 82 1d ee
00000140  5f 85 0c da 5c 31 9d b6  dc 00 72 d5 06 08 90 d4
00000150  85 60 00 15 00 af 00 00  00 00 00 00 00 00 00 00
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000200  00 00 00 00 00
```

```
    00000000  16 03 03 00 7a 02 00 00  76 03 03 82 f4 4b ce
    00000010  b3 46 18 0f 31 0b 53 1f  4d a0 e6 17 07 3a 83
    00000020  06 c0 4c a2 eb 2c a3 6f  b3 c2 f8 20 f8 43 a8
    00000030  05 41 47 7e 53 47 37 ad  39 78 32 5a f7 88 ae
    00000040  64 77 d6 51 e6 e4 ac ef  03 26 6a a2 13 02 00
    00000050  2e 00 33 00 24 00 1d 00  20 85 51 b9 c0 6e b7
    00000060  4e 79 54 6a dc f5 c2 5b  7d 0b 5e 59 a7 50 a4
    00000070  58 20 c8 6a d6 58 7d 55  31 00 2b 00 02 03 04
    00000080  03 03 00 01 01 17 03 03  10 44 ca 2d 1c 3e 06
```

```
00000205  14 03 03 00 01 01 17 03  03 00 45 86 84 73 0f 16   ........ ..E..s..
00000215  12 22 39 31 bb 38 39 3d  02 13 d0 b0 e0 dd 0a a2   ."91.89= ........
00000225  ea 6b 90 8f ab 28 f1 3c  4e f8 f8 be ba 27 d2 67   .k...(.< N....'.g
00000235  e8 e4 2e 71 28 62 13 11  7d fb a1 58 fc 0c 1d 5b   ...q(b.. }..X...[
00000245  da 7c 91 3f 6d 9f bb 1d  6c 0b 67 ce 18 23 b9 d8   .|.?m... l.g..#..
00000255  17 03 03 00 29 ae d9 ce  dc e1 eb c5 15 ed ab 31   ....)... .......1
00000265  09 28 e9 65 87 98 4a 7a  76 e9 4b 19 f7 8a 12 d9   .(.e..Jz v.K.....
00000275  07 f3 87 8d 9d e1 dc 6e  af 3e 52 bd 94 81 17 03   .......n .>R.....
00000285  03 00 2c ff 1d 93 26 f3  b8 64 16 37 40 d9 4b 87   ..,...&. .d.7@.K.
00000295  56 6a 20 78 46 14 01 12  fd 1e f8 82 8e 01 44 53   Vj xF... ......DS
000002A5  b2 e6 c8 01 ca fe 25 86  d4 b4 39 1d 18 85 f9      ......%. ..9....
000002B4  17 03 03 00 1e 2d 05 11  4e c9 af f5 05 89 07 05   .....-.. N.......
000002C4  27 55 03 a0 0b 74 35 c7  25 d9 03 89 4e 97 87 70   'U...t5. %...N..p
000002D4  a0 ba 26 17 03 03 00 39  31 66 19 54 9b d1 e9 c5   ..&....9 1f.T....
000002E4  f4 bc 2f 43 ff 0d 91 be  e8 11 ef f9 90 35 07 7e   ../C.... .....5.~
000002F4  4c de 3e 05 b5 b6 2a 34  7b 83 9d b6 48 32 e5 a9   L.>...*4 {...H2..
00000304  17 12 f2 94 3c a2 27 2c  75 da 77 8f 98 71 6a 1d   ....<.', u.w..qj.
00000314  47                                                 G
```

```
    000010CE  17 03 03 02 45 f8 f4 1d  68 b1 7e e5 a2 c6 1f ec   ....E... h.~.....
    000010DE  2a 27 d0 d9 cb 69 5d 4a  31 7b d4 54 43 e2 8f e7   *'...i]J 1{.TC...
    000010EE  e9 d0 d7 1e 8b 4f da 2a  8e 41 26 91 2a 27 d2 bc   .....O.* .A&.*'..
    000010FE  a9 de 8f 07 57 b5 72 01  11 2f 42 c4 e9 8f 41 80   ....W.r. ./B...A.
    0000110E  29 84 2b b7 8b db 8a a6  63 19 70 a3 c8 7c 28 85   ).+..... c.p..|(.
    0000111E  17 00 86 d0 ea 02 30 f3  1f 8e 6b a0 c9 19 77 de   ......0. ..k...w.
    0000112E  31 4f 61 e3 d8 4b 8e dc  c6 c7 f2 32 fa 70 f0 e1   1Oa..K.. ...2.p..
    0000113E  bb af 9c 79 e0 a9 f1 50  6c da d7 e2 36 eb 0b bb   ...y...P l...6...
    0000114E  09 f2 a3 7d a0 13 46 2e  3a 81 5c 77 d4 05 c5 2e   ...}..F. :.\w....
    0000115E  6f ba 65 49 52 1d f5 0b  1b 7d db c5 f9 1d ab ec   o.eIR... .}......
    0000116E  39 d3 40 0a 4b e3 f6 80  56 e2 e7 c5 d3 b8 df 79   9.@.K... V......y
    0000117E  b5 8f 07 48 61 30 a8 19  08 00 f5 51 d1 20 a6 b8   ...Ha0.. ...Q. ..
    0000118E  29 92 52 ae 46 89 ce 2d  43 a9 b1 ec 62 0f 69 f2   ).R.F..- C...b.i.
    0000119E  ff 34 67 5f 92 94 9f 9a  3d e6 36 0c 73 b9 8f 5a   .4g_.... =.6.s..Z
    000011AE  2c bb 91 24 fd 94 8f c4  72 f2 41 6a 49 86 f7 aa   ,..$.... r.AjI...
    000011BE  8e 17 16 c6 0e 48 92 cf  7b b3 a5 74 ee b6 f4 f4   .....H.. {..t....
    000011CE  cb 39 a6 f0 e1 15 a0 46  52 1c ab b9 ab ea d9 82   .9.....F R.......
    000011DE  fb a2 77 08 3d 05 65 20  18 7f e3 dd 44 f4 2b 38   ..w.=.e  ....D.+8
    000011EE  e7 23 9e 7f c6 29 83 dd  0b f0 e4 d0 b7 a9 fe 18   .#...).. ........
    000011FE  83 8f 77 cc 9f 88 42 df  ad a2 41 76 8f 16 38 4e   ..w...B. ..Av..8N
    0000120E  9f ea 72 24 c0 92 fd f0  b8 b3 05 2b f2 97 f4 6b   ..r$.... ...+...k
```

WHAT COULD GO WRONG

WHAT COULD GO WRONG – MEASUREMENT AT SCALE

WHY DO WE NEED A LARGE-SCALE MEASUREMENT?

- Guide us in forming research questions about the Internet practices
  - ZMap: IPv4 address space < 45-min
  - Censys: IPv4 address scans with full protocol handshakes
  - …

Port	Protocol	SubProtocol	Port Open (Hosts)	Full Handshake (Hosts)
80	HTTP	GET /	77.3 M	66.8 M
443	HTTPS	TLS	47.1 M	33.3 M
443	HTTPS	SSLv3	43.1 M	22.5 M
443	HTTPS	Heartbleed	47.1 M	33.1 M
7547	CWMP	GET /	55.1 M	44.3 M
502	MODBUS	Device ID	2.0 M	32 K
21	FTP	Banner Grab	22.9 M	14.9 M
143	IMAP	Banner Grab	7.9 M	4.9 M
993	IMAPS	Banner Grab	6.9 M	4.3 M
110	POP3	Banner Grab	8.8 M	4.1 M
995	POP3S	Banner Grab	6.6 M	4.0 M
25	SMTP	Banner Grab	14.7 M	9.0 M
22	SSH	RSA	14.3 M	14.3 M
53	DNS	OpenResolver	12.4 M	8.4 M
123	NTP	Get Time	1.6 M	1.2 M
1900	UPnP	Discovery	9.5 M	9.5 M

CENSYS FINDINGS

- Industrial control systems
  - SCADA (Supervisory control and data acquisition) systems
    - No authentication while communicating over the Internet
    - No proper security protection mechanisms

Country	Modus Devices	
United States	4723	24.7%
Spain	1,448	7.58%
Italy	1,220	6.39%
France	1,149	6.02%
Turkey	884	4.63%
Canada	822	4.30%
Denmark	732	3.83%
Taiwan	682	3.57%
Europe	615	3.22%
Sweden	567	2.97%
Total	12,842	67.23%

Table 4: **Top Countries with Modbus Devices** — We identified Modbus hosts in 117 countries, with the top 10 countries accounting for 67% of the total costs, and nearly one-quarter of all Modbus hosts we identifed are located in the United States.

Device Type	Count
Modbus Ethernet Gateway	1,440
Programmable Logic Controller	1,054
Solar Panel Controller	635
Water Flow Controller	388
Power Monitor/Controller	158
Touchscreen System Controller	79
SCADA Processor/Controller	99
Environment/Temperature Sensor	10
Cinema Controller	5
Generic Modbus Device	28,750

Table 5: **Modbus Devices** — We used Censys to categorize publicly available industrial control systems that support the Modbus protocol.

CENSYS FINDINGS – CONT'D

- Heartbleed, Poodle, and SSLv3
  - Heartbleed (https://heartbleed.com): CVE-2014-0160
    - An implementation error in OpenSSL
    - Patched quickly once known to public, but…
  - Poodle
    - A fundamental flaw in the SSLv3 protocol
    - SSL 3.0 has been disabled immediately

Vulnerability	Alexa	IPv4	IPv4 Trusted
Heartbleed	1.16%	0.96%	0.19%
SSLv3 Support	46.0%	55.8%	34.7%
SSLv3 Only	0.05%	2.9%	0.07%

Table 6: **Heartbleed and SSLv3** — We show a breakdowns for the Heartbleed vulnerability and SSLv3 support for HTTPS hosts in the IPv4 address space and the Alexa Top 1 Million.

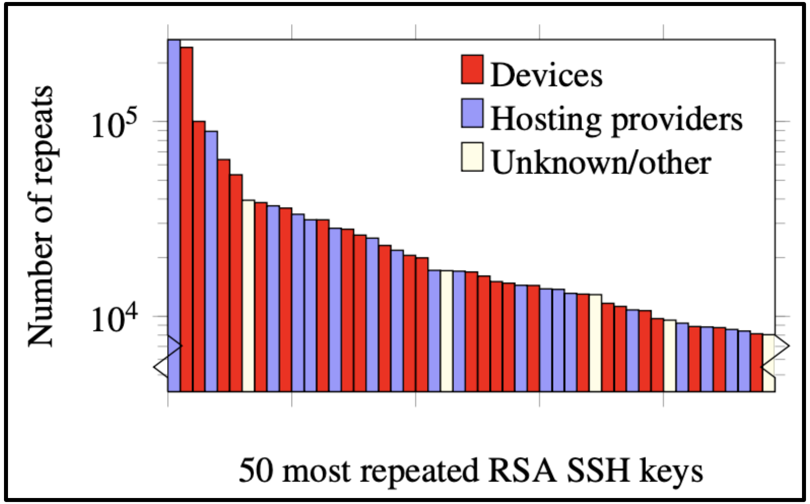
CRYPTOGRAPHY MISUSE IN THE WILD

A LARGE-SCALE MEASUREMENT ON TLS AND SSH SERVERS

- Guide us in forming research questions about cryptography misuses
  - Weak keys (insufficient entropy in key generation)
  - Reused primes
  - Improper certificate validations
  - …

Potential security problems

- TLS and SSH hosts use the same keys
  - 61% of TLS hosts and 65% of SSH hosts served the same key as another host
  - Not all of them were due to the vulnerabilities
    - 60% and 30% of the most common DSA host keys and RSA host keys are from the large hosting providers
    - Distinct TLS certificates are all belonging to the same organization



POTENTIAL SECURITY PROBLEMS – CONT'D

- Vulnerabilities keys
  - Repeated keys due to low-entropy
    - 5.23% of the TLS hosts use manufacturer-default certificates or keys
    - 0.34% of the TLS hosts served repeated keys (98% are self-signed)
    - 9.60% of the SSH hosts served repeated keys
  - Factorable RSA keys

RSA REVISITED

- Key selection
  - Choose two large prime number, p and q
    - Public key:
      - Set $N = pq$
      - Choose e (e.g., 65537) as a coprime of $\phi = (p-1)(q-1)$
    - Private key:
      - Fine d that satisfies $de == 1 \pmod{\phi}$

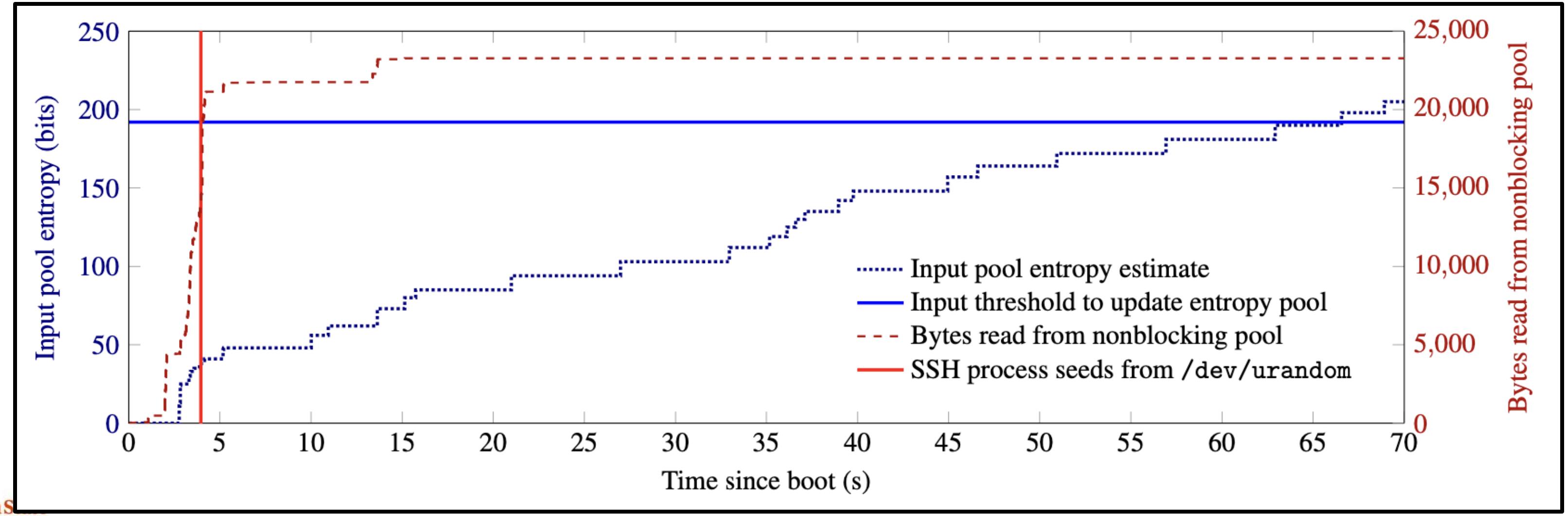
- Security
  - Concern: can an adversary guess the private key from the public key?
  - To do such an attack, the attacker needs to find ϕ
  - But we choose p and q as a large prime number; thus, it is difficult

POTENTIAL SECURITY PROBLEMS – CONT'D

- Vulnerabilities keys
  - Repeated keys due to low-entropy
    - 5.23% of the TLS hosts use manufacturer-default certificates or keys
    - 0.34% of the TLS hosts served repeated keys (98% are self-signed)
    - 9.60% of the SSH hosts served repeated keys
  - Factorable RSA keys (Mining Ps and Qs become easier)
    - Obtain private keys for 0.40% of the TLS certificates; (0.5%) of the TLS hosts
    - Obtain 0.02% of the RSA SSH host keys; 0.027% of the RSA SSH hosts
    - These vulnerable keys are:
      - System-generated certificates
      - SSH host keys used by embedded devices, e.g., routers, firewalls or remote admin cards

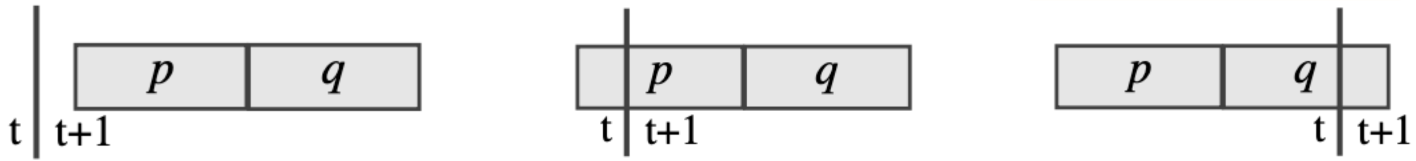
THE SOURCES OF THE VULNERABILITIES

- Weak entropy and the Linux RNG
  - Linux has entropy sources weakened under certain operating conditions
    - It uses the Nonblocking pool entropy until Input pool reaches to a certain threshold
    - The figure shows (red line) the time when the OpenSSH reads its initial PRNG
    - OpenSSH reads the PRNG before the system is ready for the secure use



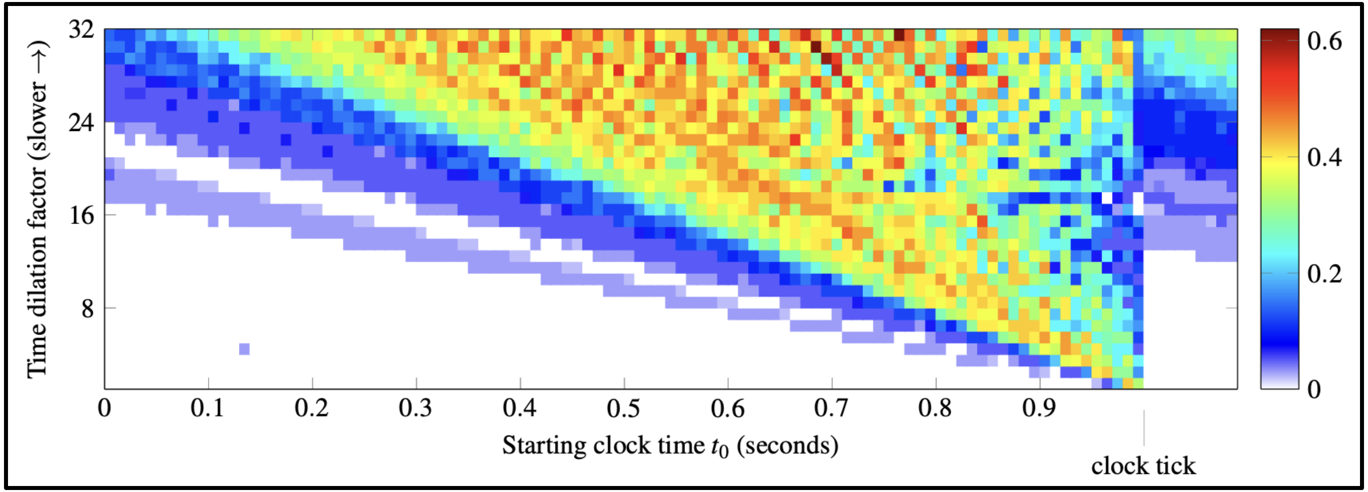
THE SOURCES OF THE VULNERABILITIES

- OpenSSH RSA key generation algorithm
  - Suppose we generate *p* and *q* pairs across many systems
    - (Left) If the *t* is the same while computing *p* and *q*, it will generate the same key
    - (Middle) If the clock ticks while generating *p,* then *p* and *q* do not share a factor
    - (Right) If the clock ticks while generating *q*, then *p* will be the same, but not *q*



THE SOURCES OF THE VULNERABILITIES

- OpenSSH RSA key generation algorithm
  - Suppose we generate *p* and *q* pairs across many systems
    - (Left) If the *t* is the same while computing *p* and *q*, it will generate the same key
    - (Middle) If the clock ticks while generating *p*, then *p* and *q* do not share a factor
    - (Right) If the clock ticks while generating *q*, then *p* will be the same, but not *q*
  - Empirical analysis



Mistakes in implementing secure protocols

BACKGROUND: SSL/TLS HANDSHAKE

Client (You)

- 1. Client hello
  - Send version, random number, available cipher suite, etc..

(google.com) Server

- 2. Server hello
  - Sends server random, version, choose cipher, etc.
- 3. Server Certificate
  - Send certificate to the client

BACKGROUND: HANDSHAKE STEP I – CLIENT HELLO

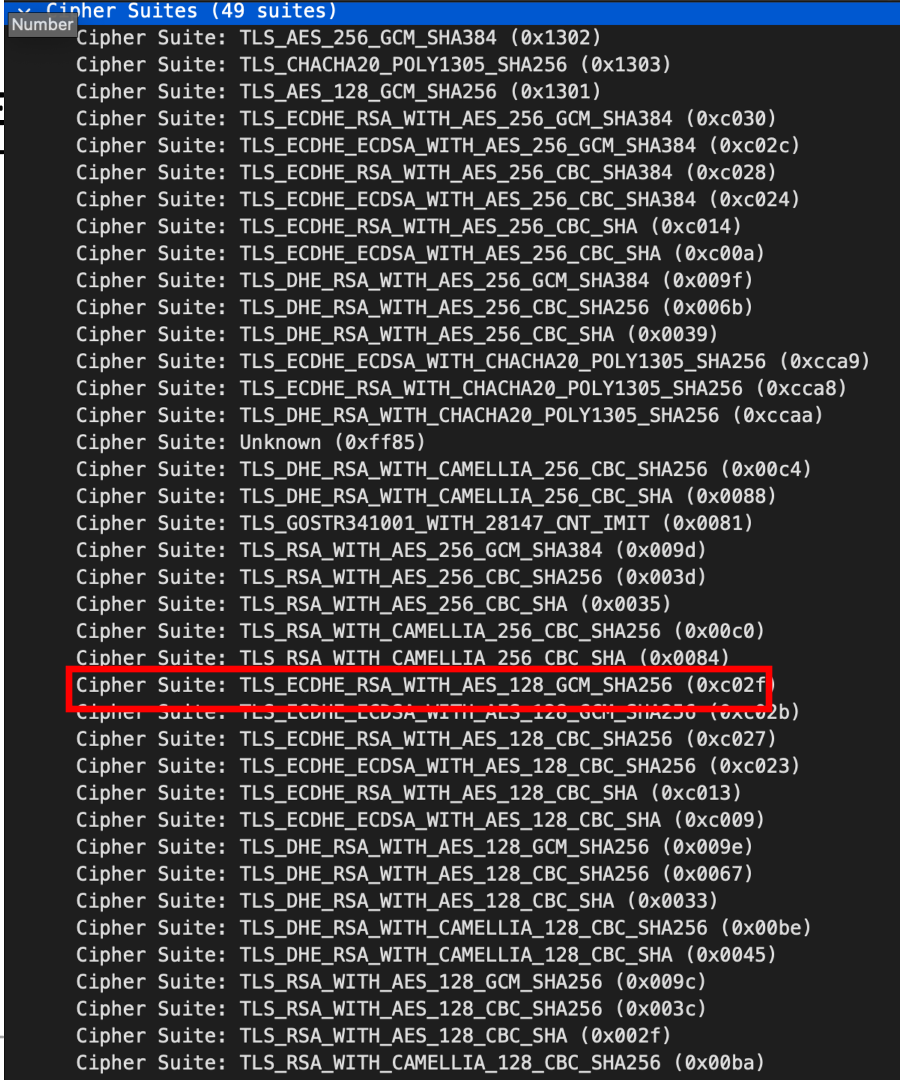
- The first message a client sends to the server
  - It sends an SSL/TLS version, a random number, an available cipher suite, …

```
00000000  16 03 01 01 44 01 00 01  40 03 03 95 8b 02 ec f4   ....D.
00000010  ca 4d 7d 98 6b 9e 3f 45  8b fa 92 10 f0 9c 2c aa   .M}.k.
00000020  bf 27 f0 52 b0 97 6c f0  6c a2 a9 20 bc b7 86 80   .'.R..
00000030  f2 f1 71 9f e0 7e 7e 4c  c2 51 88 e7 72 2d e0 3c   ..q..~
00000040  ca cc fa 2c 99 dc b9 56  d0 80 bd 91 00 62 13 02   ...,..
00000050  13 03 13 01 c0 30 c0 2c  c0 28 c0 24 c0 14 c0 0a   .....0
00000060  00 9f 00 6b 00 39 cc a9  cc a8 cc aa ff 85 00 c4   ...k.9
00000070  00 88 00 81 00 9d 00 3d  00 35 00 c0 00 84 c0 2f   ......
00000080  c0 2b c0 27 c0 23 c0 13  c0 09 00 9e 00 67 00 33   .+.'.#
00000090  00 be 00 45 00 9c 00 3c  00 2f 00 ba 00 41 c0 11   ...E..
000000A0  c0 07 00 05 00 04 c0 12  c0 08 00 16 00 0a 00 ff   ......
000000B0  01 00 00 95 00 2b 00 09  08 03 04 03 03 03 02 03   .....+
000000C0  01 00 33 00 26 00 24 00  1d 00 20 ba 53 26 b5 f2   ..3.&.
000000D0  19 5d b0 e0 b5 f4 30 0c  73 e9 2a 1d 86 72 d5 29   .]....
000000E0  6e fc 32 3f d3 0f 31 d6  e2 57 61 00 00 00 18 00   n.2?..
000000F0  16 00 00 13 77 77 77 2e  6f 72 65 67 6f 6e 73 74   ....ww
00000100  61 74 65 2e 65 64 75 00  0b 00 02 01 00 00 0a 00   ate.ed
00000110  0a 00 08 00 1d 00 17 00  18 00 19 00 0d 00 18 00   ......
00000120  16 08 06 06 01 06 03 08  05 05 01 05 03 08 04 04   ......
00000130  01 04 03 02 01 02 03 00  10 00 0e 00 0c 02 68 32   ......
00000140  08 68 74 74 70 2f 31 2e  31                        .http/
```

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 324
  ⌄ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 320
      Version: TLS 1.2 (0x0303)
    › Random: 958b02ecf4ca4d7d986b9e3f458bfa9210f09c2caabf27f052b0976cf06ca2a9
      Session ID Length: 32
      Session ID: bcb78680f2f1719fe07e7e4cc25188e7722de03ccaccfa2c99dcb956d080bd91
      Cipher Suites Length: 98
    › Cipher Suites (49 suites)
      Compression Methods Length: 1
    › Compression Methods (1 method)
      Extensions Length: 149
    ⌄ Extension: supported_versions (len=9)
        Type: supported_versions (43)
        Length: 9
        Supported Versions length: 8
        Supported Version: TLS 1.3 (0x0304)
        Supported Version: TLS 1.2 (0x0303)
        Supported Version: TLS 1.1 (0x0302)
        Supported Version: TLS 1.0 (0x0301)
```

BACKGROUND: HANDSHAKE STEP I – CLIE

- It sends supported cipher suites:
  - TLS_ECDHE_RSA_WITH AES_128_GCM_SHA256 ECDHE_RSA_AES_128_GCM_SHA256



BACKGROUND: HANDSHAKE STEP II – SERVER HELLO

- The first message a client sends to the server
  - It sends an SSL/TLS version, a random number, an available cipher suite, …

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 102
  ⌄ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 98
      Version: TLS 1.2 (0x0303)
    › Random: 7937be8da9875cf054f0ed18b7efec590e2fb8823ffb7afb87fdffed322822dc
      Session ID Length: 32
      Session ID: 6adeb8c9532bf74b3f5d9940e83f470e46ac3f49054c667dfe8255a6342bea6e
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
      Compression Method: null (0)
```

- The server choose a cipher based on the client’s availability
  - **Chosen:** TLS_ECDHE_RSA_AES_128_GCM_SHA256

BACKGROUND: HANDSHAKE STEP III – SERVER CERTIFICATE

- The first message a client sends to the server
  - It sends an SSL/TLS version, a random number, an available cipher suite, …
- The server choose a cipher based on the client's availability
  - **Chosen:** TLS_ECDHE_RSA_AES_128_GCM_SHA256
- The server next sends the certificate information to the client
  - It sends a full chain (PKI) of digital certificates

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Certificate
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 6037
  ⌄ Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 6033
      Certificates Length: 6030
    ⌄ Certificates (6030 bytes)
        Certificate Length: 1994
      › Certificate: 308207c6308206aea003020102021030bc9131f05e7eef26b3844d426b816c300d06092a… (id-at-commonName=oregonstate.edu,id-at-organizationName
        Certificate Length: 1533
      › Certificate: 308205f9308203e1a003020102021047200d0fa85461a7e17a1640291846374300d06092a… (id-at-commonName=InCommon RSA Server CA,id-at-organizat
        Certificate Length: 1413
      › Certificate: 3082058130820469a00302010202103972443af922b751d7d36c10dd313595300d06092a… (id-at-commonName=USERTrust RSA Certification Authority,
        Certificate Length: 1078
      › Certificate: 30820432308203 1aa003020102020101300d06092a864886f70d01010505000307b310b30… (id-at-commonName=AAA Certificate Services,id-at-organiz
```

Oregon State University

BACKGROUND: HANDSHAKE STEP IV – KEY EXCHANGE / VERIFYING SIGNATURE

- Key exchange
  - The client knows the server's public key written in their certificate
  - The client chooses a random key and encrypt that with the server's public key
  - The encrypted key will be sent to the server
  - It's only the server who can decrypt the key (good)

Are We Secure Now? Can We See A Potential Security Issues?

BACKGROUND: POTENTIAL SECURITY PROBLEM

- Key exchange
  - The client knows the server's public key written in their certificate
  - The client chooses a random key and encrypt that with the server's public key
  - The encrypted key will be sent to the server
  - It's only the server who can decrypt the key (good)

- Suppose:
  - 3 years later, the server's private key is stolen
  - From then, the attacker can decrypt the all the data (private key, messages, …)
  - What if the attacker also has all the encrypted messages before the breach?

BACKGROUND: HANDSHAKE REQUIRES FORWARD SECURITY

- Forward Secrecy / Perfect Forward Secrecy
  - We want to keep all the communication secure
  - Even if the server's private key (i.e., the long-term key) has been breached

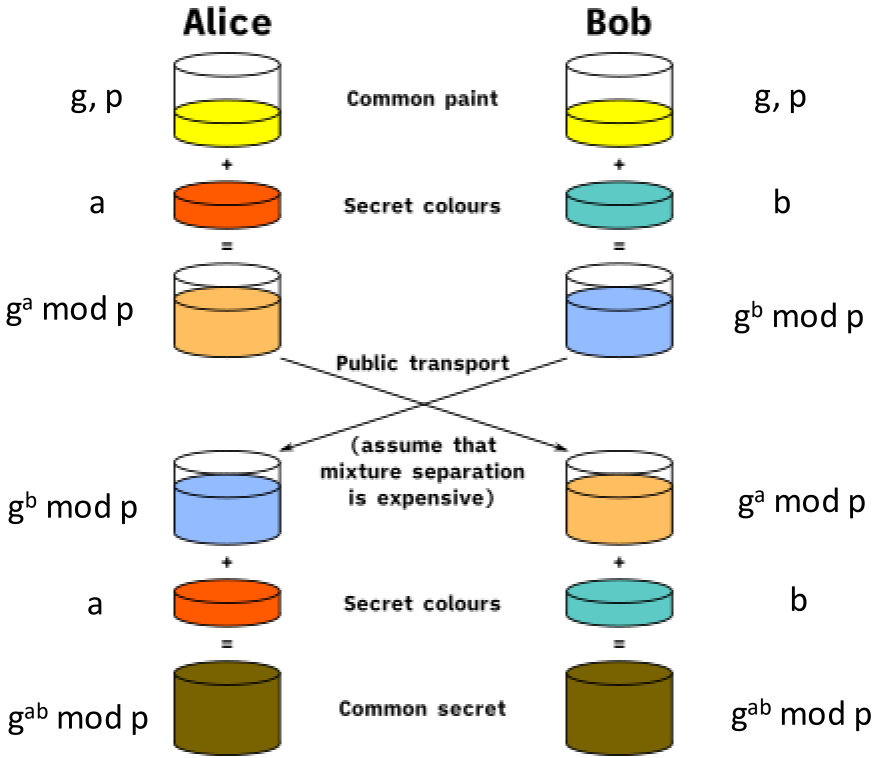
- Example of such breaches
  - Heartbleed (https://heartbleed.com/): CVE-2014-0160

BACKGROUND: SOLUTION – EPHEMERAL DIFFIE-HELLMAN

- The key idea:
  - Do not use a fixed private value for all the DH
  - This can lead to a serious information breach (stolen private key)

- Ephemeral DH
  - Generate the private value every time we make a connection
  - Never reuse the value
    - User A secretly chooses a, send $A = g^a \bmod p$
    - User B secretly chooses b, send $B = g^b \bmod p$
    - User A and B will choose different a and b for the next time

REVISITED: DIFFIE-HELLMAN KEY EXCHANGE IN GRAPHICS



BACKGROUND: ECDHE

- Elliptic-curve Diffie-Hellman Ephemeral (ECDHE)
  - Both the client and server will generate new a and b, respectively
  - Make it difficult for an adversary to infer the shared secret even if the session is compromised (they don't know b for other sessions)

BACKGROUND: HANDSHAKE STEP IV

Client (You)

- 1. Client hello

(google.com) Server

- 2. Server hello
- 3. Server Certificate
- 4. Server Key Exchange
  - Shares DH material, signed by the public key
- 5. Server Hello Done

BACKGROUND: HANDSHAKE STEP IV – KEY EXCHANGE

- The server sends ECDHE material to the client
  - ECDHE public value (pubkey) is signed by the RSA private key
  - The public key is available in the certificate

```
v Transport Layer Security
  v TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 333
    v Handshake Protocol: Server Key Exchange
        Handshake Type: Server Key Exchange (12)
        Length: 329
      v EC Diffie-Hellman Server Params
          Curve Type: named_curve (0x03)
          Named Curve: secp256r1 (0x0017)
          Pubkey Length: 65
          Pubkey: 04d3be5c83a346d31403c9803f753af4c583cd3504d550f5e1be0368c624acf4fa7e1b85…
        > Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
          Signature Length: 256
          Signature: 5fe6444e7ae294aa7815516c91c19eadd1a5edc72e1a690916a4acb89669eb219a669970…
```

BACKGROUND: HANDSHAKE STEP V – SERVER HELLO DONE

- The server sends ECDHE material to the client
  - ECDHE public value (pubkey) is signed by the RSA private key
  - The public key is available in the certificate
- The server hello done
  - Indicate that the server has finished sending required values to the client

```
v Transport Layer Security
  v TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 4
    v Handshake Protocol: Server Hello Done
        Handshake Type: Server Hello Done (14)
        Length: 0
```

BACKGROUND: HANDSHAKE STEP

Client (You)

- 1. Client hello

(google.com) Server

- 2. Server hello
- 3. Server Certificate
- 4. Server Key Exchange
  - Shares DH material, signed by the public key
- 5. Server Hello Done

Now, the Client Can Verify Server Signature and Share a Secret via DH!

BACKGROUND: HANDSHAKE STEP

Client (You)

(google.com) Server

Previous steps (omitted)

- 5. Server Hello Done

- 6. Client Key Exchange
  - Shares DH material after verifying server signature for server's DH material
- 7. Change Cipher Spec
- 8. Encrypted Handshake Message

BACKGROUND: HANDSHAKE STEP VI – CLIENT KEY EXCHANGE

- The client also sends ECDHE material to the server
  - After this, two parties will share a secret
  - We will run the encryption and MAC key by using the shared secret

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 70
  ⌄ Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 66
    ⌄ EC Diffie-Hellman Client Params
        Pubkey Length: 65
        Pubkey: 043cc5f595ea1dca4b3beb1306dec9444e5323177ef9b2c5470dd910d2ce252f672a1dc8…
```

BACKGROUND: HANDSHAKE STEP VI – CLIENT GENERATES A SESSION KEY

- Now the client knows both 'a' and 'b' of ECDHE key exchange
  - The client can compute the shared secret
  - The client then computes the following keys from the shared secret

```
To generate the key material, compute

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

until enough output has been generated.  Then, the key_block is
partitioned as follows:

   client_write_MAC_key[SecurityParameters.mac_key_length]
   server_write_MAC_key[SecurityParameters.mac_key_length]
   client_write_key[SecurityParameters.enc_key_length]
   server_write_key[SecurityParameters.enc_key_length]
   client_write_IV[SecurityParameters.fixed_iv_length]
   server_write_IV[SecurityParameters.fixed_iv_length]
```

These are from
1. Client Hello and
2. Server Hello

BACKGROUND: HANDSHAKE STEP VII – CHANGE CIPHER SPEC (CLIENT)

- Secure communication:
  - The client sends the server a message
  - that now both should use encrypted communication after this point

```
⌄ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
```

Now, We Encrypt Messages and Generate MACs for the Client's!

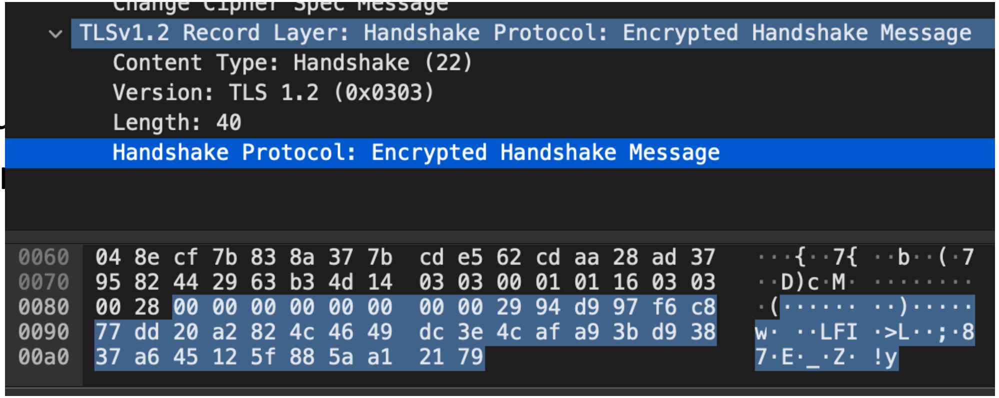
BACKGROUND: HANDSHAKE STEP VIII – ENCRYPTED HANDSHAKE MESSAGE

- The server asks
  - the encrypted versions of previous messages
  - to verify whether the client generated the keys correctly

- Compute a SHA256 hash of a concatenation of all the handshake communications (or SHA384 if the PRF is based on SHA384). This means the Client Hello, Server Hello, Certificate, Server Key Exchange, Server Hello Done and Client Key Exchange messages. Note that you should concatenate only the handshake part of each TLS message (i.e. strip the first 5 bytes belonging to the TLS Record header)
- Compute PRF(master_secret, "client finished", hash, 12) which will generate a 12-bytes hash
- Append the following header which indicates the hash is 12 bytes: 0x14 0x00 0x00 0x0C
- Encrypt the 0x14 0x00 0x00 0x0C | [12-bytes hash] (see the Encrypting / Decrypting data section). This will generate a 64-bytes ciphertext using AES-CBC and 40 bytes with AES-GCM
- Send this ciphertext wrapped in a TLS Record

BACKGROUND: HANDSHAKE STEP VIII – ENCRYPTED HANDSHAKE MESSAGE

- The server asks
  - the encrypted versions of previou
  - to verify whether the client gener



- Compute a SHA256 hash of a concatenation of all the handshake communications (or SHA384 if the PRF is based on SHA384). This means the Client Hello, Server Hello, Certificate, Server Key Exchange, Server Hello Done and Client Key Exchange messages. Note that you should concatenate only the handshake part of each TLS message (i.e. strip the first 5 bytes belonging to the TLS Record header)
- Compute PRF(master_secret, "client finished", hash, 12) which will generate a 12-bytes hash
- Append the following header which indicates the hash is 12 bytes: 0x14 0x00 0x00 0x0C
- Encrypt the 0x14 0x00 0x00 0x0C I [12-bytes hash] (see the Encrypting / Decrypting data section). This will generate a 64-bytes ciphertext using AES-CBC and 40 bytes with AES-GCM
- Send this ciphertext wrapped in a TLS Record

BACKGROUND: HANDSHAKE STEP

Client (You)

(google.com) Server

Previous steps (omitted)

- 5. Server Hello Done

- 6. Client Key Exchange
  - Shares DH material after verifying server signature for server's DH material
- 7. Change Cipher Spec
- 8. Encrypted Handshake Message

- 9. Change Cipher Spec
- 10. Encrypted Handshake Message

BACKGROUND: HANDSHAKE STEP XV – CHECK CLIENT'S ENCRYPTED MESSAGES

- The server verifies the client's encrypted handshake messages
  - After generating client_write_key
  - Decrypt the message
  - Compute the same value
  - Compare!

- Compute a SHA256 hash of a concatenation of all the handshake communications (or SHA384 if the PRF is based on SHA384). This means the Client Hello, Server Hello, Certificate, Server Key Exchange, Server Hello Done and Client Key Exchange messages. Note that you should concatenate only the handshake part of each TLS message (i.e. strip the first 5 bytes belonging to the TLS Record header)
- Compute PRF(master_secret, "client finished", hash, 12) which will generate a 12-bytes hash
- Append the following header which indicates the hash is 12 bytes: 0x14 0x00 0x00 0x0C
- Encrypt the 0x14 0x00 0x00 0x0C | [12-bytes hash] (see the Encrypting / Decrypting data section). This will generate a 64-bytes ciphertext using AES-CBC and 40 bytes with AES-GCM
- Send this ciphertext wrapped in a TLS Record

BACKGROUND: HANDSHAKE STEP XV – CHANGE CIPHER SPEC (SERVER)

- The server lets the client know
  - that we will use encrypted communication after this message

```
⌄ Transport Layer Security
  ⌄ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.2 (0x0303)
      Length: 1
      Change Cipher Spec Message
```

Now, We Encrypt Messages and Generate MACs for the Server's!

BACKGROUND: HANDSHAKE STEP X – ENCRYPTED HANDSHAKE MESSAGE

- The client asks
  - the encrypted version of previous messages
  - to verify whether the server generated keys correctly

```
⌄ TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 40
    Handshake Protocol: Encrypted Handshake Message
```

- It needs to compute a hash of the same handshake communications as the client as well as the decrypted "Encrypted Handshake Message" message sent by the client (i.e. the 16-bytes hash starting with 0x1400000C)
- It will call PRF(master_secret, "server finished", hash, 12)

BACKGROUND: HANDSHAKE STEP XI - SENDING APPLICATION DATA

- Now, the server and client
  - will send encrypted data to the client
  - both will always send [encrypted data] [MAC]
    - The server will use server_write_key and server_write_mac_key
    - The client will use client_write_key and client_write_mac_key

POTENTIAL SOURCES OF MISTAKES IN CERTIFICATION VALIDATION

- Detailed steps in client-side validation
  - Chain-of-trust validation
  - Hostname verification
  - Certificate revocation and X.509 extensions
  - …

POTENTIAL SOURCES OF MISTAKES IN CERTIFICATION VALIDATION

- SSL libraries
  - OpenSSL: applications can customize chain-of-trust verification
  - JSSE (Java): hostname verification can be optional

POTENTIAL SOURCES OF MISTAKES IN CERTIFICATION VALIDATION

- Data-transport libraries
  - Apache HTTPClient:
    - Hostname verification can be optional (and uses its own implementation)
    - HTTPS consistency checks are not strictly done
  - Weberknecht:
    - Hostname verification can be optional
  - PHP:
    - Default functionality does not check the certificate validity
    - Hostname verification can be ignored as it uses cURL
  - cURL:
    - (Unintentionally) disable hostname verification
  - Python:
    - Default functionality does not check the certificate validity

POTENTIAL SOURCES OF MISTAKES IN CERTIFICATION VALIDATION

- Misunderstanding the SSL API
  - Amazon Flexible Payments service (PHP)
  - PayPal Payments Standard and PayPal Invoicing:
    - Hostname verification can be overridden and won't be checked in that case
  - PayPal IPN in ZenCart:
    - Default, it does not check the certificate validity
  - Lynx:
    - Chain-of-trust verification is broken
  - …

POTENTIAL SOURCES OF MISTAKES IN CERTIFICATION VALIDATION

- Using insecure middleware
- Using insecure SSL libraries
- … (check the case studies in the paper)

RECOMMENDATIONS FOR SECURE INTERNET INFRASTRUCTURE

RECOMMENDATIONS

- Secure TLS/SSL connections
  - OS developers:
    - Provide RNG interface to app developers
    - Provide entropy conditions to applications
    - Test comprehensively across diverse platforms
  - App developers:
    - Generate keys on first use, not on install or first boot
    - Carefully address the warnings from crypto libraries
  - Device manufacturers:
    - Avoid factory-default keys or certificates
    - Provide sufficient entropy when manufacturing
    - Use hardware random generator if possible

Recommendations

- Secure TLS/SSL connections
  - Certificate authorities:
    - Monitor repeated, weak and factorable keys
  - End users:
    - Regenerate default or automatically generated keys
    - Check for known weak keys
  - Security and cryptography researchers:
    - True RNG
    - Primitives fail gracefully under weak entropy

RECOMMENDATIONS – CONT'D

- (Proper) certificate verification
  - Application developers:
    - Test (run fuzzing) with adversarial SSL certificates
    - Test application code with certificates with chain-of-trust (not with self-signing)
    - Check the library's configurations carefully before its use
  - SSL library developers:
    - Make SSL libraries with explicit documentations and parameters
    - Take the responsibility: manage SSL connections securely
    - Use the collective intelligence: make the error reporting platform user-friendly

Thank You!

Sanghyun Hong

https://secure-ai.systems/courses/Sec-Grad/current

Oregon State University

SAIL

Secure AI Systems Lab