32c3

December 28, 2015

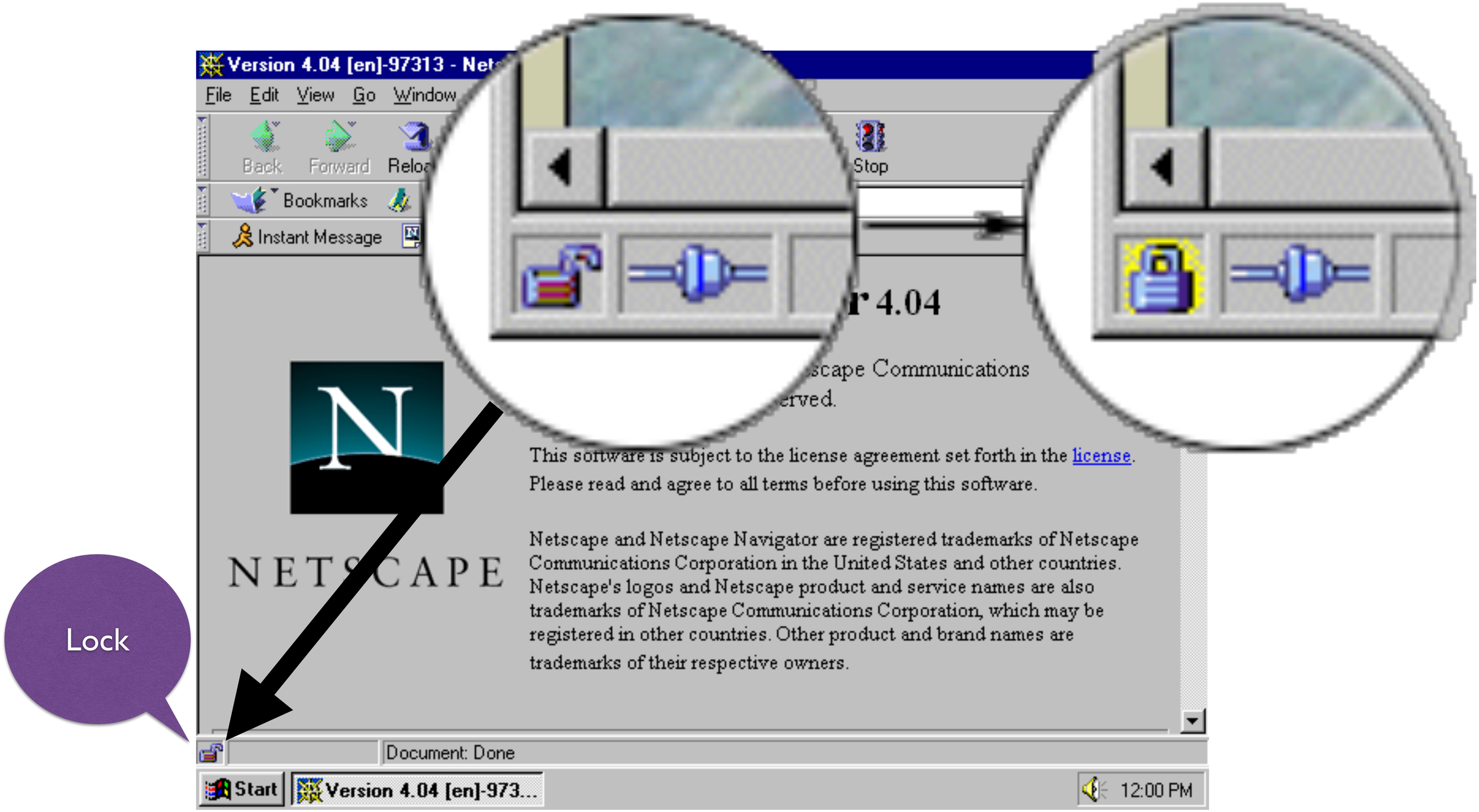
Nick Sullivan
@grittygrease
nick@cloudflare.com
https://crypto.dance

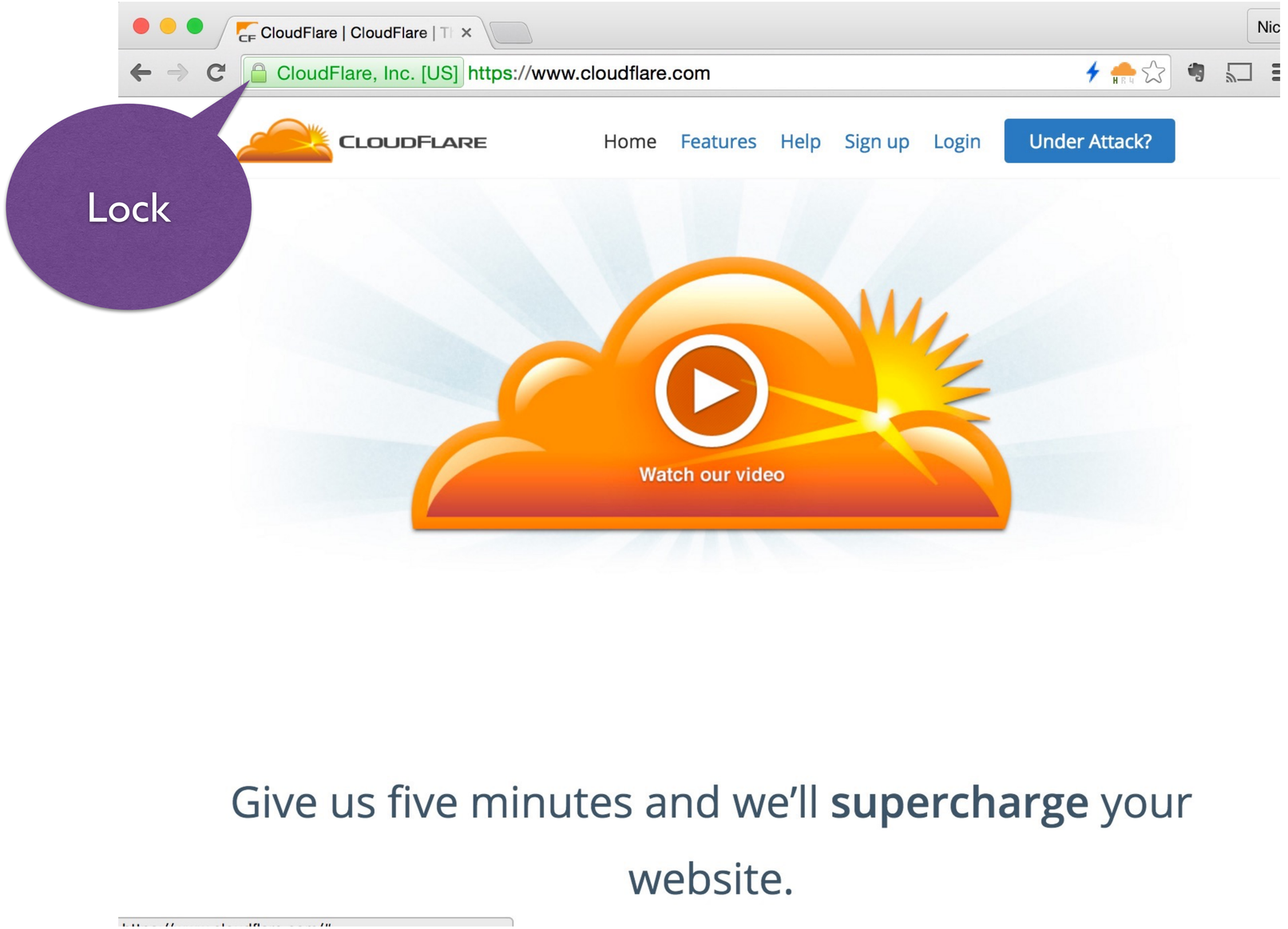
goto fail;

a compendium of transport security calamities


File Edit View Go Bookmarks Options Directory Help
7:13 PM
Netscape: Apple and Macintosh Computer Collection
Back
Forward
Home
Reload
Images
Location: file:///My%20Computer/C/Ken/Old%20Compu
What's New?
What's Cool?
Handbook
A Collector's Apple H
Introduction | Apple | Macintosh | Expandables | Golden Age | Lost Years | G3 | Peripherals | Books | Software | Collection
Introduction
In the 30 years since the advent of the personal computer, one company has stood out consistently as a symbol of quality in both hardware and software, and that company is Apple Computer [now just Apple]. Although frequently over-priced and sometimes presenting a confused range of products [e.g. the Apple III, the IIgs in competition with the Macintosh, the dizzying array of Performa machines in the mid 90s], the design, software, performance and reliability of Macintosh computers has seen Apple Computer remain a very strong presence in the computer arena, despite their long term competition with the giants of IBM and Microsoft and their market leading Windows-PC combination.
To put it another way, Apple was, and still is, "cool". In the world of billion dollar business, Apple is a counter-culture icon. In an industry inclined to produce an uninspiring series of drab, grey, impersonal boxes with clunky, amateur interfaces, Apple is seen as aesthetically driven and demanding of excellence. Well that's the myth anyway, but that there is some truth to it is reflected in the "spirit" that pervades the Macintosh experience. As Andy Herzfeld (author of much of the original
file:///My%20Computer/C/Ken/Old%20Computers/Computer%20Collection/web/Collec

Broken Key


Version 4.04 [en]-97313 - Net
File Edit View Go Window
Back Forward Reloa
Stop
Bookmarks
Instant Message
r 4.04
scape Communications
erved.
This software is subject to the license agreement set forth in the license.
Please read and agree to all terms before using this software.
Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corporation in the United States and other countries. Netscape's logos and Netscape product and service names are also trademarks of Netscape Communications Corporation, which may be registered in other countries. Other product and brand names are trademarks of their respective owners.
NETSCAPE
Document: Done
Start
Version 4.04 [en]-973...
12:00 PM
Lock


CloudFlare | CloudFlare |
CloudFlare, Inc. [US] https://www.cloudflare.com
Lock
CLOUDFLARE
Home
Features
Help
Sign up
Login
Under Attack?
Watch our video
Give us five minutes and we'll **supercharge** your website.

Welcome to Amazon.com Books!

One million titles, consistently low prices.

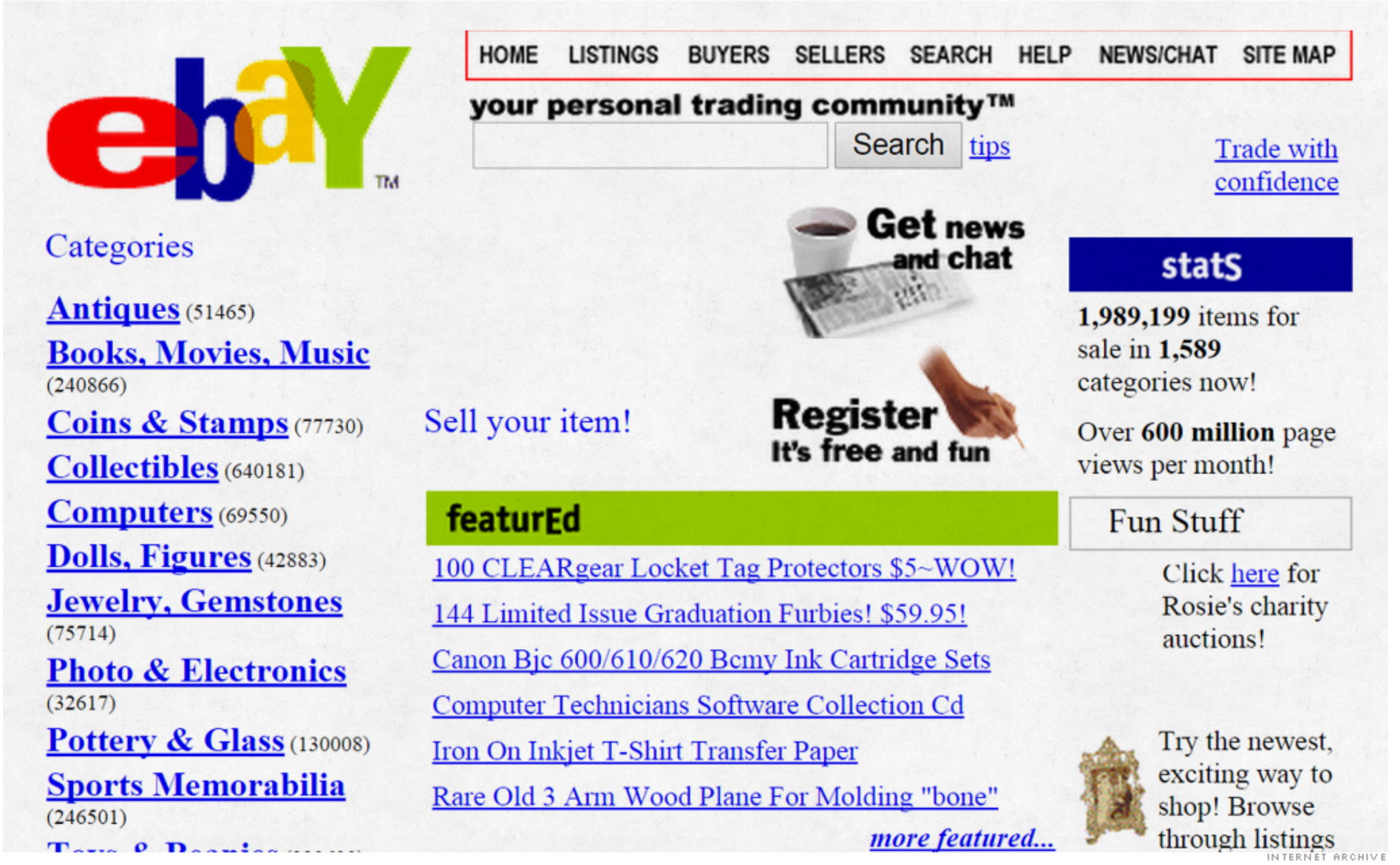
(If you explore just one thing, make it our personal notification service. We think it's very cool!)

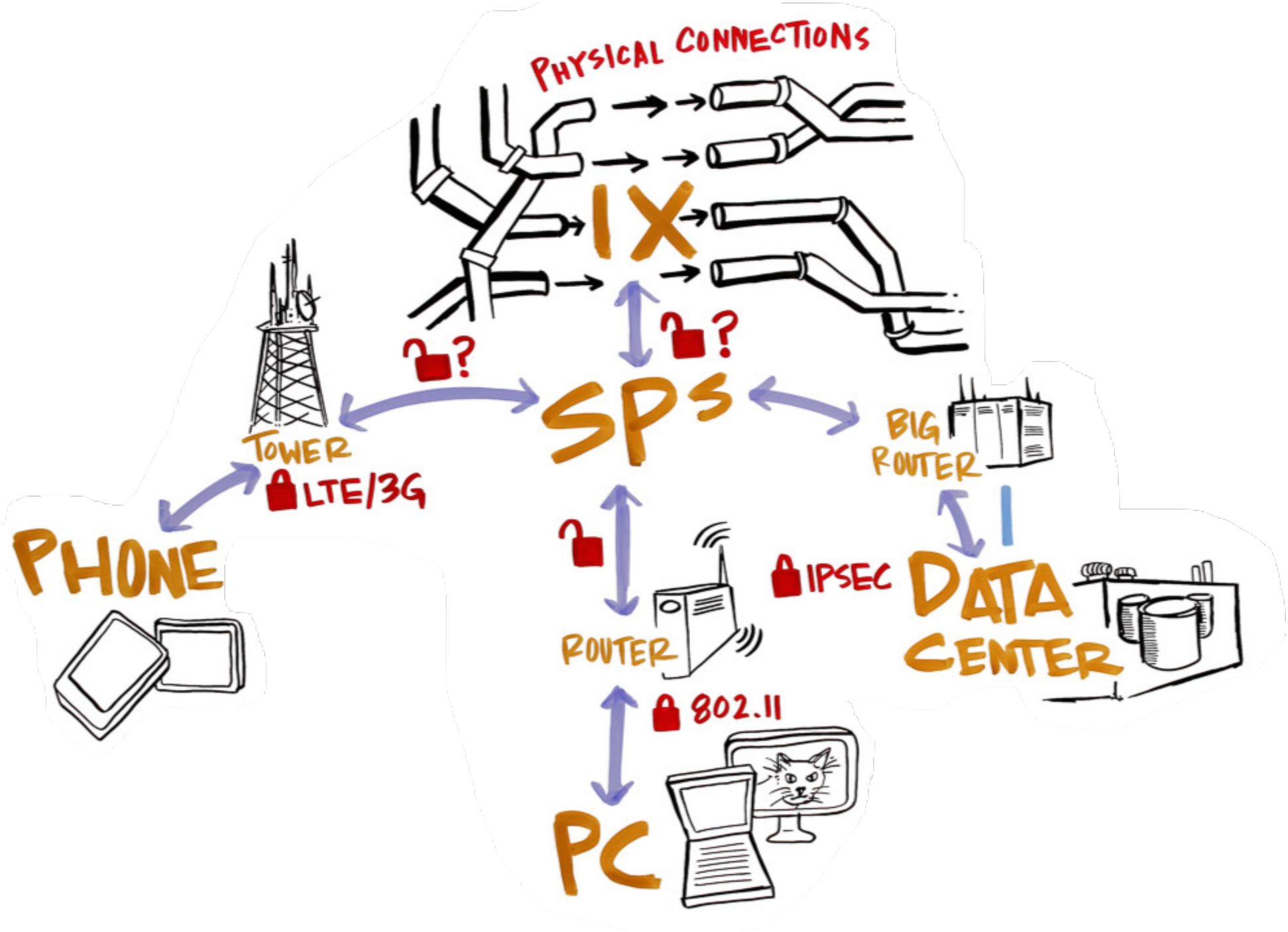
SPOTLIGHT! -- AUGUST 16TH

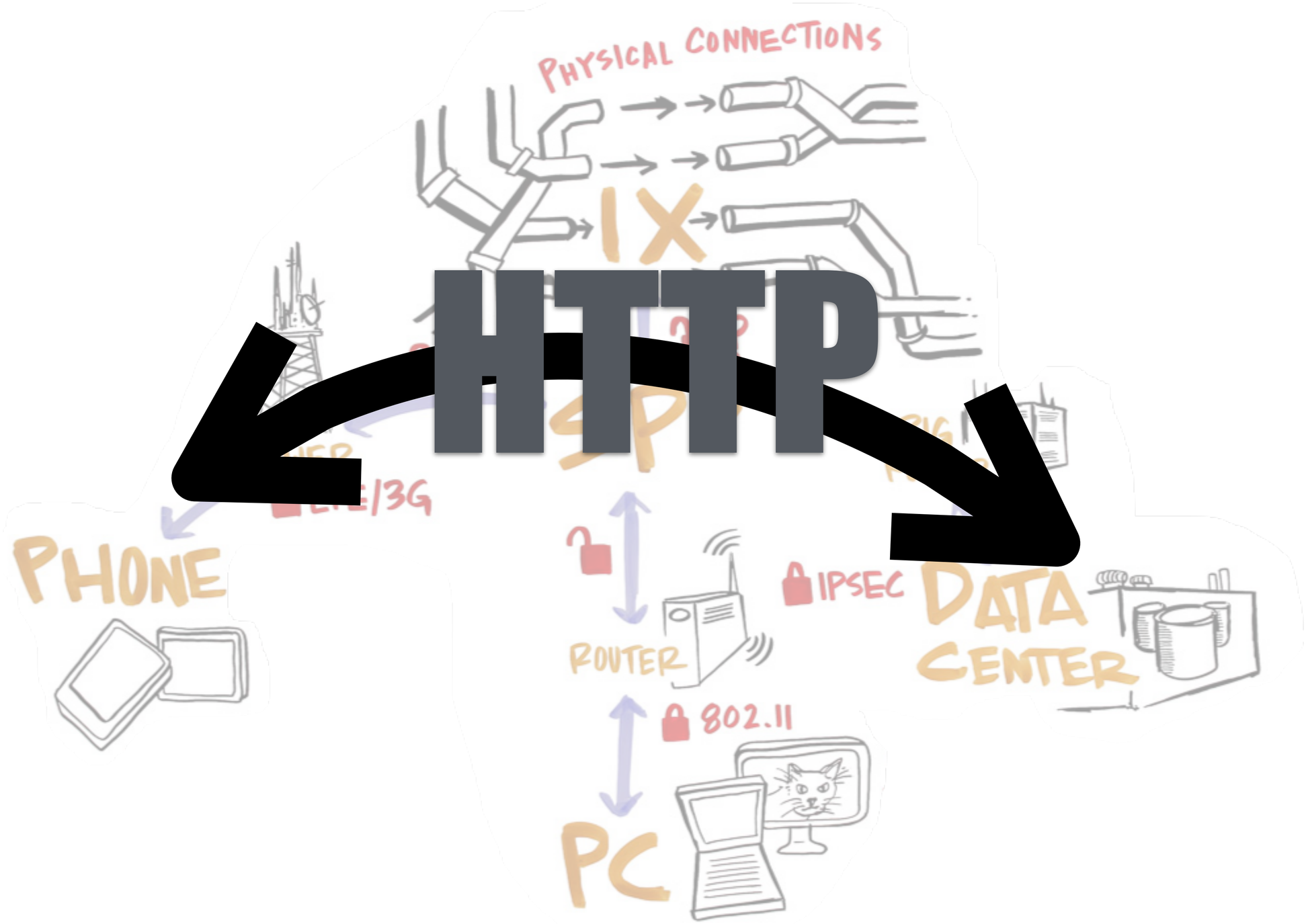
These are the books we love, offered at Amazon.com low prices. The spotlight moves **EVERY** day so please come often.

ONE MILLION TITLES

Search Amazon.com's million title catalog by author, subject, title, keyword, and more... Or take a look at the books we recommend in over 20 categories... Check out our customer reviews and the award winners from the Hugo and Nebula to the Pulitzer and Nobel... and bestsellers are 30% off the publishers list...


HOME LISTINGS BUYERS SELLERS SEARCH HELP NEWS/CHAT SITE MAP
your personal trading community™
Search
tips
Trade with confidence
Categories
Antiques (51465)
Books, Movies, Music (240866)
Coins & Stamps (77730)
Collectibles (640181)
Computers (69550)
Dolls, Figures (42883)
Jewelry, Gemstones (75714)
Photo & Electronics (32617)
Pottery & Glass (130008)
Sports Memorabilia (246501)
Get news and chat
Sell your item!
Register
It's free and fun
stats
1,989,199 items for sale in 1,589 categories now!
Over 600 million page views per month!
featurEd
100 CLEARgear Locket Tag Protectors $5~WOW!
144 Limited Issue Graduation Furbies! $59.95!
Canon Bjc 600/610/620 Bcmy Ink Cartridge Sets
Computer Technicians Software Collection Cd
Iron On Inkjet T-Shirt Transfer Paper
Rare Old 3 Arm Wood Plane For Molding "bone"
more featured...
Fun Stuff
Click here for Rosie's charity auctions!
Try the newest, exciting way to shop! Browse through listings
INTERNET ARCHIVE


PHYSICAL CONNECTIONS
IX
?
?
SPS
TOWER
LTE/3G
PHONE
BIG ROUTER
DATA CENTER
IPSEC
ROUTER
802.11
PC


PHYSICAL CONNECTIONS
IX
HTTP
PHONE
LTE/3G
ROUTER
IPSEC
DATA CENTER
802.11
PC

HTTPS

The “S” stands for Secure

HTTPS = HTTP + Security

- Transport Layer Security (TLS)
- Data **encryption** and **integrity**
- Server **authentication**
- Negotiation of keys happens in the “handshake”

A bit of history

SSL version 1

Kipp E.B. Hickman at Netscape in 1993-1994

A bit of history

“Marc Andressen presented it to about six of us. John Klensin, Tim Berners-Lee, Alan Schiffman and myself were the only ones I can remember, but there were two other seats filled.”

- Phillip Hallam-Baker

A bit of history

“The original protocol had no authenticity checks in whatsoever.”

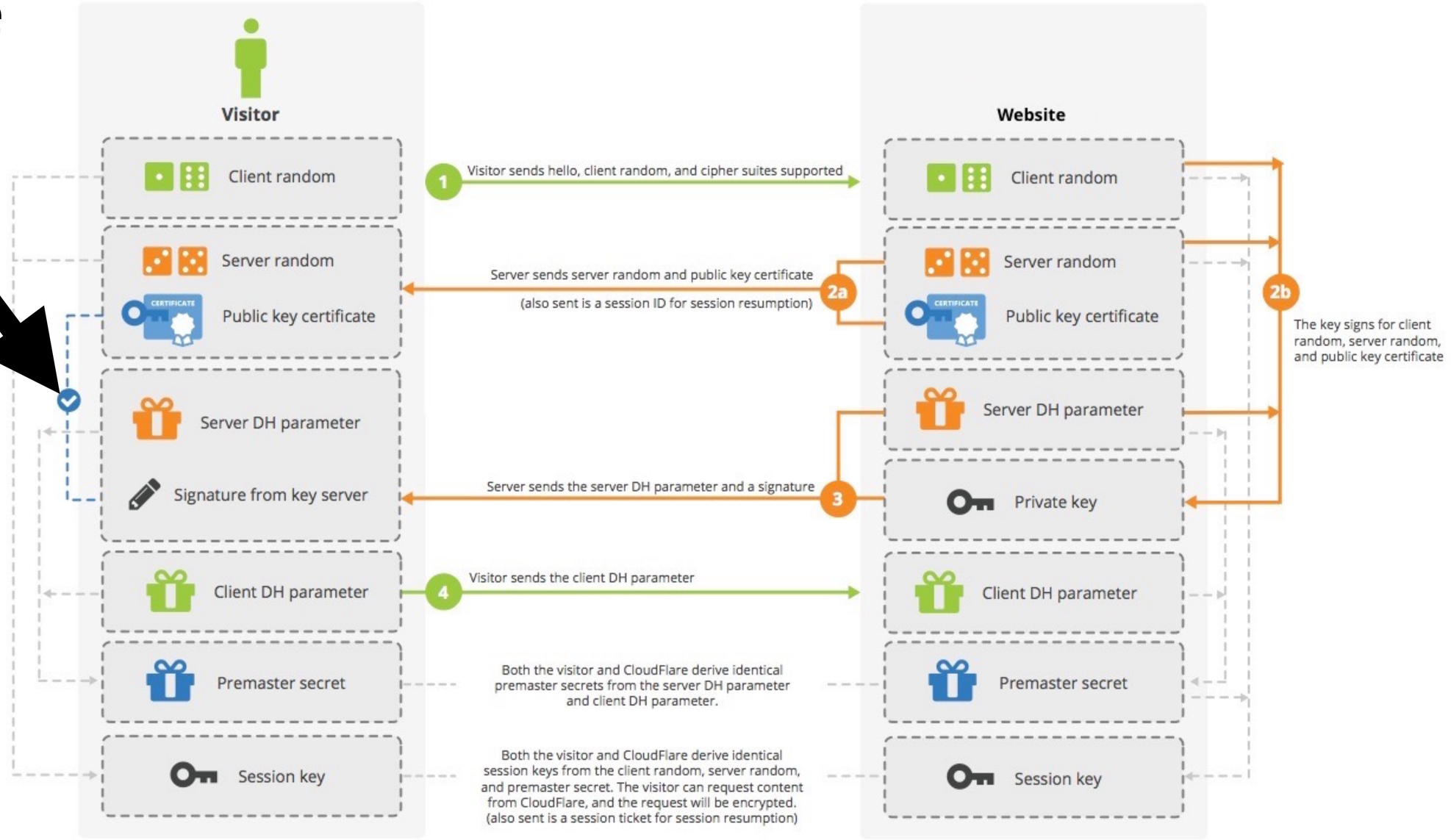
- Phillip Hallam-Baker

How does it work?

Public-key crypto: establish shared keys, server identity

Data encapsulation: encrypt and HMAC with shared keys

Certificate Validation

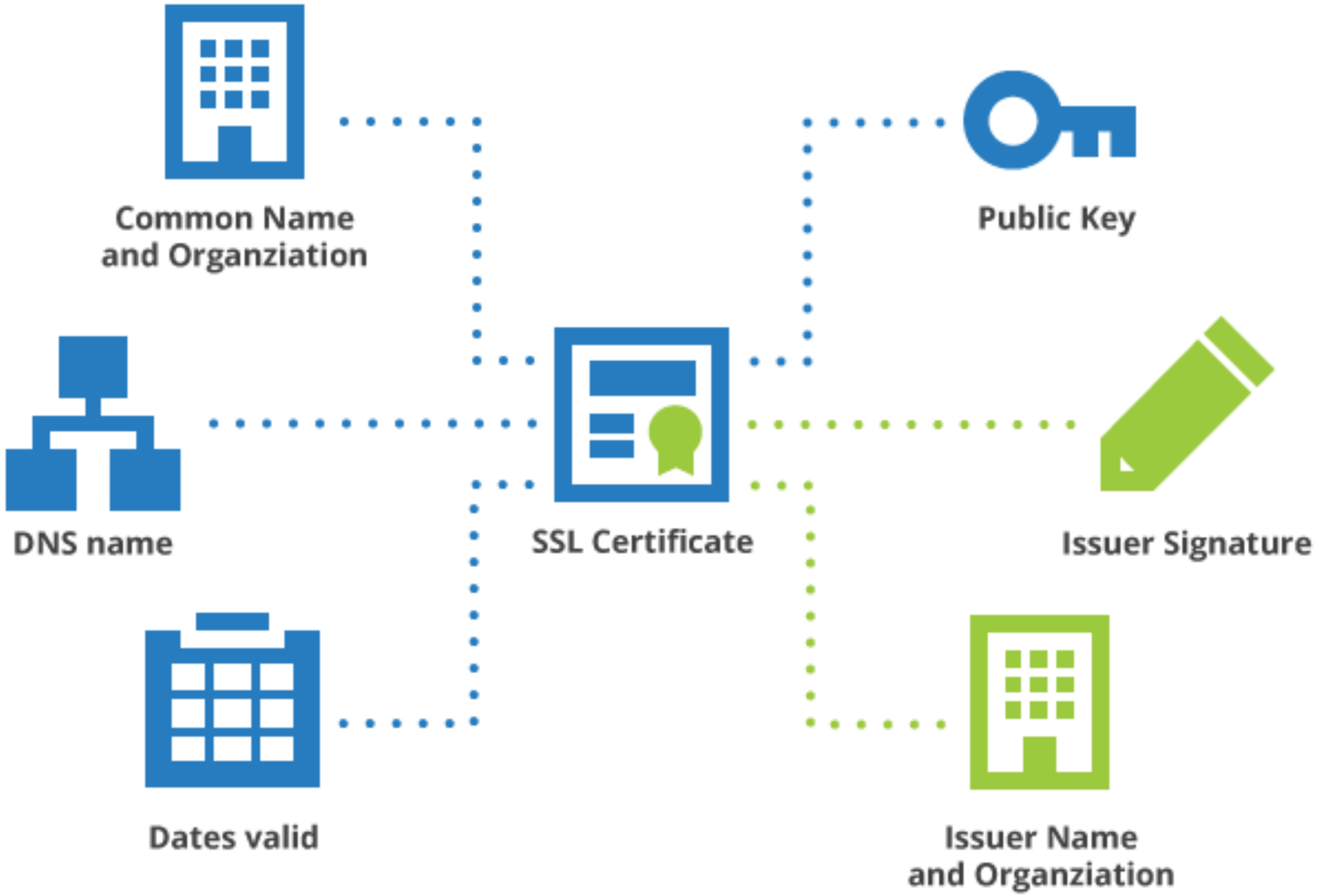


How does TLS provide authenticity?

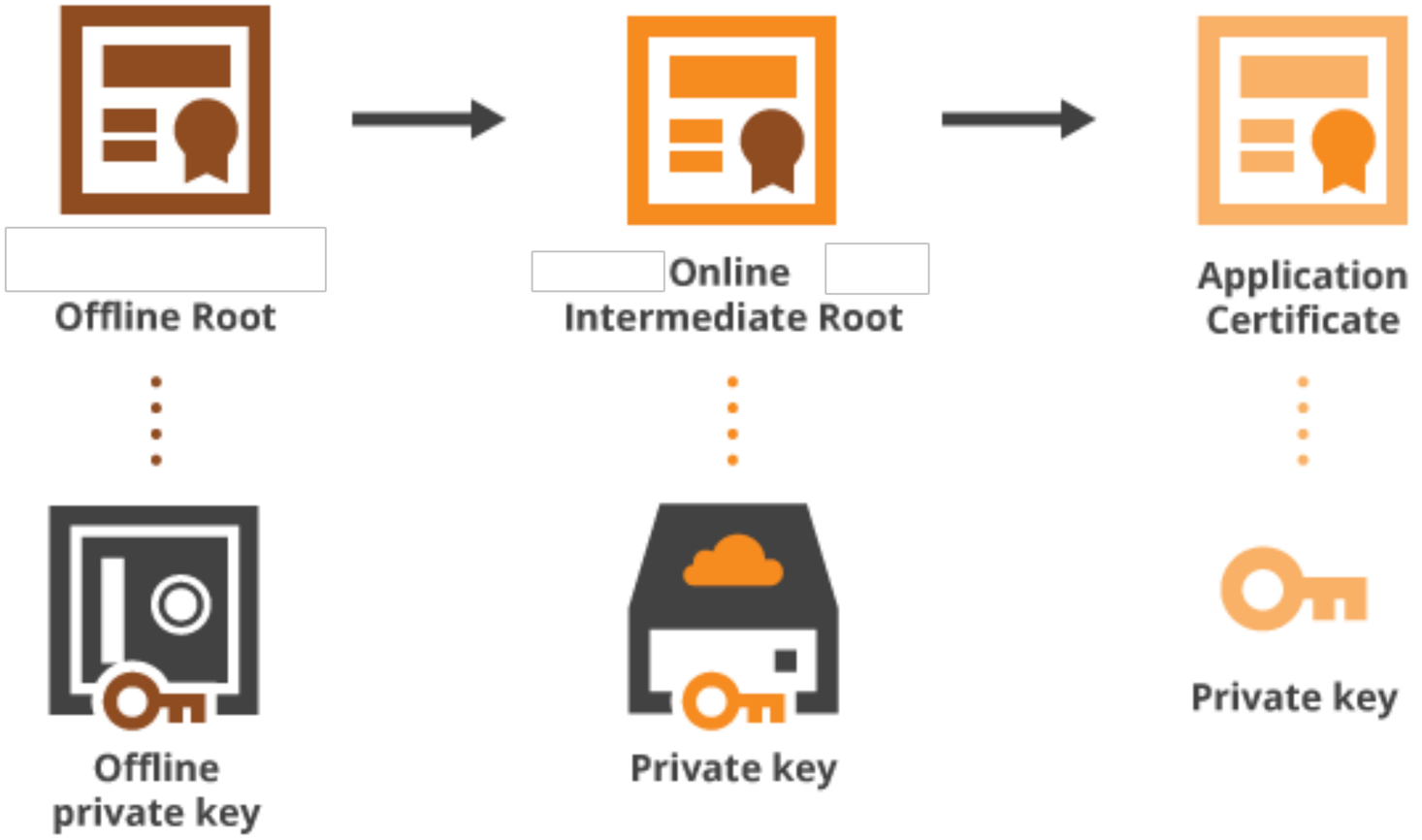
Why the Public Key Infrastructure (PKI) of course…

X.509 Certificates

The anatomy of a certificate



Certificate chain of trust



Implementation bugs

Intentional flaws

Issues of trust

Client Logic

Phase 1: Validate certificate

1. Parse the certificate
2. Find the parent certificate
3. Validate signature against parent’s public key
4. If parent is in trust store, accept
5. If not, repeat

Client Logic

Phase 2: Tie certificate to channel

For RSA:

- Encrypt the key material with public key
- Verify shared key using final message

For Diffie-Hellman:

- Server signs the key derivation parameters
  (Server DH Parameter, Client Random, Server Random)
- Verify signature with certificate public key

Man-in-the-middle attack

Phase 1 (trust verification fail):

- Use untrusted cert

Phase 2 (channel verification fail):

- Use a real certificate, but fake the signature

check_if_ca

```
/* Checks if the issuer of a certificate is a
 * Certificate Authority, or if the certificate is the same
 * as the issuer (and therefore it doesn't need to be a CA).
 *
 * Returns true or false, if the issuer is a CA,
 * or not.
 */
static int
check_if_ca (gnutls_x509_crt_t cert, gnutls_x509_crt_t issuer,
             unsigned int flags)
{
    int result;
    result =
        _gnutls_x509_get_signed_data (issuer->cert, "tbsCertificate",
                                      &issuer_signed_data);
    if (result < 0)
        {
        gnutls_assert ();
        goto cleanup;
        }

    …

    result = 0;

cleanup:
    // cleanup type stuff
    return result;
}
```

SSLVerifySignedServerKeyExchange

```
static OSStatus SSLVerifySignedServerKeyExchange(
        SSLContext *ctx, bool isRsa, SSLBuffer signedParams, uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    ...
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

PKCS #1 v1.5

RSA Signature Format

Encrypted with private key

DigestInfo: ASN-1 data containing Digest Type

Bleichenbacher 2006

RSA Signature Verification allows:

00 01 FF FF FF FF FF FF FF 00	DigestInfo	MessageDigest	**Garbage**

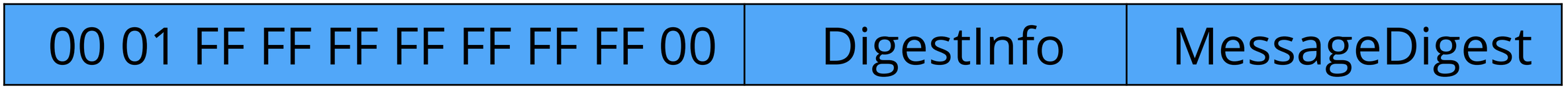
Pick garbage so that:

DigestInfo + MessageDigest + Garbage =

Cube of forged message

BERserk (2014)

RSA Signature Format:



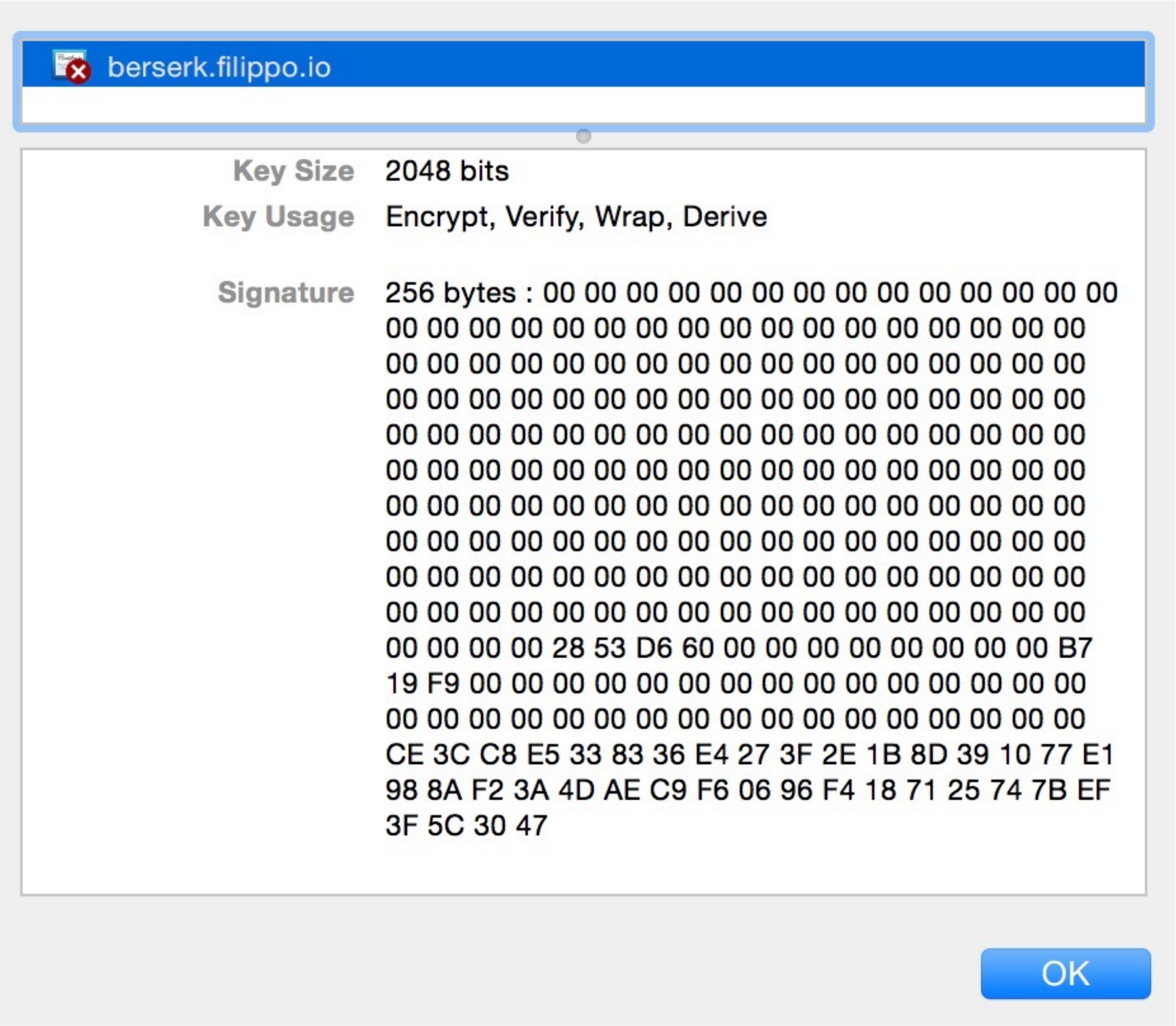
Bug in NSS: Integer overflow in ASN-1 length:

initial bytes of length are ignored

DigestInfo (with garbage length) + MessageDigest =

Cube of forged message

BERserk


berserk.filippo.io
Key Size 2048 bits
Key Usage Encrypt, Verify, Wrap, Derive
Signature 256 bytes : 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 28 53 D6 60 00 00 00 00 00 00 00 B7
19 F9 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
CE 3C C8 E5 33 83 36 E4 27 3F 2E 1B 8D 39 10 77 E1
98 8A F2 3A 4D AE C9 F6 06 96 F4 18 71 25 74 7B EF
3F 5C 30 47
OK

Issues of Trust

How many CAs do you trust?

How many countries have a valid CA?

How many CAs do you trust?

How many countries have a valid CA?

46

(EFF’s SSL Observatory)

Superfish and friends

Local man-in-the-middle proxy installed by your

OEM

anti-virus software

corporate IT department

country-level inspection proxy

Superfish and friends

Some proxies don’t validate certificates correctly.

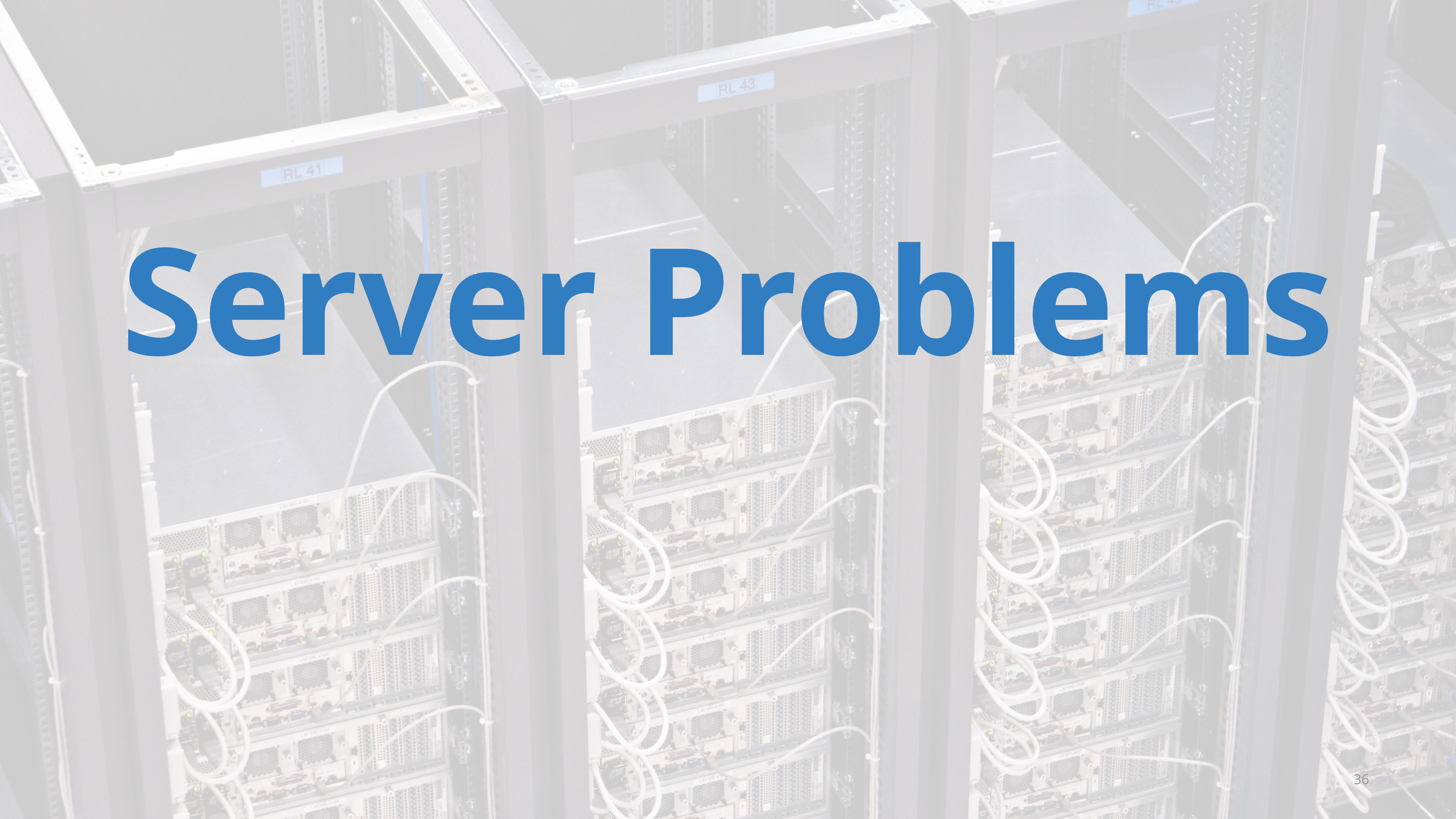
Oops!

Bonus: Locally installed roots bypass key pinning!

The TLS software ecosystem

Client	TLS	PKI
Chrome	NSS/ BoringSSL	Native
Firefox	NSS	mozilla::pkix
Safari	Common Crypto	Common Crypto
Internet Explorer	schannel	schannel

OS	PKI
Windows	schannel
Linux	OpenSSL/GnuTLS
OS X/iOS	Common Crypto



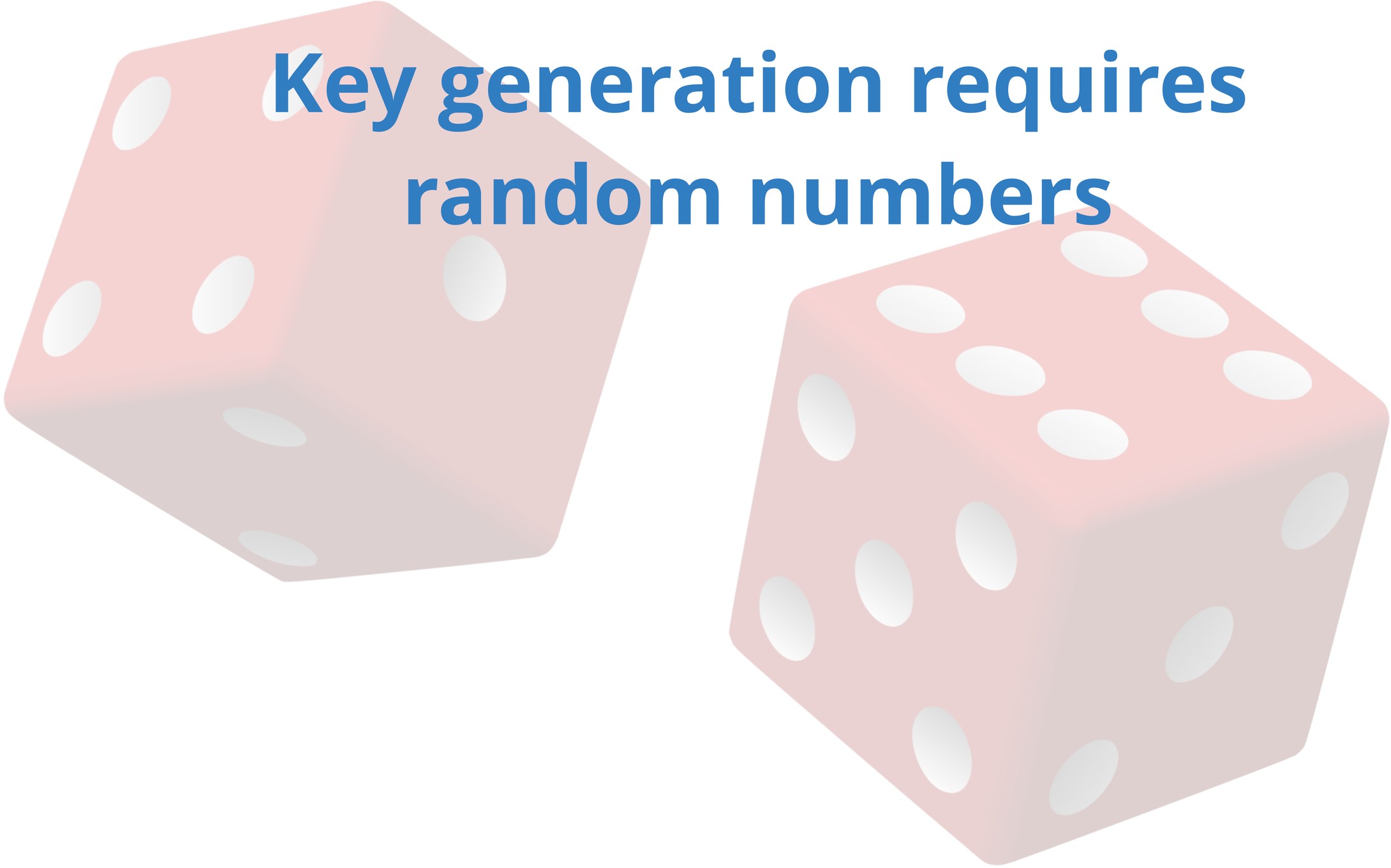
Server Problems

Server Libraries

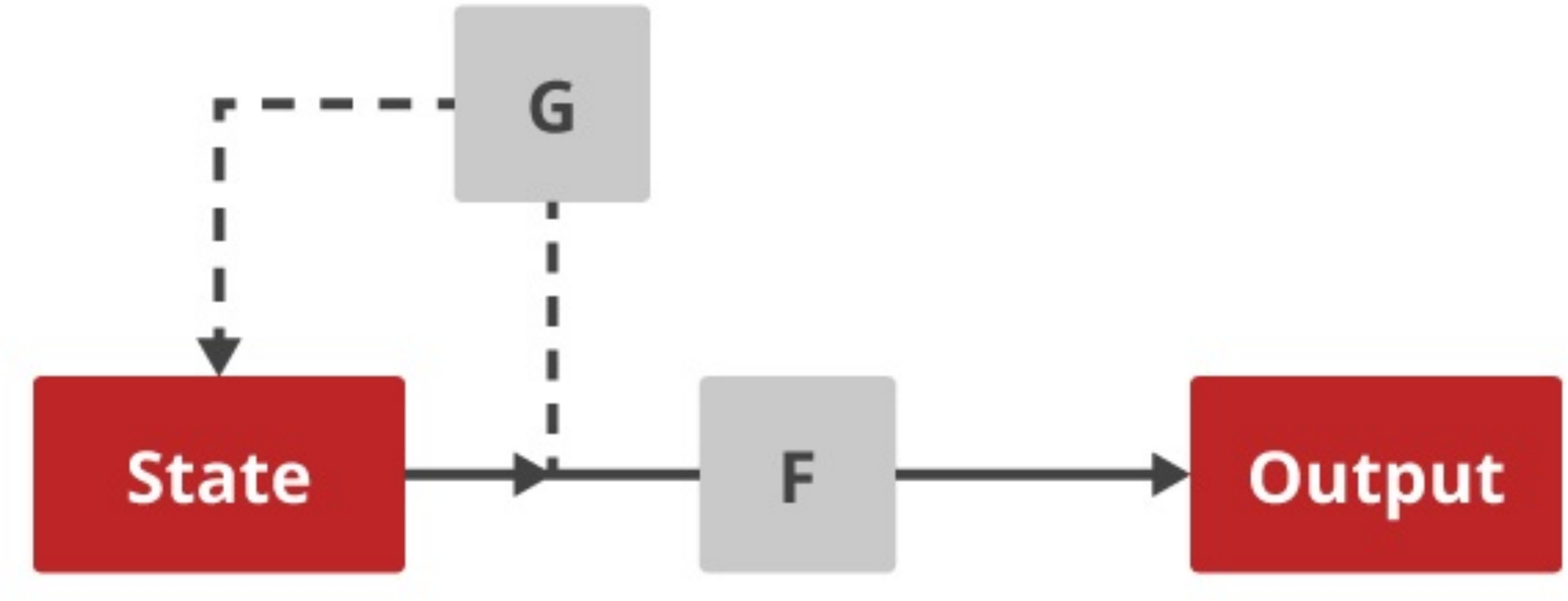
Developer	October 2015	Percent
Apache	303,234,897	34.53%
Microsoft	267,012,322	30.40%
nginx	146,229,307	16.65%
Google	19,931,862	2.27%

Server	TLS
Apache	OpenSSL
nginx	OpenSSL
Microsoft IIS	Schannel

Key generation requires random numbers

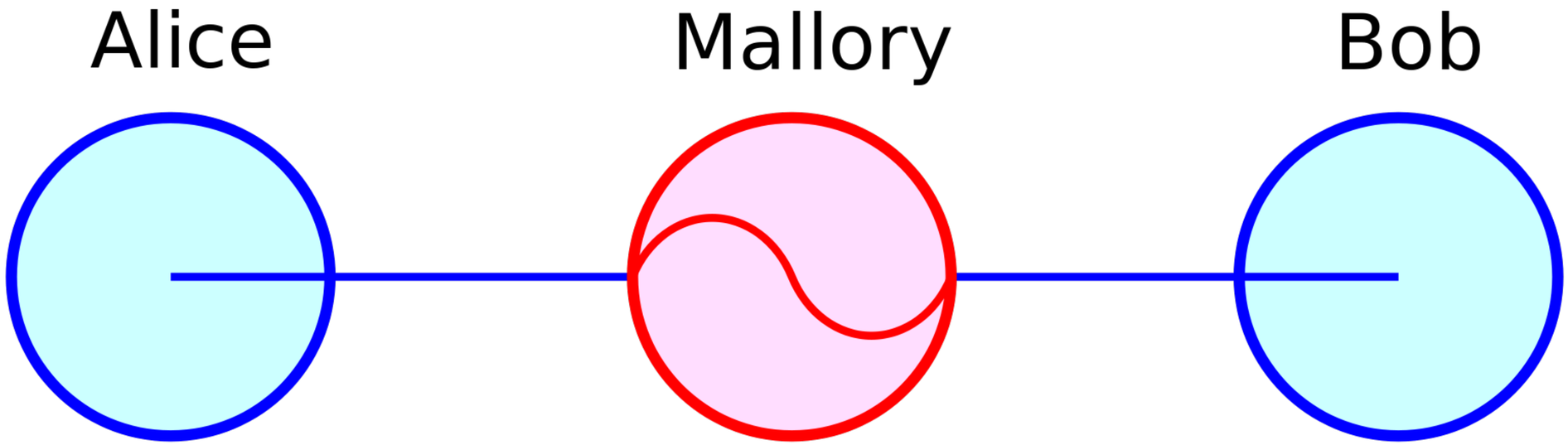


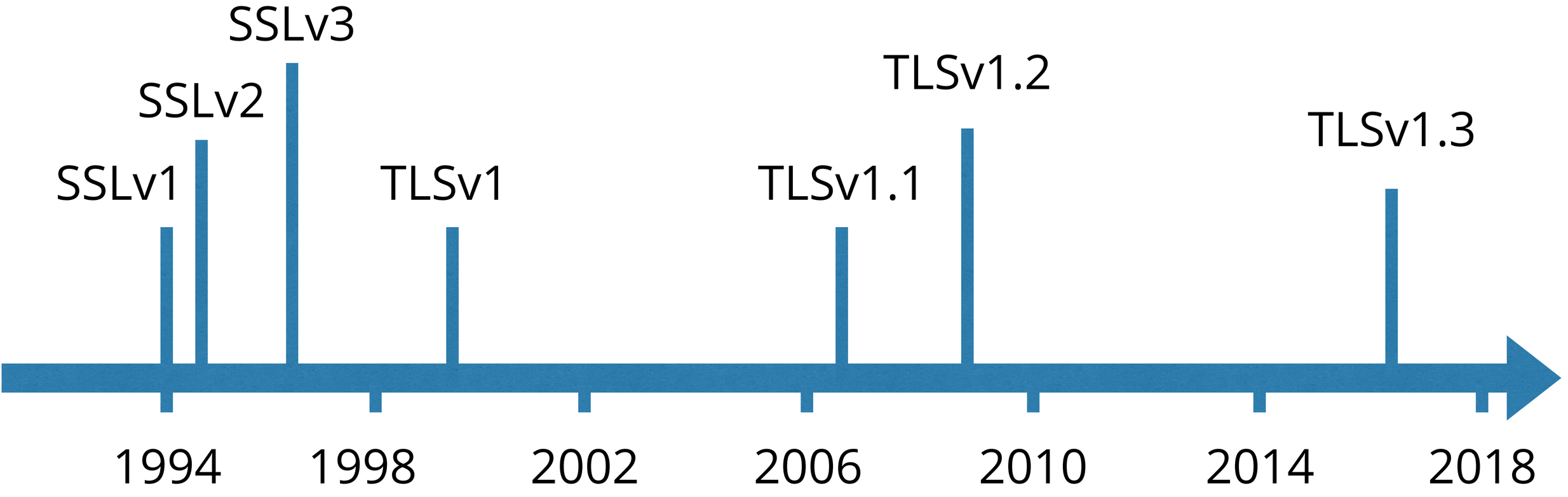
Dual EC DRBG



Heartbleed (2014)

Protocol Bugs



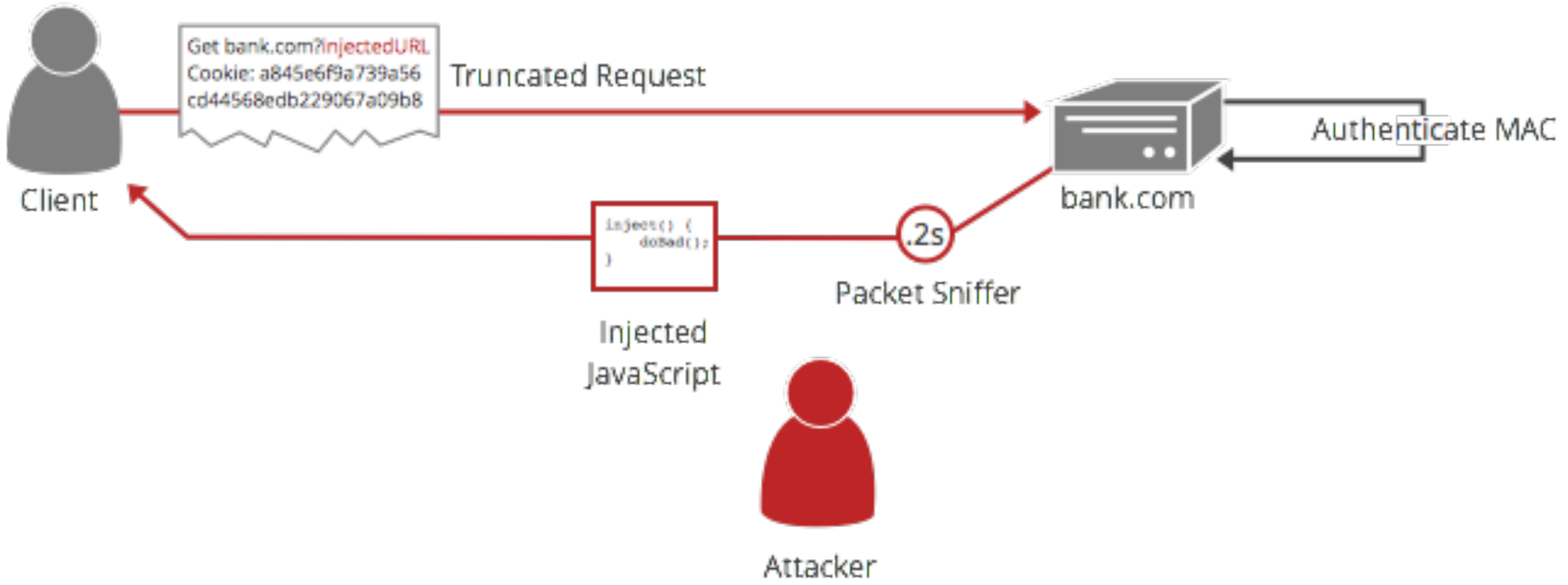

SSLv3
SSLv2
SSLv1
TLSv1
TLSv1.1
TLSv1.2
TLSv1.3
1994
1998
2002
2006
2010
2014
2018

HTTP is great for crypto attacks

“repeated plaintext” = Cookies, passwords, CSRF tokens

“chosen plaintext” = URI

```
GET /<chosen plaintext>
Cookie: <repeated plaintext>
```



Attacker on the local network

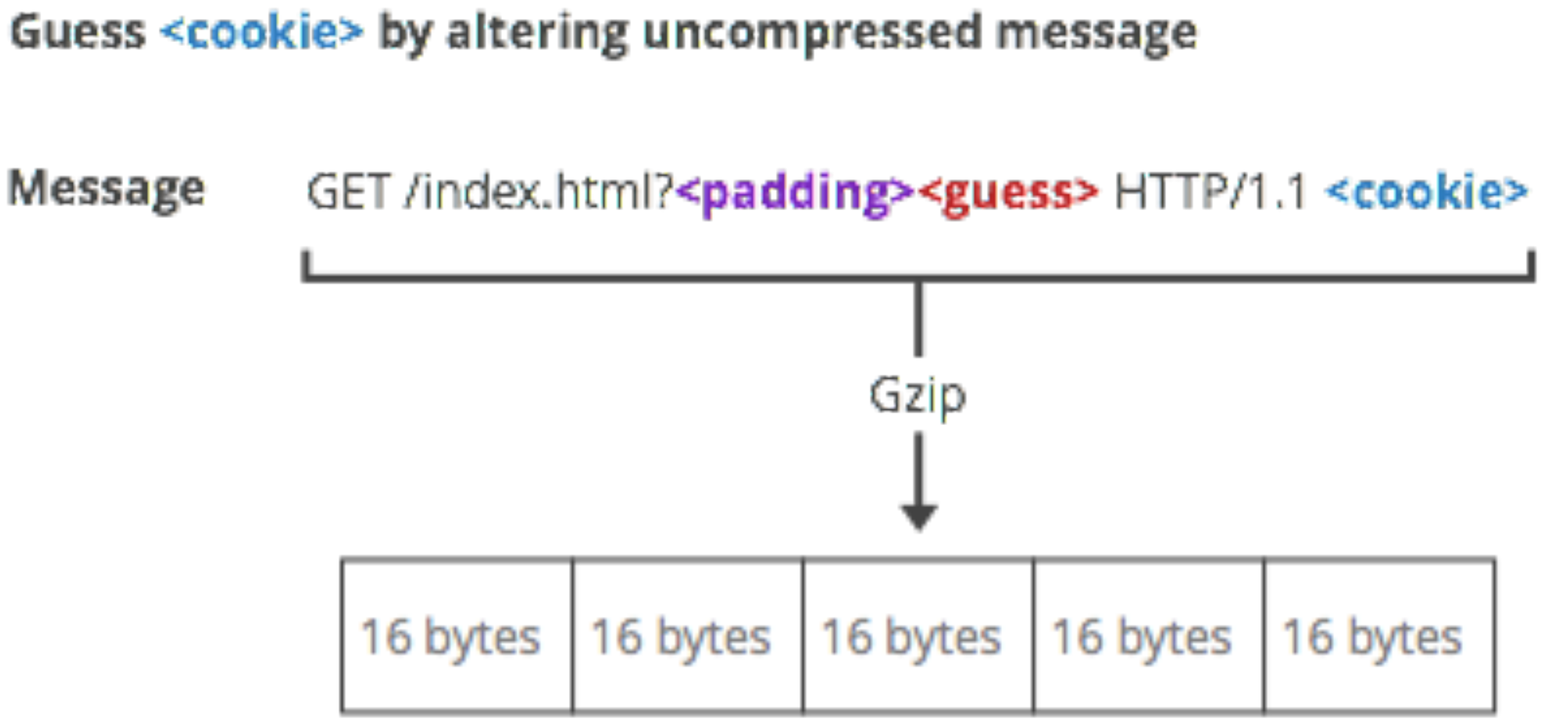
ARP spoofing to get MiTM position

Inject random JS to make cross-origin request

TLS errors can trigger browsers to re-send request

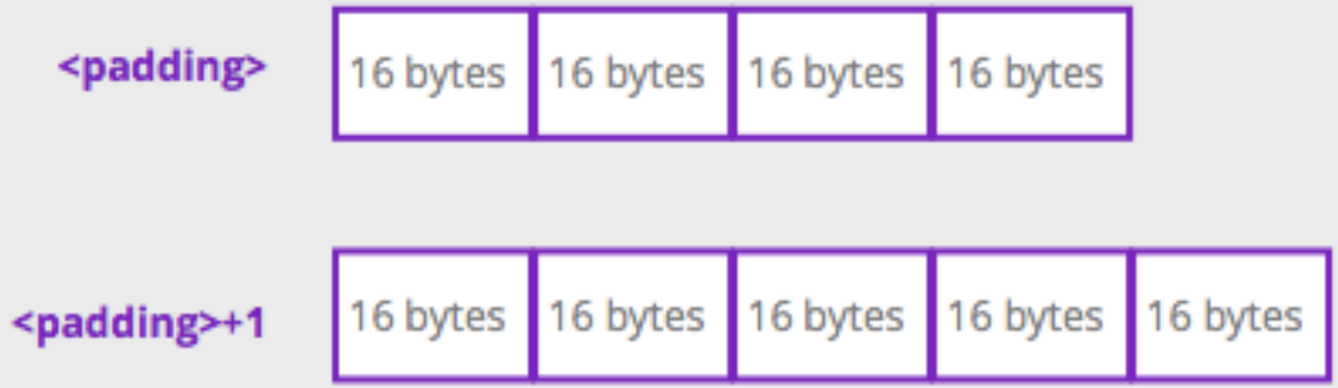
Compression Oracles

CRIME & BREACH

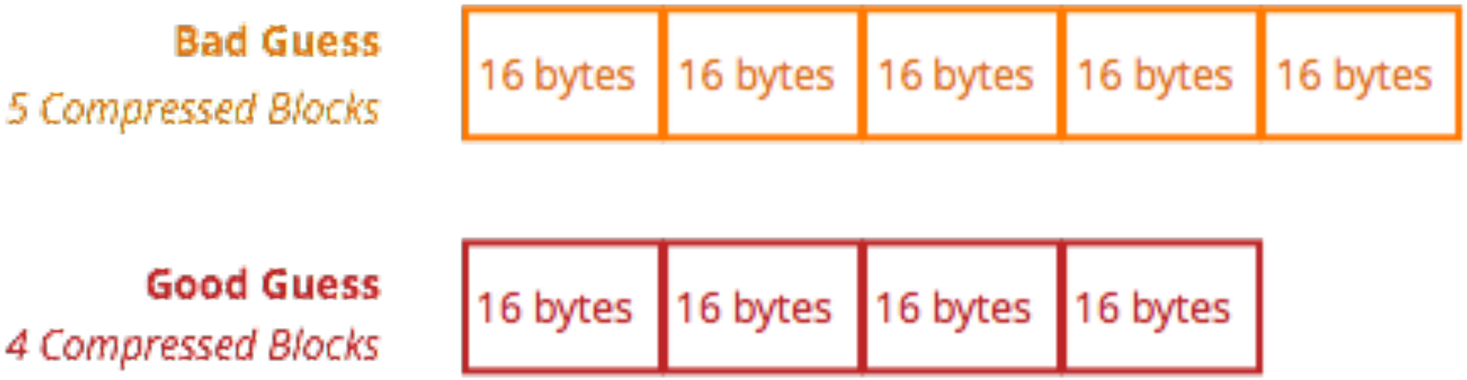


CRIME & BREACH

Choose <padding> to align compressed bytes on 16-byte boundary



Guess all potential bytes in <cookie>



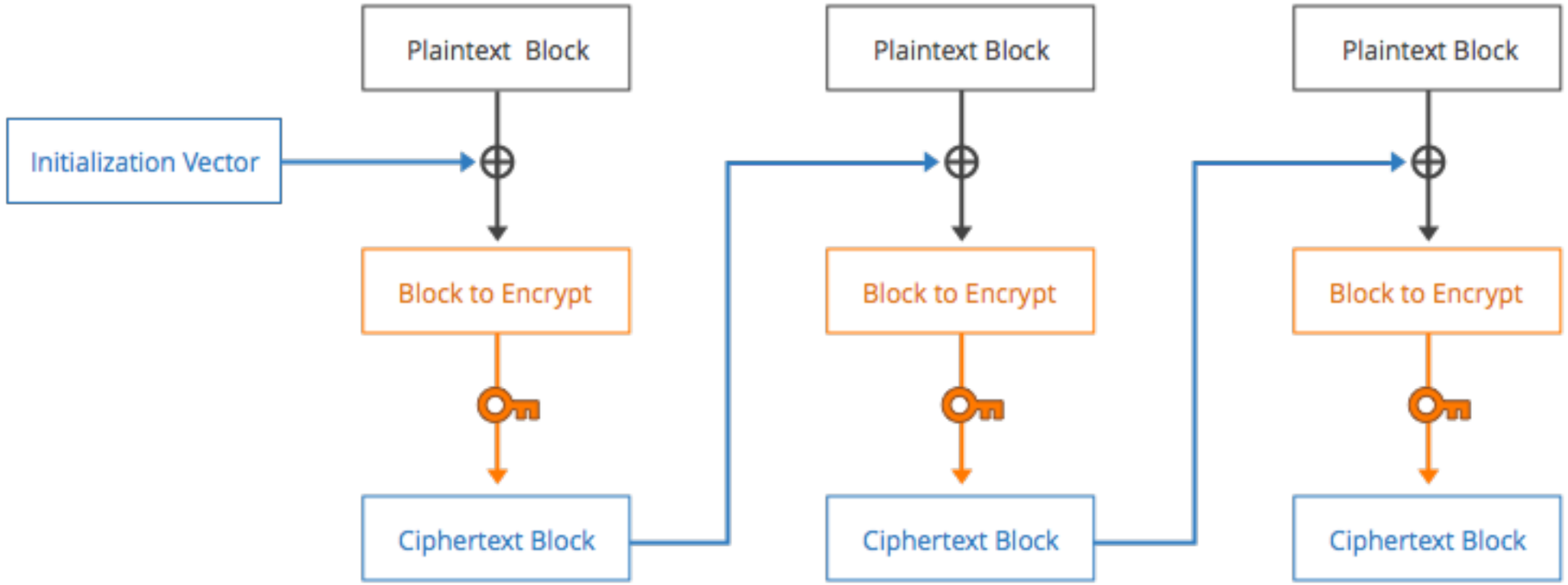
Padding Oracles

CBC mode

MAC-then-Encrypt

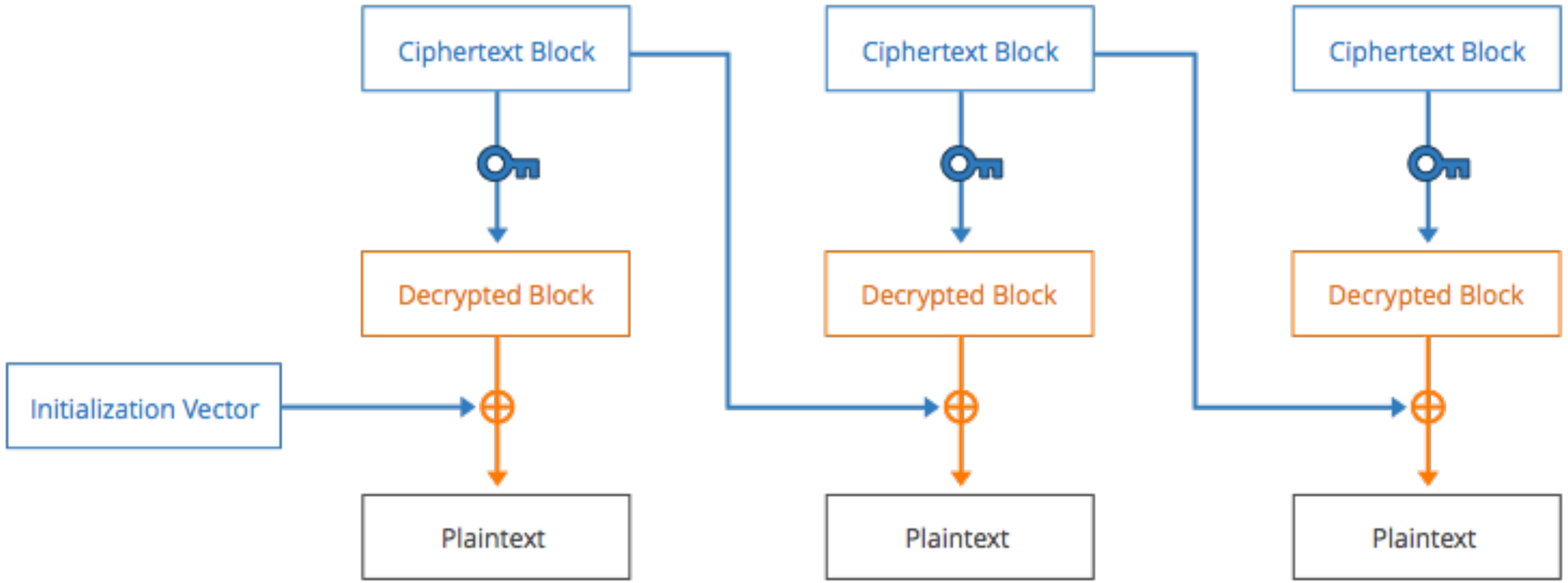
Cipher Block Chaining

CBC Mode Encryption

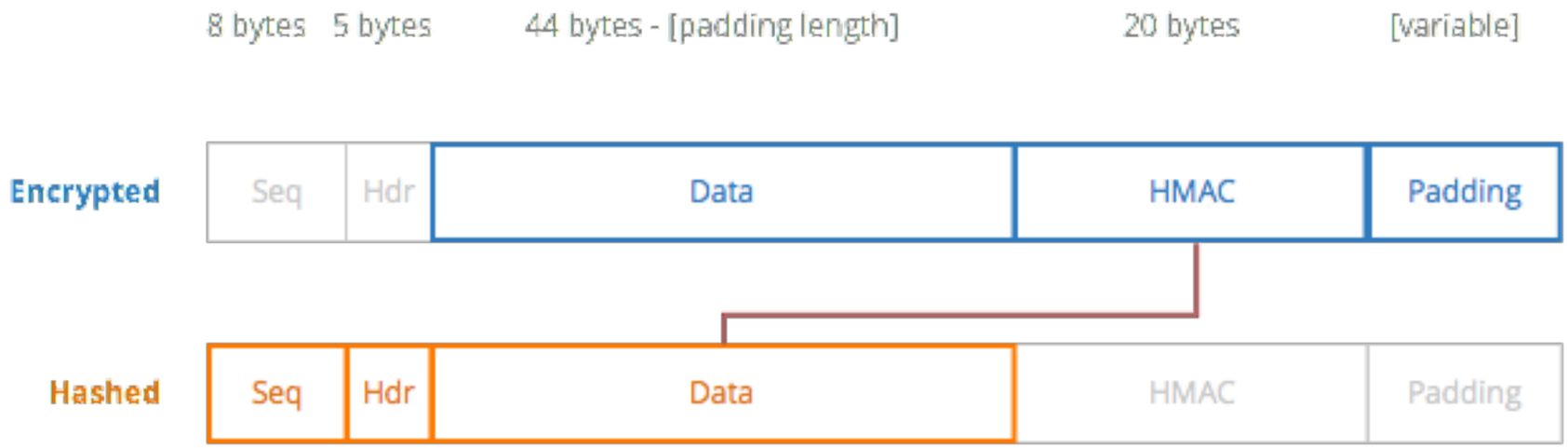


Cipher Block Chaining

CBC Mode Decryption



MAC-then-encrypt



MAC-then-encrypt

Valid padding SSLv3:

0x00

0xXX, 0x01

0xXX, 0xXX, 0x02

etc.

MAC-then-encrypt

Valid padding TLS:

0x00

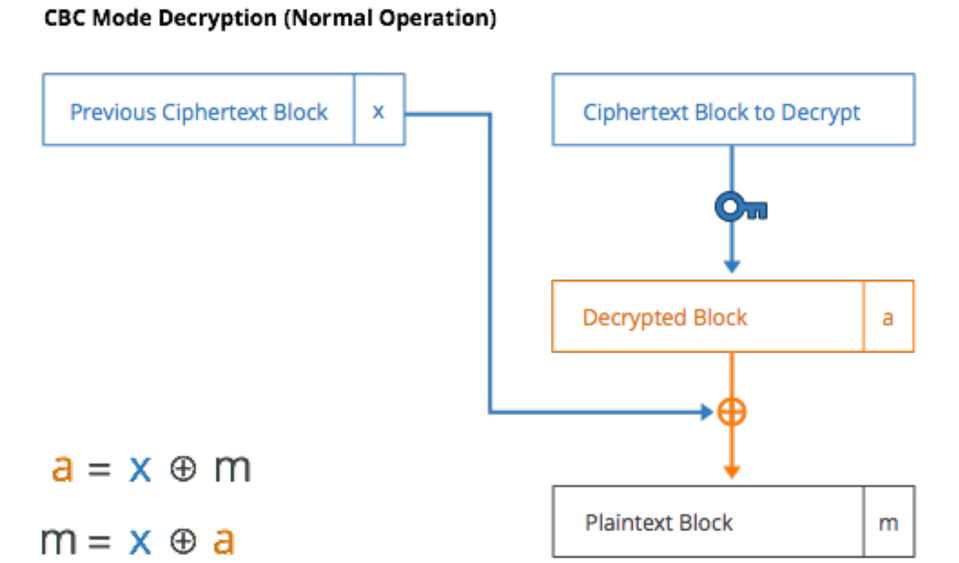
0x01, 0x01

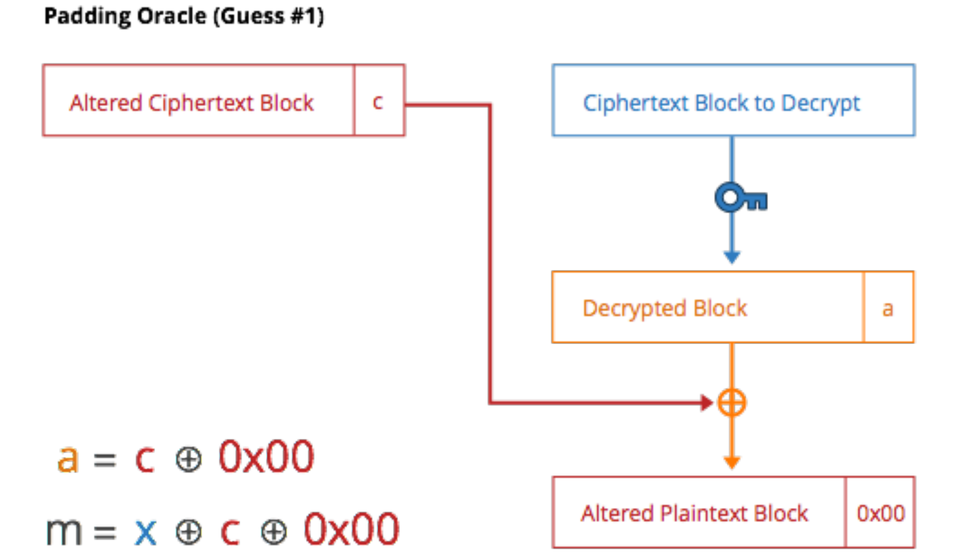
0x02, 0x02, 0x02

etc.

Padding Oracle

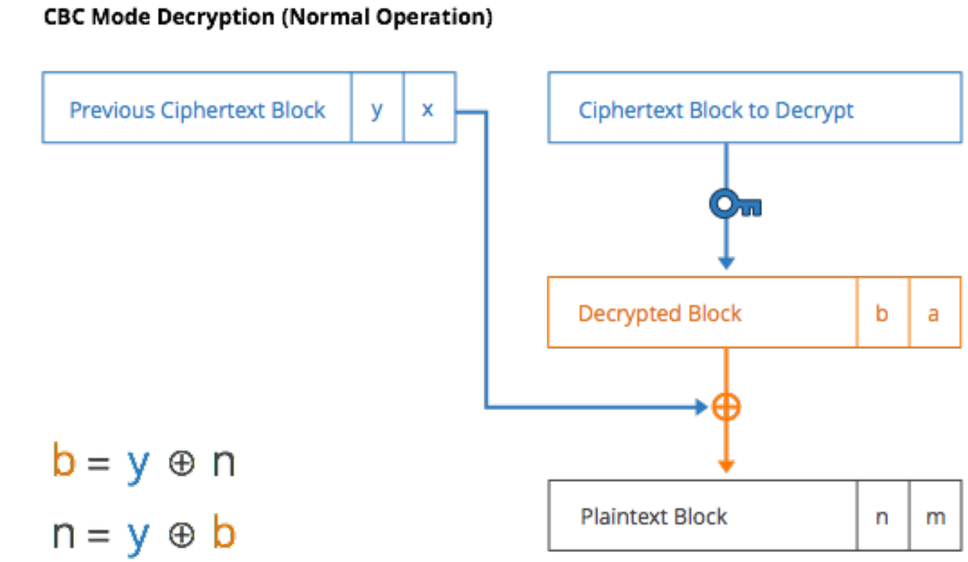
Step 1

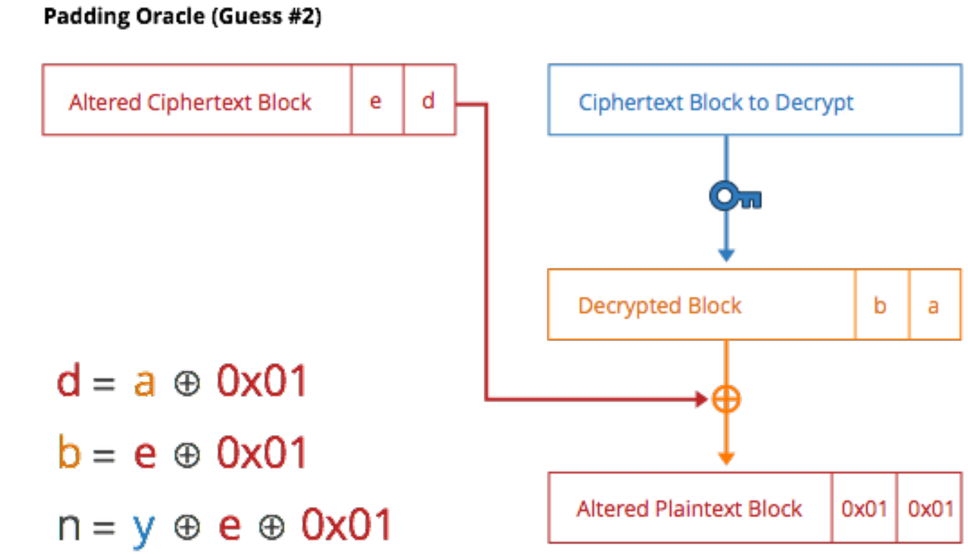




Padding Oracle

Step 2





Padding Oracle History

Error code as a side channel

Timing as a side channel

Timing redux: Lucky 13

Error code side channel

incorrect guess:

bad padding

correct guess:

bad HMAC

Timing side-channel

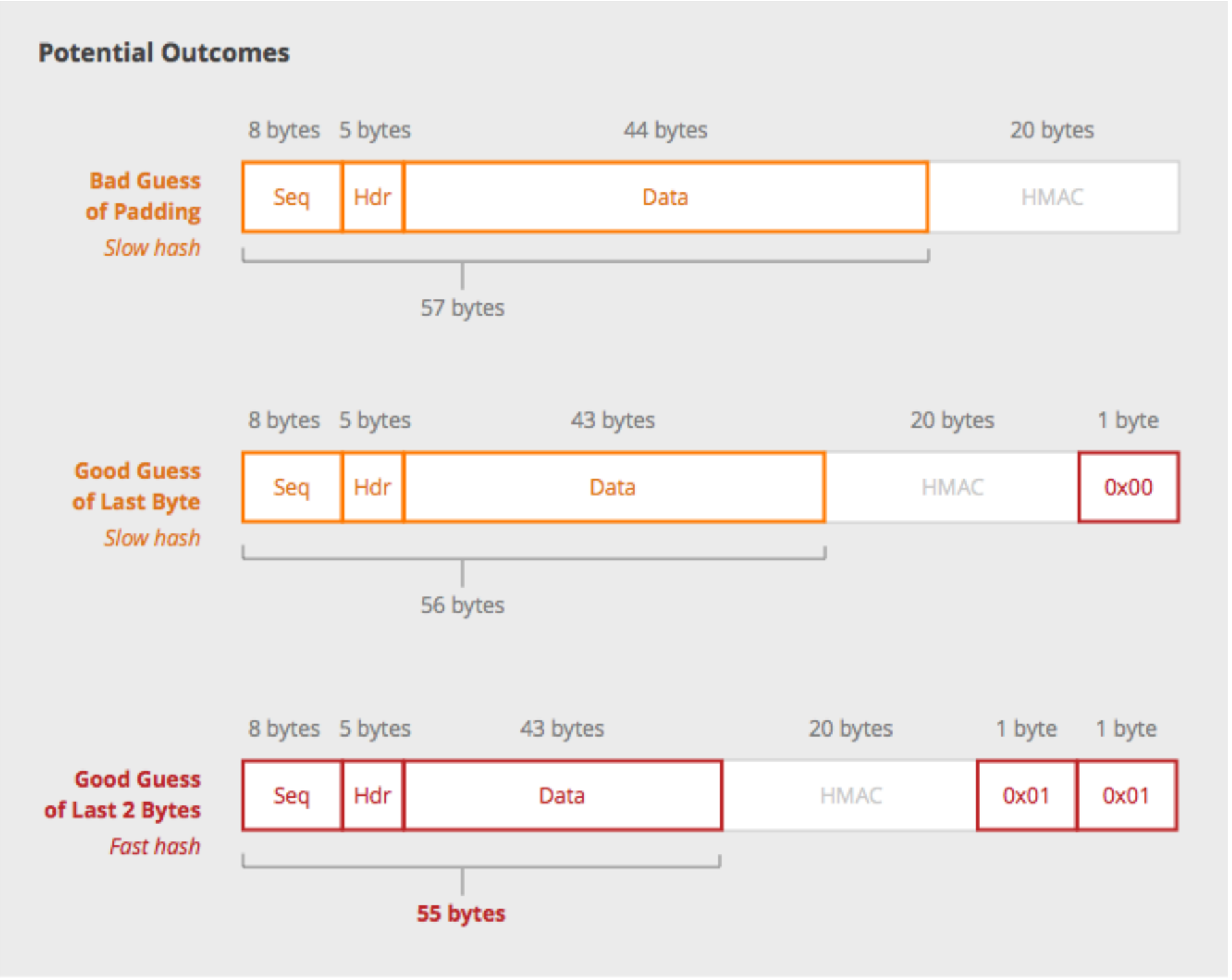
incorrect guess:

no HMAC computation (fast)

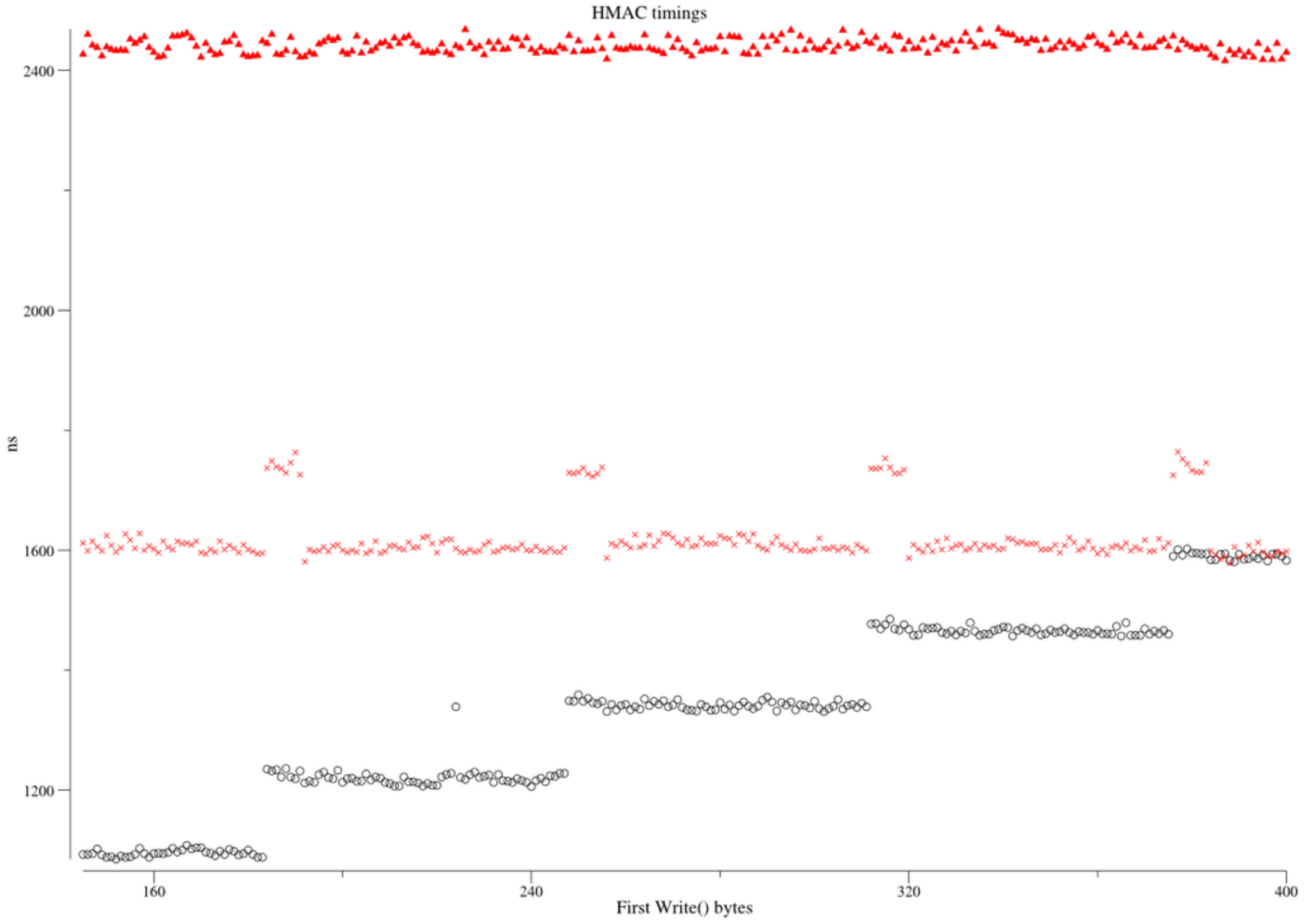
correct guess:

HMAC computation (slow)

Lucky 13


Potential Outcomes
8 bytes
5 bytes
44 bytes
20 bytes
Bad Guess of Padding
Slow hash
Seq
Hdr
Data
HMAC
57 bytes
8 bytes
5 bytes
43 bytes
20 bytes
1 byte
Good Guess of Last Byte
Slow hash
Seq
Hdr
Data
HMAC
0x00
56 bytes
8 bytes
5 bytes
43 bytes
20 bytes
1 byte
1 byte
Good Guess of Last 2 Bytes
Fast hash
Seq
Hdr
Data
HMAC
0x01
0x01
55 bytes

Lucky 13


HMAC timings
2400
2000
1600
1200
ns
160
240
320
400
First Write() bytes

Downgrade attacks

Downgrade attacks

If you support something old,

someone’s going to trick you into using it.

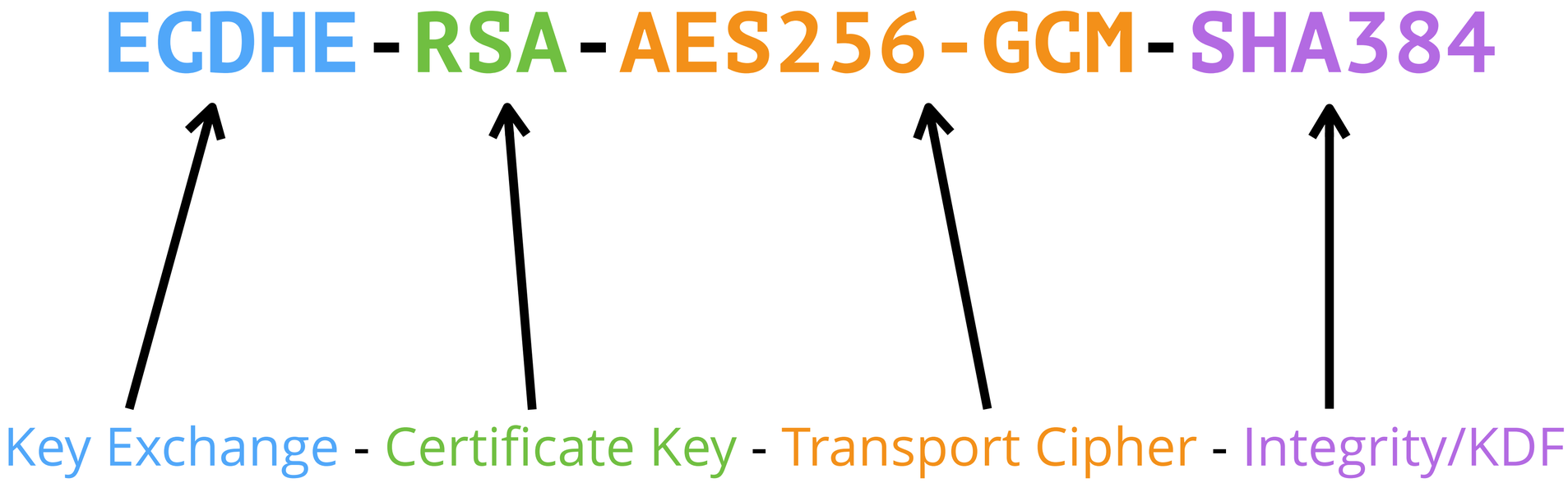
Cipher Suites

- Complicated string describing the type of crypto used
- Check your ciphers with `$ openssl ciphers`

Example:

```
ECDHE-RSA-AES128-GCM-SHA256:
ECDHE-ECDSA-AES128-GCM-SHA256:
ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-ECDSA-AES256-GCM-SHA384:
DHE-RSA-AES128-GCM-SHA256:
```

Cipher Suites



Cipher Suite Negotiation

AES256-GCM-SHA384:
ECDHE-RSA-AES128-GCM-SHA256:
ECDHE-RSA-AES256-GCM-SHA384:

ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-RSA-AES256-SHA:
ECDHE-RSA-AES128-SHA256:
AES256-GCM-SHA384:

ECDHE-RSA-AES256-GCM-SHA384
AES256-GCM-SHA384

ECDHE-RSA-AES256-GCM-SHA384

Export Ciphers

RSA-EXPORT-WITH-RC4-40-MD5

RSA-EXPORT-WITH-DES40-CBC-SHA

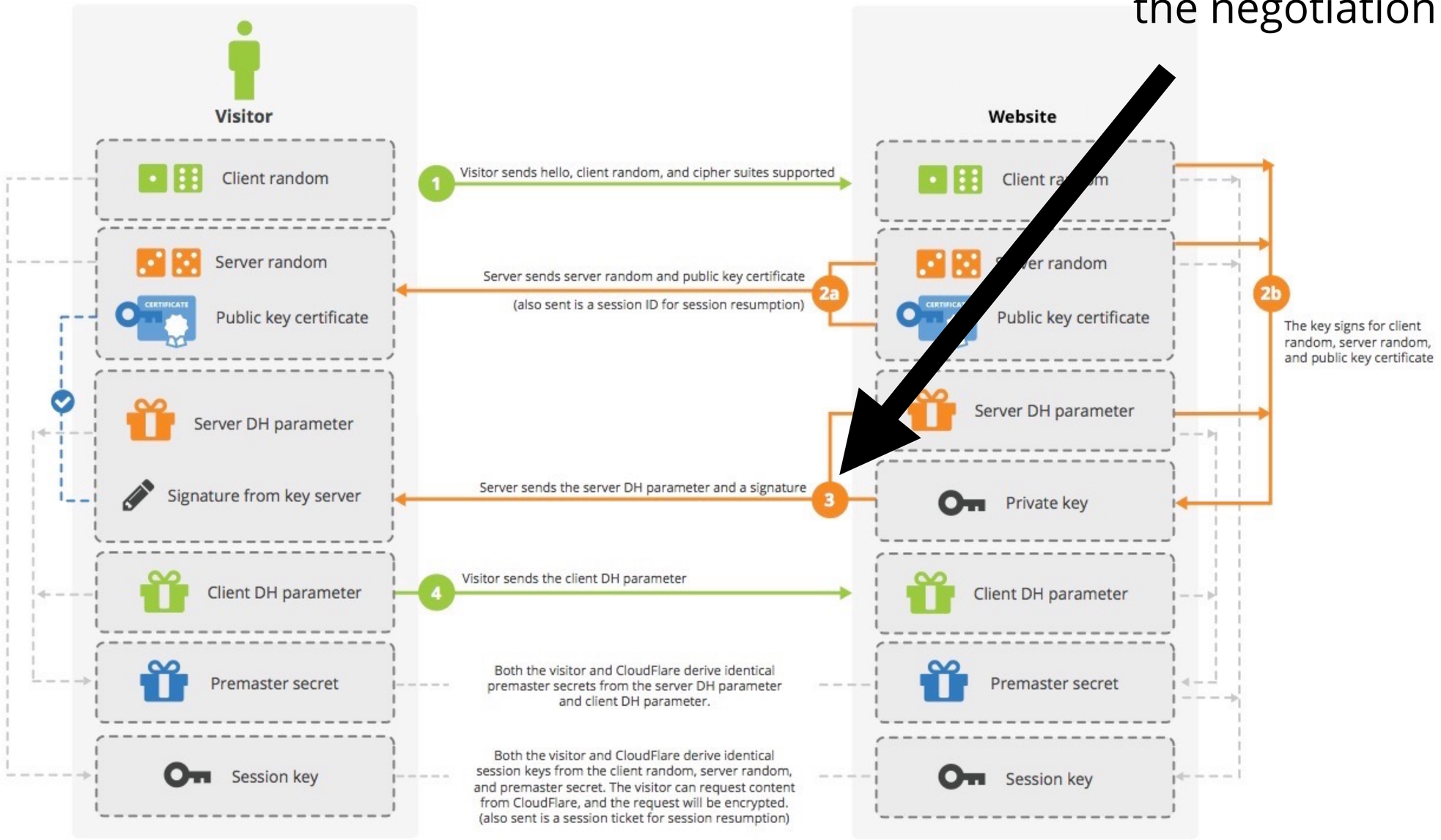
DHE-DSS-EXPORT-WITH-RC4-56-SHA

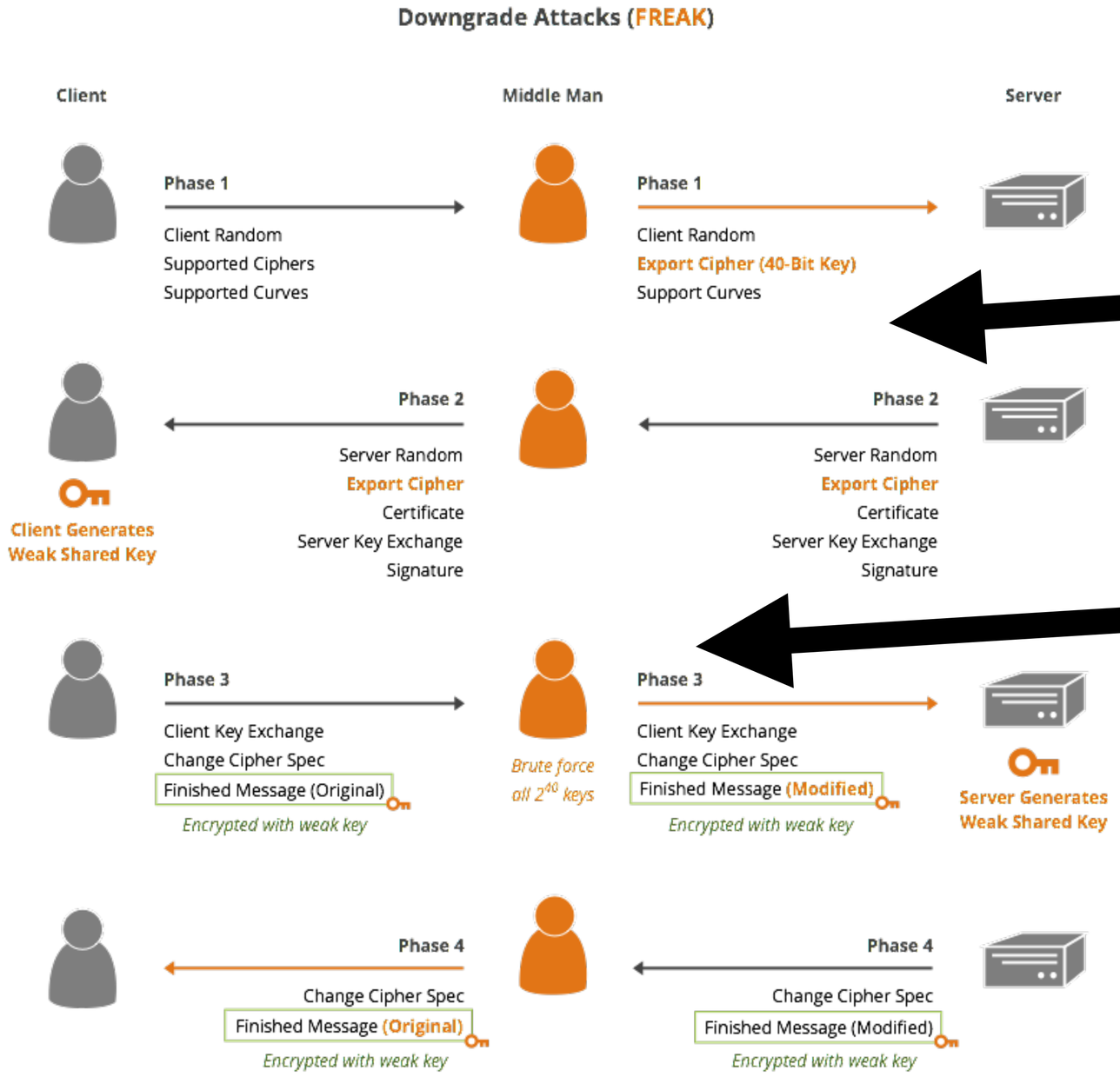
Enter FREAK, LogJam, WeakDH

SSL Handshake (Diffie-Hellman)

Handshake

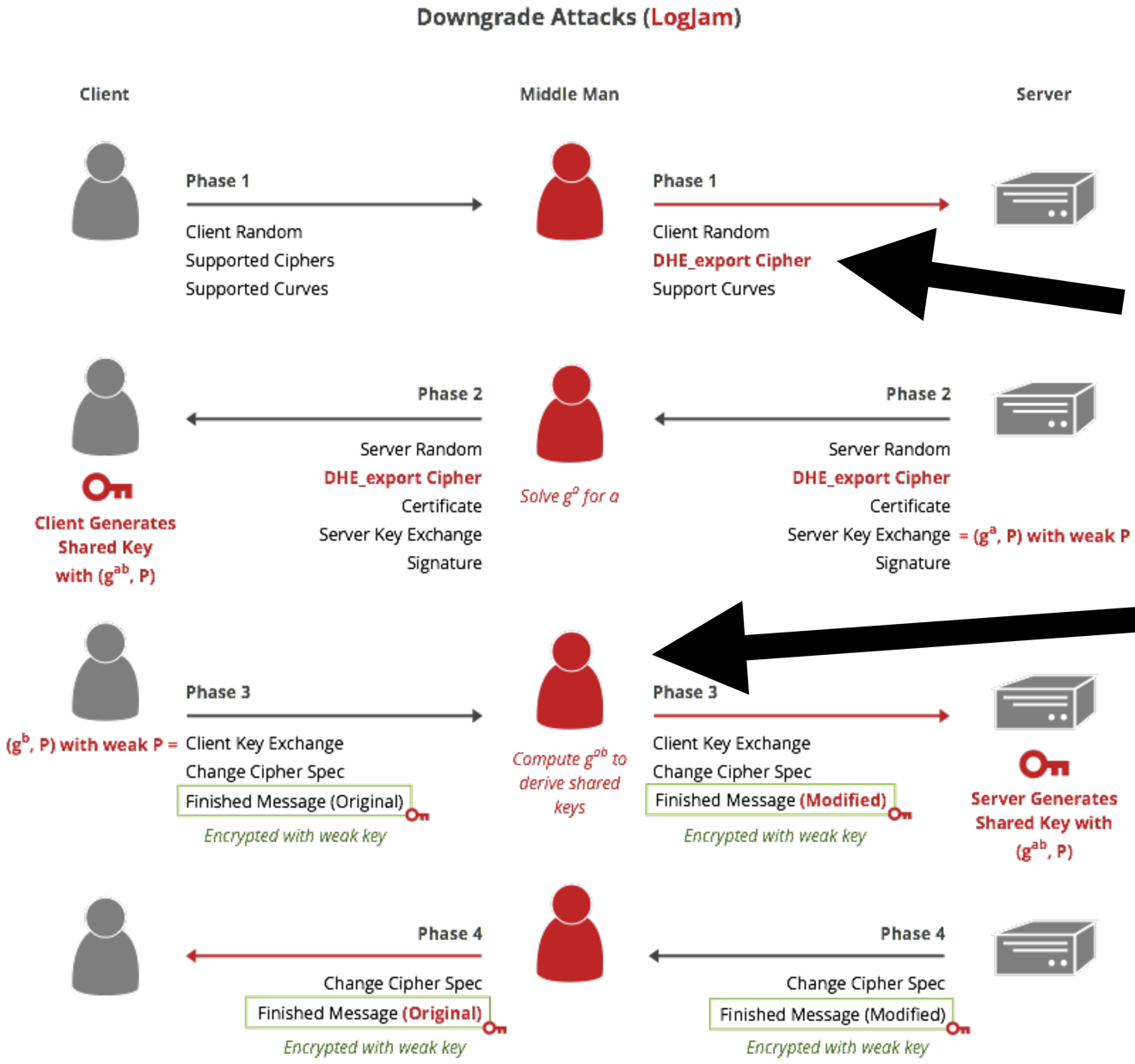
Only the key generation material is signed, not the negotiation





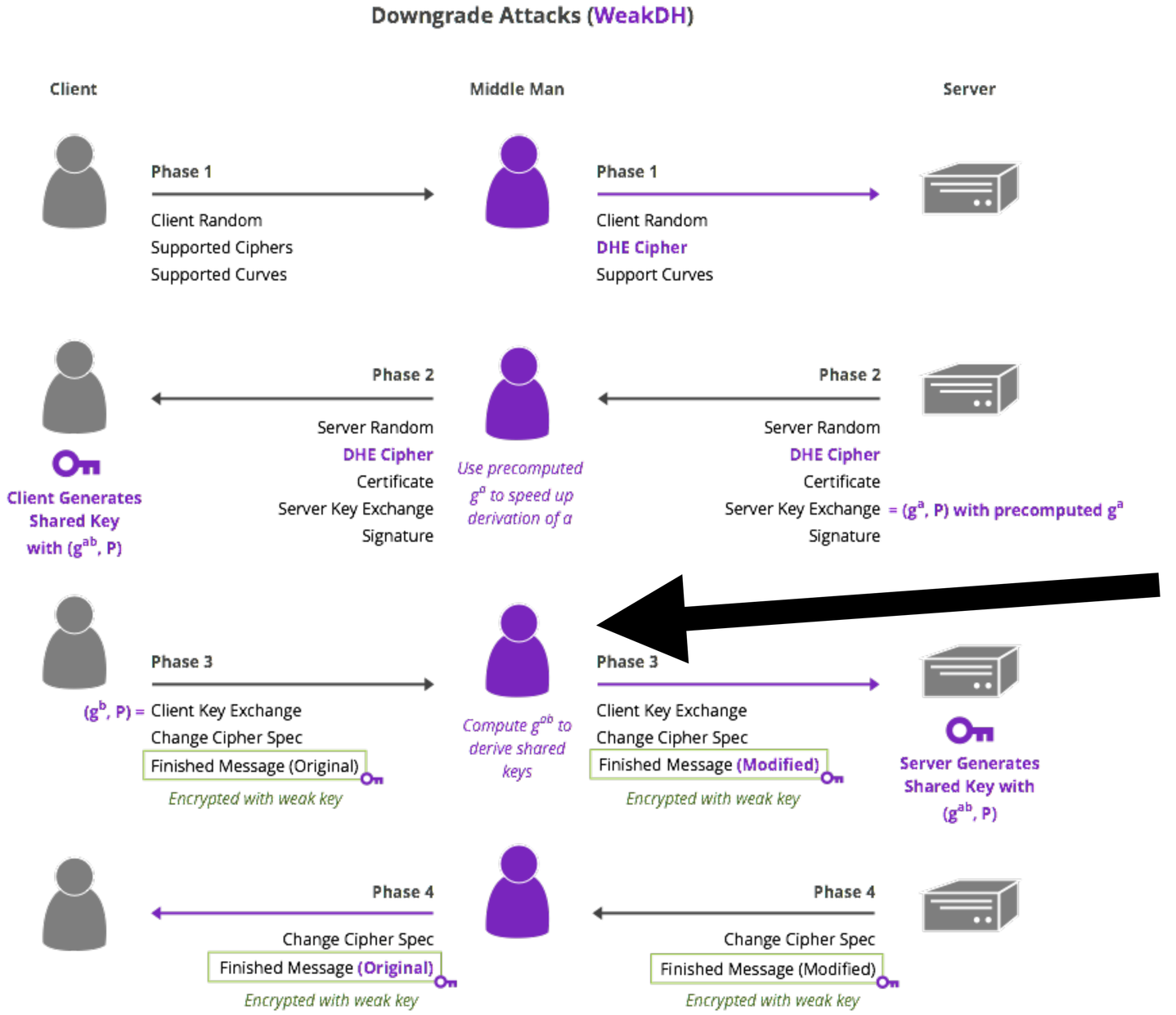
Swap Supported Ciphers with Export Ciphers

Crack Export Key



Swap Supported Ciphers with DHE Export Ciphers

Crack Export DH Param



Crack DH Param

LogJam, WeakDH, FREAK

Unauthenticated protocol data in handshake

More to come on this…

POODLE

Padding oracle and a downgrade attack

- Downgrade dance (thanks browsers!)
- Line things up so that padding is in last block
- Swap target block with padding block
- Around 256 guesses per byte

TLS POODLE

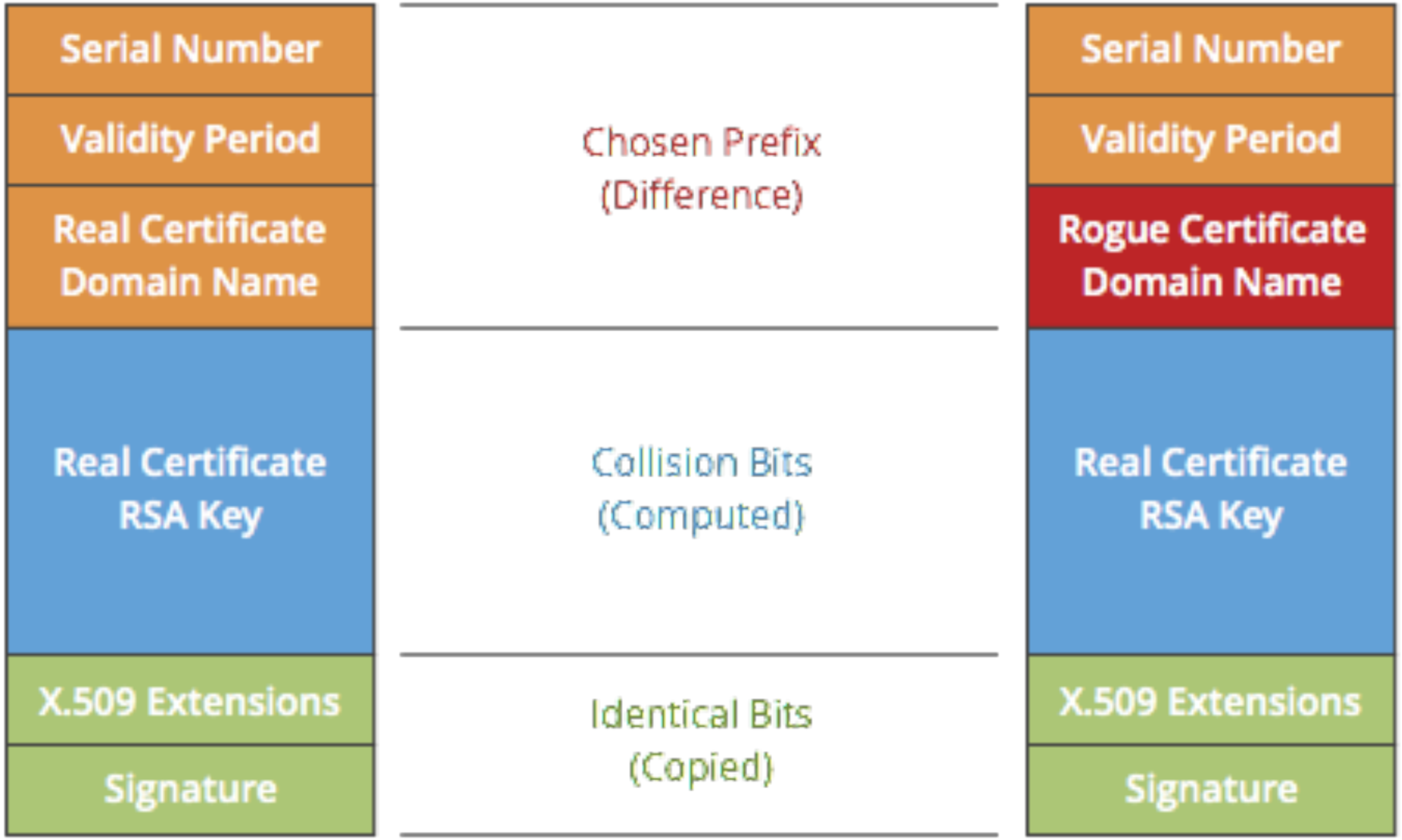
Padding oracle,

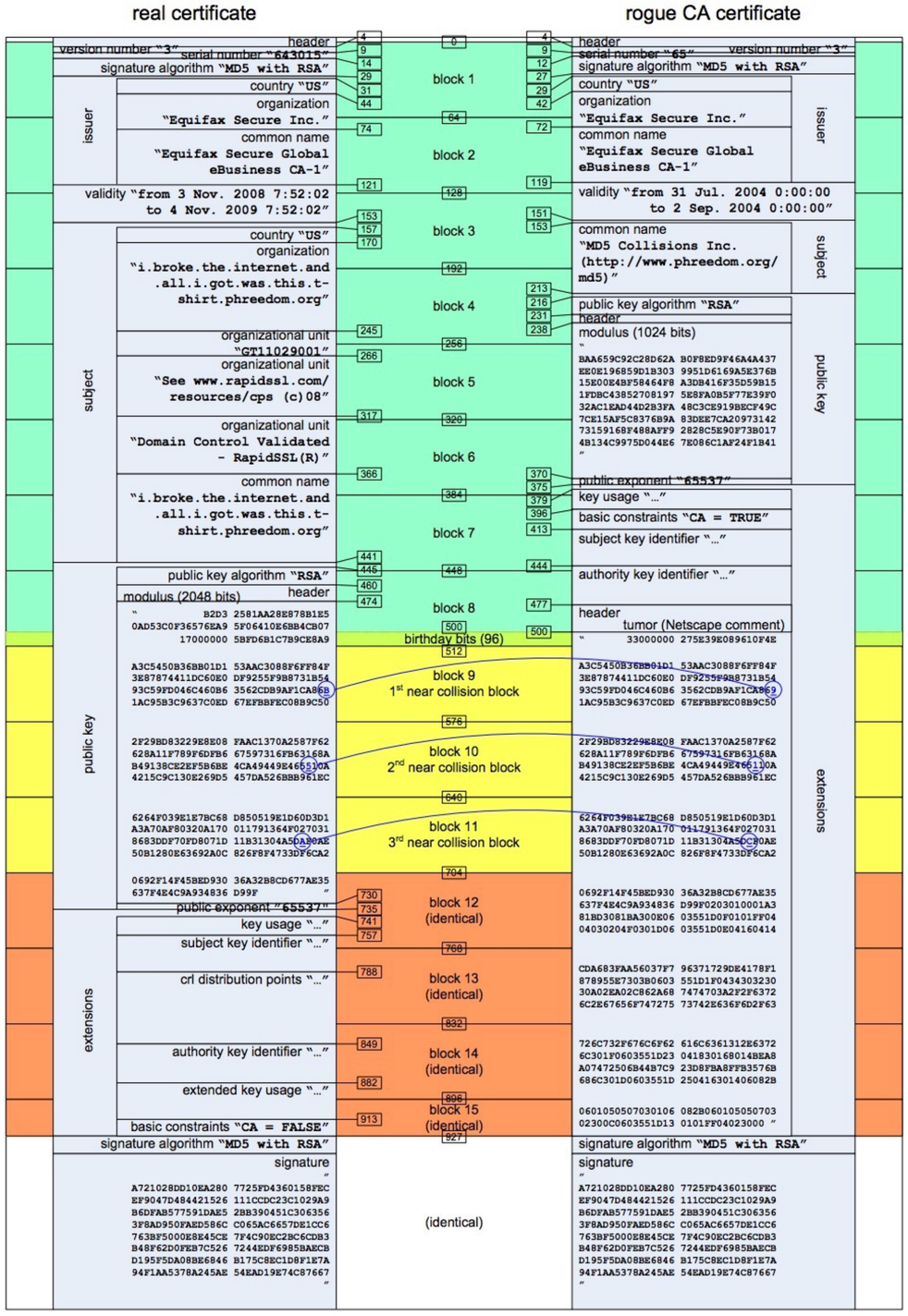
Downgrade attack,

Implementation bug

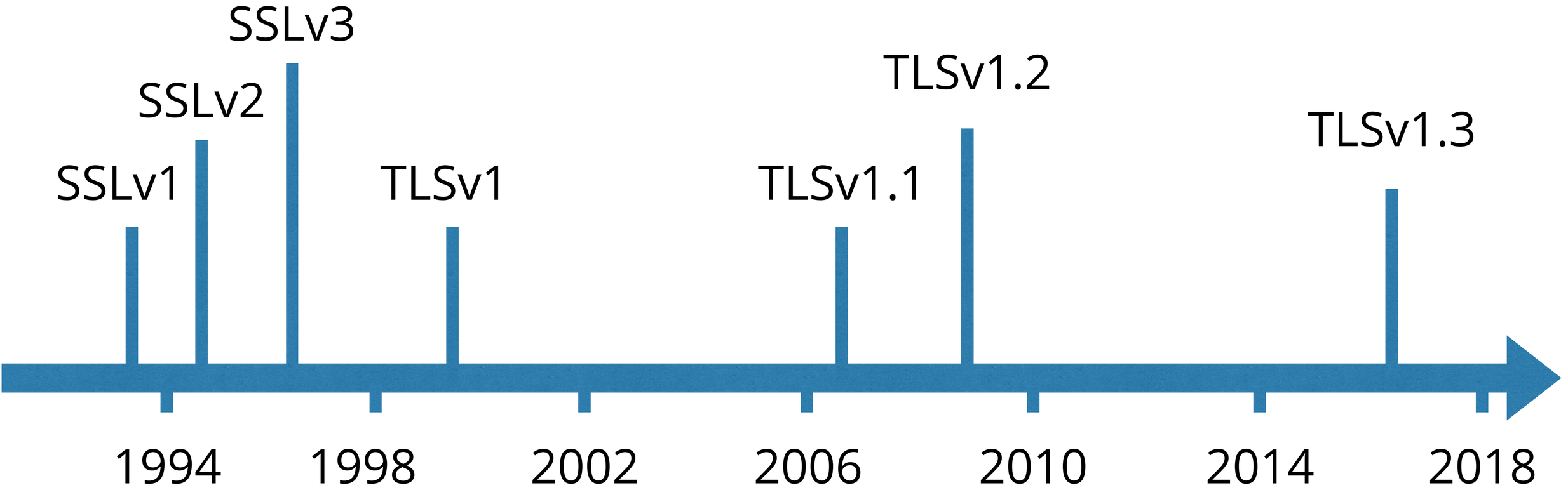
Aging Crypto

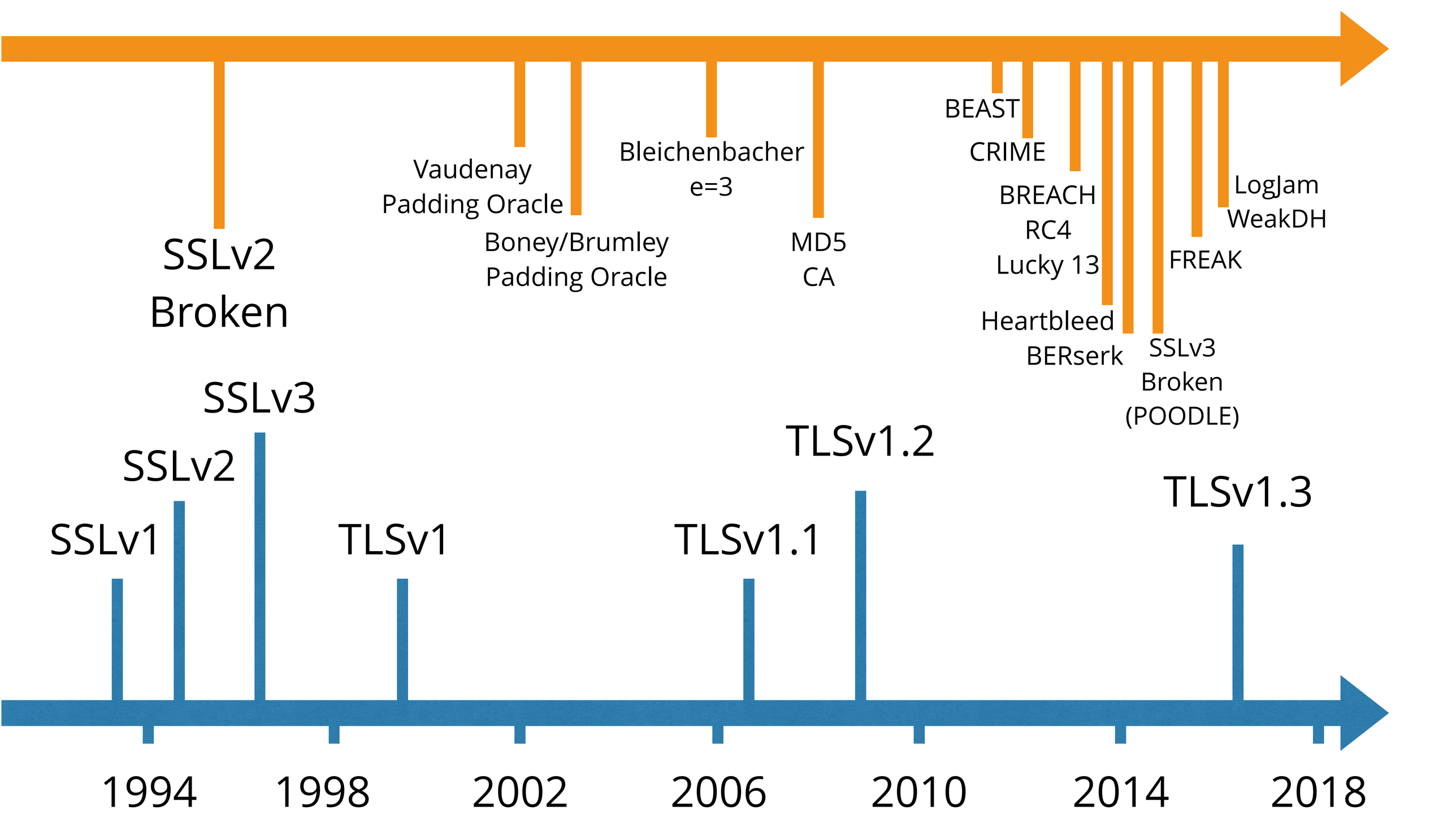
MD5 Collision

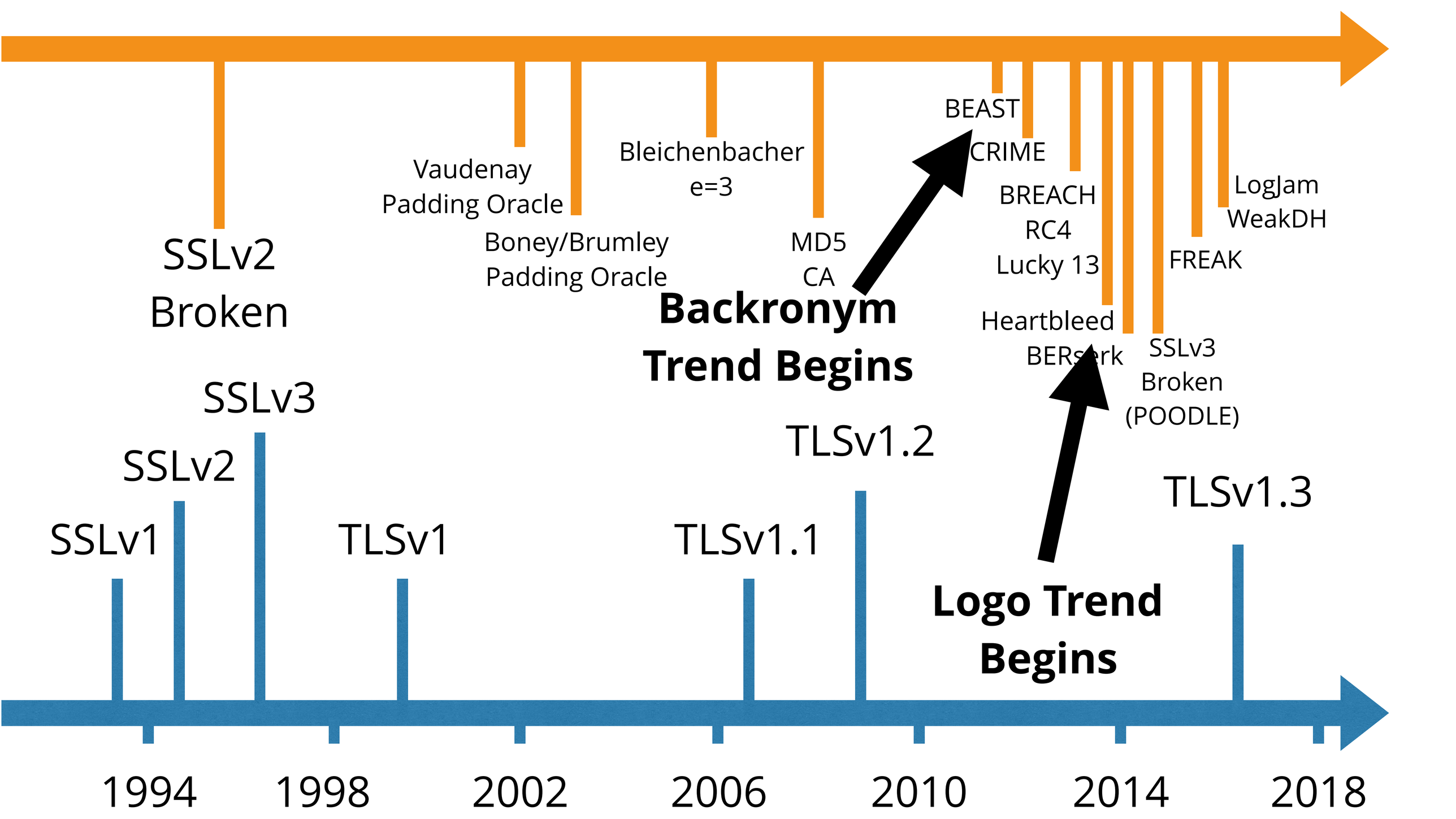


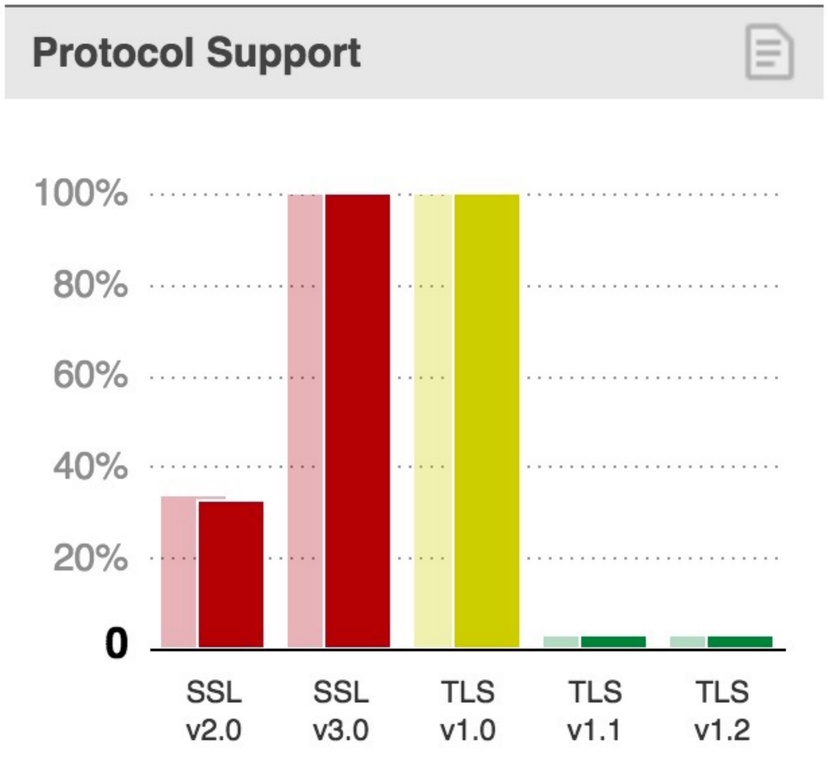

real certificate
rogue CA certificate
header
version number "3"
serial number "643015"
signature algorithm "MD5 with RSA"
issuer
country "US"
organization "Equifax Secure Inc."
common name "Equifax Secure Global eBusiness CA-1"
validity "from 3 Nov. 2008 7:52:02 to 4 Nov. 2009 7:52:02"
subject
country "US"
organization "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
organizational unit "GT11029001"
organizational unit "See www.rapidssl.com/resources/cps (c)08"
organizational unit "Domain Control Validated - RapidSSL(R)"
common name "i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom.org"
public key
public key algorithm "RSA"
modulus (2048 bits)
B2D3 2581AA28E878B1E5
0AD53C0F36576EA9 5F06410E6BB4CB07
17000000 5BFD6B1C7B9CE8A9
A3C5450B36BB01D1 53AAC3088F6FF84F
3E87874411DC60E0 DF9255F9B8731B54
93C59FD046C460B6 3562CDB9AF1CA86B
1AC95B3C9637C0ED 67EFBBFEC08B9C50
2F29BD83229E8E08 FAAC1370A2587F62
628A11F789F6DFB6 67597316FB63168A
B49138CE2EF5B6BE 4CA49449E46511OA
4215C9C130E269D5 457DA526BBB961EC
6264F039E1E7BC68 D850519E1D60D3D1
A3A70AF80320A170 011791364F027031
8683DDF70FD8071D 11B31304A5DC30AE
50B1280E63692A0C 826F8F4733DF6CA2
0692F14F45BED930 36A32B8CD677AE35
637F4E4C9A934836 D99F
public exponent "65537"
extensions
key usage "..."
subject key identifier "..."
crl distribution points "..."
authority key identifier "..."
extended key usage "..."
basic constraints "CA = FALSE"
signature algorithm "MD5 with RSA"
signature
A721028DD10EA280 7725FD4360158FEC
EF9047D484421526 111CCDC23C1029A9
B6DFAB577591DAE5 2BB390451C306356
3F8AD950FAED586C C065AC6657DE1CC5
763BF5000E8E45CE 7F4C90EC2BC6CDB3
B48F62D0FEB7C526 7244EDF6985BAECB
D195F5DA08BE6846 B175C8EC1D8F1E7A
94F1AA5378A245AE 54EAD19E74C87667
block 1
block 2
block 3
block 4
block 5
block 6
block 7
block 8
birthday bits (96)
block 9
1st near collision block
block 10
2nd near collision block
block 11
3rd near collision block
block 12
(identical)
block 13
(identical)
block 14
(identical)
block 15
(identical)
(identical)
header
serial number "65"
version number "3"
signature algorithm "MD5 with RSA"
issuer
country "US"
organization "Equifax Secure Inc."
common name "Equifax Secure Global eBusiness CA-1"
validity "from 31 Jul. 2004 0:00:00 to 2 Sep. 2004 0:00:00"
subject
common name "MD5 Collisions Inc. (http://www.phreedom.org/md5)"
public key
public key algorithm "RSA"
header
modulus (1024 bits)
BAA659C92C28D62A B0F8ED9F46A4A437
EE0E196859D1B303 9951D6169A5E376B
15E00E48F50464F8 A3DB416F35D59B15
1FDBC43852708197 5E8FA0B5F77E39F0
32AC1EAD44D2B3FA 48C3CE919BECF49C
7CE15AF5C8376B9A 83DEE7CA20973142
73159168F488AFF9 2828C5E90F73B017
4B134C9975D044E6 7E086C1AF24F1B41
public exponent "65537"
extensions
key usage "..."
basic constraints "CA = TRUE"
subject key identifier "..."
authority key identifier "..."
header
tumor (Netscape comment)
33000000 275E39E089610F4E
A3C5450B36BB01D1 53AAC3088F6FF84F
3E87874411DC60E0 DF9255F9B8731B54
93C59FD046C460B6 3562CDB9AF1CA869
1AC95B3C9637C0ED 67EFBBFEC08B9C50
2F29BD83229E8E08 FAAC1370A2587F62
628A11F789F6DFB6 67597316FB63168A
B49138CE2EF5B6BE 4CA49449E46511OA
4215C9C130E269D5 457DA526BBB961EC
6264F039B1E7BC68 D850519E1D60D3D1
A3A70AF80320A170 011791364F027031
8683DDF70FD8071D 11B31304A5DC30AE
50B1280E63692A0C 826F8F4733DF6CA2
0692F14F45BED930 36A32B8CD677AE35
637F4E4C9A934836 D99F0203010001A3
81BD3081BA300E06 03551D0F0101FF04
04030204F0301D06 03551D0E04160414
CDA683FAA56037F7 96371729DE4178F1
878955E7303B0603 551D1F0434303230
30A02EA02C862A68 7474703A2F2F6372
6C2E67656F747275 73742E636F6D2F63
726C732F676C6F62 616C6361312E6372
6C301F0603551D23 04183016801 4BEA8
A07472506B44B7C9 23D8FBA8FFB3576B
686C301D0603551D 250416014060828
0601050507030106 082B060105050703
02300C0603551D13 0101FF04023000
signature algorithm "MD5 with RSA"
signature
A721028DD10EA280 7725FD4360158FEC
EF9047D484421526 111CCDC23C1029A9
B6DFAB577591DAE5 2BB390451C306356
3FBAD950FAED586C CD65AC6657DE1CC5
763BF5000E8E45CE 7F4C90EC2BC6CDB3
B48F62D0FEB7C526 7244EDF6985BAECB
D195F5DA08BE6846 B175C8EC1D8F1E7A
94F1AA5378A245AE 54EAD19E74C87667

How do these fit on a timeline?

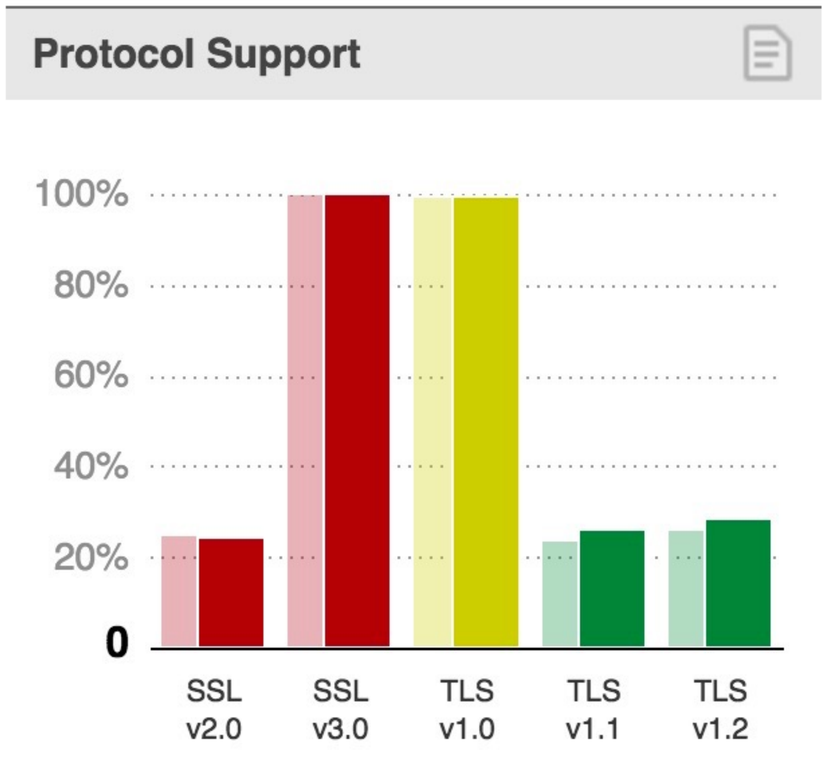

SSLv3
SSLv2
SSLv1
TLSv1
TLSv1.1
TLSv1.2
TLSv1.3
1994
1998
2002
2006
2010
2014
2018


SSLv2
Broken
Vaudenay
Padding Oracle
Boney/Brumley
Padding Oracle
Bleichenbacher
e=3
MD5
CA
BEAST
CRIME
BREACH
RC4
Lucky 13
Heartbleed
BERserk
SSLv3
Broken
(POODLE)
FREAK
LogJam
WeakDH
SSLv1
SSLv2
SSLv3
TLSv1
TLSv1.1
TLSv1.2
TLSv1.3
1994
1998
2002
2006
2010
2014
2018

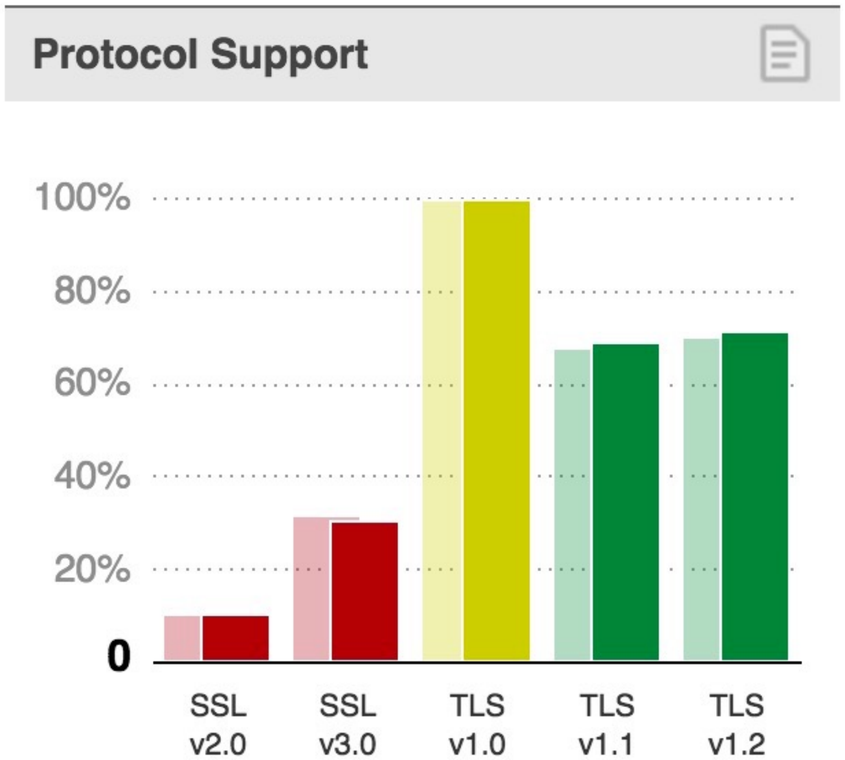

SSLv2
Broken
Vaudenay
Padding Oracle
Boney/Brumley
Padding Oracle
Bleichenbacher
e=3
MD5
CA
BEAST
CRIME
BREACH
RC4
Lucky 13
Heartbleed
BERserk
FREAK
LogJam
WeakDH
SSLv3
Broken
(POODLE)
Backronym
Trend Begins
SSLv3
SSLv2
SSLv1
TLSv1
TLSv1.1
TLSv1.2
TLSv1.3
Logo Trend
Begins
1994
1998
2002
2006
2010
2014
2018



May 2012



Feb 2014



Dec 2015

Data: SSL Pulse

TLS 1.2 Client Support

Google

- Chrome 30 and later
- Google Android Browser for Android 5.0 and later

Mozilla:

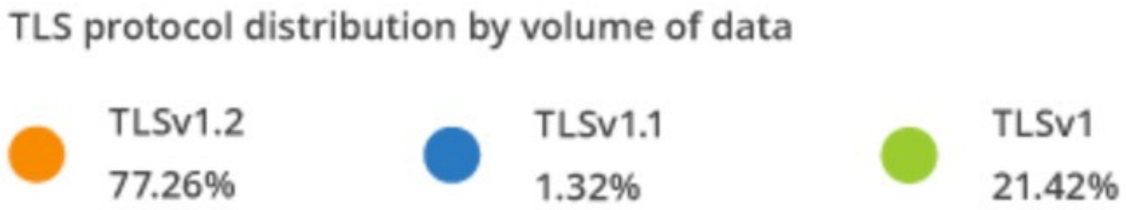
- Firefox 27 and and later

Microsoft:

- Internet Explorer 11 and later
- Internet Explorer Mobile 11 and later
- Microsoft Edge all versions

Apple

- Safari on OS X 10.9 and later
- Safari on iOS 5 and later



Lessons Learned?

- If an attacker can identify ***one*** bit of information, it's over
- Copious side channels
- Trusting unauthenticated data is a bad idea
- Don't MAC-then-encrypt - use AEADs
- X.509 and ASN-1 are hard to implement correctly
- Support insecure crypto/protocols for backwards compatibility at your own risk

Issues Skipped

BEAST

Blechenbacher RSA Decryption oracle

Schannel RCE

Triple Handshake

CA problems (DigiNotar, Comodo, Symantec)

RC4 weaknesses

Bignum vulnerabilities

Forward Secrecy

More…

One more thing…

Other unauthenticated negotiations in the handshake

NPN/ALPN

Supported elliptic curves

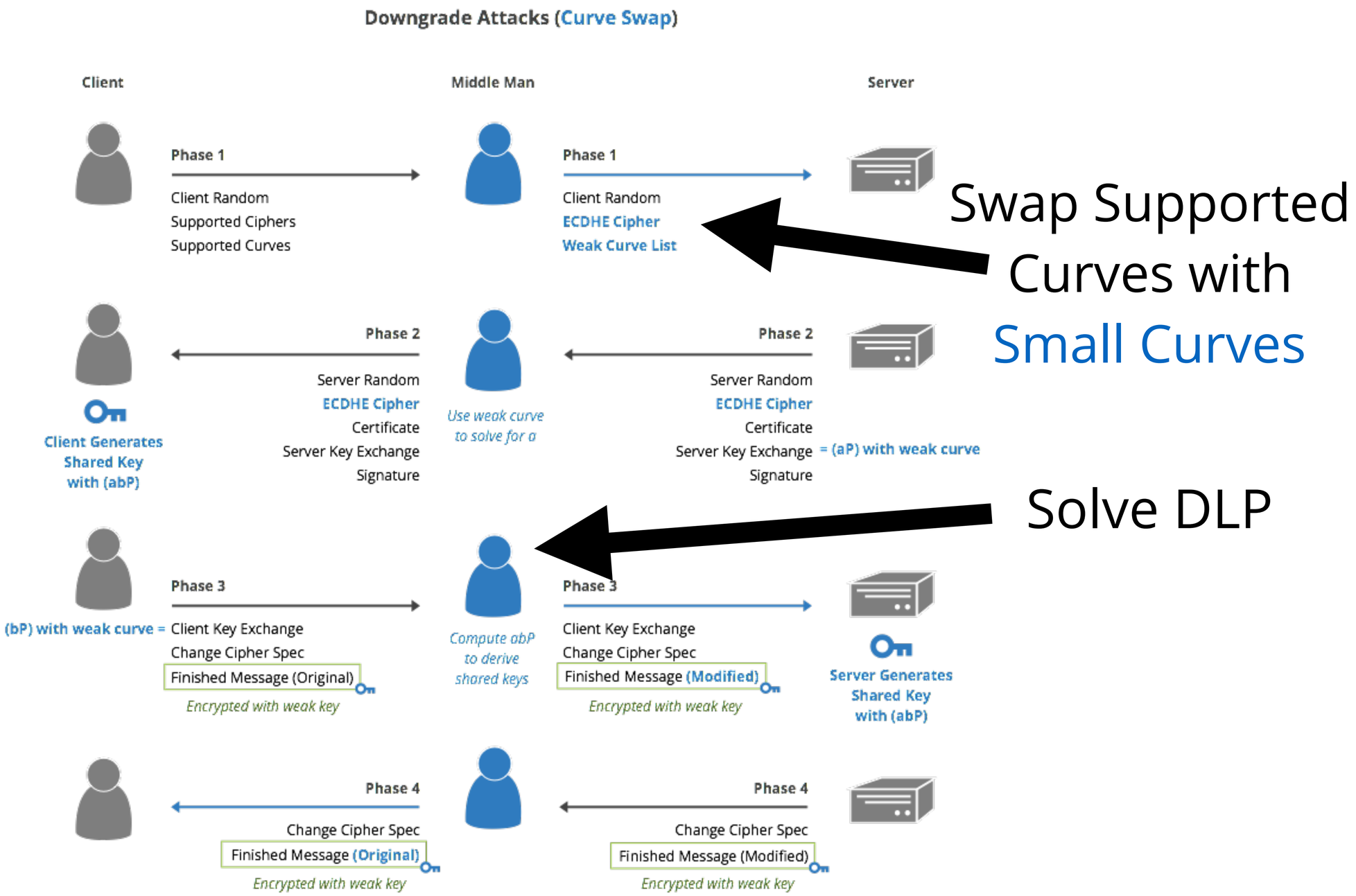
Other unauthenticated negotiations in the handshake

NPN/ALPN

Supported elliptic curves

Introducing:

CurveSwap


Downgrade Attacks (Curve Swap)
Client
Middle Man
Server
Phase 1
Client Random
Supported Ciphers
Supported Curves
Phase 1
Client Random
ECDHE Cipher
Weak Curve List
Swap Supported Curves with Small Curves
Phase 2
Server Random
ECDHE Cipher
Certificate
Server Key Exchange
Signature
Phase 2
Server Random
ECDHE Cipher
Certificate
Server Key Exchange
Signature
Client Generates Shared Key with (abP)
Use weak curve to solve for a
= (aP) with weak curve
Solve DLP
(bP) with weak curve =
Phase 3
Client Key Exchange
Change Cipher Spec
Finished Message (Original)
Encrypted with weak key
Compute abP to derive shared keys
Phase 3
Client Key Exchange
Change Cipher Spec
Finished Message (Modified)
Encrypted with weak key
Server Generates Shared Key with (abP)
Phase 4
Change Cipher Spec
Finished Message (Original)
Encrypted with weak key
Phase 4
Change Cipher Spec
Finished Message (Modified)
Encrypted with weak key
All traffic beyond this point is encrypted with the compromised keys.
The middle man can read or modify all messages between client and server.

Practicality

What is the weakest curve supported by clients and servers

sect163k?

Client support: **4.3%**

Alexa top 100,000: **0.13%**

The good news

Nobody has publicly broken DLP for ~160bit curves

But…

There’s a reason we don’t use binary curves

The Future

TLS 1.3 ?

32c3

December 28, 2015

Nick Sullivan
@grittygrease
nick@cloudflare.com
https://crypto.dance

goto fail;

a compendium of transport security calamities