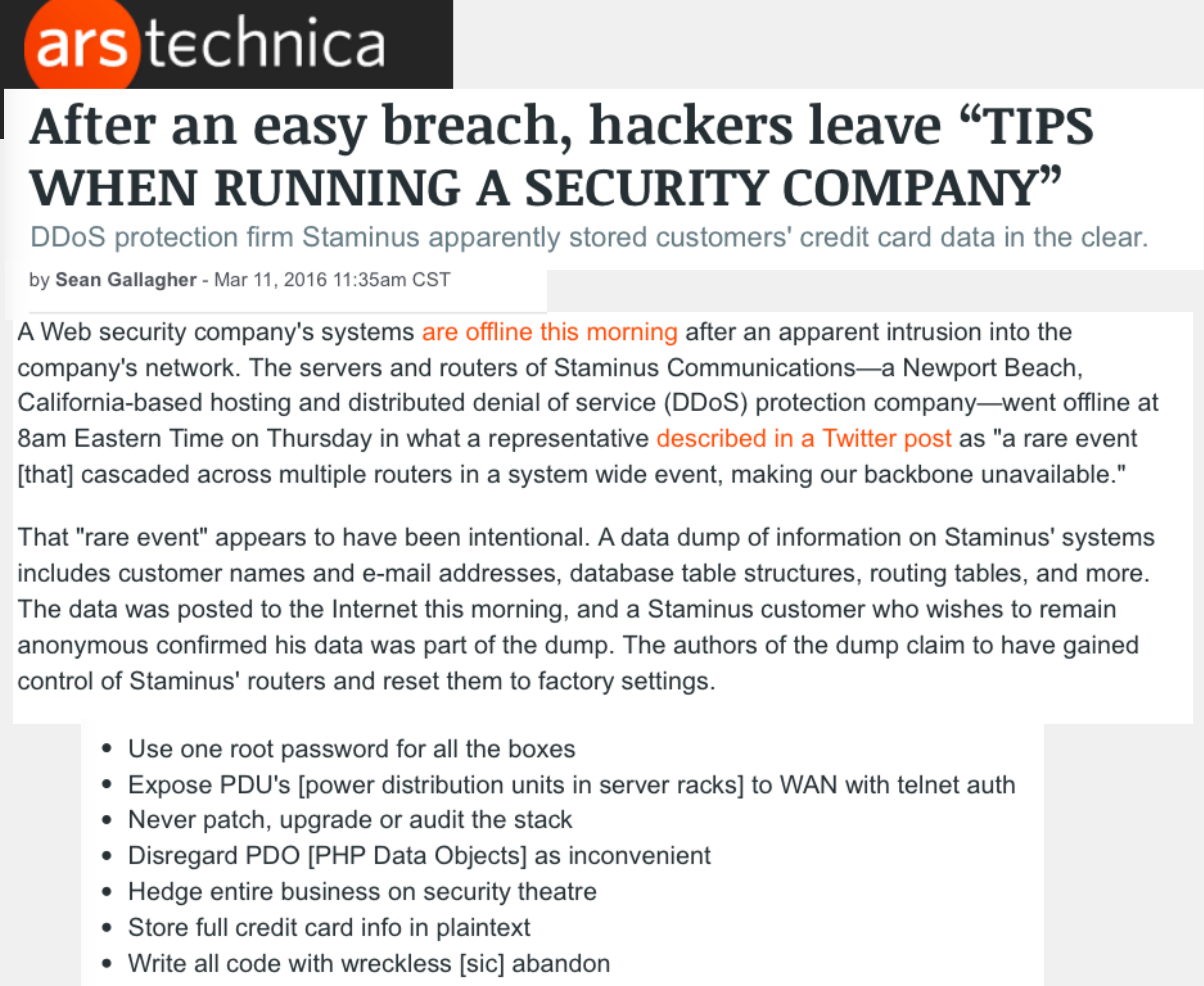


ars technica

After an easy breach, hackers leave “TIPS WHEN RUNNING A SECURITY COMPANY”

DDoS protection firm Staminus apparently stored customers' credit card data in the clear.

by **Sean Gallagher** - Mar 11, 2016 11:35am CST

A Web security company's systems are offline this morning after an apparent intrusion into the company's network. The servers and routers of Staminus Communications—a Newport Beach, California-based hosting and distributed denial of service (DDoS) protection company—went offline at 8am Eastern Time on Thursday in what a representative described in a Twitter post as "a rare event [that] cascaded across multiple routers in a system wide event, making our backbone unavailable."

That "rare event" appears to have been intentional. A data dump of information on Staminus' systems includes customer names and e-mail addresses, database table structures, routing tables, and more. The data was posted to the Internet this morning, and a Staminus customer who wishes to remain anonymous confirmed his data was part of the dump. The authors of the dump claim to have gained control of Staminus' routers and reset them to factory settings.

- Use one root password for all the boxes
- Expose PDU's [power distribution units in server racks] to WAN with telnet auth
- Never patch, upgrade or audit the stack
- Disregard PDO [PHP Data Objects] as inconvenient
- Hedge entire business on security theatre
- Store full credit card info in plaintext
- Write all code with wreckless [sic] abandon

public key *cryptography*

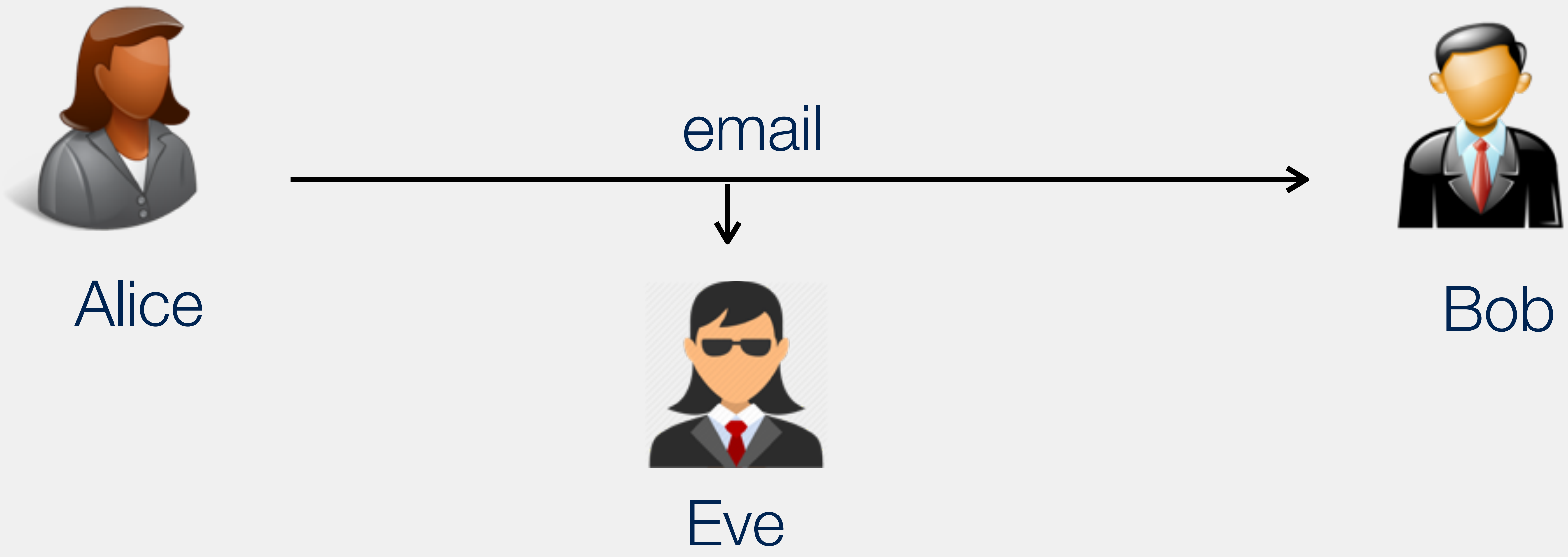
cs642

computer security

adam everspaugh
ace@cs.wisc.edu

today

- Hybrid encryption
- Digital signatures, certificates
- TLS overview
- Passwords



email

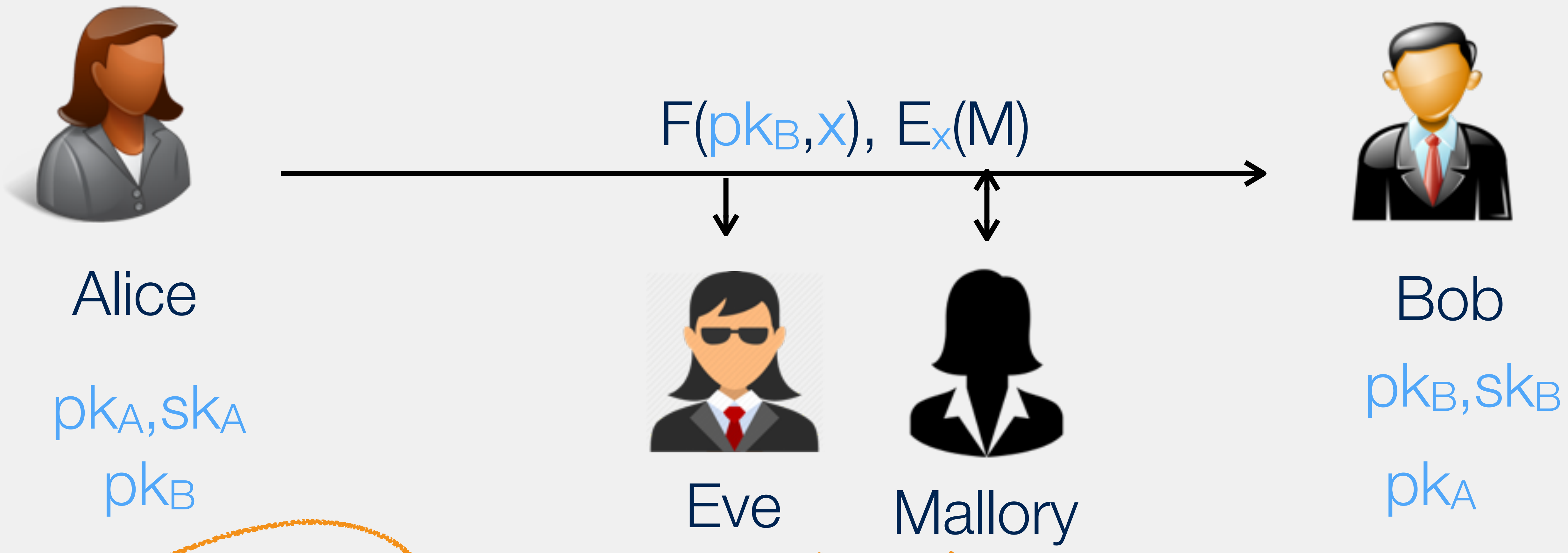
Alice

Bob

Eve

- Security goals?
  / Confidentiality, integrity, authenticity
- Symmetric encryption: fast, hard to distribute keys
- Public key encryption: slow, easy to distribute public keys

hybrid encryption



$F(pk_B,x), E_x(M)$

Alice

pk_A, sk_A

pk_B

Eve

Mallory

Bob

pk_B, sk_B

pk_A

random key for this message

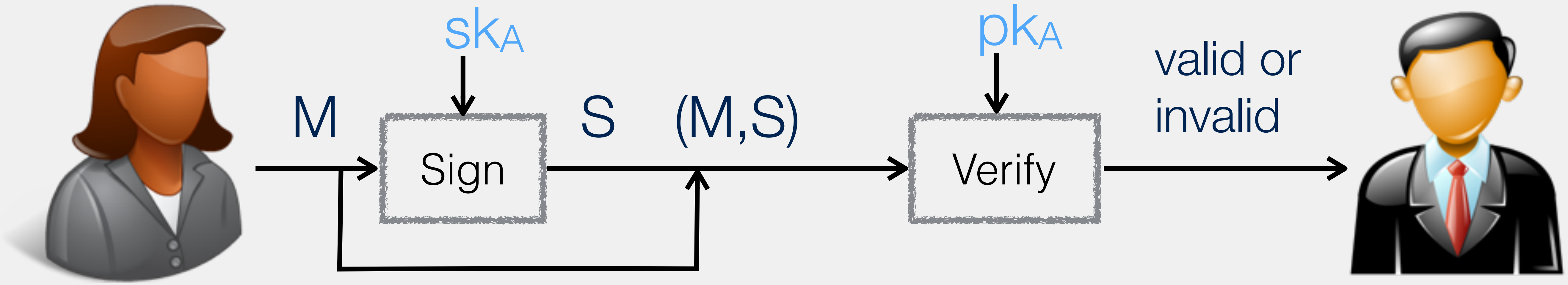
$x \leftarrow\$ \{0,1\}^k$

$F(pk_B, x), E_x(M)$

Authenticated encryption scheme

Encrypt under Bob's pubkey

hybrid encryption



Alice

pk_A, sk_A

Trapdoor permutation

F_{pk}: X→X

F^{-1}_{sk}: X→X

Hash Fn

H: $\{0,1\}^* \rightarrow X$

Bob

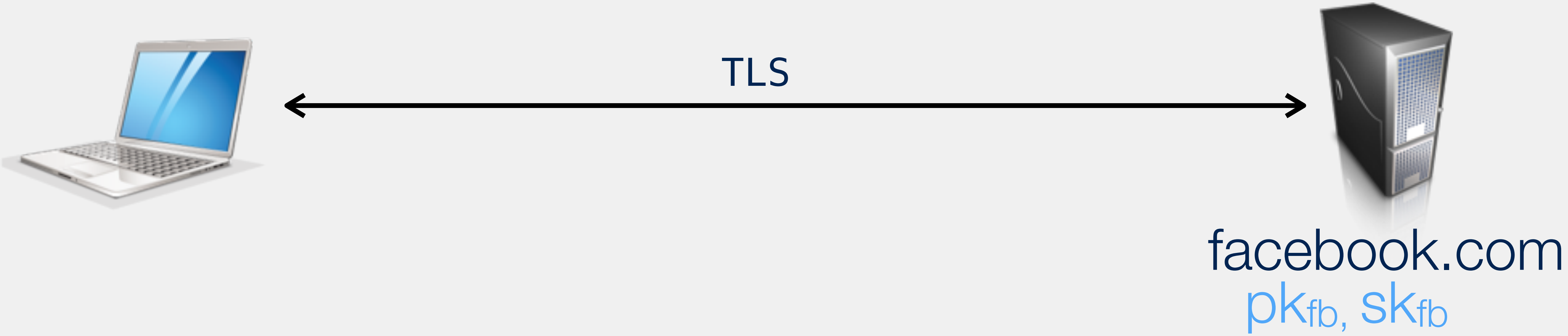
pk_A

think-*pair*-share

```
Sign($sk_A$, M):
  $d = H(M)$
  $S = F^{-1}(sk_A, d)$
```

```
Verify($pk_A$, M, S):
  $d' = F(pk_A, S)$
  if $d' = H(M)$:
    ret VALID
  else:
    ret INVALID
```

digital signatures



TLS

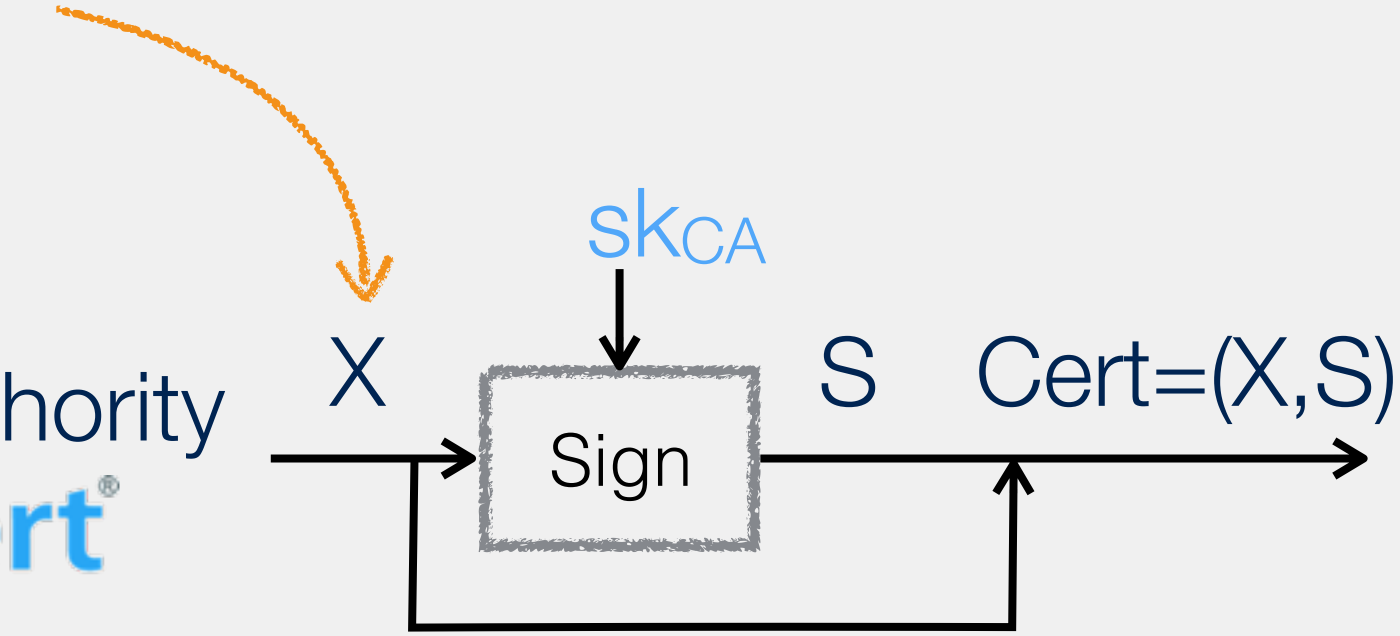
facebook.com

pk_{fb}, sk_{fb}

- Problem: How does a client get the public key for a website?

certificates

Domain: *.facebook.com
Pubkey: 04 DB D1 77 …



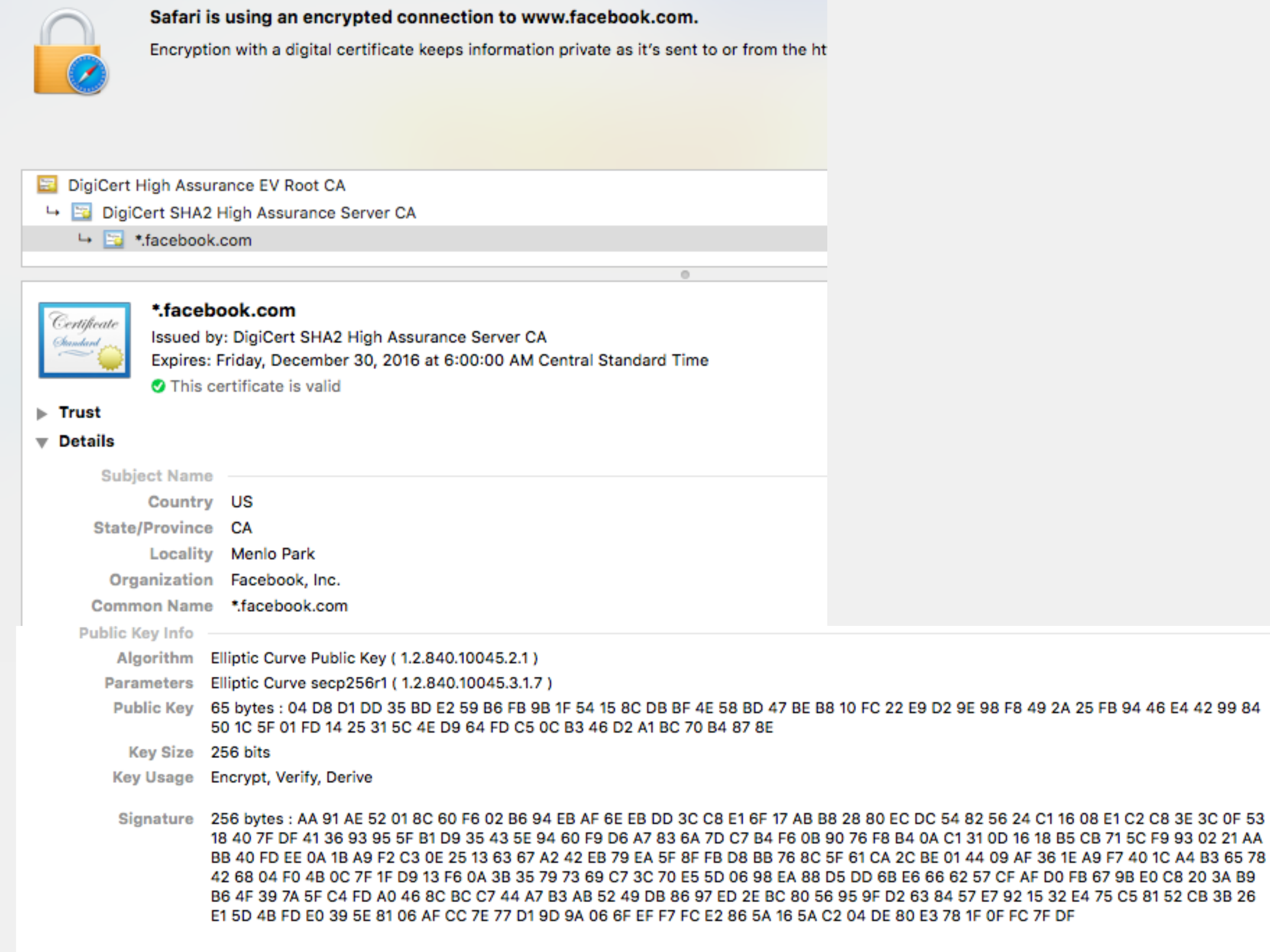
Certificate Authority

digicert

pk_{CA}, sk_{CA}

pk_{CA}

cert signing



Safari is using an encrypted connection to www.facebook.com.

Encryption with a digital certificate keeps information private as it's sent to or from the ht

DigiCert High Assurance EV Root CA
↳ DigiCert SHA2 High Assurance Server CA
↳ *.facebook.com

***.facebook.com**
Issued by: DigiCert SHA2 High Assurance Server CA
Expires: Friday, December 30, 2016 at 6:00:00 AM Central Standard Time
✓ This certificate is valid

▶ **Trust**

▼ **Details**

Subject Name

Country US
State/Province CA
Locality Menlo Park
Organization Facebook, Inc.
Common Name *.facebook.com

Public Key Info

Algorithm Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key 65 bytes : 04 D8 D1 DD 35 BD E2 59 B6 FB 9B 1F 54 15 8C DB BF 4E 58 BD 47 BE B8 10 FC 22 E9 D2 9E 98 F8 49 2A 25 FB 94 46 E4 42 99 84 50 1C 5F 01 FD 14 25 31 5C 4E D9 64 FD C5 0C B3 46 D2 A1 BC 70 B4 87 8E
Key Size 256 bits
Key Usage Encrypt, Verify, Derive

Signature 256 bytes : AA 91 AE 52 01 8C 60 F6 02 B6 94 EB AF 6E EB DD 3C C8 E1 6F 17 AB B8 28 80 EC DC 54 82 56 24 C1 16 08 E1 C2 C8 3E 3C 0F 53 18 40 7F DF 41 36 93 95 5F B1 D9 35 43 5E 94 60 F9 D6 A7 83 6A 7D C7 B4 F6 0B 90 76 F8 B4 0A C1 31 0D 16 18 B5 CB 71 5C F9 93 02 21 AA BB 40 FD EE 0A 1B A9 F2 C3 0E 25 13 63 67 A2 42 EB 79 EA 5F 8F FB D8 BB 76 8C 5F 61 CA 2C BE 01 44 09 AF 36 1E A9 F7 40 1C A4 B3 65 78 42 68 04 F0 4B 0C 7F 1F D9 13 F6 0A 3B 35 79 73 69 C7 3C 70 E5 5D 06 98 EA 88 D5 DD 6B E6 66 62 57 CF AF D0 FB 67 9B E0 C8 20 3A B9 B6 4F 39 7A 5F C4 FD A0 46 8C BC C7 44 A7 B3 AB 52 49 DB 86 97 ED 2E BC 80 56 95 9F D2 63 84 57 E7 92 15 32 E4 75 C5 81 52 CB 3B 26 E1 5D 4B FD E0 39 5E 81 06 AF CC 7E 77 D1 9D 9A 06 6F EF F7 FC E2 86 5A 16 5A C2 04 DE 80 E3 78 1F 0F FC 7F DF

Let's Encrypt is a new Certificate Authority:

It's free, automated, and open.

Get Started (Public Beta)

http://letsencrypt.org

certificates

digicert

pk_{CA}, sk_{CA}

- What does having a trusted TLS certificate prove?
  - That someone paid at least $0
  - Proved to an intermediate CA that they controlled a given domain name for at least 5 minutes
  - If TLS established, proves they know the corresponding private key to the pub key in cert

think-*pair*-share

- What could possibly go wrong?
  - Any CA secret key in chain could be compromised
  - Server secret key could be compromised
  - Typo-squatting domain (gmal.com)
  - Malicious root CA key installed on client
  - DNS chicanery during verification process

DigiNotar

- Dutch CA DigiNotar compromised in 2011
- Attackers generated fake certificates
- Twitter.com was redirected to fake site
- Attackers eavesdropped with man-in-the-middle attacks
  / Iranian govt eavesdropping on dissidents

DigiNotar
Internet Trust Services

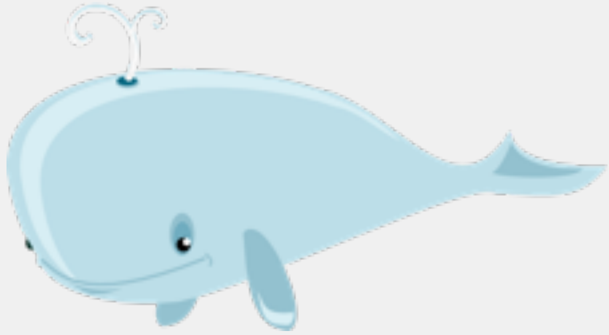
DigiNotar
Internet Trust Services

- How did compromise occur?
- DigiNotar had crappy security
  - Out-of-date antivirus software
  - Poor software patching
  - Weak passwords
  - No auditing of logs
  - Poorly designed local network

DigiNotar

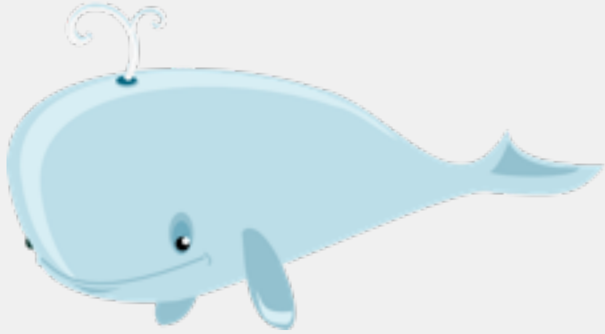
eDellRoot

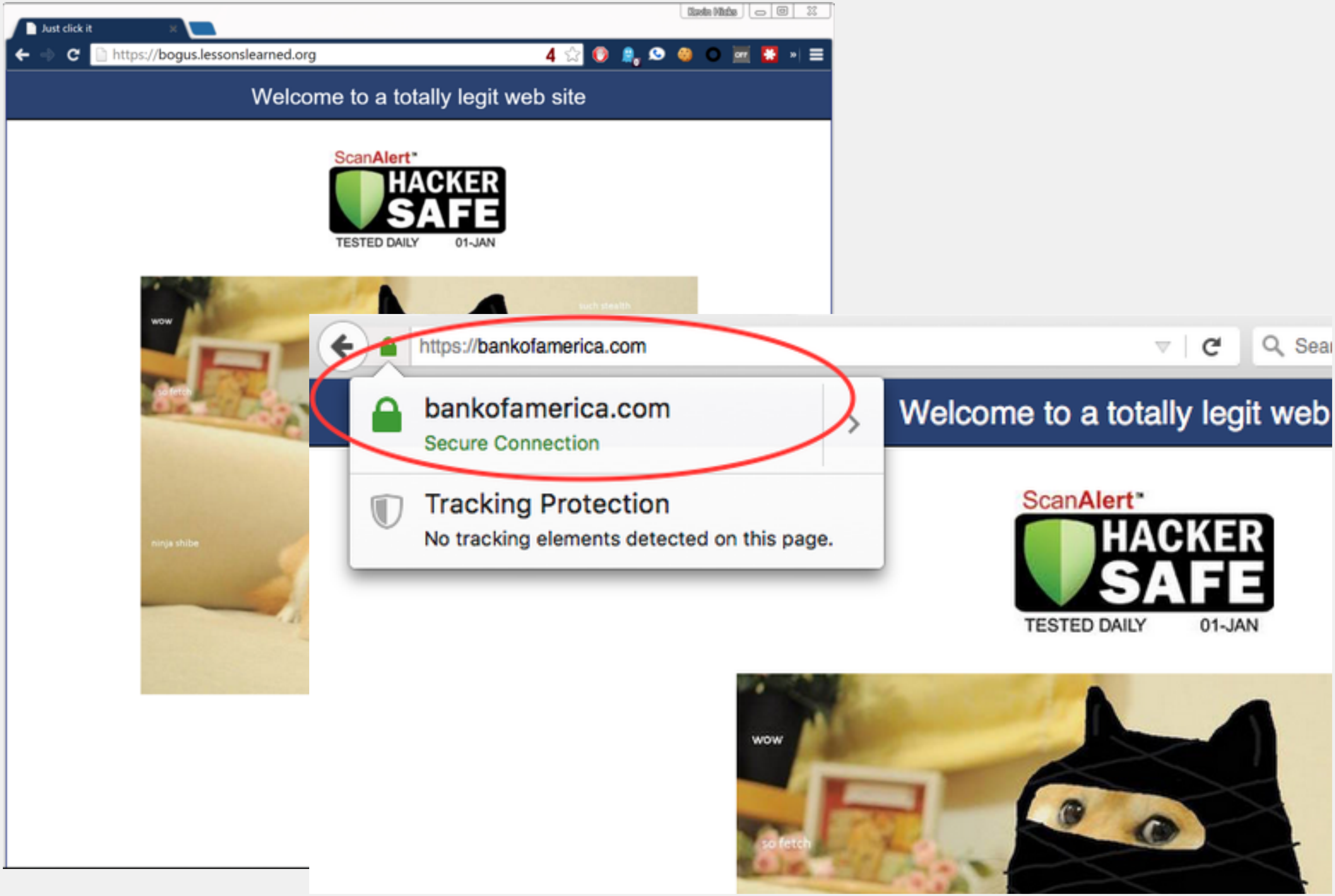
- Dell shipped several computer systems with a self-signed root CA certificate preinstalled
  / The cert also contained the CA secret key



- Intended purpose: something to do with automated support software

- If certificate removed, automatically reinstalls on reboot





Welcome to a totally legit web site

bankofamerica.com

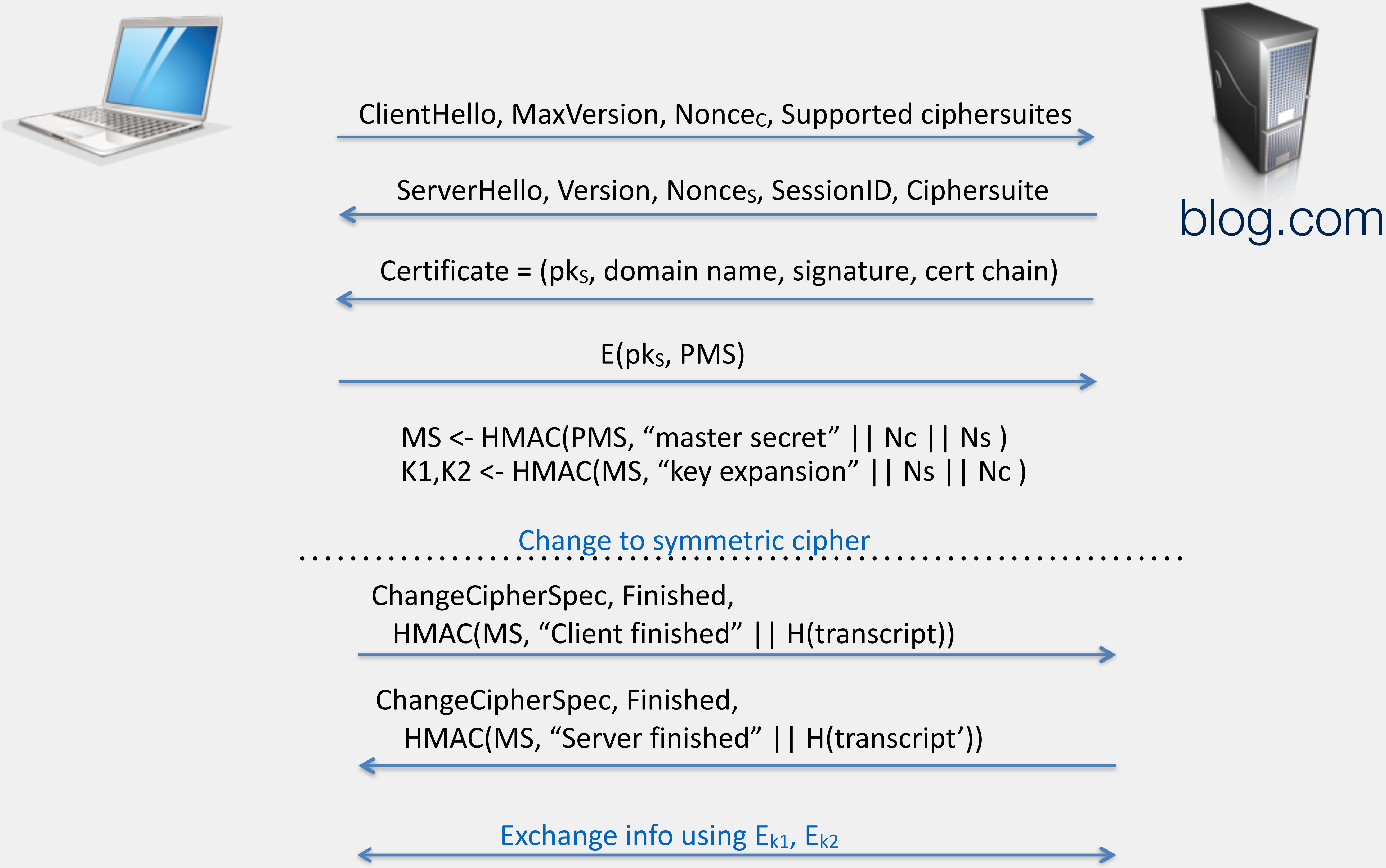
Secure Connection

Tracking Protection

No tracking elements detected on this page.

Welcome to a totally legit web

eDellRoot



blog.com

ClientHello, MaxVersion, Nonce$_C$, Supported ciphersuites

ServerHello, Version, Nonce$_S$, SessionID, Ciphersuite

Certificate = (pk_S, domain name, signature, cert chain)

E(pk_S, PMS)

MS <- HMAC(PMS, “master secret” || Nc || Ns)
K1,K2 <- HMAC(MS, “key expansion” || Ns || Nc)

Change to symmetric cipher

ChangeCipherSpec, Finished,
HMAC(MS, “Client finished” || H(transcript))

ChangeCipherSpec, Finished,
HMAC(MS, “Server finished” || H(transcript’))

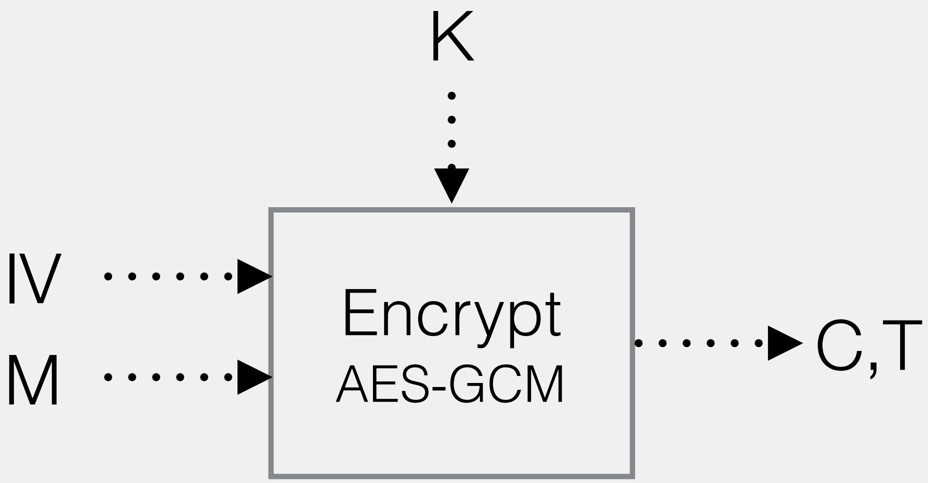
Exchange info using E_{k1}, E_{k2}

tls

INTERMISSION

passwords

pw use cases



K

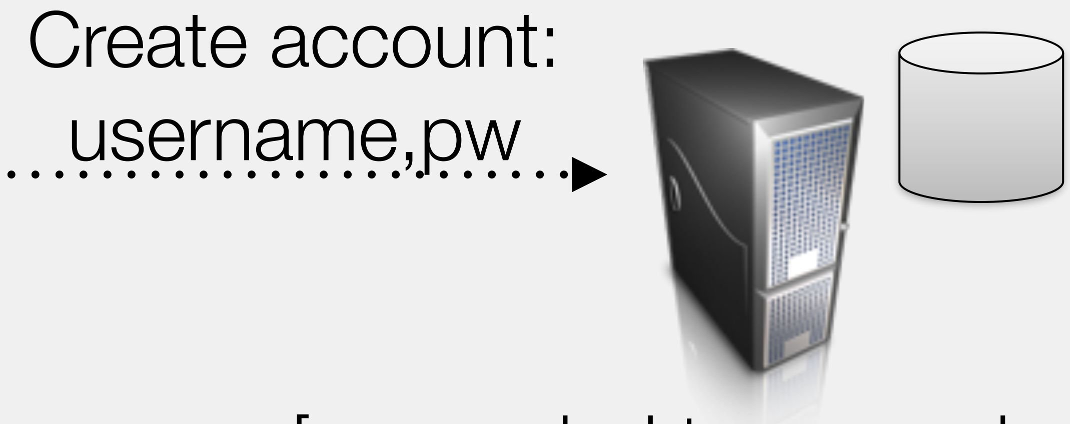
IV

M

Encrypt

AES-GCM

C,T

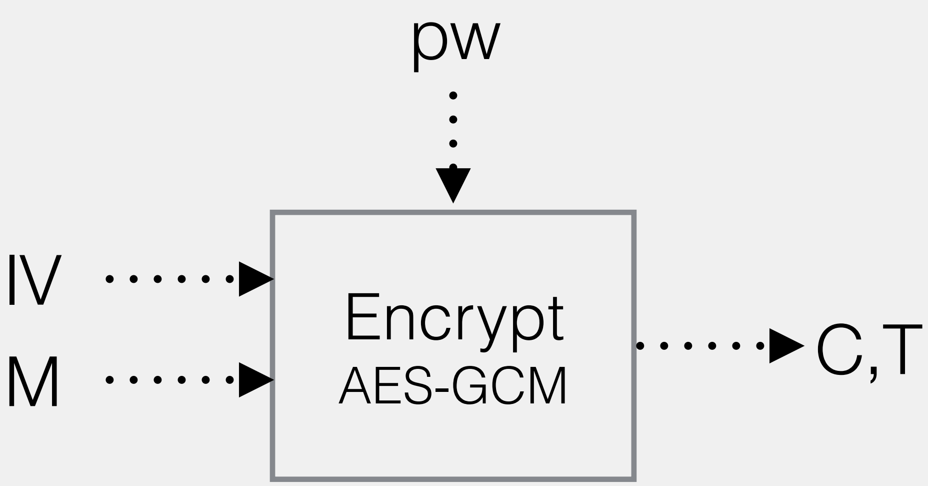


Create account:

username,pw

[server, desktop, or web service]

How does the server store the pw?



pw

IV

M

Encrypt

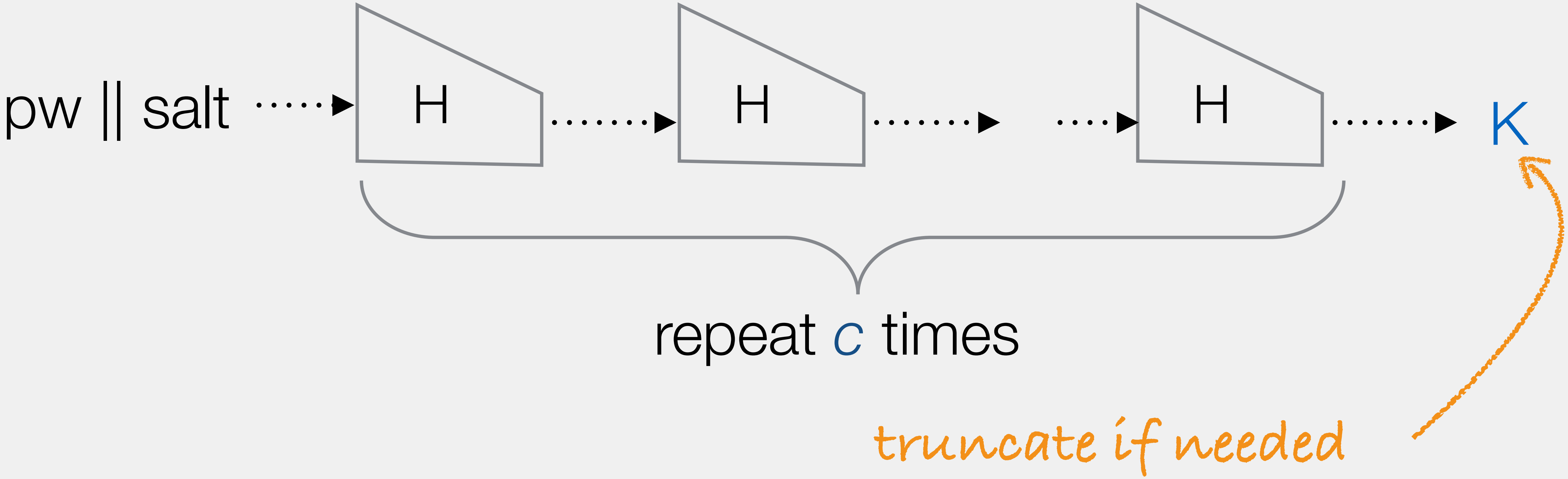
AES-GCM

C,T

Password-based symmetric encryption

[password-based key derivation function]

PBKDF(pw, salt):



pbkdf

pw-based encryption

```
Enc(pw,M,R):
salt || R’ = R
K = PBKDF(pw,salt)
C = Enc’(K,M,R’)
Return (salt,C)
```

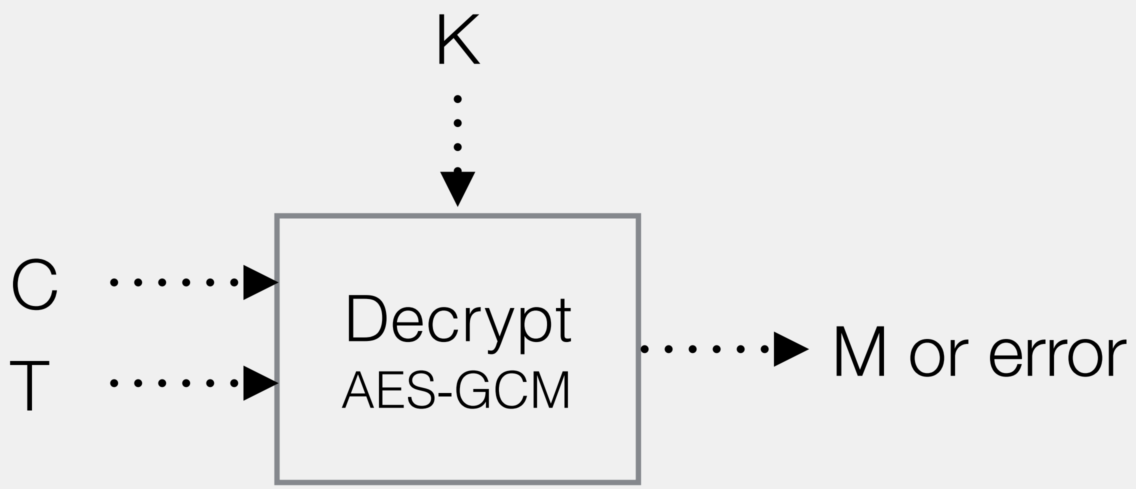
```
Dec(pw,C):
salt || C’ = C
K = PBKDF(pw,salt)
M = Dec’(K,C’)
Return M
```

Enc'/Dec' is some authenticated encryption scheme,
like AES-GCM

PBKDF + symmetric encryption → pw-based encryption

Attacks?

dictionary attack



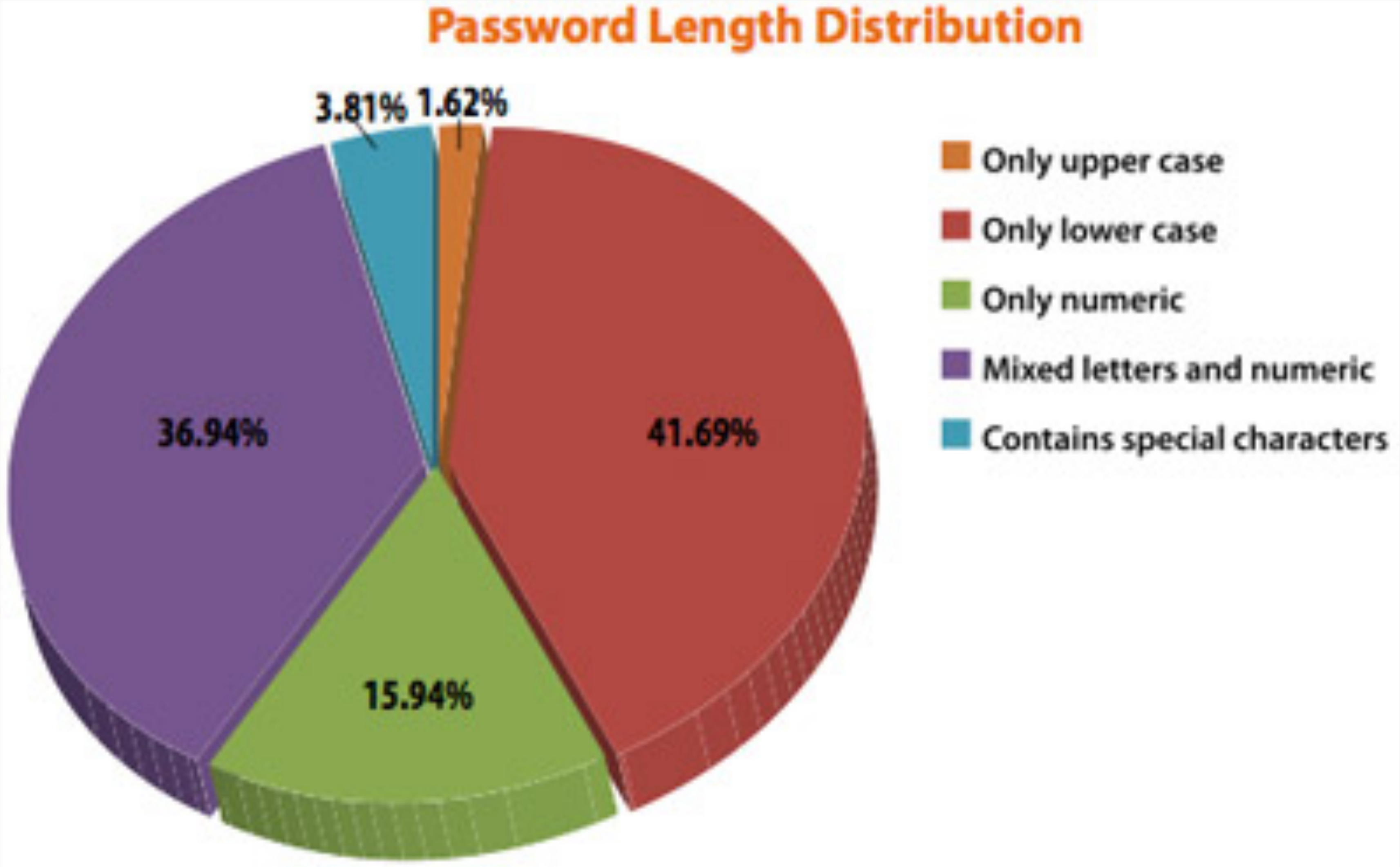
```
DictionaryAttack(D,C,T):
for pw* in D:
    M* = Dec(pw*,C,T)
    if M* ≠ error:
      return pw,M*
```

- Given an authenticated encryption output (C,T), dictionary *D* of possible password

- Enumerate *D* in order of likelihood

- Test each candidate password

pw distribution



From an Imperva study of released RockMe.com password database (2010)

Facebook’s Password Onion

```
$cur  = ‘password’
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```

- Hybrid encryption

- Digital signatures

- Certificates, problems

- Password-based key derivation
  / Dictionary attacks

- Exit slips
  / 1 thing you learned
  / 1 thing you didn't understand

recap