MACs, Passwords and Asymmetric encryption

CS642: Computer Security

Asymmetric encryption

The RSA algorithm

Digital signing & public-key infrastructure

The Fundamental Equation

$$Z = Y^X \bmod N$$

One-Way Functions

$$Z = Y^{X} \bmod N$$

When X is unknown, the problem is known as the *discrete logarithm* and is generally believed to be hard to solve.

One-Way Trap-Door Functions

$$Z = Y^X \bmod N$$

This equation is solvable for Y if the factorization of N is known, but is *believed* to be hard otherwise.

RSA Public-Key Cryptosystem

Alice

Anyone

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.

Anyone

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.
- Publish the product $N = PQ$.

Anyone

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.
- Publish the product $N = PQ$.

Anyone

- To send message Y to Alice, compute $Z = Y^X \bmod N$.

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.
- Publish the product $N = PQ$.

Anyone

- To send message Y to Alice, compute $Z = Y^X \bmod N$.
- Send Z and X to Alice.

RSA Public-Key Cryptosystem

Alice

- Select two large random primes P & Q.
- Publish the product $N = PQ$.

- Use knowledge of P & Q to compute Y.

Anyone

- To send message Y to Alice, compute $Z = Y^X \bmod N$.
- Send Z and X to Alice.

Some RSA Details

When $N = PQ$ is the product of distinct primes,

$$Y^X \bmod N = Y$$

whenever

$X \bmod (P-1)(Q-1) = 1$ and $0 \leq Y < N$.

Some RSA Details

When $N = PQ$ is the product of distinct primes,

$$Y^X \bmod N = Y$$

whenever

$X \bmod (P-1)(Q-1) = 1$ and $0 \leq Y < N$.

Alice can easily **select** integers E and D such that $(E \times D) \bmod (P-1)(Q-1) = 1$.

Some RSA Details

Encryption: $E(Y) = Y^E \bmod N$.

Decryption: $D(Y) = Y^D \bmod N$.

$$
\begin{aligned}
& D(E(Y)) \\
&= (Y^E \bmod N)^D \bmod N \\
&= Y^{ED} \bmod N \\
&= Y
\end{aligned}
$$

Some RSA Details

In practice, the encryption exponent E is almost always fixed to be

$$E = 65537 = 2^{16} + 1.$$

Some RSA Details

In practice, the encryption exponent E is almost always fixed to be

$$E = 65537 = 2^{16} + 1.$$

The decryption exponent D is then computed as

$$D = (1 \div E) \bmod (P - 1)(Q - 1).$$

Public-Key Directory

Name	Public Key	Encryption
Alice	N_A	$E_A(Y) = Y^E \bmod N_A$
Bob	N_B	$E_B(Y) = Y^E \bmod N_B$
Carol	N_C	$E_C(Y) = Y^E \bmod N_C$
⋮	⋮	⋮

Public-Key Directory

Name	Public Key	Encryption
Alice	N_A	$E_A(Y) = Y^E \bmod N_A$
Bob	N_B	$E_B(Y) = Y^E \bmod N_B$
Carol	N_C	$E_C(Y) = Y^E \bmod N_C$
⋮	⋮	⋮

(Recall that E is commonly fixed to be $E = 65537$.)

RSA Signatures

RSA Signatures

An additional property

RSA Signatures

An additional property

$$D(E(Y)) = Y^{ED} \bmod N = Y$$

RSA Signatures

An additional property

$$D(E(Y)) = Y^{ED} \bmod N = Y$$

$$E(D(Y)) = Y^{DE} \bmod N = Y$$

RSA Signatures

An additional property

$$D(E(Y)) = Y^{ED} \bmod N = Y$$

$$E(D(Y)) = Y^{DE} \bmod N = Y$$

Only Alice (knowing the factorization of N) knows D. Hence only Alice can compute $D(Y) = Y^{D} \bmod N$.

RSA Signatures

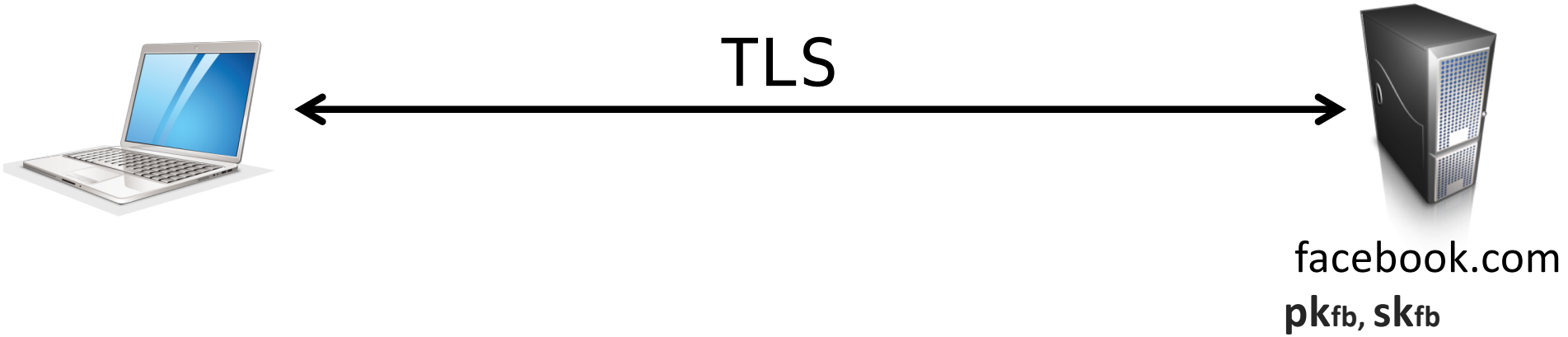
An additional property

$$D(E(Y)) = Y^{ED} \bmod N = Y$$

$$E(D(Y)) = Y^{DE} \bmod N = Y$$

Only Alice (knowing the factorization of N) knows D. Hence only Alice can compute $D(Y) = Y^{D} \bmod N$.

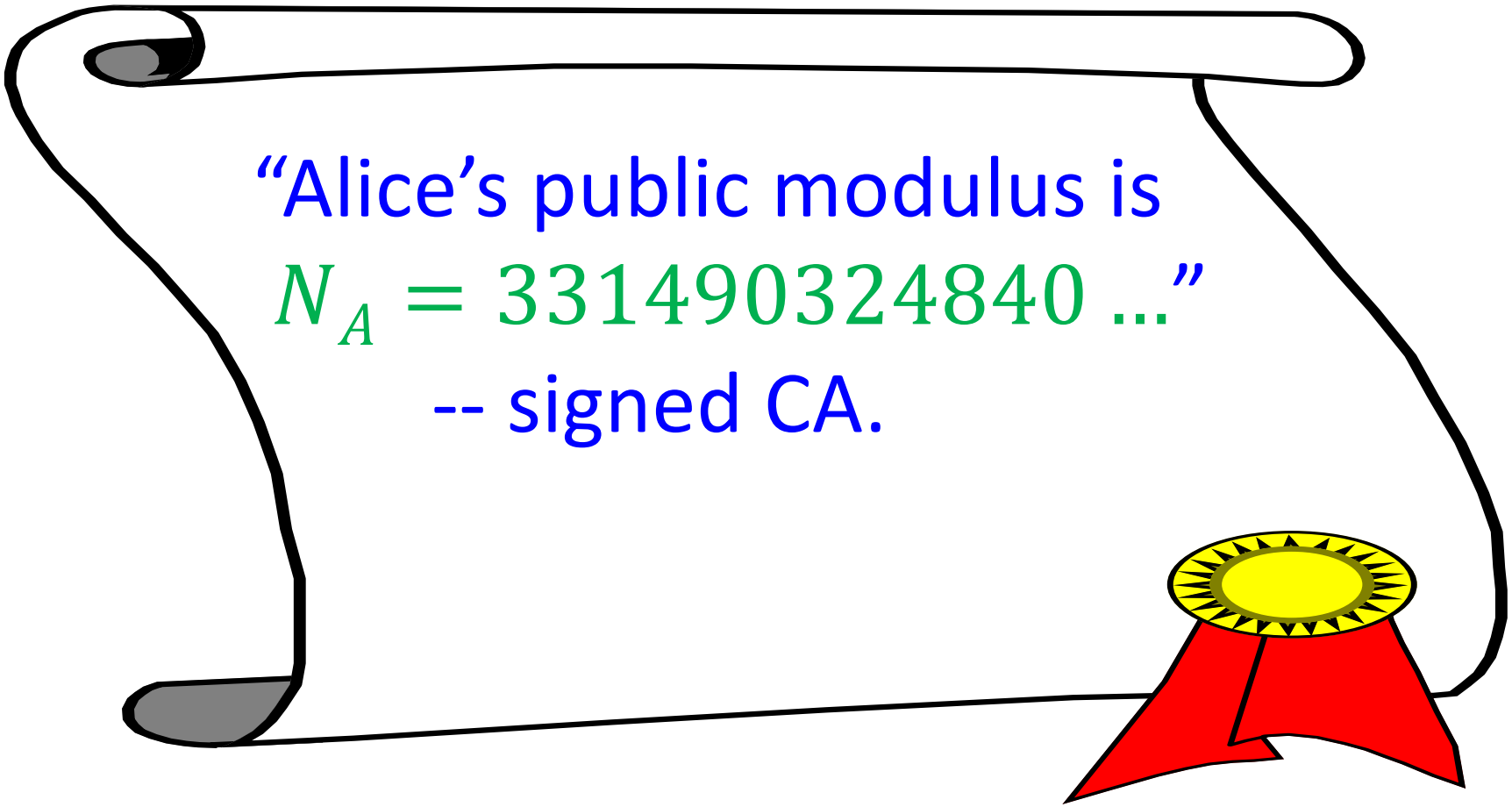
This $D(Y)$ serves as Alice's signature on Y.



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

certificates

Certificate Authority

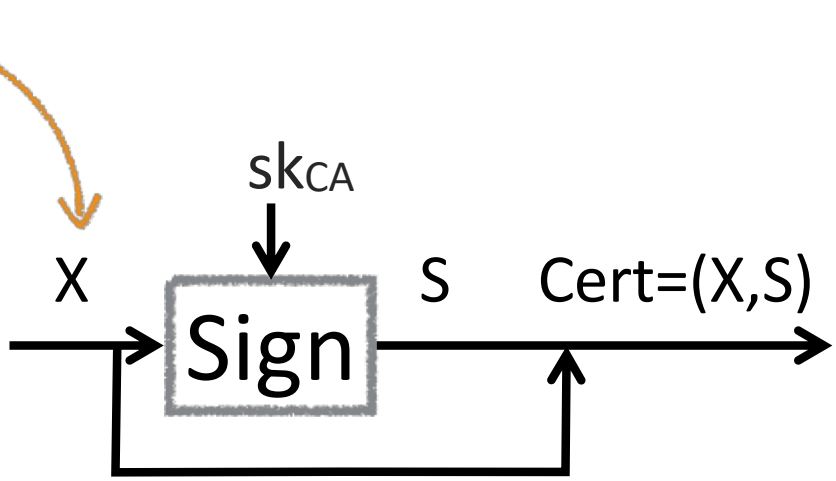


Domain: *.facebook.com
Pubkey: 04 DB D1 77 …

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

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

- 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

certificates

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

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

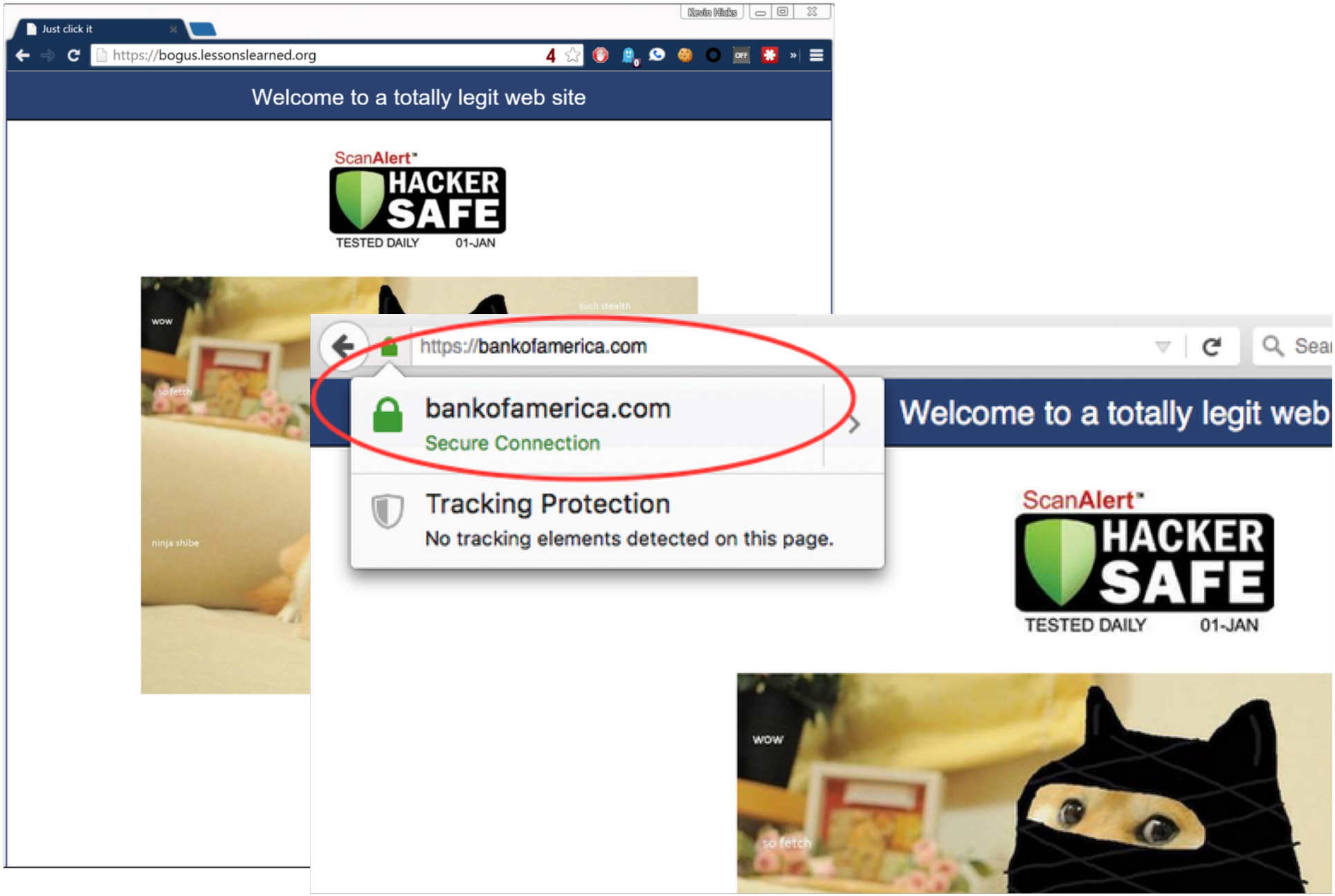
How did compromise occur?

DigiNotar had crappy security

- Out-of-date antivirus software
- Poor software patching
- Weak passwords
- No auditing of logs
- Poorly designed local network

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


Just click it
Kevin Hicks
https://bogus.lessonslearned.org
Welcome to a totally legit web site
ScanAlert
HACKER SAFE
TESTED DAILY 01-JAN
wow
so fetch
ninja shibe
such stealth
https://bankofamerica.com
bankofamerica.com
Secure Connection
Tracking Protection
No tracking elements detected on this page.
Welcome to a totally legit web
ScanAlert
HACKER SAFE
TESTED DAILY 01-JAN
wow
so fetch

eDellRoot