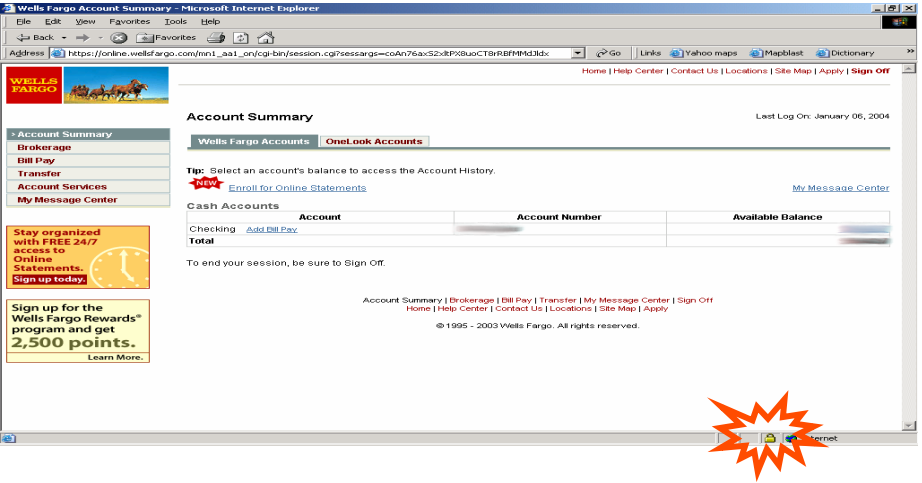
Public Key Crypto

Slides credit: Dan Boneh

Administrative Issues

- Security is a fast-changing field
- We cover a broad spectrum of areas in computer security
- Hence, there're no suitable textbooks
- Going to class is required
  - Best way to learn the material

Secure communication



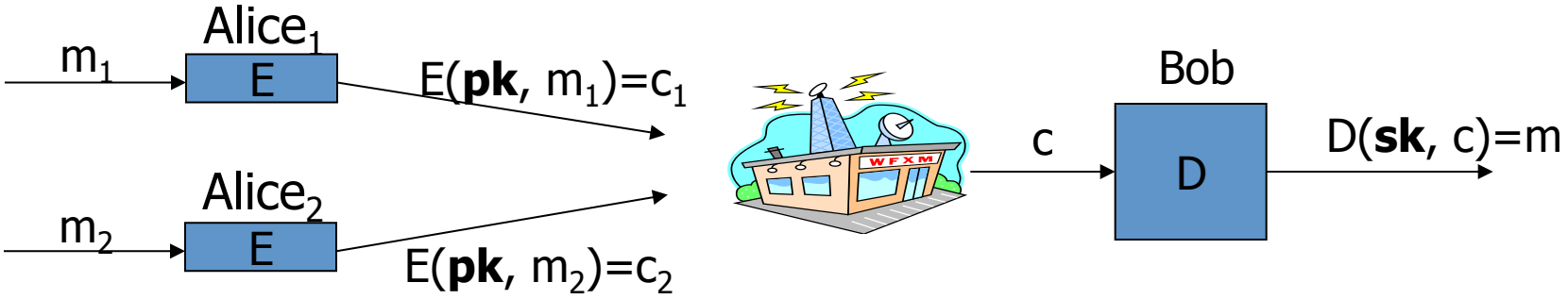
Authenticated channel
privacy + integrity

This section: how do we generate session key?

Need two concepts: **public-key encryption** and **digital signatures**

Public-key encryption

Tool for managing or generating symmetric keys



- E – Encryption alg. pk – Public encryption key
- D – Decryption alg. sk – Private decryption key

Algorithms E, D are publicly known.

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk, c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$ (pk, sk) output by G :

$\forall m \in M$: D(sk, E(pk, m)) = m

Building Block: Trapdoor Functions (TDF)

Def: a trapdoor function over X is a triple of efficient algs. (G, F, F^{-1})

- **G**(): randomized alg. outputs a key pair (pk, sk)
- **F**(pk, $\cdot$): det. alg. that defines a function $X \longmapsto Y$
- $\mathbf{F^{-1}}$(sk, $\cdot$): defines a function $X \longmapsto Y$ that inverts F(pk, $\cdot$)

for all x in X: $F^{-1}(sk, F(pk, x)) = x$

Security: (G, F, F^{-1}) is secure if F(pk, $\cdot$) is a "one-way" function:

given **F(pk, x)** it is difficult to find **x** (just given pk)

Example: RSA

- alg. G(): generate two equal length primes p, q

  set $N \leftarrow p \cdot q$ (3072 bits $\approx$ 925 digits)

  set $e \leftarrow 2^{16}+1 = 65537$; $d \leftarrow e^{-1} \pmod{\varphi(N)}$

  pk = (N, e) ; sk = (N, d)

- RSA(pk, x) : $x \rightarrow (x^e \bmod N)$

  Inverting this function is believed to be as hard as factoring N

- RSA^{-1}(sk, y) : $y \rightarrow (y^d \bmod N)$

Public Key Encryption with a TDF

G(): generate pk and sk

E(pk, m):

- choose random $x \in$ domain(F) and set $k \leftarrow H(x)$
- $c_0 \leftarrow F(pk, x)$, $c_1 \leftarrow E(k, m)$ (E: symm. cipher)
- send $c = (c_0, c_1)$

D(sk, $c=(c_0,c_1)$): $x \leftarrow F^{-1}(sk, c_0)$, $k \leftarrow H(x)$, $m \leftarrow D(k, c_1)$

Suppose the message m to encrypt is short.

Can we directly encrypt m using the TDF as $c \leftarrow F(pk, m)$?

- Yes, this would not hurt security
- No, this would be insecure because encrypting the same message twice results in the same ciphertext
- It depends on the specific TDF F used

Digital Signature

Recall: Trapdoor Functions (TDF)

Def: a trapdoor function over X is a triple of efficient algs. (G, F, F^{-1})

- **G**(): randomized alg. outputs a key pair (pk, sk)
- **F**(pk, $\cdot$): det. alg. that defines a function $X \longmapsto Y$
- $\mathbf{F^{-1}}$(sk, $\cdot$): defines a function $X \longmapsto Y$ that inverts F(pk, $\cdot$)

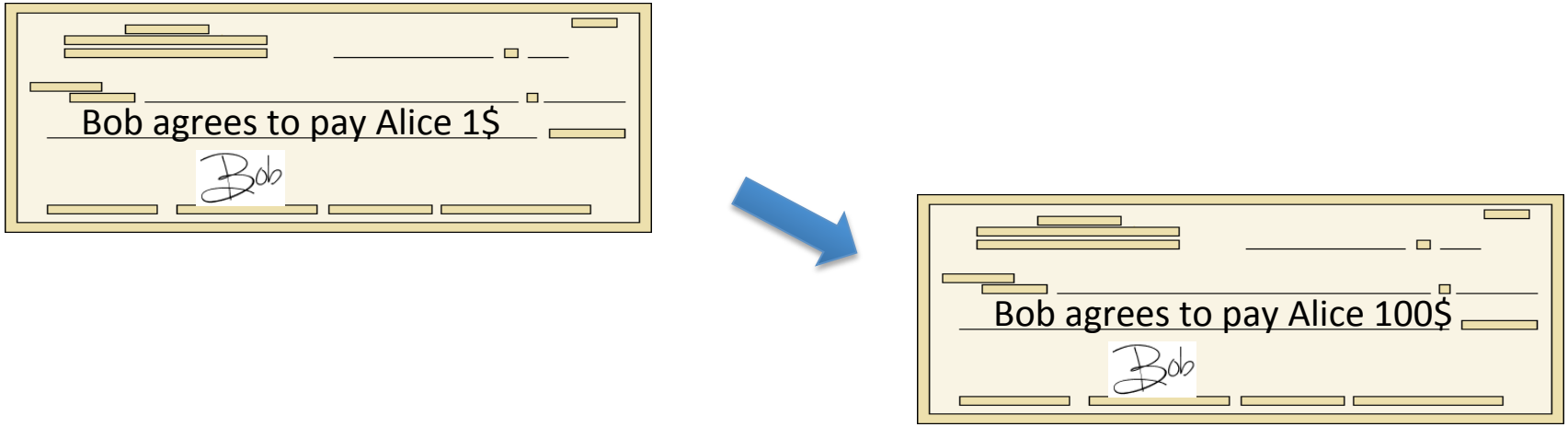
for all x in X: $F^{-1}(sk, F(pk, x)) = x$

Security: (G, F, F^{-1}) is secure if F(pk, $\cdot$) is a "one-way" function:

given **F(pk, x)** it is difficult to find **x** (just given pk)

Digital signatures

Goal: bind document to author



Problem: attacker can copy Bob's sig from one doc to another

Digital signatures

Solution: make signature depend on document

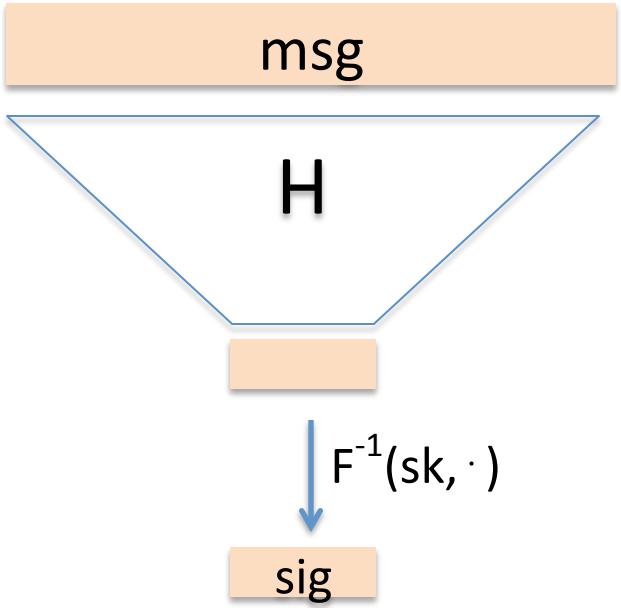
Example: signatures from trapdoor functions (e.g. RSA)

sign(sk, m) := F^{-1} (sk, H(m))

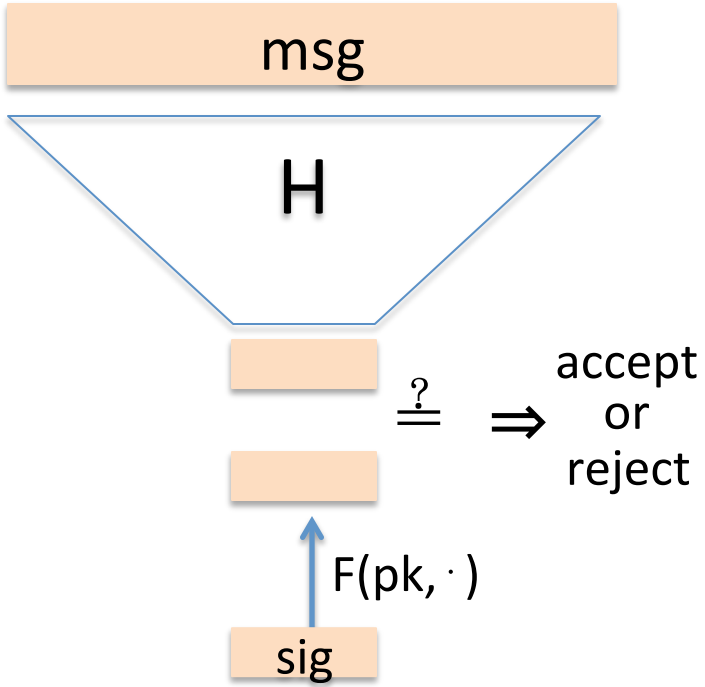
Verify(pk, m, sig) := accept if F(pk, sig) = H(m)

Digital Sigs. from Trapdoor Functions

sign(sk, msg):



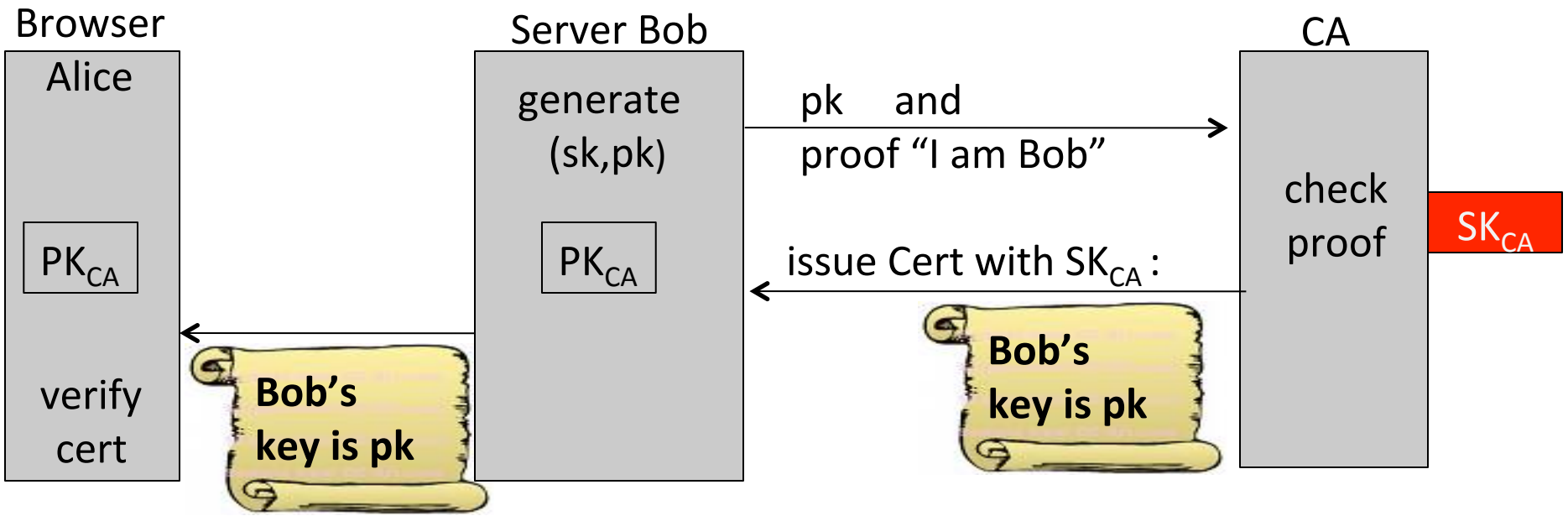
verify(pk, msg, sig):



Certificates

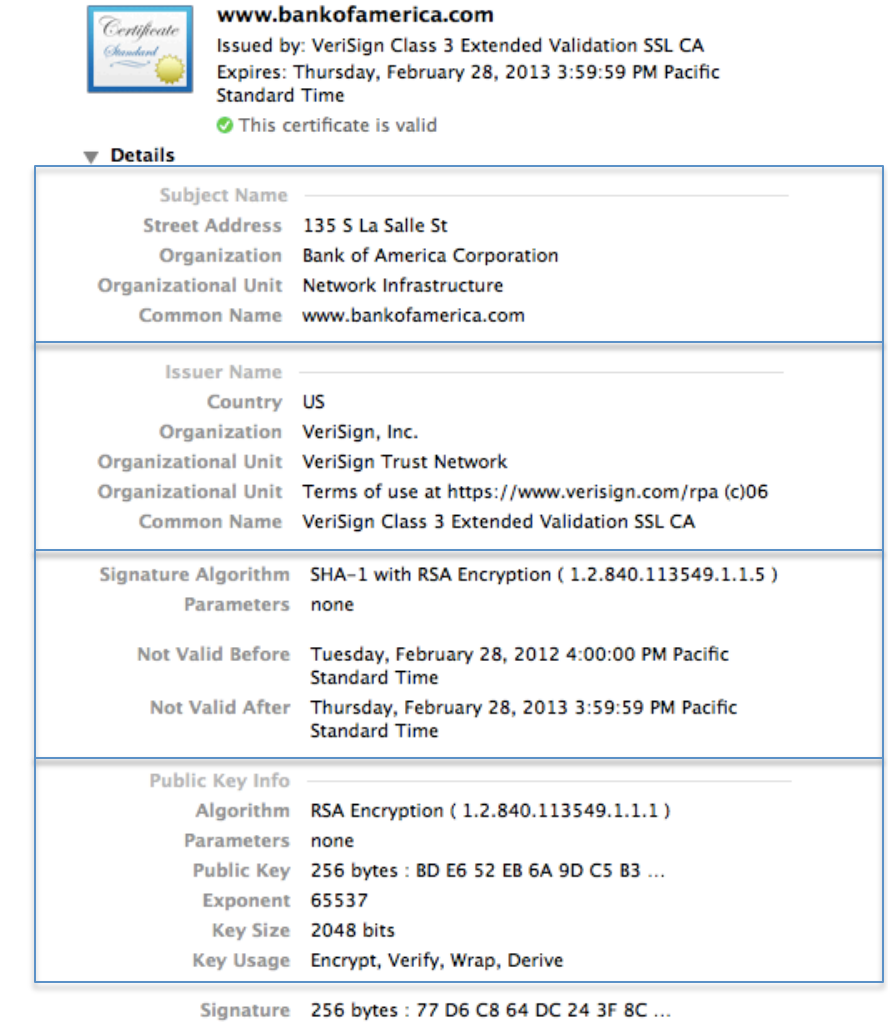
Certificates: bind Bob's ID to his PK

How does Alice (browser) obtain Bob's public key pk_{Bob} ?



Bob uses Cert for an extended period (e.g. one year)

Sample certificate:



www.bankofamerica.com

Issued by: VeriSign Class 3 Extended Validation SSL CA

Expires: Thursday, February 28, 2013 3:59:59 PM Pacific Standard Time

This certificate is valid

Details

Subject Name	
Street Address	135 S La Salle St
Organization	Bank of America Corporation
Organizational Unit	Network Infrastructure
Common Name	www.bankofamerica.com
Issuer Name	
Country	US
Organization	VeriSign, Inc.
Organizational Unit	VeriSign Trust Network
Organizational Unit	Terms of use at https://www.verisign.com/rpa (c)06
Common Name	VeriSign Class 3 Extended Validation SSL CA
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Tuesday, February 28, 2012 4:00:00 PM Pacific Standard Time
Not Valid After	Thursday, February 28, 2013 3:59:59 PM Pacific Standard Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : BD E6 52 EB 6A 9D C5 B3 ...
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes : 77 D6 C8 64 DC 24 3F 8C ...

Certificate Issuance Woes

Wrong issuance:

2011: Comodo and DigiNotar CAs hacked,
incorrectly issue certs for

gmail.com, yahoo.com, and many others

2009: Etisalat CA signs software patch on behalf of RIM

⇒ Resulting signatures are incorrectly trusted by verifier

Certificate revocation

What happens if Bob loses his secret key sk?

- Certificate on pk_{bob} must be revoked

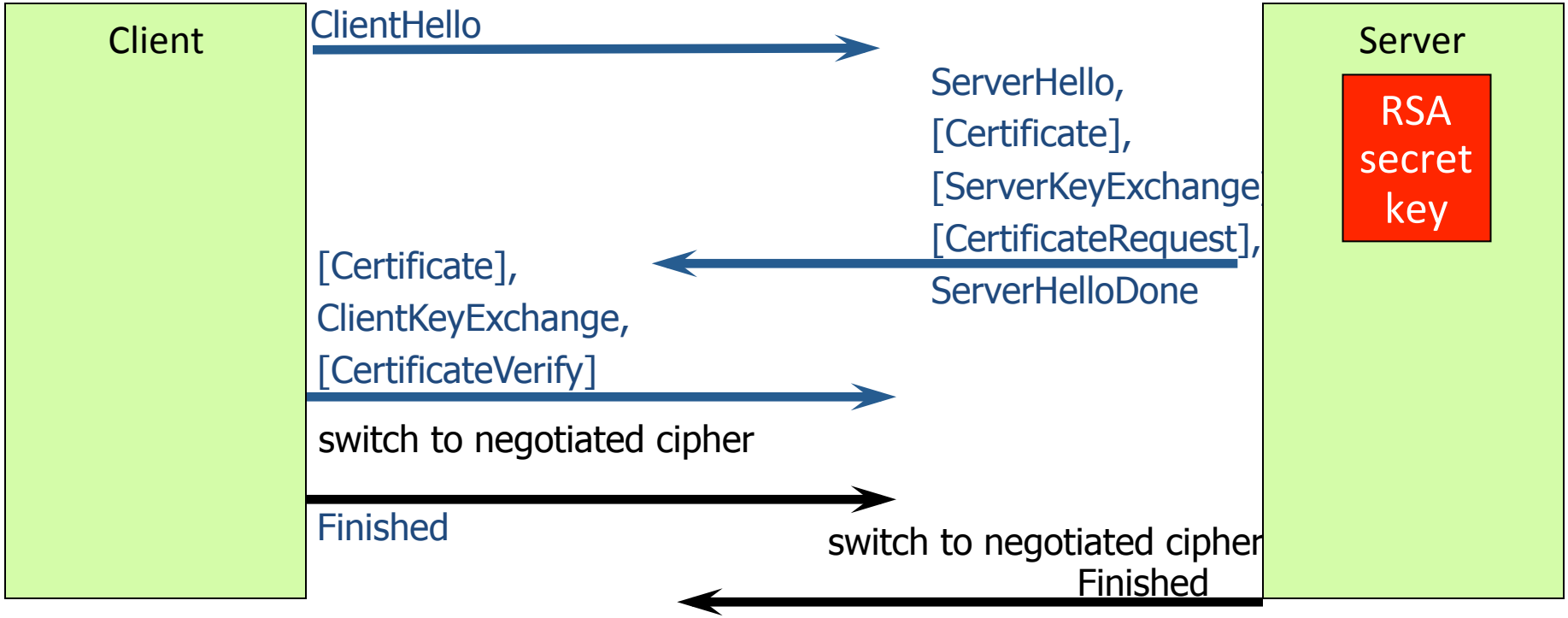
Revocation methods:

- Expiration: certificates active in fixed time window (one year)
- Certificate Revocation Lists (CRLs):

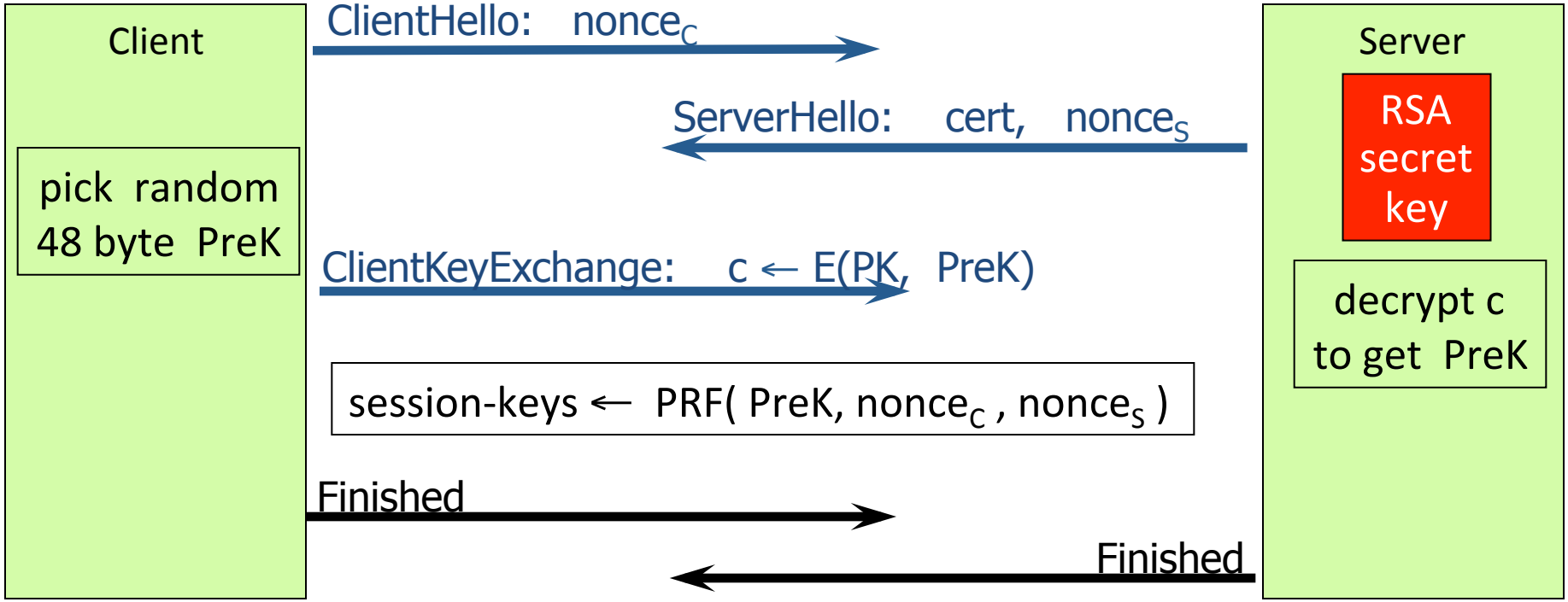
  CA publishes a list of revoked certificates
- Online Certificate Status Protocol (OCSP)

Example Key Exchange: TLS

TLS session setup



Abstract TLS (simplified)

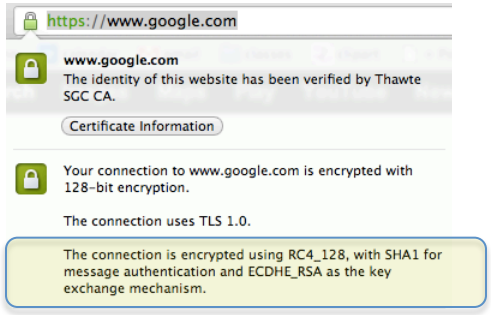


Properties

Nonces: prevent replay of an old session

No forward secrecy:

- Compromise of server secret key exposes old sessions
- TLS has support for forward secrecy



Properties

One sided identification:

- Browser identifies server using server-cert
- Server leans nothing about client's id

TLS has support for mutual identification

- Rarely used: requires a client sk and client-cert