Practical Cryptographic Systems

Asymmetric Cryptography IV/Protocols

Instructor: Matthew Green

Housekeeping

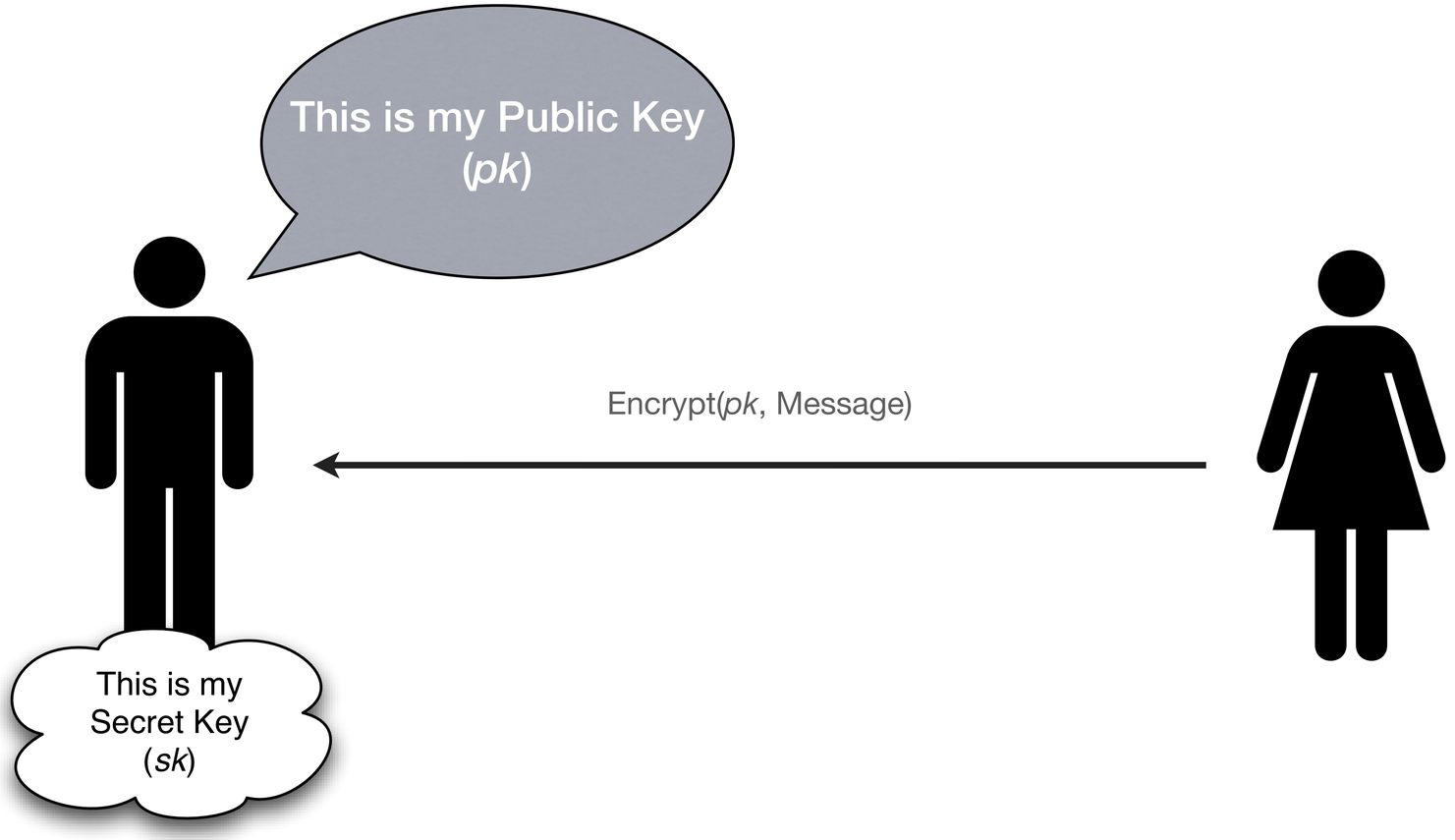
- A2 will be out on Monday
- Reading assignment (short!) out tonight Weds
- TA is working on grades
  - (Reminder: we have a late-day policy, 120 hours.)

News?

Review

- RSA

Public Key Encryption



Quick reminder: Euler/Fermat's little theorem

$$a^{\phi(N)} \equiv 1 \ mod \ N$$

$$\forall a, N : gcd(a, N) = 1$$

Implies….

$$a^{\phi(N)+1} \equiv 1 \cdot a \equiv a \ mod \ N$$

Quick reminder: Euler/Fermat's little theorem

$$a^{\phi(N)} \equiv 1 \ mod \ N$$

$$\forall a, N : gcd(a, N) = 1$$

Implies….

$$a^{\phi(N)+1} \equiv 1 \cdot a \equiv a \ mod \ N$$

Q: Can we split this into two separate keys (e, d)?

RSA Cryptosystem

Choose large primes: p, q

$$N = p \cdot q$$

$$\phi(N) = (p-1)(q-1)$$

Choose:

$$e \; : \; gcd\,(e, \phi(N)) = 1$$

$$d \; : \; ed \; mod \; \phi(N) = 1$$

Output:

$$pk = (e, N)$$

$$sk = d$$

Encryption

$$c = m^e \; mod \; N$$

Decryption

$$m = c^d \; mod \; N$$

“Textbook RSA”

- In practice, we don’t use Textbook RSA
  - Fully deterministic (not semantically secure)
  - Malleable
  - Might be partially invertible

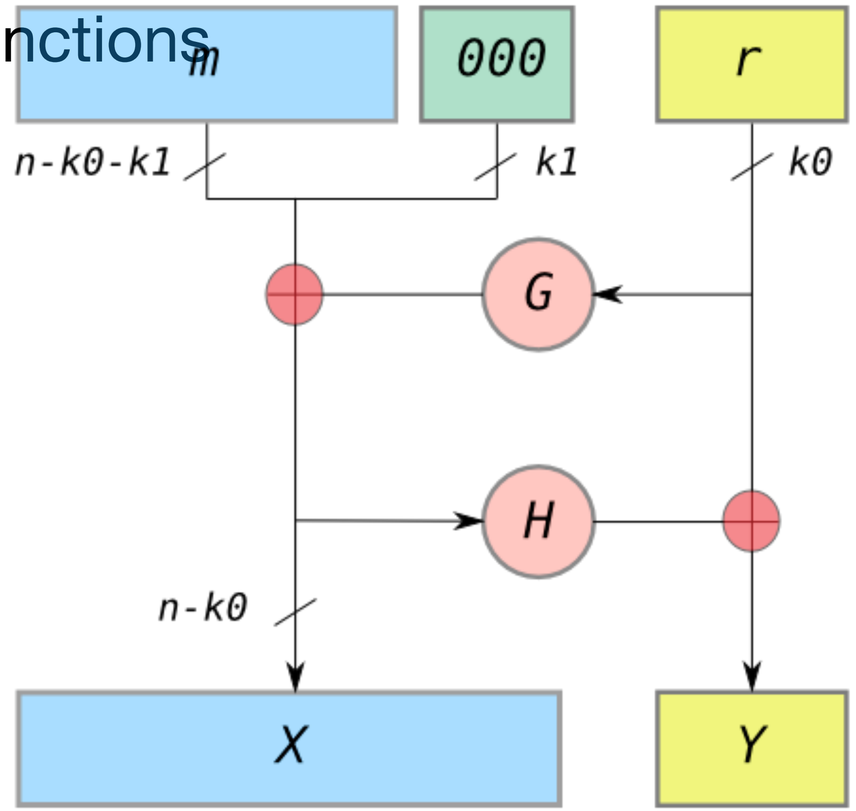
•Coppersmith’s attack: recover part of plaintext (when *m* and *e* are small)

RSA Padding

- Early solution (RSA PKCS #1 v1.5):
  - Add “padding” to the message before encryption
  - Includes randomness
  - Defined structure to mitigate malleability
  - PKCS #1 v1.5 badly broken (Bleichenbacher)

RSA Padding

- Better solution (RSA-OAEP):
  - G and H are hash functions



Hybrid Encryption

- Mixed Approach
  - Use PK encryption to encrypt a symmetric key
  - Use (fast) symmetric encryption on data

Digital Signatures

- Similar to MACs, with public keys
  - Secret key used to sign data
  - Public key can verify signature
  - Advantages over MACs?

Digital Signatures

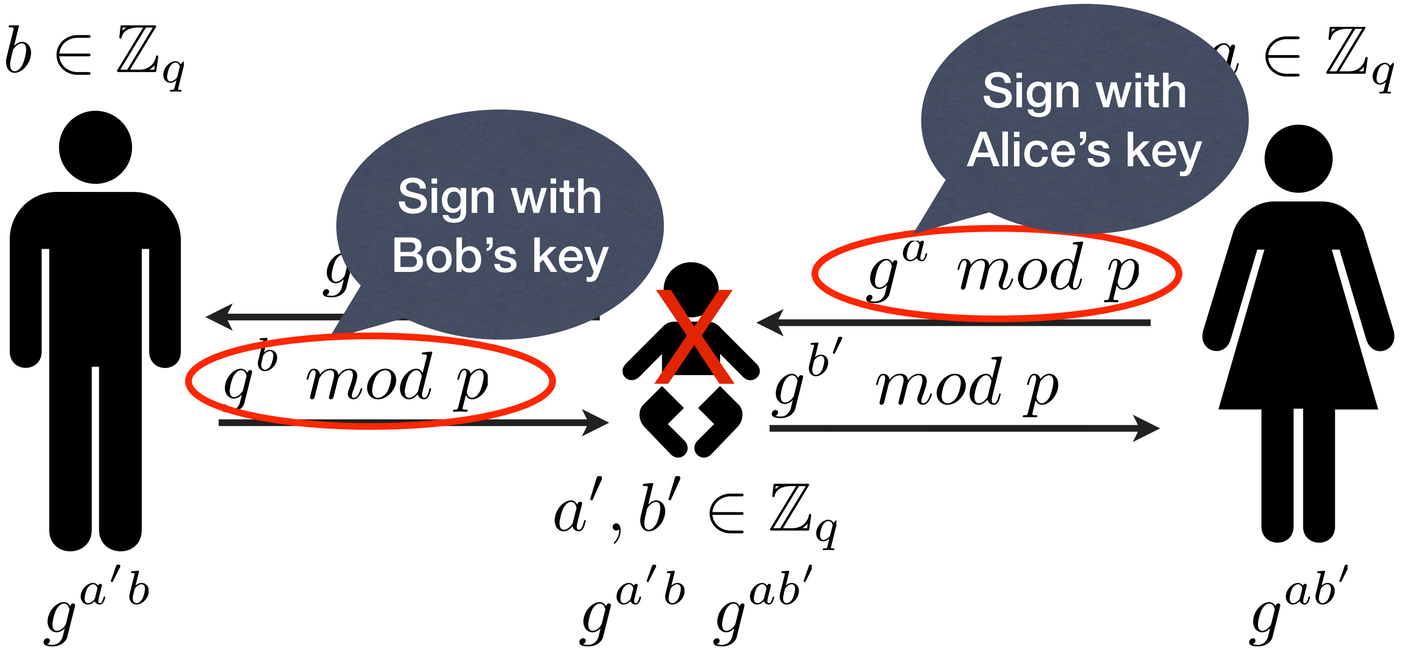
- Three algorithms:
  - Keygen(1^k) -> sk, vk
  - Sign(pk, sk, M) -> sig
  - Verify(pk, M, sig) -> True/False

How do we build signatures?

- RSA signature
- Schnorr signature
  - EdDSA signature
  - DSA/ECDSA (related)

Preventing MitM

- Assume an active adversary:



PKI & Certificates

- How do I know to trust your public key?
  - Put it into a file with some other info, and get someone else to sign it!

SSL/TLS

- Transport-layer security protocol
  - Often used to secure reliable protocols (TCP)
  - Does not require pre-shared keys
  - Most common usage: https

-E-commerce ($200bn/2008), Banking, etc.

Bank of America

Bank of Opportunity™

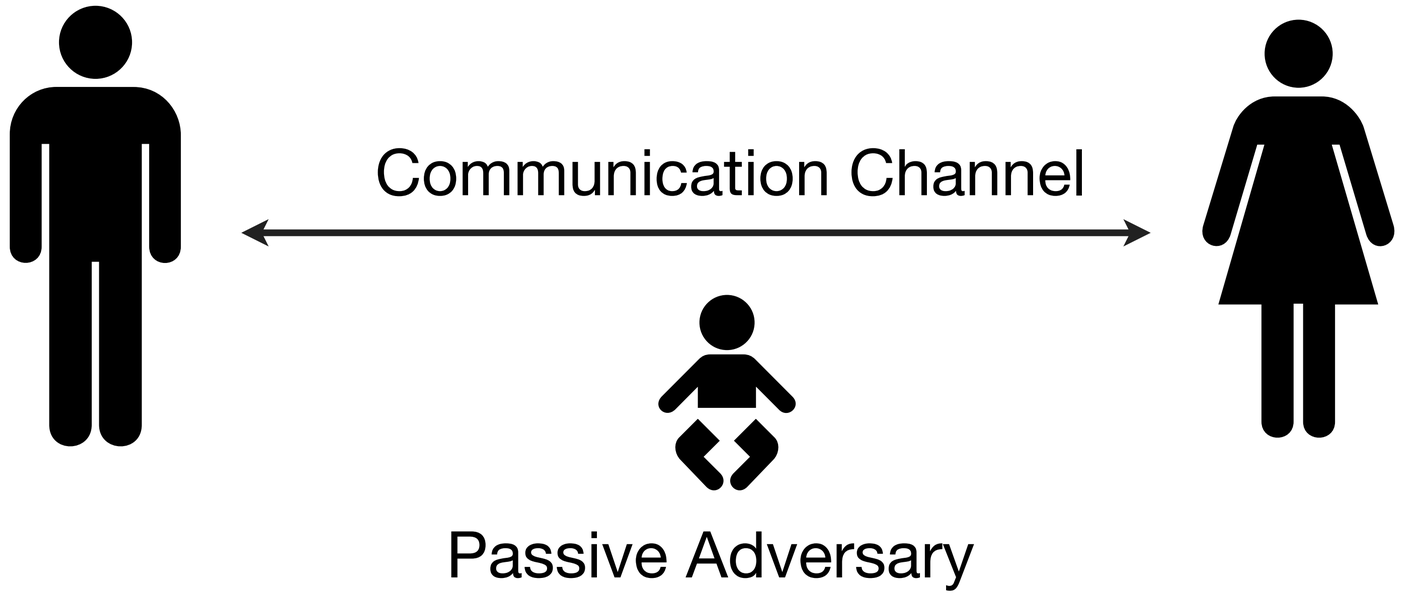
Protocols (definition)

- Definition
  - “A set of rules or procedures for transmitting data between electronic devices, such as computers”
  - “A security protocol (cryptographic protocol or encryption protocol) is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods”

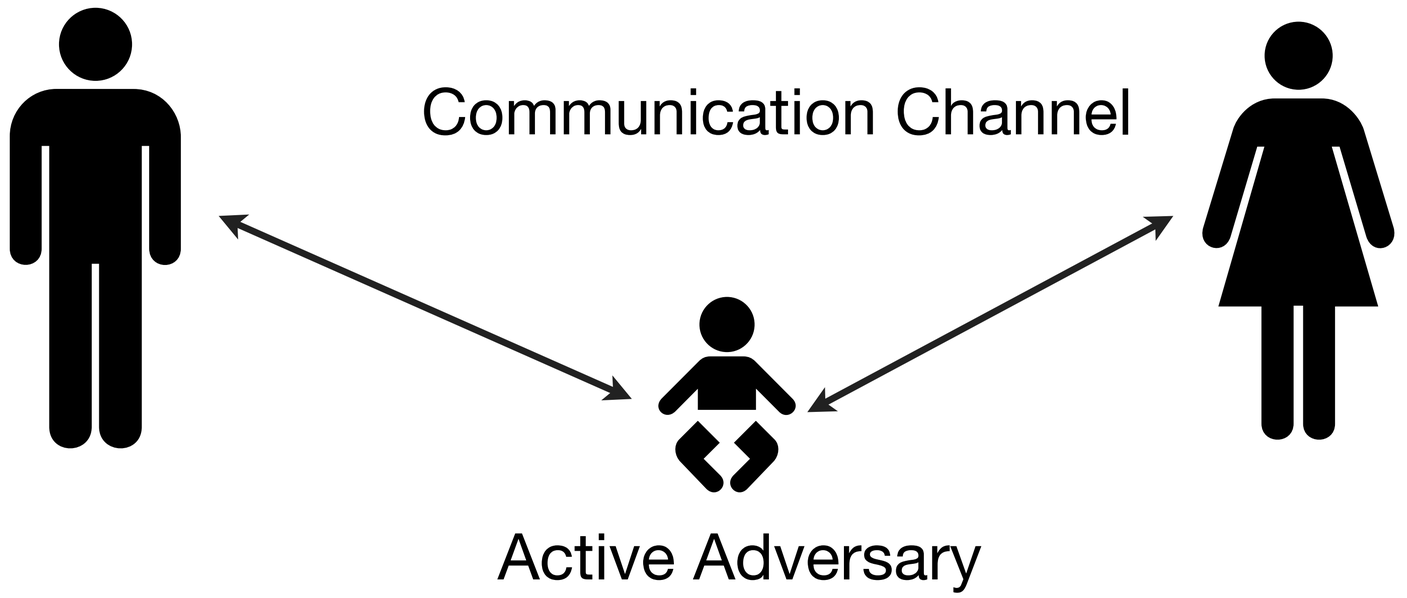
Why not just use primitives?

- A primitive (algorithm) can sometimes be a “protocol”
- But generally there’s more to a protocol
- E.g., TLS:
  - Negotiation (what version are you running?)
  - Authentication (who are you?)
  - Key exchange (let’s get a shared key)
  - Authenticated Encryption (let’s exchange data)

Threat Model

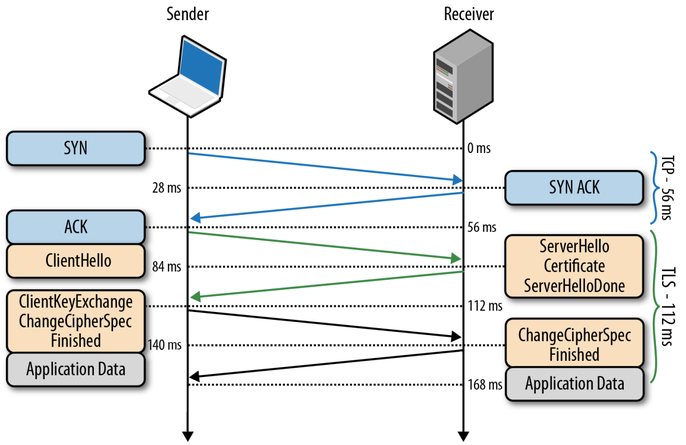


Threat Model



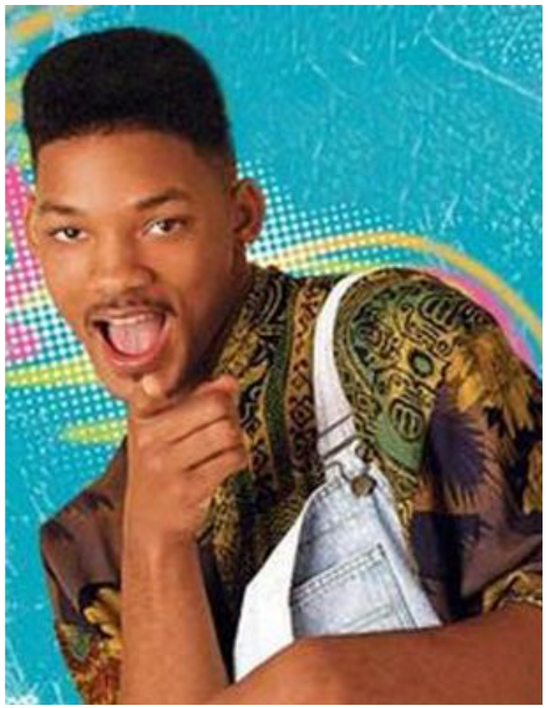
SSL/TLS

- Most important security protocol on the Internet
  - Allows secure connections between clients & servers
  - Current version: TLS 1.3 (RFC 8446)
    - (But browsers still support SSL 3
  - Not just web browsing!



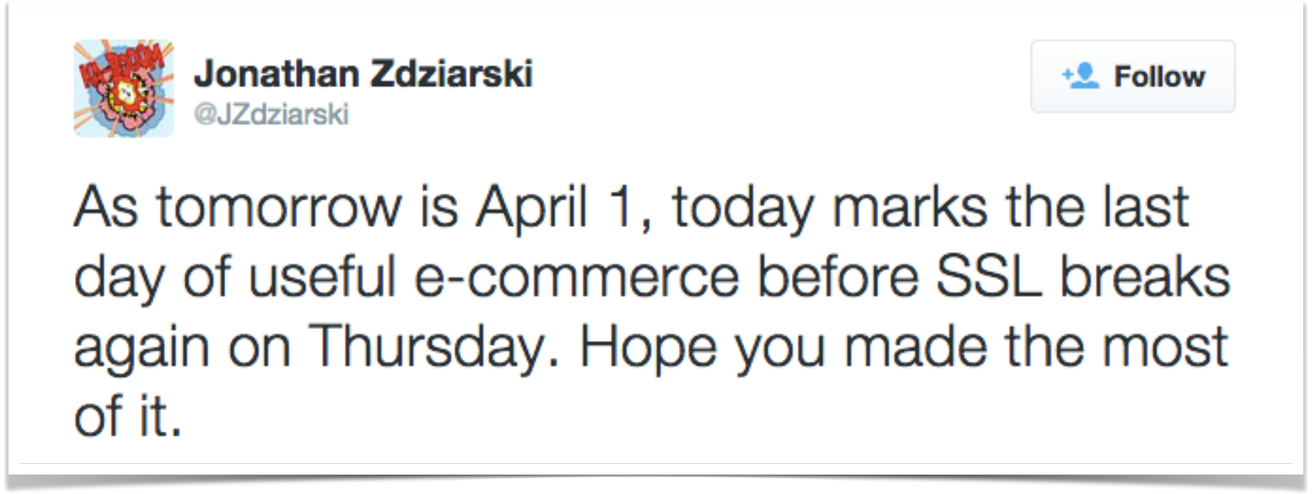
A brief history

- **SSLv1 born at Netscape. Never released. (~1994)**
- **SSLv2 released one year later**
- **SSLv3 (1996)**
- **TLS 1.0 (1998)**
  - Still widely deployed
- **TLS 1.1 (2006)**
- **TLS 1.2 (2008)**



How secure is TLS?

- **Many active attacks and implementation vulnerabilities**
  - Heartbleed, Lucky13, FREAK, CRIME, BEAST, RC4



Why these problems?

- Many problems result from TLS's use of "*pre-historic cryptography*" (- Eric Rescorla)
  - Export grade encryption
  - RSA-PKCS#1v1.5 encryption padding
  - RC4
  - DH parameter generation
  - Horrifying backwards compatibility requirements

Quite a bit

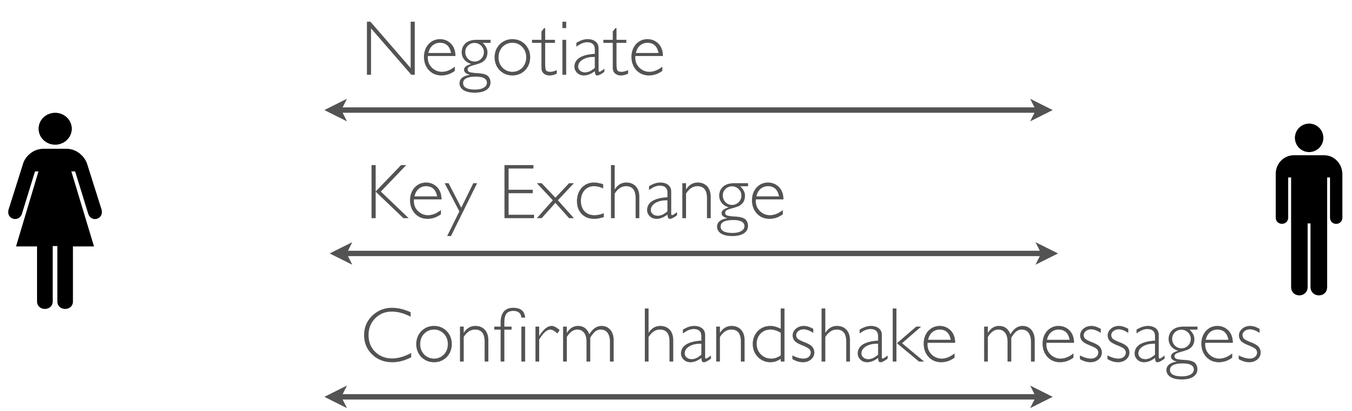
- Many problems result from TLS's use of "*pre-historic cryptography*" (- Eric Rescorla)
  - **Export**
  - RSA-P
  - RC4
  - DH par
  - Horrify

1995-~2000 (and onward)

Weakened "ciphersuites" with limited security
(e.g., 512-bit DH/RSA, 40-bit RC4)

TLS Negotiation

Each TLS handshake begins with a cipher suite negotiation that determines which key agreement protocol (etc.) will be used.



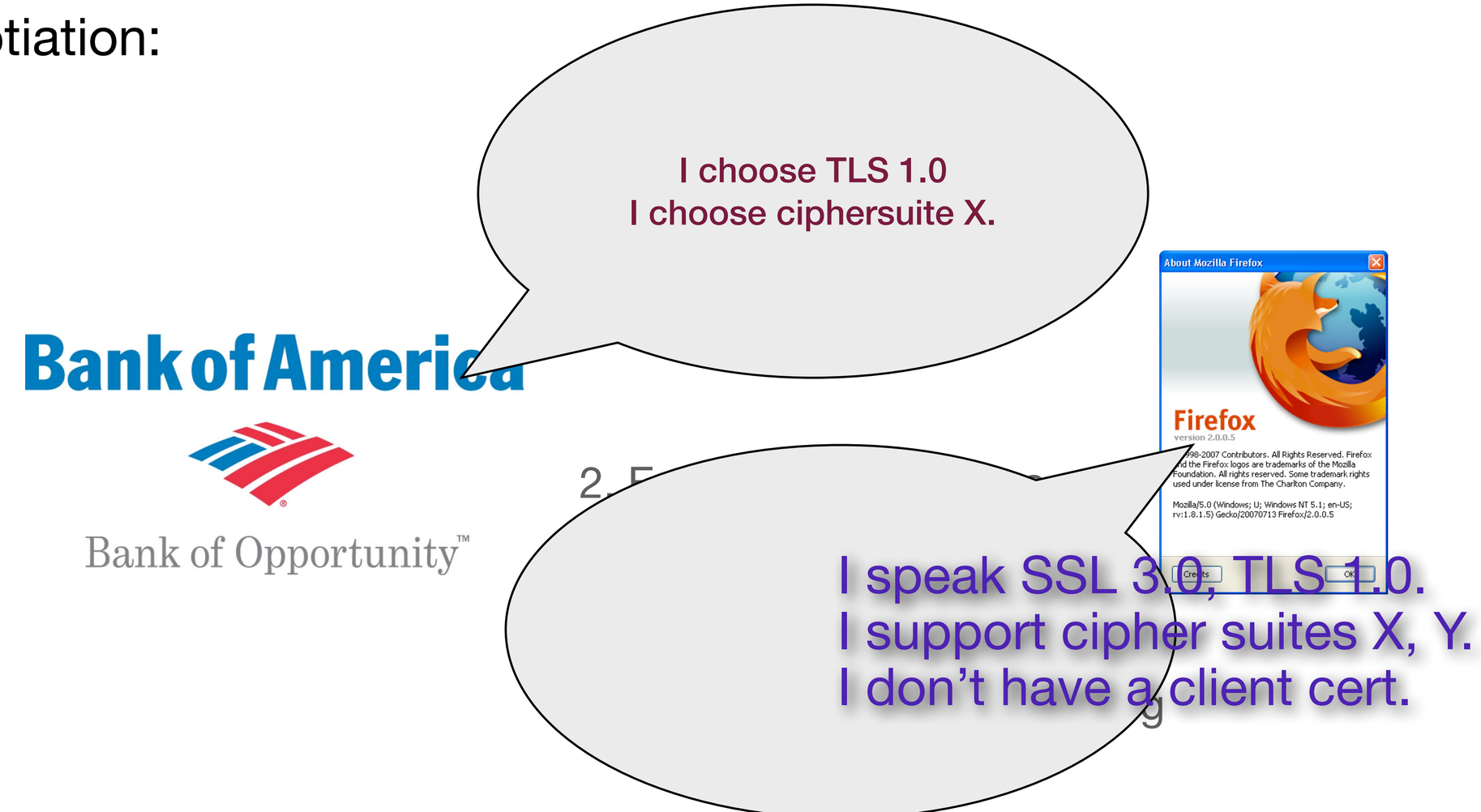
SSL/TLS

Bank of America

Bank of Opportunity™

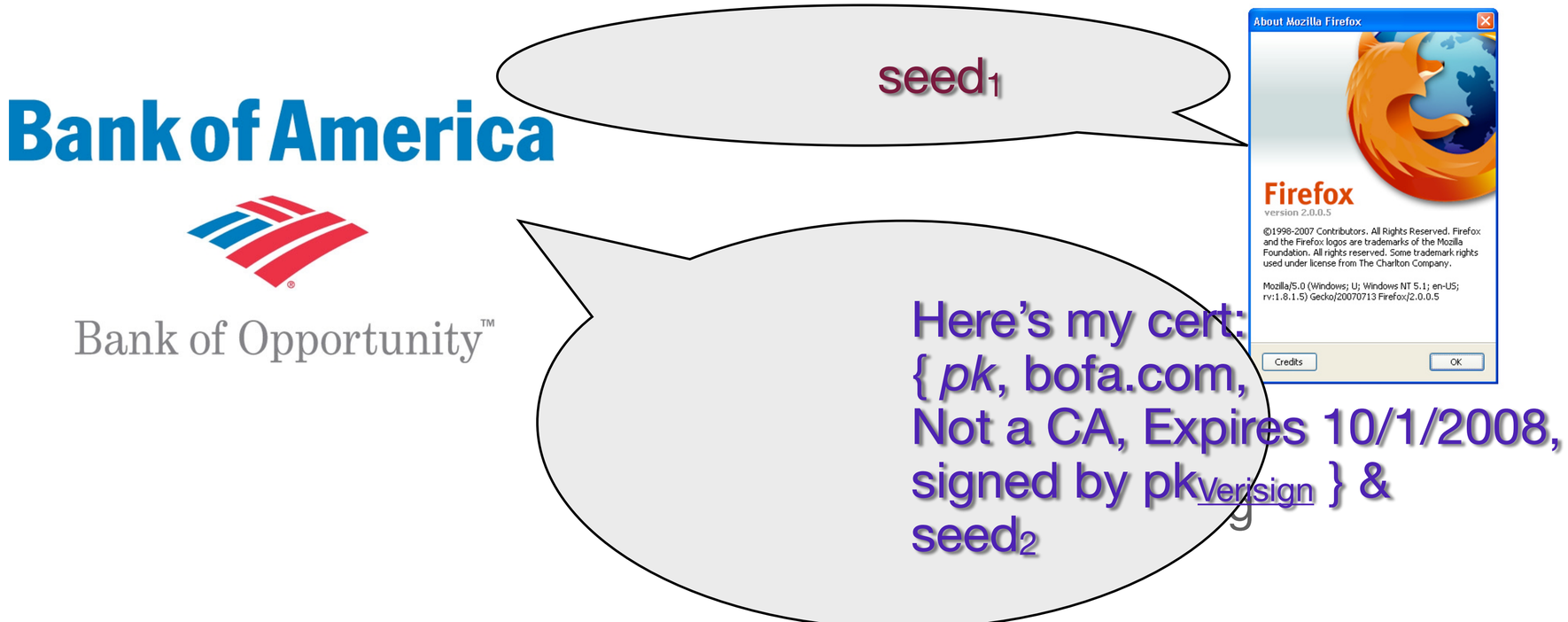
SSL/TLS

- Negotiation:



SSL/TLS

- Certificate Exchange



SSL/TLS

- Session key establishment
  - Various options
  - Common approach: RSA based

$C = RSA\text{-}ENC_{pk}(seed_3)$

1. Negotiate peer capabilities
2. Exchange certificates
3. Secure communi
4. Session expira

$seed_3 = RSA\text{-}DEC(sk, C)$

$k_s = H(seed_1 \,||\, seed_2 \,||\, seed_3)$

$k_s = H(seed_1 \,||\, seed_2 \,||\, seed_3)$

SSL/TLS

- Secure communication
  - In practice, we derive separate MAC & encryption keys

Bank of America

Bank of Opportunity™

1. Negotiate peer capabilities
2. Exchange certificates
3. Session key establishment

5. Session expiration/rekeying

SSL/TLS

- Key expiration/rekeying
  - Key has a defined lifetime
  - If session drops within that lifetime, we restart:

•This shortcut saves PK operations

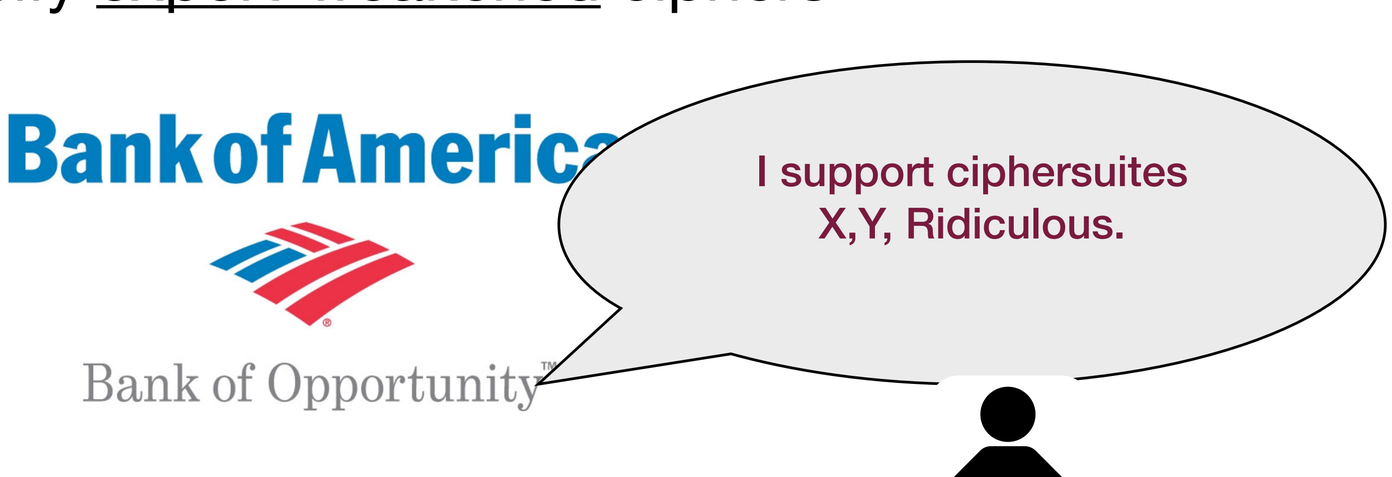
Bank of America

Bank of Opportunity™

1. Negotiate peer capabilities
2. Exchange certificates
3. Session key establishment
4. Secure communications

Attacks on SSL2

- Many and varied...
- Major vulnerability:
  - Ciphersuite list not authenticated
  - Active attacker could modify the message to specify export-weakened ciphers



SSL3

- All of the problems with SSL2 fixed!
- Well, not quite:
  - Ciphersuite rollback attack (weaker)
  - Key-exchange algorithm rollback
  - Version rollback
  - (Weak) traffic analysis
  - Also, uses some non-standard primitives

CCS Rollback

- Most messages sent during client/server handshake are authenticated
  - Final MAC is sent at finish message
  - However, [change cipher spec] message is not included in the MAC
  - Tells the other party to start using encryption/authentication
  - Attacker can modify/drop this message!

CCS Rollback

- Normal protocol:

$$
\begin{array}{lll}
\ldots & & \\
1. & C \rightarrow S: & [\textbf{change cipher spec}] \\
2. & C \rightarrow S: & [\textbf{finished:}]\,\{a\}_k \\
3. & S \rightarrow C: & [\textbf{change cipher spec}] \\
4. & S \rightarrow C: & [\textbf{finished:}]\,\{a\}_k \\
5. & C \rightarrow S: & \{m\}_k \\
\ldots & &
\end{array}
$$

CCS Rollback

- MITM attack:

$$
\begin{array}{lll}
\ldots & & \\
1. & C \rightarrow M: & [\textbf{change cipher spec}] \\
2. & C \rightarrow M: & [\textbf{finished:}]\, \{a\}_k \\
2'. & M \rightarrow S: & [\textbf{finished:}]\, a \\
3. & S \rightarrow M: & [\textbf{change cipher spec}] \\
4. & S \rightarrow M: & [\textbf{finished:}]\, \{a\}_k \\
4'. & M \rightarrow C: & [\textbf{finished:}]\, a \\
5. & C \rightarrow M: & \{m\}_k \\
5'. & M \rightarrow S: & m \\
\ldots & &
\end{array}
$$

Key-Exchange Rollback

- SSL3 standard supports two ephemeral key exchange modes:
  - 1. Server publishes ephemeral RSA parameters (signed under its certified signing key)
  - 2. Server publishes ephemeral DH parameters
  - Client may be able to pick which to use
- Why ephemeral key exchange?
  - Advantages of Diffie-Hellman? RSA?