CS 5830
Cryptography

Instructor: Tom Ristenpart

TAs: Yan Ji, Sanketh Menda

CORNELL TECH

HOME OF THE JACOBS INSTITUTE

TLS handshake for RSA transport

Client

Server

Pick random Nc — ClientHello, MaxVer, Nc, Ciphers/CompMethods →

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod — Pick random Ns

Check CERT using CA public verification key ← CERT = (pk , signature over it)

Pick random PMS
C <- Enc(pk,PMS) — C → — PMS <- Dec(sk,C)

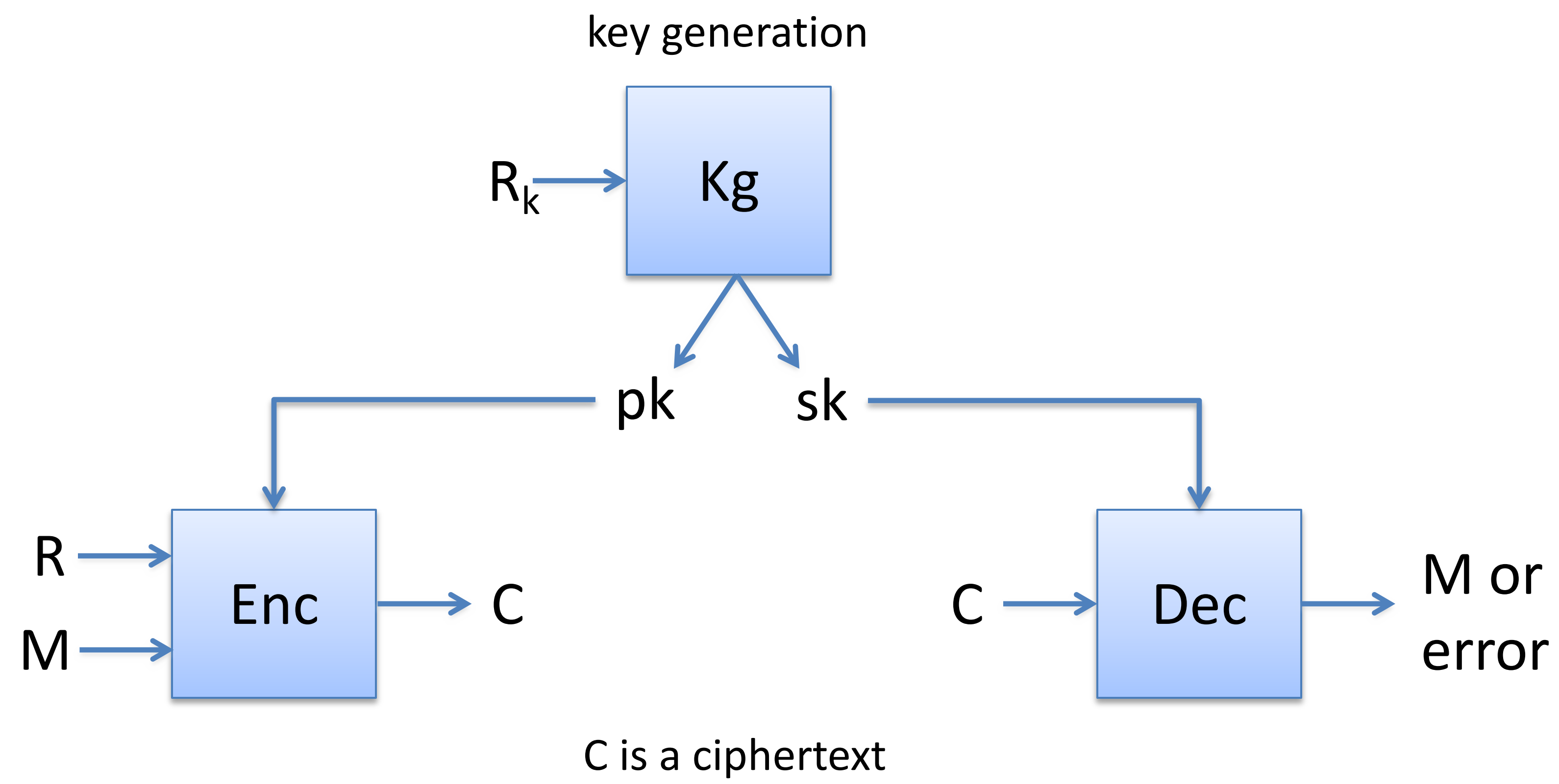
ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, “master secret” || Nc || Ns)

Public-key encryption



Correctness: D(sk , E(pk,M,R)) = M with probability 1 over randomness used

Recap of RSA trapdoor permutation

- $\mathbf{Z}_N^*$ with multiplication mod N is a group
- Find 2 large primes p, q . Let N = pq
  - random integers + primality testing
- Choose e (usually 65,537)
  - Compute d using $\phi(N) = (p-1)(q-1)$
- pk = (N,e) and sk = (N,d)
  - Often store p,q with sk to use Chinese Remainder Theorem
- $f_{N,e}(x) = x^e \bmod N$
- $g_{N,d}(y) = y^d \bmod N$

Permutation on $\mathbf{Z}_N^*$

Textbook RSA (also called "raw RSA")

- Why not just use $M^e \bmod N$ directly to encrypt?
- Lots of reasons:
  - It is deterministic! If M falls in smallish set, can brute-force recover M
  - Ciphertext leaks Jacobi symbol of M (partial information revealed)
  - Meet-in-the-middle attack for short M that requires $2^{|M|/2}$ time and space (e.g., $|M| = 64$)
  - Malleable (chosen-ciphertext attacks!)
  - Small e attacks.
    - $e = 3$ and M smallish, $M^3 < N$, solve by taking 3rd root of M^3

https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf

Raw RSA "meet-in-the-middle" attack

- Suppose m-bit message M can be written as $M = M_1M_2$ where $M_1 \leq 2^{m1}$ and $M_2 \leq 2^{m2}$.
  - Say |M| = 64 bits then probability of random M satisfying above for m1 = m2 = 32 is 18% [Boneh, Joux, Nguyen 2000]
- Given $C = M^e \bmod N$, we have that:

$$C / (M_2)^e \bmod N = (M_1)^e \bmod N$$

- What can we do? Compute all possibilities for left hand side and possibilities for right hand side, find match

Raw RSA "meet-in-the-middle" attack

Step 1: build a table

M_1	M_1^e mod N
0	0
1	1
…	…
2^{m1}	2^{m1e} mod N

2^{m1} time and $O(2^{m1})$ space

Step 2: look for collision

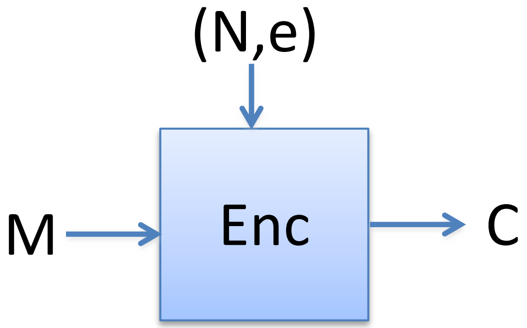
For i = 0 to 2^{m2}
  y <- C/ i^e mod N
  If y in table row j then
    Return i*j
Return ⊥

2^{m2} time

PKCS #1 RSA encryption

Kg outputs (N,e),(N,d)   where $|N|_8 = n$

Want to encrypt messages of length $|M|_8 = m$

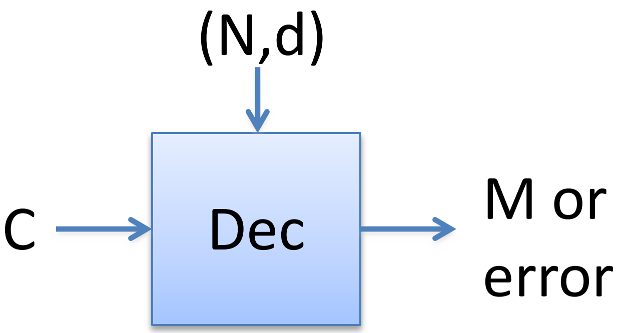


<u>Enc((N,e), M)</u>

pad = $n - m - 3$ random non-zero bytes

X = 00 || 02 || pad || 00 || M

Return X^e mod N



<u>Dec((N,d), C)</u>

X = C^d mod N   ; aa||bb||w = X

If (aa ≠ 00) or (bb ≠ 02) or (00 $\notin$ w)

    Return error

pad || 00 || M = w

Return M

Security of RSA PKCS#1

- Passive adversary sees (N,e),C
- Attacker would like to invert C
- Possible attacks?

Factoring records

Challenge	Year	Algorithm	Time
RSA-400	1993	QS	830 MIPS years
RSA-478	1994	QS	5000 MIPS years
RSA-515	1999	NFS	8000 MIPS years
RSA-768	2009	NFS	~2.5 years
RSA-512	2015	NFS	$75 on EC2 / 4 hours
RSA-795	2019	NFS	4000 core-years (Xeon Gold 6130 CPU

RSA-x is an RSA challenge modulus of size x bits

Formalizing security: IND-CPA

- Indistinguishability under chosen-plaintext attack formal notion can be adapted to PKE setting
  - Only difference: provide public key to adversary
- No formal reduction showing PKCS#1 v1.5 is IND-CPA under RSA one-wayness assumption
  - Coppersmith low-exponent attacks
  - [Coron et al. 2000] attacks for some message formats
- Seems ok in key transport for CPA attacks (if implemented right)

<u>IND-CPA(SE,$\mathcal{A}$):</u>
(M_0,M_1) <-$ $\mathcal{A}$
(pk,sk) <-$ Kg ; b <-$ {0,1}
C <-$ Enc(pk,M_b)
b’ <-$ $\mathcal{A}$(pk,C)
Return (b = b’)

Bleichenbacher attack

Given ciphertext C,
learn $X = C^d \bmod N$

C_1

OK

$\underline{\mathrm{Dec}((N,d), C\,)}$

$X' = C^d \bmod N \quad ; \; aa\,||\,bb\,||\,w = X'$

If $(aa \neq 00)$ or $(bb \neq 02)$ or $(00 \notin w)$

Return FAIL

pad || 00 || M = w

Return OK

$C_1 = C\, s1^e \bmod N$

Response OK:

$X' = (C\, s1^e) \bmod N = X\, s1 \bmod N$

So we know that:

$2 * 2^{8(n-2)} \leq X * s1 \bmod N < 3 * 2^{8(n-2)}$

Leaks some information about X!

Bleichenbacher attack

Given ciphertext C,
learn $X = C^d \bmod N$

C_1

padding error?

C_2

padding error?

...

$C_1 = C\, s1^e \bmod N$

$C_2 = C\, s2^e \bmod N$

$\vdots$

<u>Dec((N,d), C)</u>
$X' = C^d \bmod N$; $aa \| bb \| w = X'$
If $(aa \neq 00)$ or $(bb \neq 02)$ or $(00 \notin w)$
  Return FAIL
$pad \| 00 \| M = w$
Return OK

We can take a target C and decrypt it using a sequence of carefully chosen ciphertexts $C_1, \ldots, C_q$ where $q \approx 1$ million

[Bardou et al. 2012] q = 9400 ciphertexts on average

Hybrid encryption

Hybrid encryption combines asymmetric and symmetric crypto

Kg outputs (pk,sk)

<u>Enc(pk, M)</u>
K <-$ $\{0,1\}^k$
C = Enc(pk,K)
C' = Enc(K,M)
Return (C,C')

Sometimes referred to as
key encapsulation mechanism (KEM)

Sometimes referred to as
data encapsulation mechanism
(DEM)

<u>Dec(sk, (C,C'))</u>
K = Dec(sk,C)
M = Dec(K,C')
Return M

Why hybrid encryption?

- Don't need to extend asymmetric encryption to arbitrary message lengths
- Speed: symmetric DEM faster for long messages

An insecure example: QQ Browser

- QQ browser popular in China, 100s millions of users
- 1024-bit RSA key (N,e),(N,d). Choose 128-bit AES key K at random for symmetric encryption. To encrypt message M:

$C = K^e \bmod N$ $\qquad$ $C' = \mathrm{Enc}(K,M)$

C , C’ →

← OK

QQ servers

$X = C^d \bmod N$
K’ = Low128bits(X)
If Dec(K’,C’) fails then Ret FAIL
Ret OK

https://citizenlab.ca/2016/03/privacy-security-issues-qq-browser/ https://arxiv.org/pdf/1802.03367.pdf

An insecure example: QQ Browser

$C = K^e \bmod N$ $\quad$ $C' = \mathrm{Enc}(K, M^*)$

C1 , C1’ →

← OK

C2 , C2’ →

← FAIL

⋮

$X = C^d \bmod N$
$K' = \mathrm{Low128bits}(X)$
If $\mathrm{Dec}(K', C')$ fails then Ret FAIL
Ret OK

First bit of K is 1 if return OK
First bit of K is 0 if return FAIL

$C1 = 2^{127e}\, C \bmod N$
$C1' = \mathrm{Enc}(10^{127}, M)$

$C2 = 2^{126e}\, C \bmod N$
$C2' = \mathrm{Enc}(110^{126}, M)$

⋮

K = …01
Recover full key in 128 queries

https://citizenlab.ca/2016/03/privacy-security-issues-qq-browser/ https://arxiv.org/pdf/1802.03367.pdf

An insecure example: QQ Browser

So many problems!

- Earlier version: used RSA with 128 bit modulus

    245406417573740884710047745869965023463

- Used ms-precision timestamp as randomness source to generate K
- Responses to requests actually didn't use K, used hard-coded key K*

https://citizenlab.ca/2016/03/privacy-security-issues-qq-browser/

Response to CCA attacks

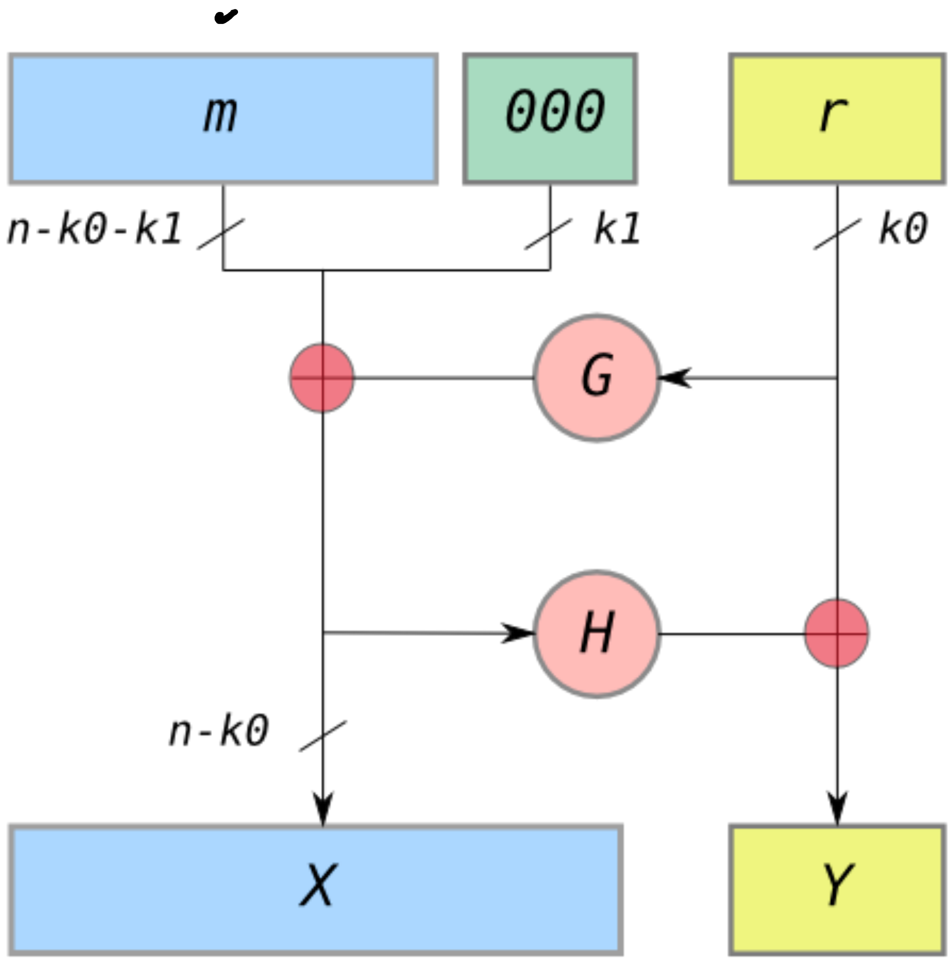
- Ad-hoc fix to Bleichanbacher:
  - Don't leak whether padding was wrong or not
  - This is harder than it looks (timing attacks, control-flow side channel attacks, etc.)
- Better:
  - use chosen-ciphertext secure encryption
  - OAEP is common choice

RSA-OAEP (optimal asymmetric encryption padding)

- Provide better padding scheme than PKCS#1v1.5

- OAEP is such a padding scheme
  - r chosen randomly
  - G,H hash functions
  - C = $(X||Y)^e \bmod N$

- RSA one-wayness implies CCA security



Formalizing security: IND-CPA & IND-CCA for PKE

- Indistinguishability under chosen-plaintext attack formal notion can be adapted to PKE setting
  - Only difference: provide public key to adversary
- Semantic security [Goldwasser, Micali 1984]:
  - Can't learn any predicate over message
  - Equivalent to IND-CPA
- Neither model chosen-ciphertext attacks (like Bleichanbacher's attack)

IND-CPA(SE, $\mathcal{A}$):
(M_0, M_1) <-$ $\mathcal{A}$
(pk,sk) <-$ Kg ; b <-$ {0,1}
C <-$ Enc(pk, M_b)
b' <-$ $\mathcal{A}$(pk,C)
Return (b = b')

Formalizing security: IND-CPA & IND-CCA for PKE

- Can formalize chosen-ciphertext attack (CCA) security for PKE: IND-CCA

<u>IND-CPA(SE,$\mathcal{A}$):</u>
(M_0,M_1) <-\$ $\mathcal{A}$
(pk,sk) <-\$ Kg ; b <-\$ {0,1}
C <-\$ Enc(pk,M_b)
b' <-\$ $\mathcal{A}$(pk,C)
Return (b = b')

Formalizing security: IND-CPA & IND-CCA for PKE

- Can formalize chosen-ciphertext attack (CCA) security for PKE: IND-CCA
- This is different than authenticity in AEAD
  - Why?
  - Anyone can encrypt: ciphertext forgeries trivial!
- Combine digital signatures (stay tuned) with PKE to achieve authenticity in asymmetric setting
- Reduction showing RSA uninvertability => OAEP is IND-CCA

<u>IND-CCA(SE,$\mathcal{A}$):</u>
(M_0,M_1) <-\$ $\mathcal{A}$
(pk,sk) <-\$ Kg ; b <-\$ {0,1}
C <-\$ Enc(pk,M_b)
b' <-\$ $\mathcal{A}^{Dec}$(pk,C)
Return (b = b')

<u>Dec(C')</u>
If C' = C then Return ⊥
M <- Dec(sk,C)
Return M

Summary

- RSA is example of trapdoor one-way function
  - Security conjectured. Relies on factoring being hard
- RSA security scales somewhat poorly with size of primes
- Never use “raw” RSA encryption
- RSA PKCS#1 v1.5 is insecure due to padding oracle attacks. Don’t use it in new systems.
  - Use OAEP instead
- Hybrid encryption (OAEP + AEAD scheme) good for increasing efficiency

TLS handshake for Diffie-Hellman Key Exchange

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod ←

Pick random x
$X = g^x$

$PMS = g^{xy}$

CERT = (pk_s , signature over it) ←

p , g , X , σ = Sign(sk_s, p || g || X) ←

Check CERT
using CA public
verification key
Check σ

Pick random y
$Y = g^y$

$PMS = g^{xy}$

Y →

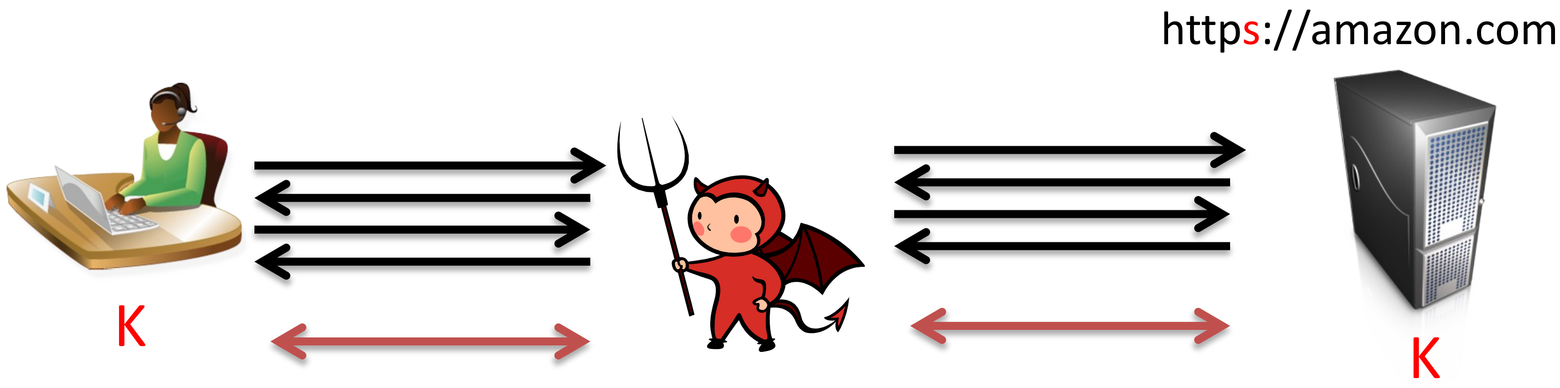
ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

Bracket notation means contents encrypted

ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) } ←

MS <- PRF(PMS, "master secret" || Nc || Ns)

Security goals for TLS



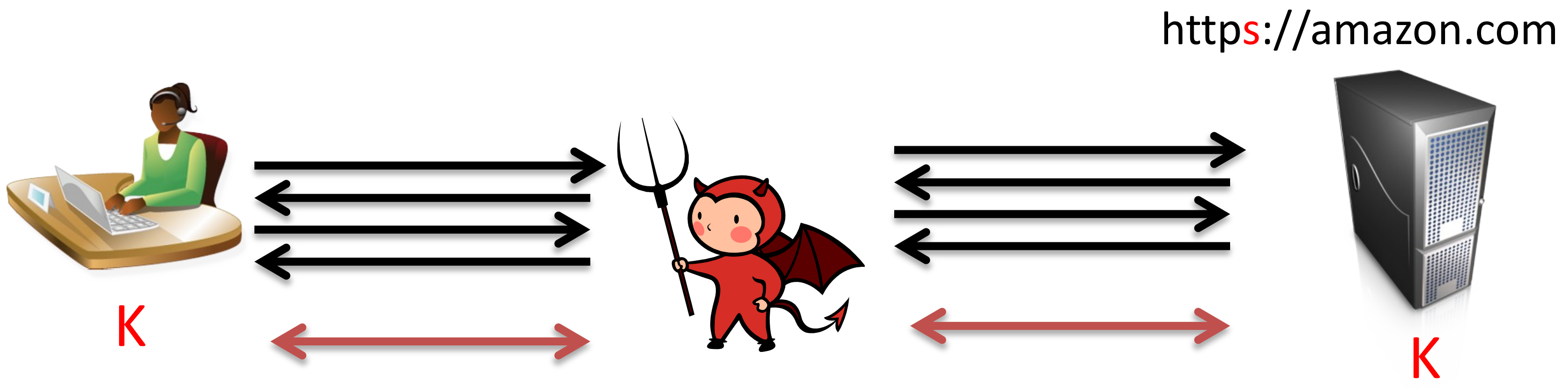
Security goals:

1. Resist passive adversaries
2. Resist active attacker-in-the-middle (often called man-in-the-middle)
3. Forward secrecy: old sessions not decryptable even if server later compromised

Computational Diffie-Hellman problem

$$\begin{tikzcd} X = g^{x}\,,\ Y = g^{y} \arrow[r, bend left, "\text{easy given x or y}"] & g^{xy} \arrow[l, bend left, "\text{hard given just X, Y}"] \end{tikzcd}$$

Security goals for TLS



Security goals:

1. Resist passive adversaries
2. Resist active attacker-in-the-middle (often called man-in-the-middle)
3. Forward secrecy: old sessions not decryptable even if server later compromised

Diffie-Hellman provides forward secrecy
Public-key encryption does not. Why?

Client

TLS handshake for Anonymous Diffie-Hellman

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod ←

Pick random x

$X = g^x$

CERT = (pk_s , signature over it) ←

p , g , X , σ = Sign(sk_s, p || g || X) ←

Check CERT using CA public verification key

Check σ

Pick random y

$Y = g^y$

Y →

PMS = g^{xy}

PMS = g^{xy}

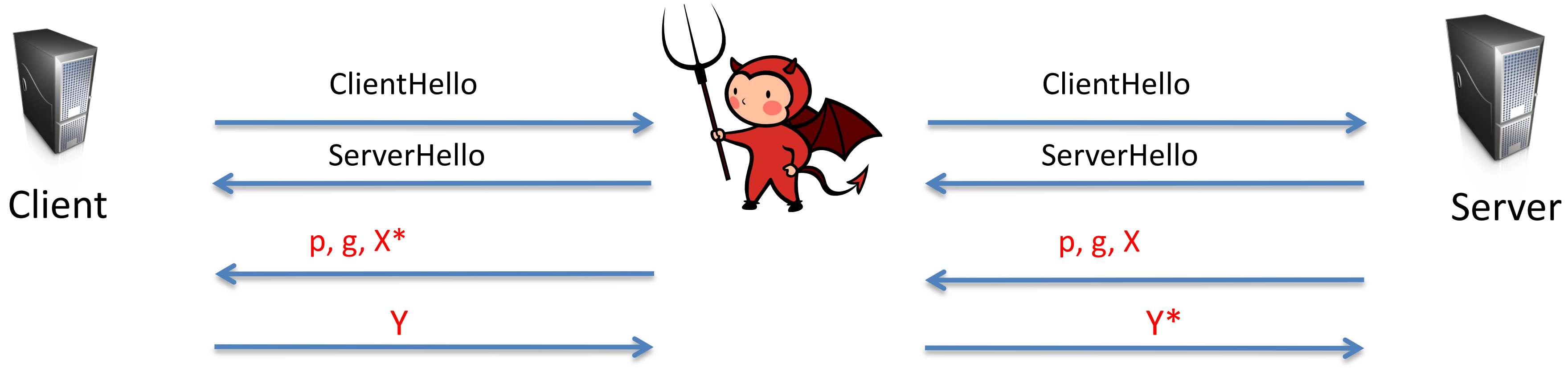
ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) } →

ChangeCipherSpec,
{ Finished, PRF(MS, “Server finished” || H(transcript’)) } ←

Bracket notation means contents encrypted

MS <- PRF(PMS, “master secret” || Nc || Ns)

Adversary-in-the-middle attacks



Attacker can choose X*, Y*, so it knows discrete logs

Completes handshake on both sides

Client thinks its talking to Server

All communications decrypted by adversary, re-encrypted and forwarded to server

Next time

- Will go through Diffie-Hellman more thoroughly
- Discuss meddler-in-the-middle attacks