

## The background of the slide is a collage of various architectural blueprints and technical drawings. These drawings include dimension lines, circles, and text in multiple languages, including English, German, and Russian. The blueprints are layered and cut out to form a geometric, abstract shape that resembles a stylized 'C' or a series of interlocking triangles. The colors are primarily blue and white, with some yellow and red lines visible in the background drawings. The Cornell Tech logo is positioned in the upper right corner, featuring the text 'CORNELL TECH' in a bold, sans-serif font. Below this, the text 'HOME OF THE JACOBS INSTITUTE' is written in a smaller, all-caps font. At the bottom right, there are two circular logos: the Technion logo on the left, which includes the year '1912' and a central emblem, and the Cornell University logo on the right, which includes the text 'CORNELL UNIVERSITY' and 'FOUNDED A.D. 1826'.

HOME OF THE  
**JACOBS**  
**INSTITUTE**



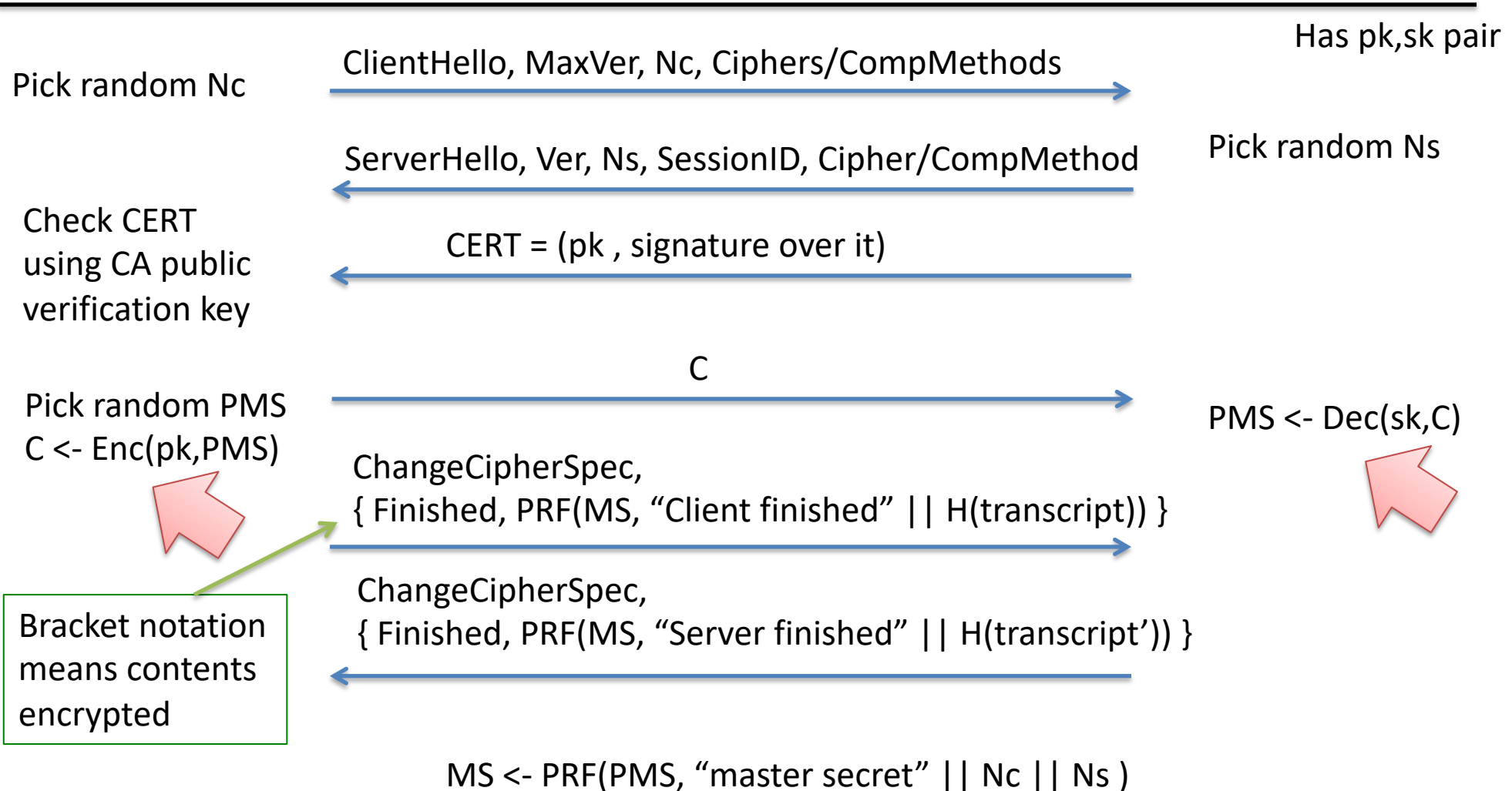


Client

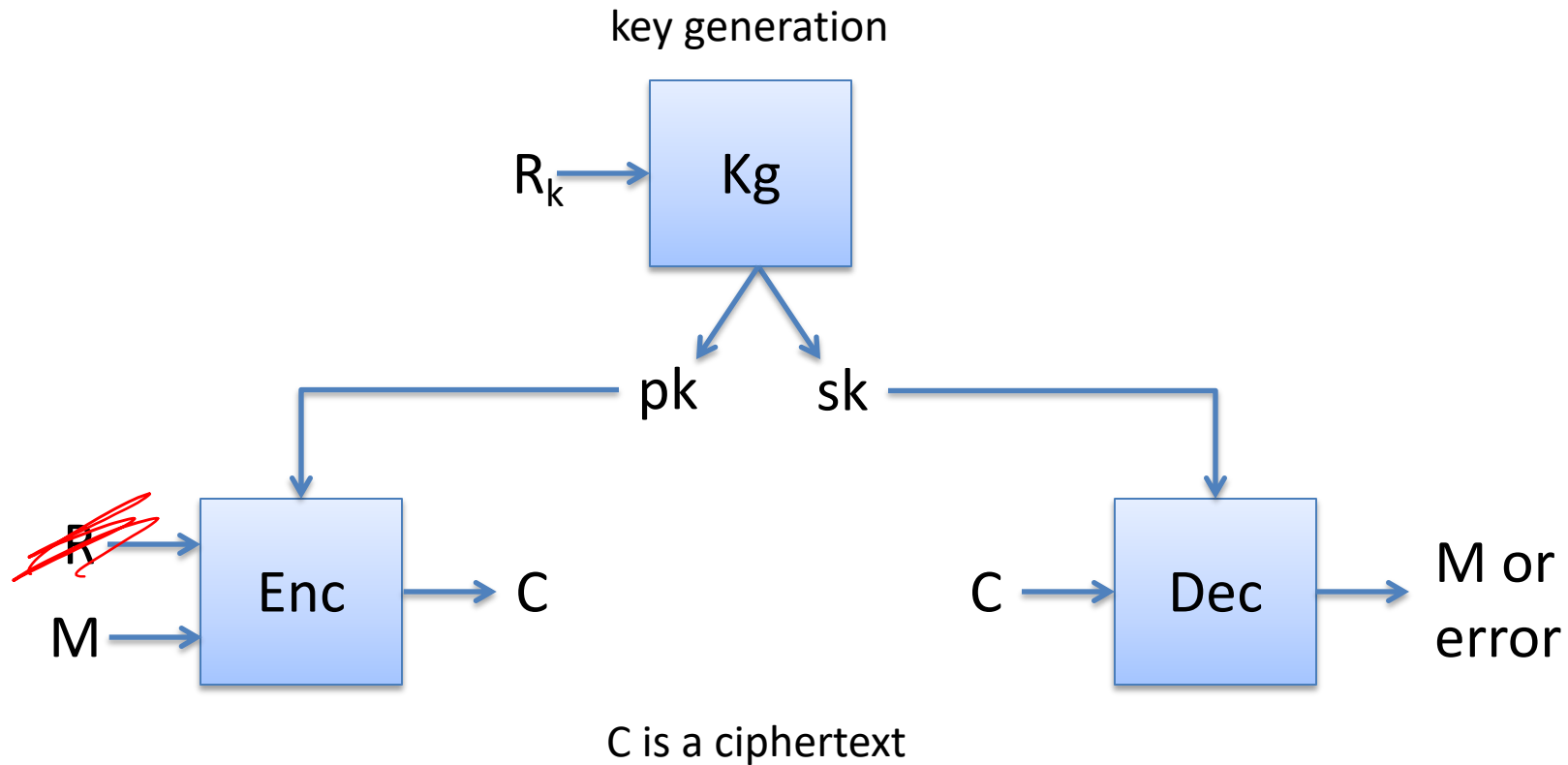
# TLS 1.2 handshake for RSA transport



Server



# Public-key encryption



Correctness:  $\text{Dec}(sk, \text{Enc}(pk, M, R)) = M$  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  $M$
  - 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  $M^3$

$$C = M^e \bmod N$$

$$\tilde{C} \leftarrow C \cdot Z^e \bmod N$$

$$C \bmod N$$

$$(C \cdot Z^e)^d \bmod N$$

$$(M^e \cdot Z^e)^d \bmod N$$

$$(Z \cdot M)^e \bmod N$$

# Raw RSA “meet-in-the-middle” attack

- Suppose  $m$ -bit message  $M$  can be written as  $M = M_1 M_2$  where  $M_1 \leq 2^{m_1}$  and  $M_2 \leq 2^{m_2}$ .
  - Say  $|M| = 64$  bits then probability of random  $M$  satisfying above for  $m_1 = m_2 = 32$  is 18% [Boneh, Joux, Nguyen 2000]
- Given  $C = M^e \bmod N$ , we have that:
  - $C / (M_2)^e \bmod N = (M_1)^e \bmod N$
- Can we recover  $M$  efficiently given  $(N, e)$  and  $C$ ?
  - Compute all possibilities for left hand side and possibilities for right hand side, find match

$M = m_1 m_2 m_3$

$$\begin{aligned} C &= m^e \bmod N \\ &= (m_1 m_2)^e \bmod N \\ &= m_1^e m_2^e \bmod N \end{aligned}$$

# Raw RSA “meet-in-the-middle” attack

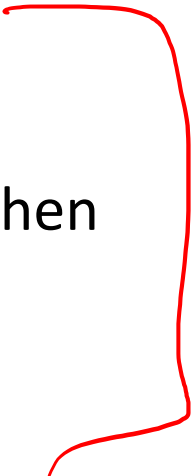
Step 1: build a table

$M_1$	$M_1^e \bmod N$
0	0
1	1
...	...
$2^{m_1}$	$2^{m_1 e} \bmod N$

$2^{m_1}$  time and  $O(2^{m_1})$  space

Step 2: look for collision

For  $i = 0$  to  $2^{m_2}$   
   $y \leftarrow C / i^e \bmod N$   
  If  $y$  in table row  $j$  then  
    Return  $i*j$   
Return  $\perp$

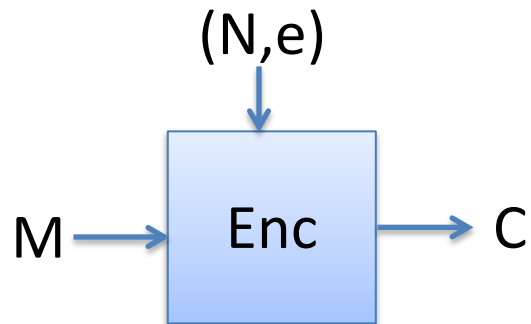


$2^{m_2}$  time

# PKCS #1 ver 1.5 RSA encryption

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

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

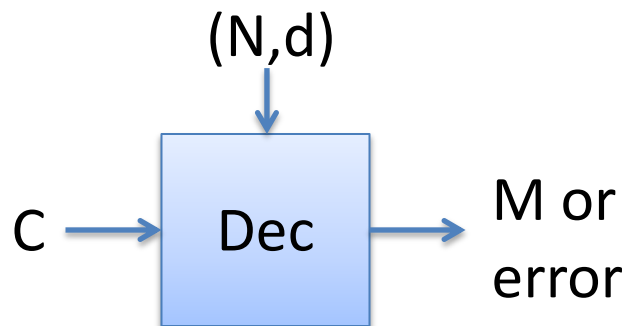


$\text{Enc}((N,e), M)$

$\text{pad} = n - m - 3$  random non-zero bytes

$X = \underline{00 || 02 || \text{pad} || 00 || M}$

Return  $X^e \bmod N$



$\text{Dec}((N,d), C)$

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

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

Return error

$\text{pad} || 00 || M = w$

Return M



# Security of RSA PKCS#1

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

# Factoring algorithms

*W. from Alpha*

Algorithm	Time to factor N
Naïve	$O(e^{0.5 \ln(N)})$
Quadratic sieve (QS)	$O(e^c)$ $c = d (\ln N)^{1/2} (\ln \ln N)^{1/2}$
Number Field Sieve (NFS)	$O(e^c)$ $c = 1.92 (\ln N)^{1/3} (\ln \ln N)^{2/3}$

$\log_2(e^{(1.92 * ((\ln(2^{2048}))^{1/3}) * (\ln(\ln(2^{2048})))^{2/3})})$

NATURAL LANGUAGE  $\int_2^{\pi}$  MATH INPUT

Assuming the principal root | Use [the real-valued root](#) instead

Input

$$\log_2 \left( e^{1.92 \sqrt[3]{\log(2^{2048})} \log^{2/3}(\log(2^{2048}))} \right)$$

Result

116.702...

Also best known attack for RSA one-wayness  
(uninvertibility) problem

Given  $(N, e)$ ,  $X^e \bmod N$  for random  $X$   
Find  $X$

# 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 as reference)
→ RSA-829	2020	NFS	2700 core-years (same as above)


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

MIPS = million instructions per second

Recent academic paper: <https://eprint.iacr.org/2020/697.pdf>

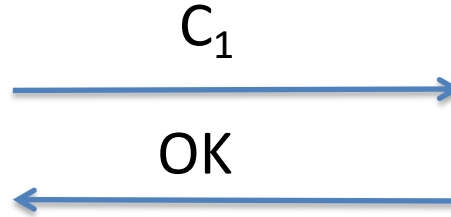
# Formalizing security: IND-CPA

- Indistinguishability under chosen-plaintext attack formal notion can be adapted to PKE setting
  - Except need to 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
- No practical CPA-only attacks against PKCS#1 v1.5 (excluding side channel attacks, and assuming correct implementation)

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

# Bleichenbacher attack

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



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

Response OK:

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

So we know that:

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

Leaks some information about  $X$ !



Dec((N,d), C)

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

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

Return FAIL ←

pad || 00 || M = w

Return OK ←

# Bleichenbacher attack

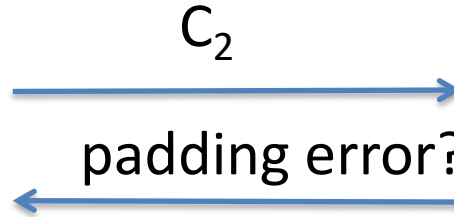
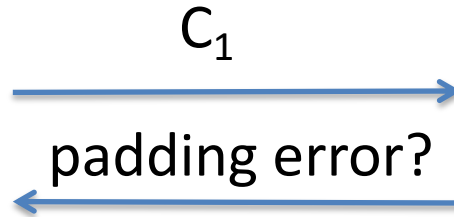
Given ciphertext  $C$ ,  
learn  $X = C^d \bmod N$



$$C_1 = C s_1^e \bmod N$$

$$C_2 = C s_2^e \bmod N$$

⋮



...



Dec((N,d), C)

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

If (aa ≠ 00) or (bb ≠ 02) or (00 ∉ 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  $C_1, \dots, C_q$  where  $q \approx 1$  million

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

# Response to CCA attacks

- Ad-hoc fix to Bleichenbacher:
  - Don't leak whether padding was wrong or not
  - This is much harder than it looks (timing attacks, control-flow side channel attacks, etc.)
- Better use chosen-ciphertext secure encryption
  - Will give example later, first one more example of an CCA-insecure RSA-based scheme

# Hybrid encryption

Public-key encryption that combines asymmetric and symmetric crypto

Kg outputs (pk,sk)

Enc(pk, M)

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

$C = \text{Enc}(pk, K)$

$C' = \text{Enc}(K, M)$

Return (C, C')

Sometimes referred to as key encapsulation mechanism (KEM)

Examples: RSA PKCS #1, raw RSA

*KEM / DEM*

Sometimes referred to as data encapsulation mechanism (DEM)

Example: AES-GCM

Dec(sk, (C, C'))

$K = \text{Dec}(sk, C)$

$M = \text{Dec}(K, C')$

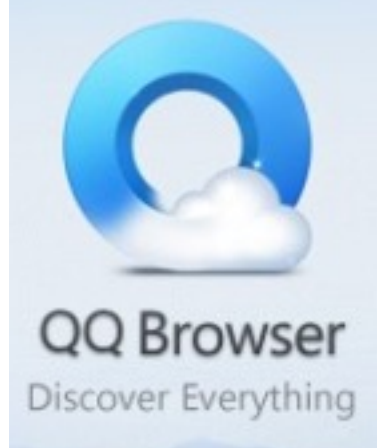
Return M

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 circa 2018



- 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  $M$ :

$$C = K^e \bmod N$$

$$C' = \text{Enc}(K,M)$$



$(2^{127}, C, C')$

OK



QQ servers

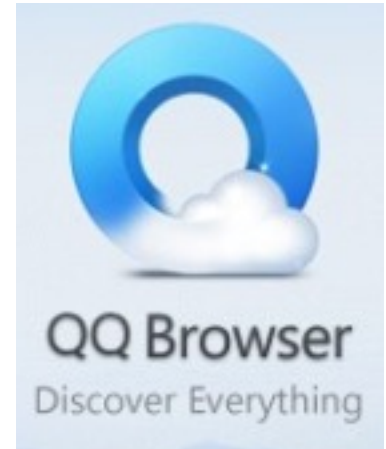
$$X = C^d \bmod N$$

$$K' = \text{Low128bits}(X)$$

If  $\text{Dec}(K', C')$  fails then Ret FAIL

Ret OK

# An insecure example: QQ Browser circa 2018



$$C = K^e \bmod N$$

$$C' = \text{Enc}(K, M^*)$$



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

$C1, C1'$

OK

$C2, C2'$

FAIL

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

:

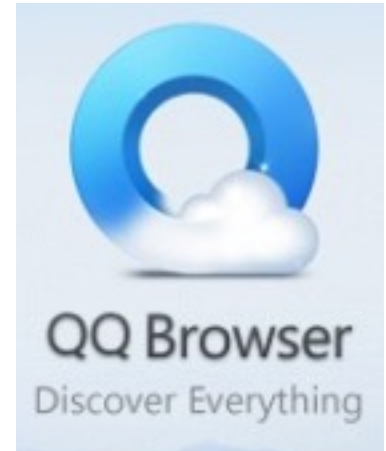
$K = \dots 01$

Recover full key in 128 queries

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

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

# An insecure example: QQ Browser circa 2018

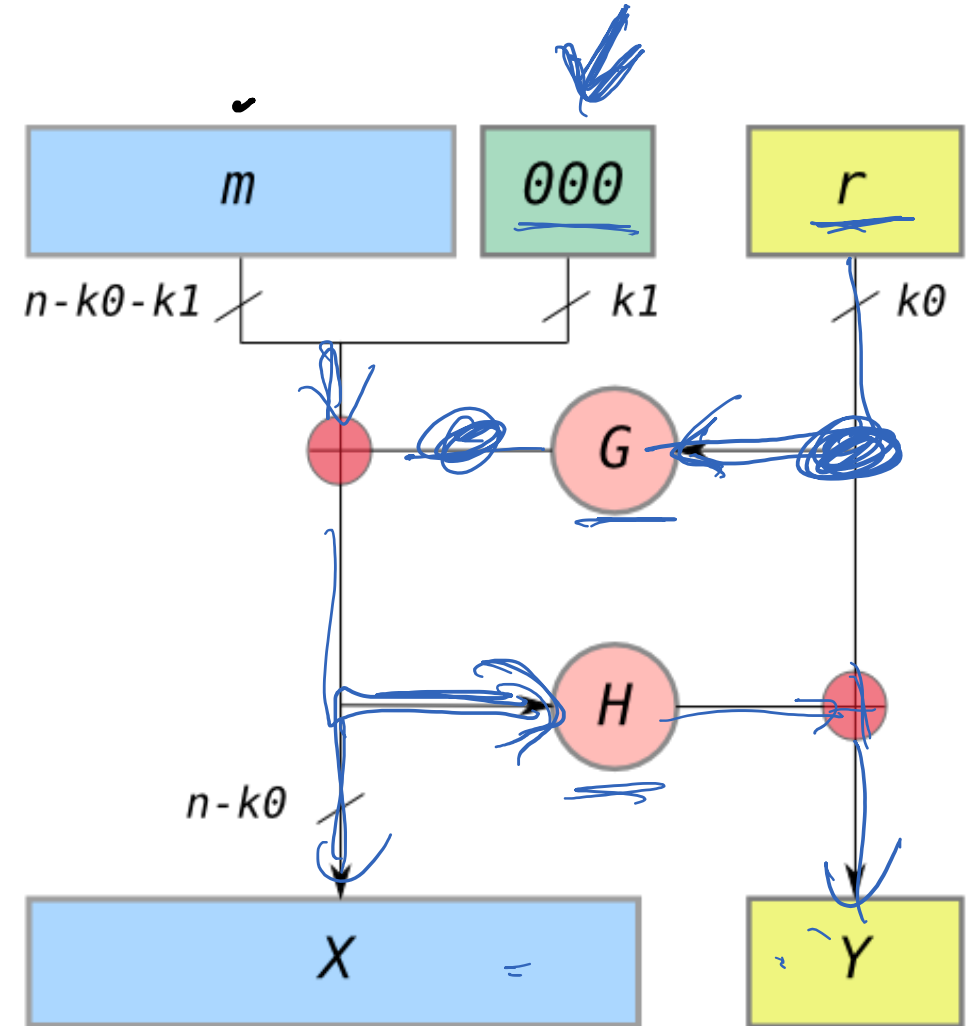


## So many problems!

- 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  $K^*$

# 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

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

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

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

IND-CPA(SE,  $\mathcal{A}$ ):  
 $(M_0, M_1) \leftarrow \$ \mathcal{A}$   
 $(pk, sk) \leftarrow \$ Kg$  ;  $b \leftarrow \$ \{0, 1\}$   
 $C \leftarrow \$ Enc(pk, M_b)$   
 $b' \leftarrow \$ \mathcal{A}(pk, C)$   
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 one-wayness  $\Rightarrow$  OAEP is IND-CCA [Fujisaki et al. 2001]

IND-CCA( $\mathcal{SE}, \mathcal{A}$ ):

$(M_0, M_1) \leftarrow \$ \mathcal{A}$

$(pk, sk) \leftarrow \$ Kg$  ;  $b \leftarrow \$ \{0, 1\}$

$C \leftarrow \$ Enc(pk, M_b)$

$b' \leftarrow \$ \mathcal{A}^{Dec}(pk, C)$

Return  $(b = b')$

Dec( $C'$ )

If  $C' = C$  then Return  $\perp$

$M \leftarrow Dec(sk, C)$

Return  $M$

# Summary

- RSA is example of trapdoor one-way permutation
  - 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



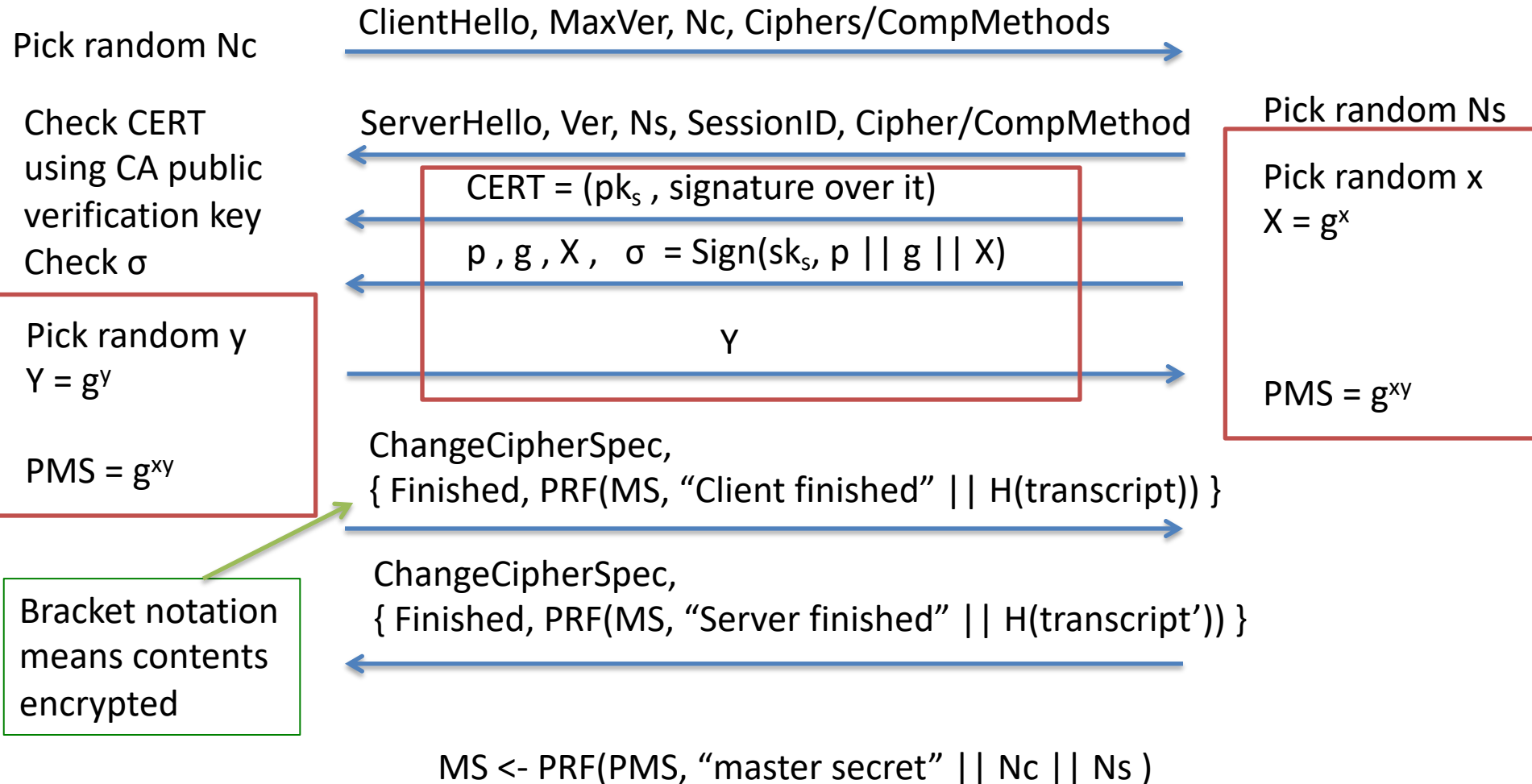


Client

# TLS handshake for Diffie-Hellman Key Exchange

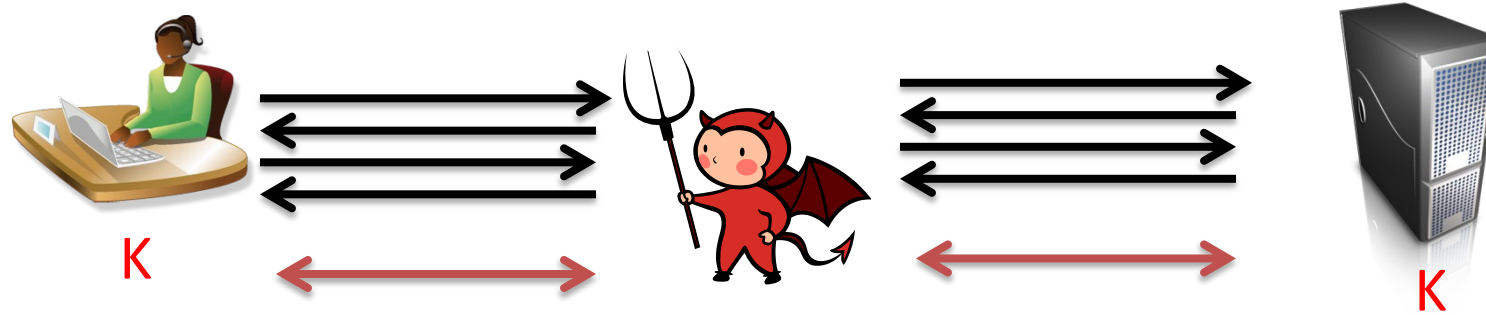


Server



# Security goals for TLS

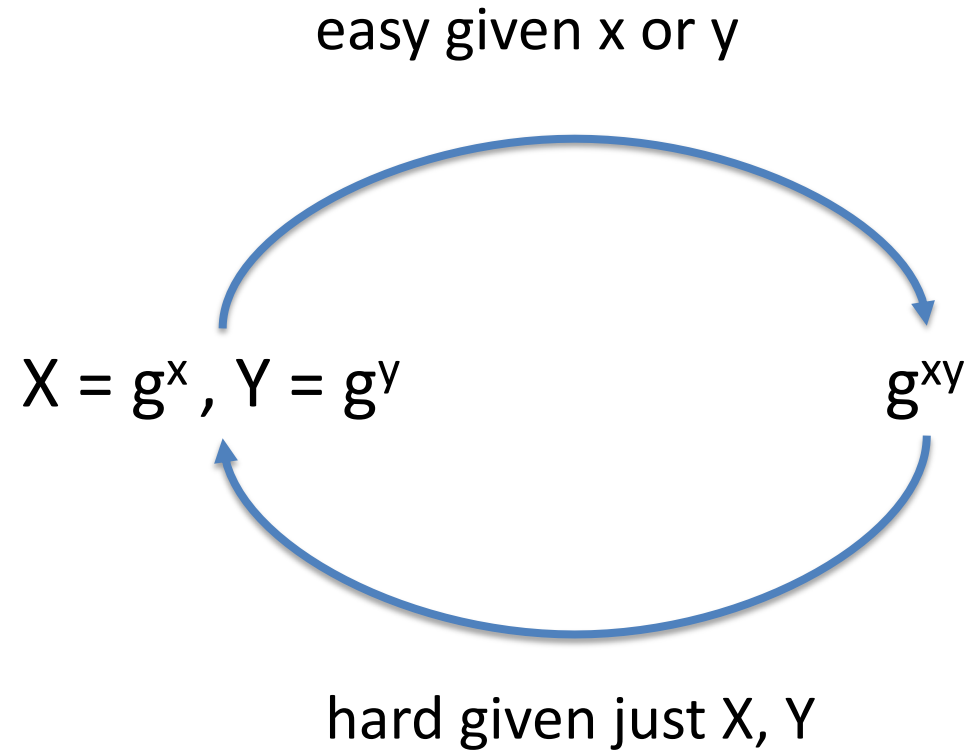
<https://amazon.com>



Security goals:

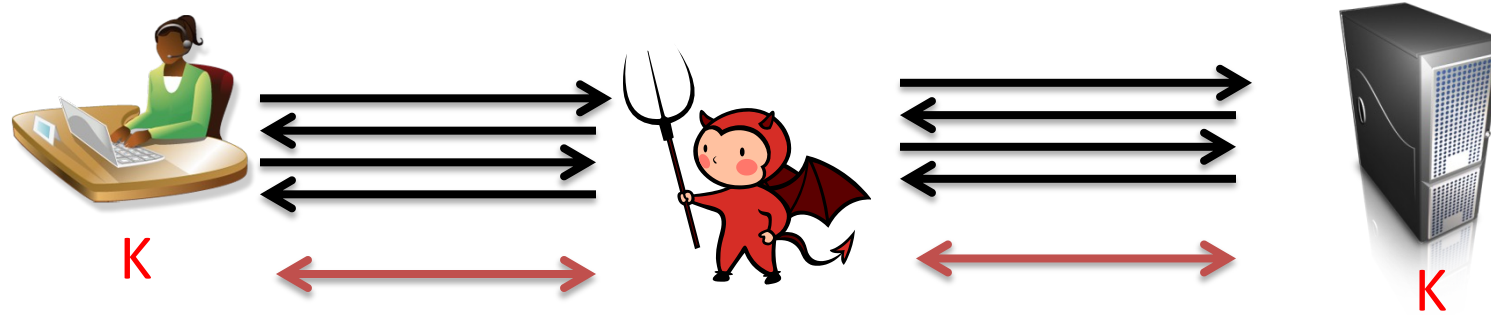
1. Resist passive adversaries
2. Resist active attacker-in-the-middle (often called man-in-the-middle)
3. Forward secrecy: old sessions not decryptable even if server later compromised

# Computational Diffie-Hellman problem



# Security goals for TLS

<https://amazon.com>



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?

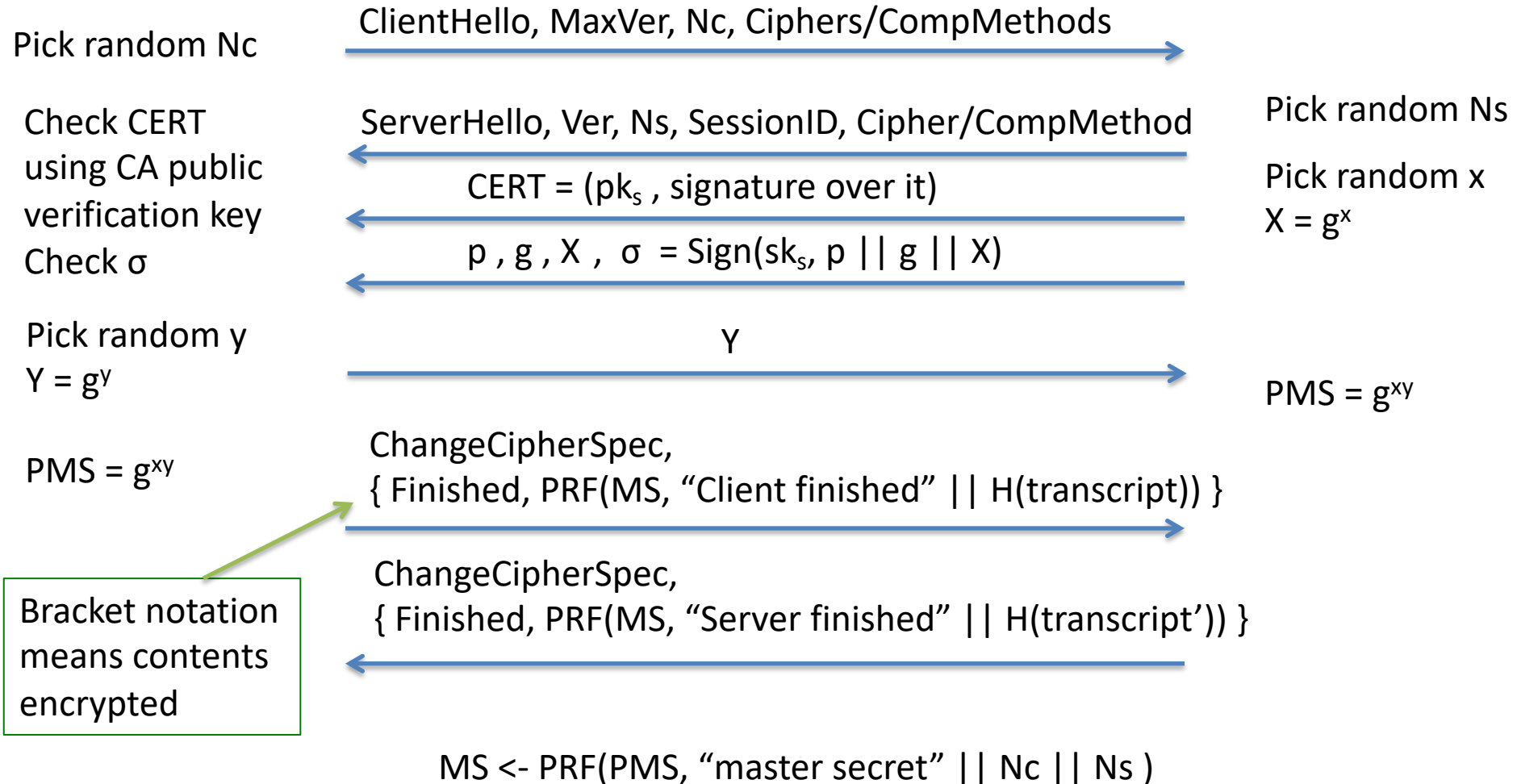


Client

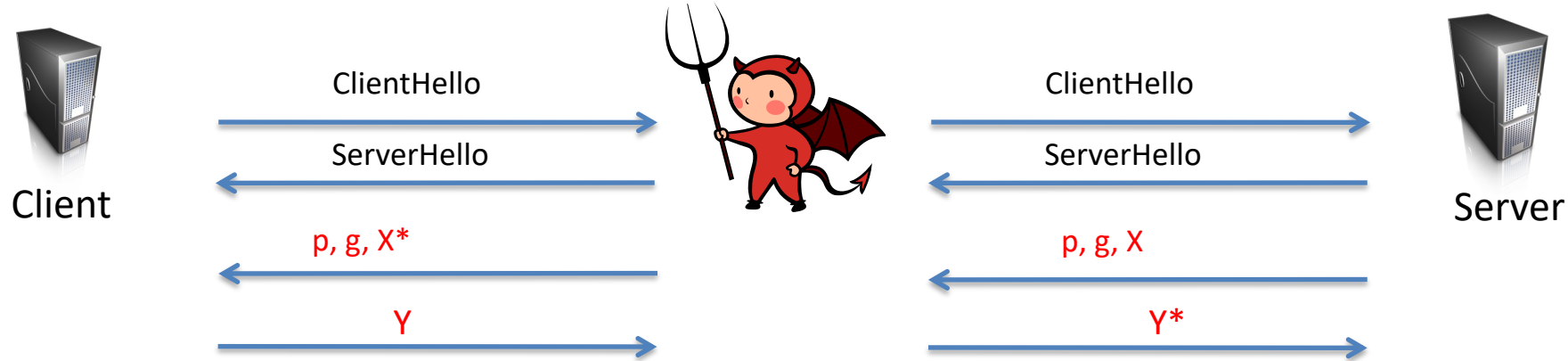
# TLS handshake for Anonymous Diffie-Hellman



Server



# 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