Asymmetric encryption

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu

Asymmetric encryption

Basic setting

The RSA algorithm

PKCS #1 encryption

Digital signing & public-key infrastructure

Hybrid encryption

TLS handshake for RSA transport

Bank customer

Bank

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods

Pick random Ns

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

CERT = (pk of bank, signature over it)

Pick random PMS
C <- E(pk,PMS)

C

PMS <- D(sk,C)

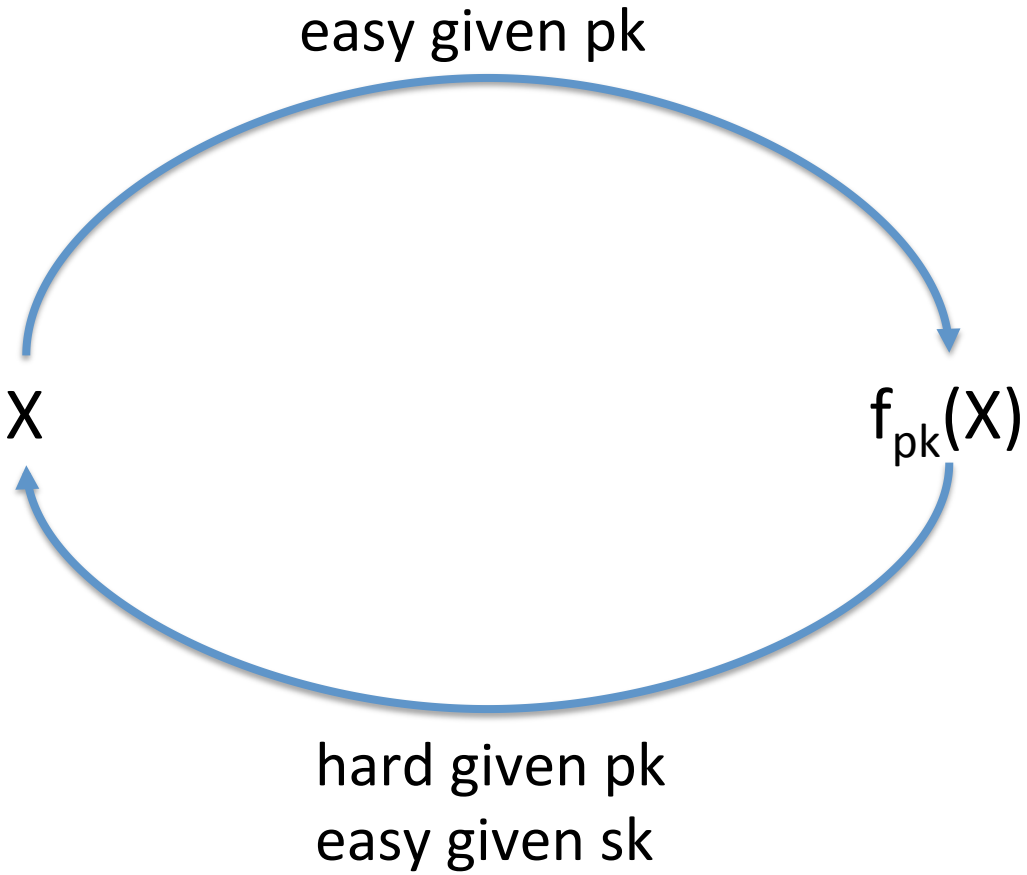
ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) }

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

Bracket notation means contents encrypted

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

Trapdoor functions help us build PKE



The RSA trapdoor function

- Rivest, Shamir, Adleman 1978
- Garnered them a Turing award

RSA math

p and q be large prime numbers
N = pq
N is called the modulus

p = 7, q = 13, gives N = 91

p = 17, q = 53, gives N = 901

RSA math

p and q be large prime numbers

N = pq

N is called the modulus

$\mathbf{Z}_N = \{0,1,2,3,\ldots, N-1\}$

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \text{ and } i < N\}$

gcd(X,Y) = 1 if greatest common divisor of X,Y is 1

RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

$N = 13 \qquad \mathbf{Z}_{13}^* = \{ 1,2,3,4,5,6,7,8,9,10,11,12 \}$

$N = 15 \qquad \mathbf{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \}$

The size of a set S is denoted by $|S|$

Def. $\phi(N) = |\mathbf{Z}_N^*|$ (This is Euler's totient function)

$\phi(13) = 12$

$\phi(15) = 8$

$\mathbf{Z}_{\phi(15)}^* = \mathbf{Z}_8^* = \{ 1,3,5,7 \}$

RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

$\mathbf{Z}_N^*$ is a group under modular multiplication

Fact. For any a,N with N > 0, there exists unique q,r such that

$$a = Nq + r \qquad \text{and} \qquad 0 \le r < N$$

$17 \bmod 15 = 2$

Def. $a \bmod N = r \in \mathbf{Z}_N$ $\qquad 105 \bmod 15 = 0$

Def. $a \equiv b \pmod N$ iff $(a \bmod N) = (b \bmod N)$

RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

$\mathbf{Z}_N^*$ is a group under modular multiplication

$$\mathbf{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \}$$

$2 \bullet 7 \equiv 14 \pmod{15}$

$4 \bullet 8 \equiv 2 \pmod{15}$

Closure: for any $a,b \in \mathbf{Z}_N^*$ $\quad a \bullet b \bmod N \in \mathbf{Z}_N^*$

Def. $a^i \bmod N = \underbrace{a \bullet a \bullet a \bullet \ldots \bullet a}_{i \text{ times}} \bmod N$

Textbook exponentiation

Let G be a group.
How do we compute h^x for any $h \in G$?

Exp(h,x)
X' = h
For i = 2 to x do
    X' = X'*h
Return X'

Requires time O(|G|) in worst case.

SqrAndMulExp(h,x)
$b_k,\ldots,b_0 = x$
f = 1
For i = k down to 0 do
    $f = f^2$
    If $b_i = 1$ then
        f = f*h
Return f

Requires time O(k) multiplies and squares in worst case.

SqrAndMulExp(h,x)

$b_k,\ldots,b_0 = x$

f = 1

For i = k down to 0 do

    $f = f^2$

    If $b_i = 1$ then

        f = f*h

Return f

$$x = \sum_{b_i \neq 0} 2^i$$

$$h^x = h^{\sum_{b_i \neq 0} 2^i} = \prod_{b_i \neq 0} h^{2^i}$$

$h^{11} = h^{8+2+1} = h^8 \bullet h^2 \bullet h$

$b_3 = 1$ $f_3 = 1 \bullet h$

$b_2 = 0$ $f_2 = h^2$

$b_1 = 1$ $f_1 = (h^2)^2 \bullet h$

$b_1 = 1$ $f_0 = (h^4 \bullet h)^2 \bullet h = h^8 \bullet h^2 \bullet h$

Don't implement this algorithm: side-channel attacks

RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

Claim: Suppose $e,d \in \mathbf{Z}^*_{\phi(N)}$ satisfying $ed \bmod \phi(N) = 1$
then for any $x \in \mathbf{Z}_N^*$ we have that

$$(x^e)^d \bmod N = x$$

$$(x^e)^d \bmod N = x^{(ed \bmod \phi(N))} \bmod N$$
$$= x^1 \bmod N$$
$$= x \bmod N$$

First equality is
by Euler's Theorem

RSA math

$\mathbf{Z}_N^* = \{ i \mid \gcd(i,N) = 1 \}$

Claim: Suppose $e,d \in \mathbf{Z}_{\phi(N)}^*$ satisfying $ed \bmod \phi(N) = 1$ then for any $x \in \mathbf{Z}_N^*$ we have that

$$(x^e)^d \bmod N = x$$

$\mathbf{Z}_{15}^* = \{ 1,2,4,7,8,11,13,14 \}$ $\mathbf{Z}_{\phi(15)}^* = \{ 1,3,5,7 \}$

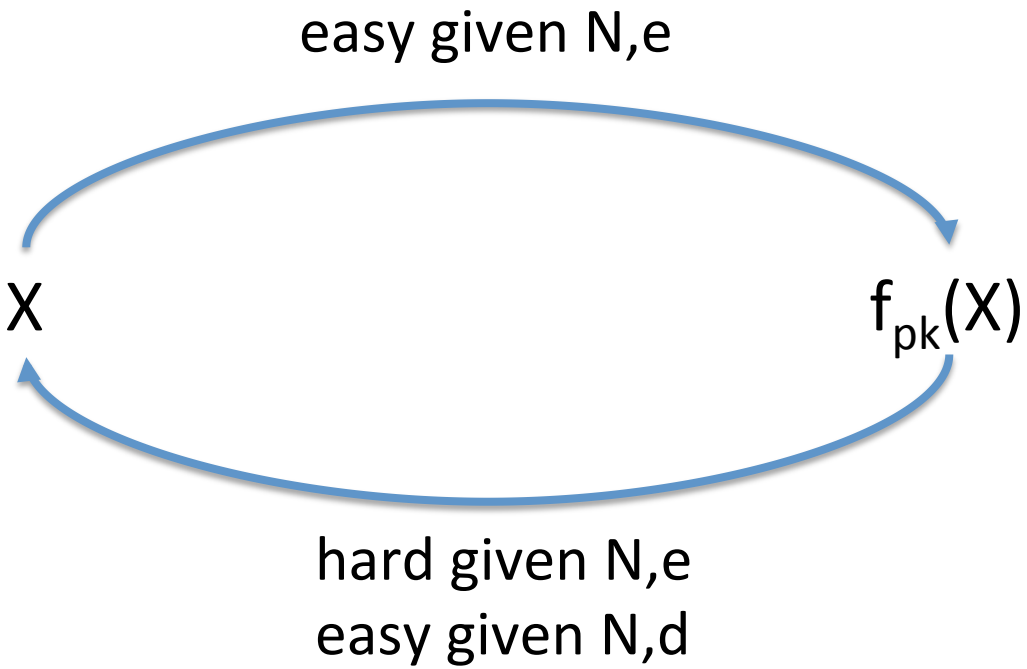
$e = 3$, $d = 3$ gives $ed \bmod 8 = 1$

x	1	2	4	7	8	11	13	14
$x^3 \bmod 15$	1	8	4	13	2	11	7	14
$y^3 \bmod 15$	1	2	4	7	8	11	13	14

The RSA trapdoor permutation

pk = (N,e) sk = (N,d) with ed mod $\phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$ $g_{N,d}(y) = y^d \bmod N$



The RSA trapdoor permutation

pk = (N,e) sk = (N,d) with ed mod $\phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$ $g_{N,d}(y) = y^d \bmod N$

But how do we find suitable N,e,d ?

If p,q distinct primes and N = pq then $\phi(N) = (p-1)(q-1)$

Why?

$$\begin{aligned}\phi(N) &= |\{1,\ldots,N-1\}| - |\{ip : 1 \le i \le q-1\}| - |\{iq : 1 \le i \le p-1\}| \\ &= N-1 - (q-1) - (p-1) \\ &= pq - p - q + 1 \\ &= (p-1)(q-1)\end{aligned}$$

The RSA trapdoor permutation

pk = (N,e) sk = (N,d) with ed mod $\phi(N) = 1$

$f_{N,e}(x) = x^e \bmod N$ $g_{N,d}(y) = y^d \bmod N$

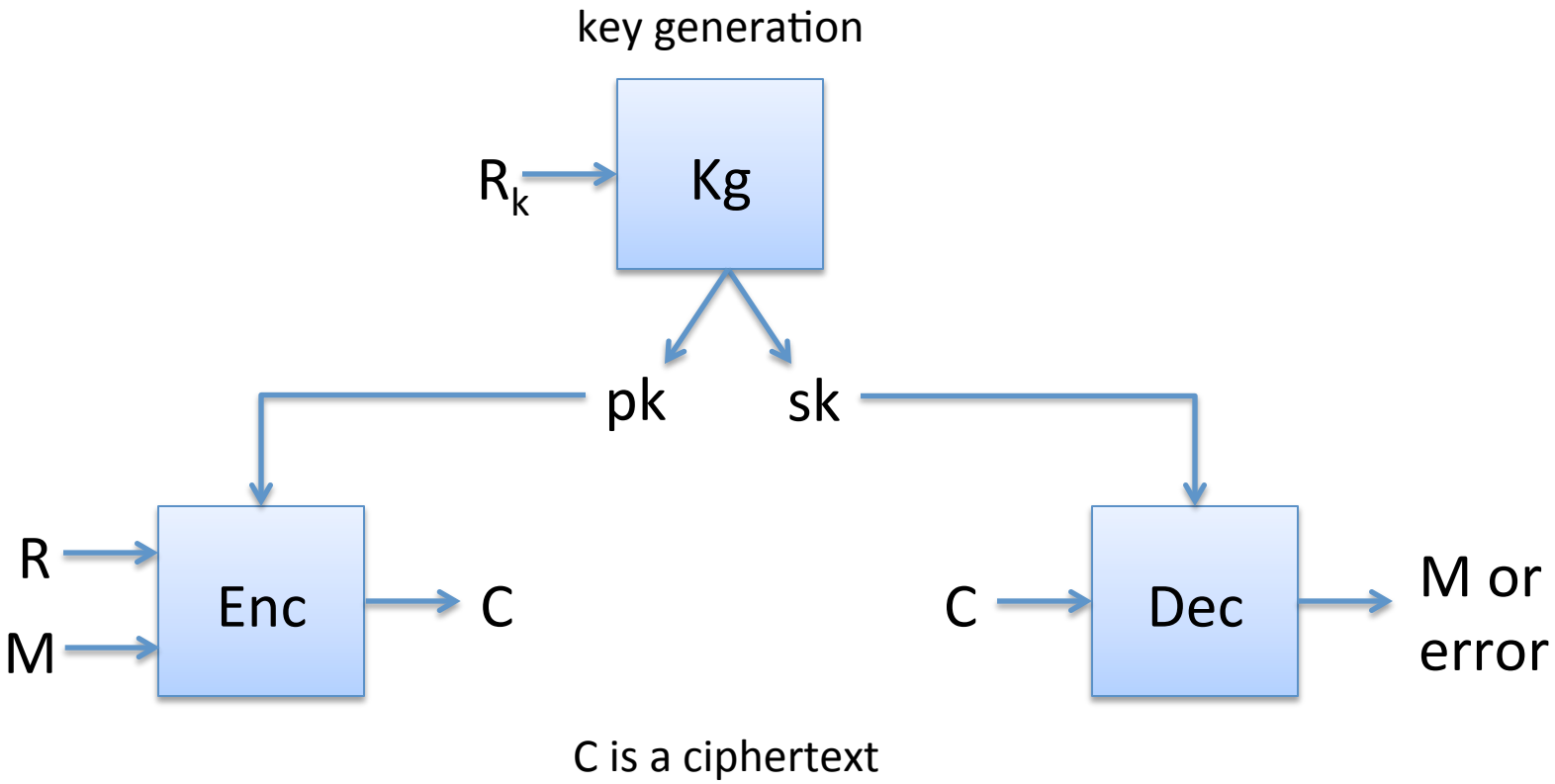
But how do we find suitable N,e,d ?

If p,q distinct primes and N = pq then $\phi(N) = (p-1)(q-1)$

Given $\phi(N)$, choose $e \in \mathbf{Z}^*_{\phi(N)}$ and calculate

$d = e^{-1} \bmod \phi(N)$

Public-key encryption



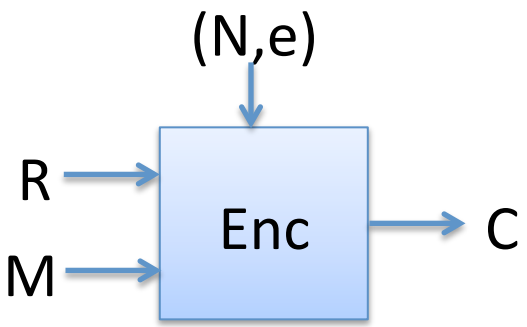
Correctness: D(sk , E(pk,M,R)) = M with probability 1 over randomness used

PKCS #1 RSA encryption

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

Let $B = \{0,1\}^8 / \{00\}$ be set of all bytes except 00

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

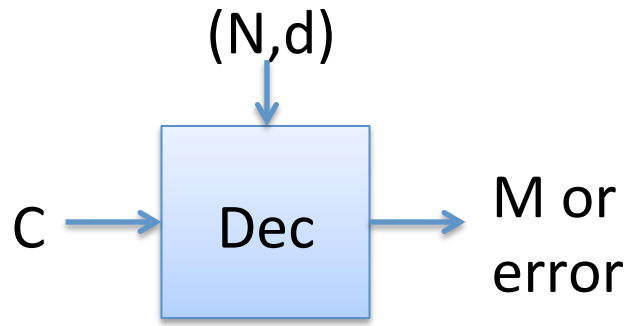


Enc((N,e), M, R)

pad = first n - m - 3 bytes from R that are in B

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

Return X^e mod N



Dec((N,d), C)

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

If (aa ≠ 00) or (bb ≠ 02) or (00 ∉ w)

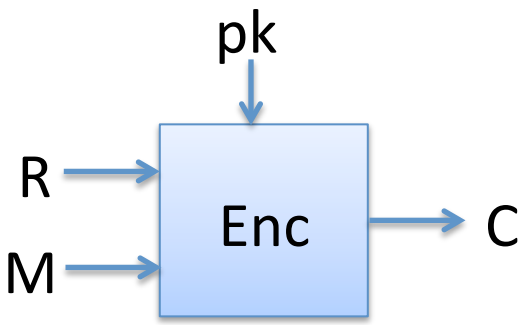
Return error

pad || 00 || M = w

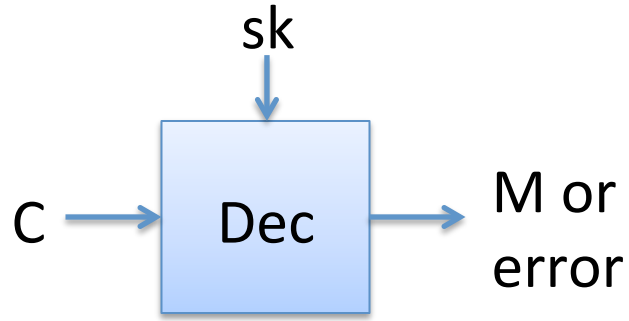
Return M

Hybrid encryption

Kg outputs (pk,sk)



Enc(pk, M, R)
K||R1||R2 = R
C1 = Enc(pk,K,R1)
C2 = Enc(K,M,R2)
Return (C1,C2)



Dec(sk, (C1,C2))
K = Dec(sk,C1)
M = Dec(K,C2)
Return M

Bank customer

TLS handshake for RSA transport

Bank

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns

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

Check CERT using CA public verification key

← CERT = (pk of bank, signature over it)

Pick random PMS
C <- E(pk,PMS)

C →

PMS <- D(sk,C)

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

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

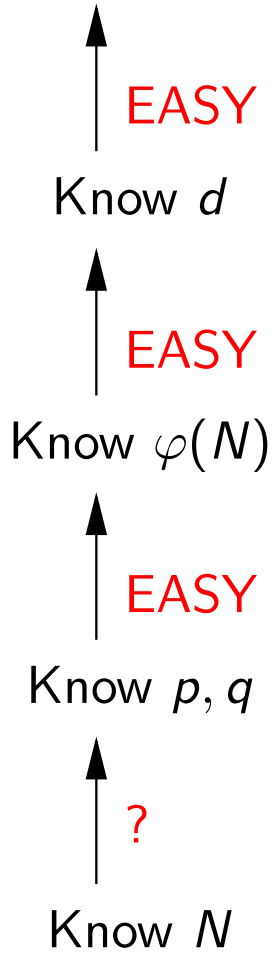
Bracket notation means contents encrypted

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

Security of RSA PKCS#1

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

Inverting RSA : given N, e, y find x such that $x^e \equiv y \pmod{N}$



because $f^{-1}(y) = y^d \bmod N$

because $d = e^{-1} \bmod \varphi(N)$

because $\varphi(N) = (p-1)(q-1)$

Learning p,q from N is the factoring problem

We don't know if inverse is true, whether inverting RSA implies ability to factor

Factoring composites

- What is p,q for N = 901?

Factor(N):
for i = 2 , … , sqrt(N) do
    if N mod i = 0 then
        p = i
        q = N / p
        Return (p,q)

Woops… we can always factor

But not always efficiently:
Run time is sqrt(N)

$O(\text{sqrt}(N)) = O(e^{0.5 \ln(N)})$

Factoring composites

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)	$O(e^{c})$ $c = 1.92\ (\ln N)^{1/3}\ (\ln \ln N)^{2/3}$

Factoring records

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

Security of RSA PKCS#1

- Passive adversary sees (N,e),C
- Attacker would like to invert C
- Possible attacks?
  - Pick |N| > 1024 and factoring will fail
  - Active attacks?

Bleichanbacher attack

I've just learned some information about $C_1^d \bmod N$

C_1 →

← padding error?

C_2 →

← padding error?

…

Dec((N,d), C)

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

If (aa ≠ 00) or (bb ≠ 02) or (00 ∉ w)

Return error

pad || 00 || M = w

Return M

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

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

Response to this attack

- Ad-hoc fix: 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

Diffie-Hellman math

Let p be a large prime number
Fix the group G = $\mathbf{Z}_p^*$ = {1,2,3,…, p-1}

Then G is *cyclic*. This means one can give a member g $\in$ G , called the generator, such that

$$G = \{ g^0, g^1, g^2, \ldots, g^{p-1} \}$$

Example: p = 7. Is 2 or 3 a generator for $\mathbf{Z}_7^*$?

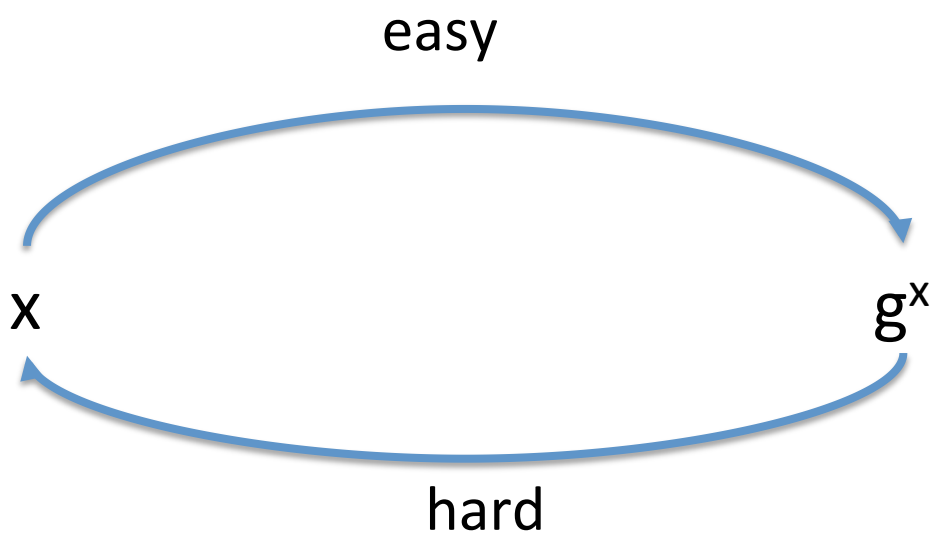
x	0	1	2	3	4	5	6
2^x mod 7	1	2	4	1	2	4	1
3^x mod 7	1	3	2	6	4	5	1

The discrete log problem

Fix a cyclic group G with generator g

Pick x at random from $\mathbf{Z}_{|G|}$

Give adversary g, $X = g^x$. Adversary's goal is to compute x



The discrete log problem

Fix a cyclic group G with generator g

Pick x at random from $\mathbf{Z}_{|G|}$

Give adversary g, $X = g^x$. Adversary's goal is to compute x

$\mathcal{A}(X)$:
for i = 2 , … , |G|-1 do
    if $X = g^i$ then
        Return i

Very slow for large groups!

$O(|G|)$

Baby-step giant-step is better:

$O(|G|^{0.5})$

Nothing faster is known for some groups.

Diffie-Hellman Key Exchange

X

Y

Pick random x from $\mathbf{Z}_{|G|}$
$X = g^x$

Pick random y from $\mathbf{Z}_{|G|}$
$Y = g^y$

$K = H(Y^x)$

$K = H(X^y)$

Get the same key. Why? $Y^x = g^{yx} = g^{xy} = X^y$

What type of security does this protocol provide?

Computational Diffie-Hellman Problem

Fix a cyclic group G with generator g

Pick x,y both at random $\mathbf{Z}_{|G|}$

Give adversary g, $X = g^x$, $Y = g^y$.
Adversary must compute g^{xy}

For most groups, best known algorithm finds discrete log of X or Y.
But we have no proof that this is best approach.

TLS handshake for Diffie-Hellman Key Exchange

Client

Server

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Check CERT using CA public verification key
Check σ

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

Pick random Ns

← CERT = (pk_s , signature over it)

Pick random x
$X = g^x$

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

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)

Security of RSA PKCS#1

- Passive adversary sees (N,e),C
- Attacker would like to invert C
- Possible attacks?
  - Pick |N| > 1024 and factoring will fail
  - Active attacks?
    - Some implementations seem ok
  - Man-in-the-middle: replace (N,e) with our own key

TLS handshake for RSA transport

Bank customer

Bank

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods

Pick random Ns

ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

CERT = (pk of bank, signature over it)

C

Pick random PMS
C <- E(pk,PMS)

PMS <- D(sk,C)

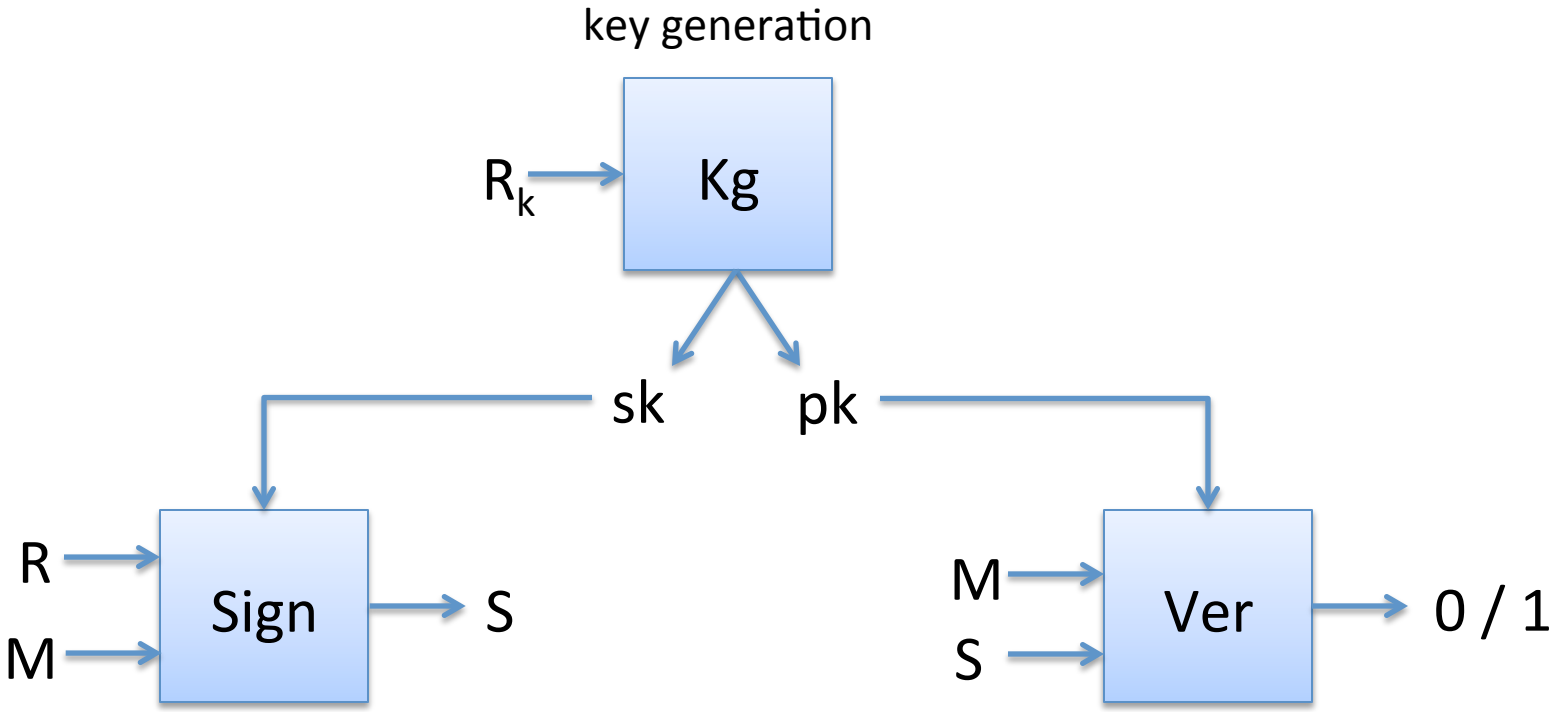
ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) }

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

Bracket notation means contents encrypted

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

Digital signatures

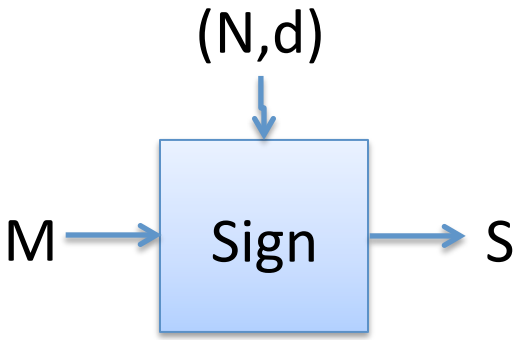


Anyone with public key can verify a signature
Only holder of secret key should be able to generate a signature

Full Domain Hash RSA

Kg outputs pk = (N,e) , sk = (N,d)

H is a hash function

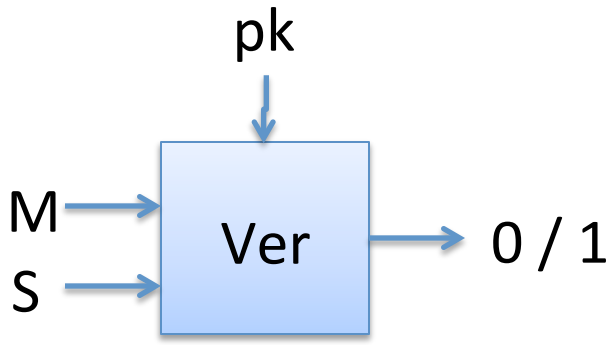


Sign((N,d), M)

X = 00 || H(1||M) || … || H(k||M)

$S = X^d \bmod N$

Return S



Ver((N,e), M, S)

$X = S^e \bmod N$

X' = 00 || H(1||M) || … || H(k||M)

If X = X' then

    Return 1

Return 0

Certificate Authorities and Public-key Infrastructure

(pk,sk)

Give me a certificate for pk', please

pk

M = (pk',data)
S = Sign(sk,M)

S

http://amazon.com

pk', data, S

M = (pk',data)
If Ver(pk,M,S) then
trust pk'

(pk',sk')

This prevents man-in-the-middle (MitM) attacks