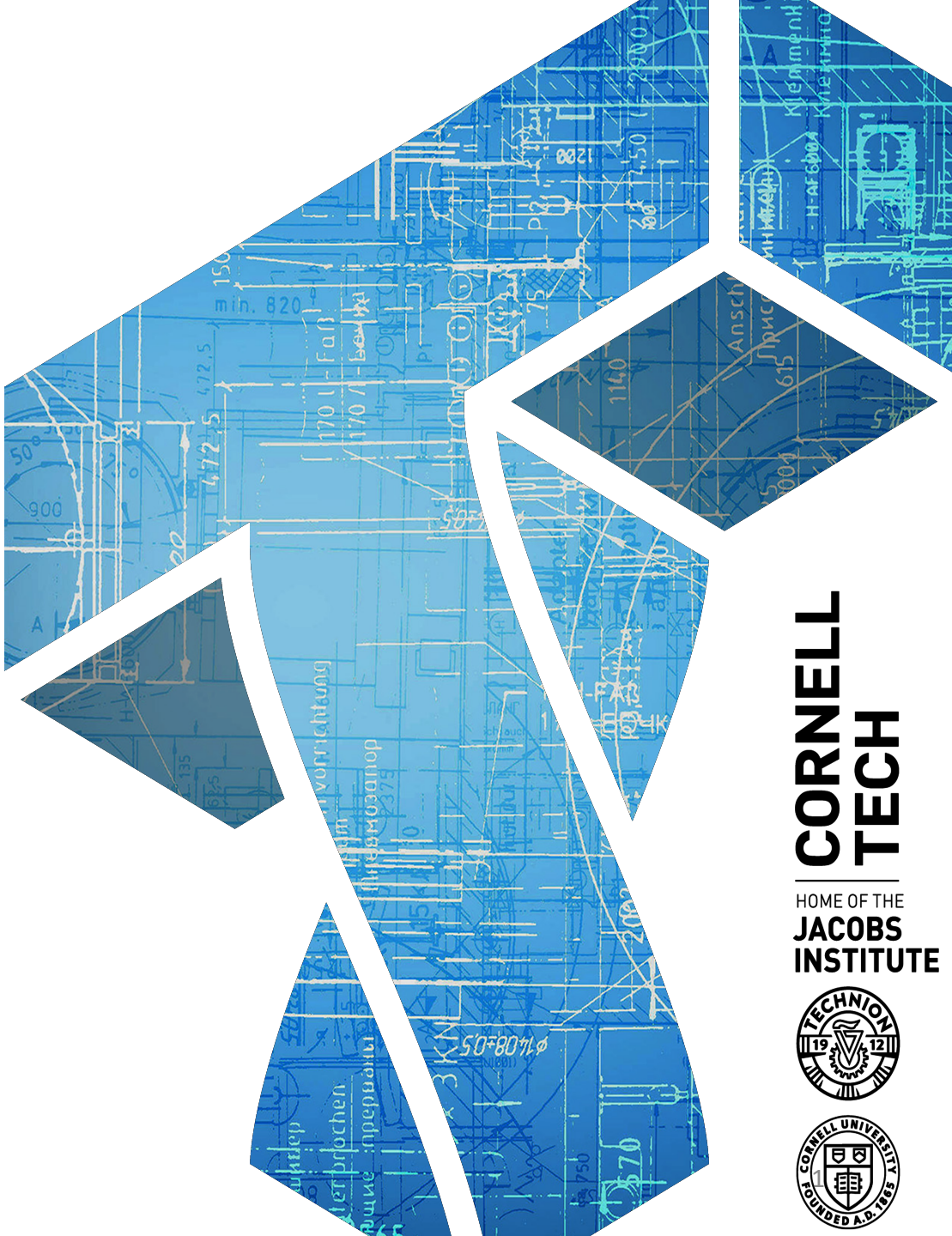
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda

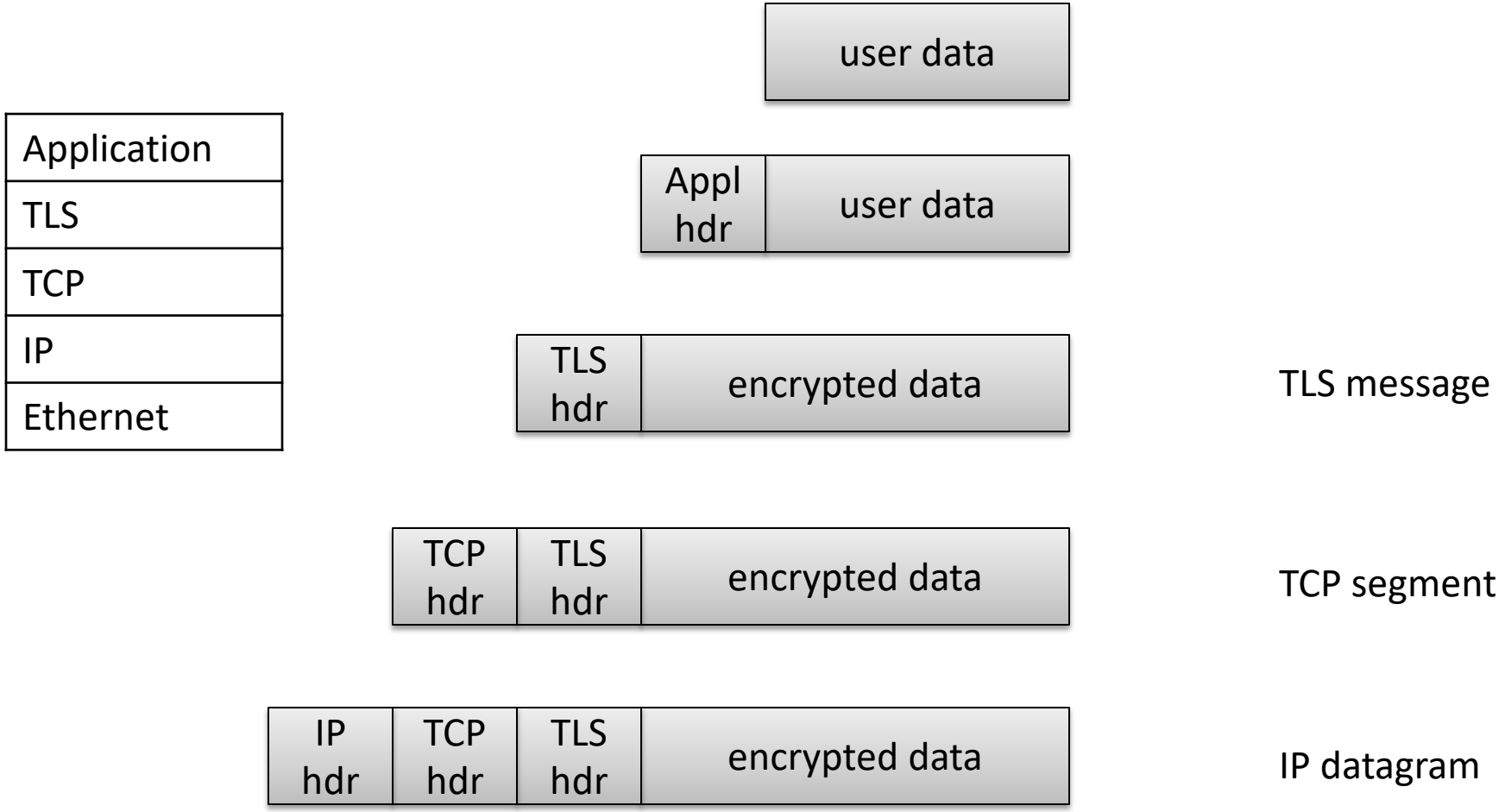


CORNELL TECH
HOME OF THE JACOBS INSTITUTE

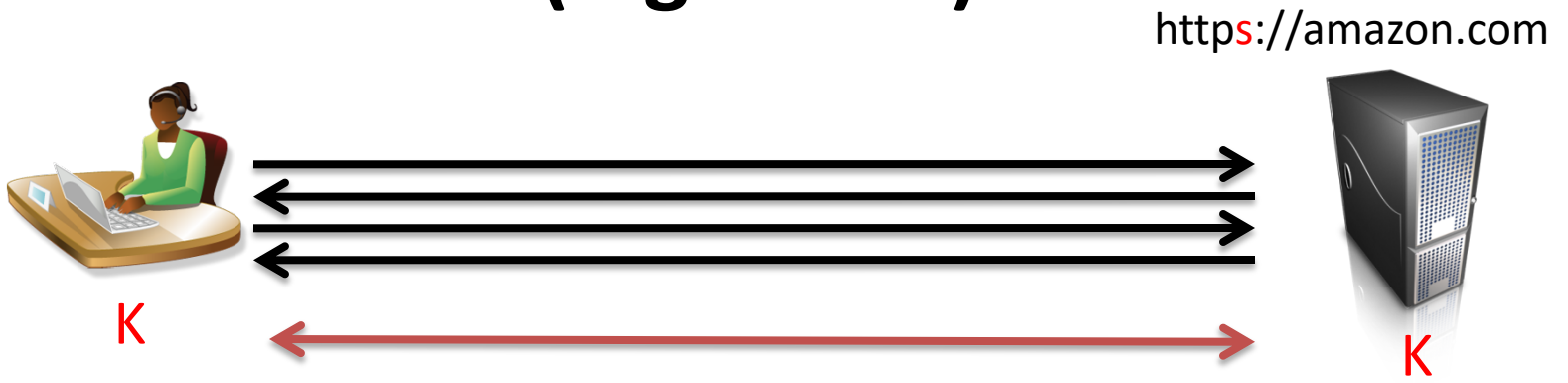
Background on TLS

- Secure Sockets Layer (SSL) designed at Mozilla in 1990s
  - SSL 1.0, 2.0, 3.0
  - Provide point-to-point encryption solution that can be used to protect sensitive data on the web (and elsewhere)
- Renamed to Transport Layer Security (TLS) in late 1990s
  - TLS 1.0, 1.1, 1.2, 1.3
  - 1.2 and before: multiple key exchange mechanisms including RSA key transport, Diffie-Hellman key exchange
  - 1.3: DH-only plus lower latency handshakes

TLS sits between application layer (HTTP) and TCP



How does TLS work (high level)?



Step 1:
Key exchange protocol to share secret K

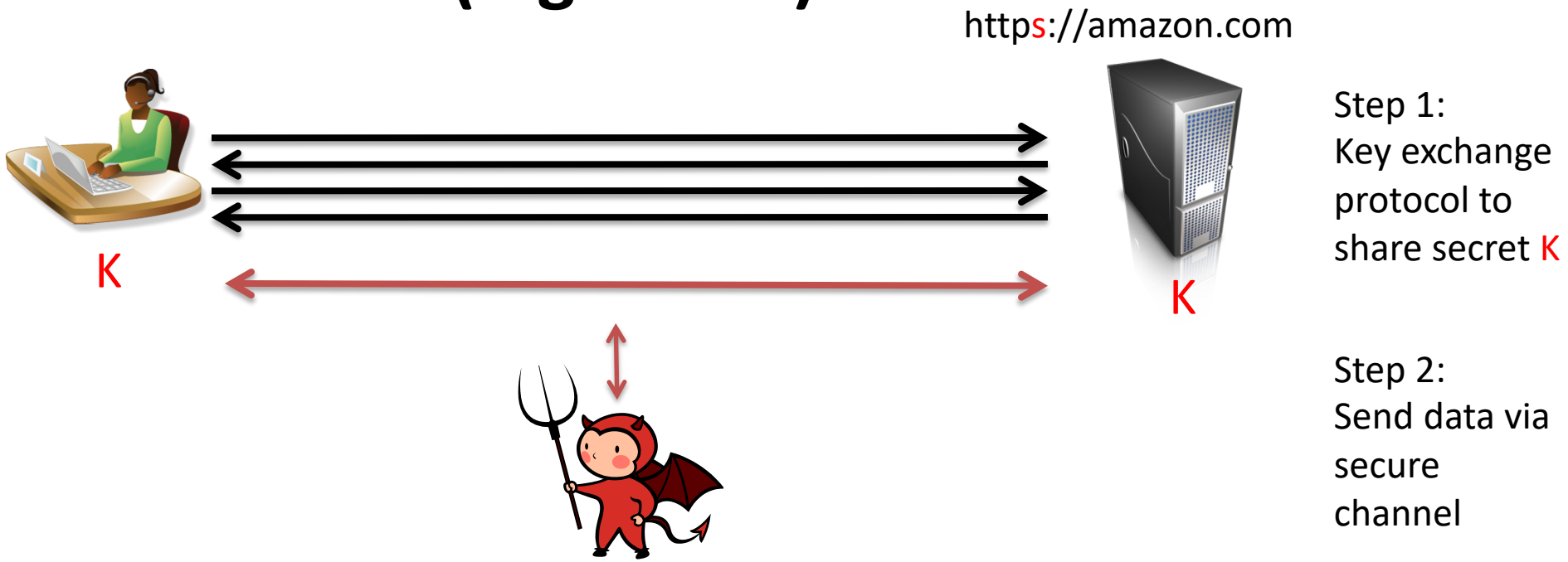
Step 2:
Send data via secure channel

The secure channel is implemented via our now familiar symmetric encryption primitives

Goals of handshake:

- Negotiate version
- Negotiate parameters (crypto algorithms to use)
- Authenticate server (Is server actually Amazon.com?)
  - Digital signatures and certificates
- Establish shared secret
  - Asymmetric cryptography

How does TLS work (high level)?



Security goals:

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

TLS 1.2 handshake for RSA transport

Client

Server

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_s , 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(PMS, “master secret” || Nc || Ns)

TLS 1.2 key derivation & use

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

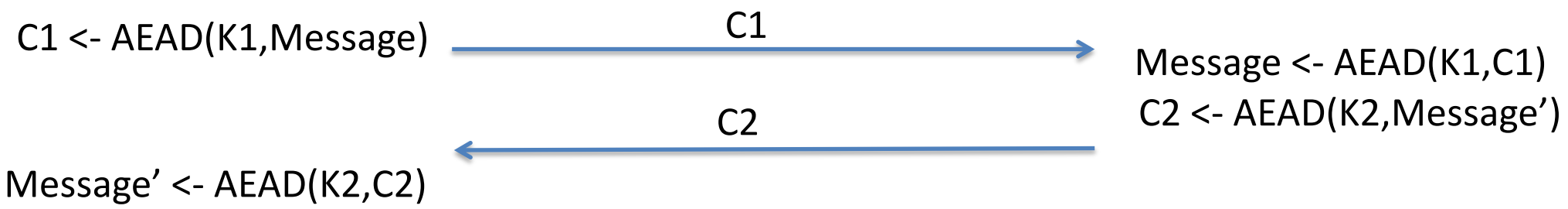
K1,K2 <- PRF(MS, “key expansion” || Ns || Nc)

concatenation

PRF(secret, seed) = HMAC-HASH(secret, A(1) + seed) +
HMAC-HASH(secret, A(2) + seed) +
HMAC-HASH(secret, A(3) + seed) + ...

Replaced with HKDF in TLS 1.3

Where A(0) = seed and A(i) = HMAC_hash(secret, A(i-1))



Record layer details

- Fragmentation
  - Maximum TLS ciphertext handles 2^{14} bytes of message data
  - Split longer message into multiple fragments
  - Encrypt each fragment separately
- Sequence numbers keep track of count of fragments sent in each direction
- Compression methods
- AEAD used in TLS 1.2 and before vulnerable to padding oracle attacks, or RC4-abuse attacks
  - HMAC-then-CBC
  - HMAC-then-RC4

TLS record layer attacks

Attack	Year	Vulnerability	Countermeasure
Vaudenay	2002	Padding oracle (theoretical)	---
Rogaway	2002	IV chaining (theoretical)	---
Kelsey	2002	Compression before encryption (theoretical)	---
Canval et al.	2003	Padding oracle via timing	Always compute HMAC
BEAST (Duong & Rizzo)	2011	IV chaining	Dedicated IVs
CRIME (Duong & Rizzo)	2012	TLS compression before encryption	Turn off TLS compression
Lucky13 (AlFardan & Paterson)	2013	Padding oracle via HMAC timing	Constant-time decryption attempted; move to RC4
RC4 attack (AlFardan et al.)	2013	RC4 cryptanalysis made practical	Move to CBC-based cipher suites
BREACH (Prado et al.)	2013	HTTP compression before encryption	Turn off HTTP compression (if possible)

TLS handshake for RSA transport

Client

Server

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_s , signature over it)

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

C →

PMS <- Dec(sk,C)

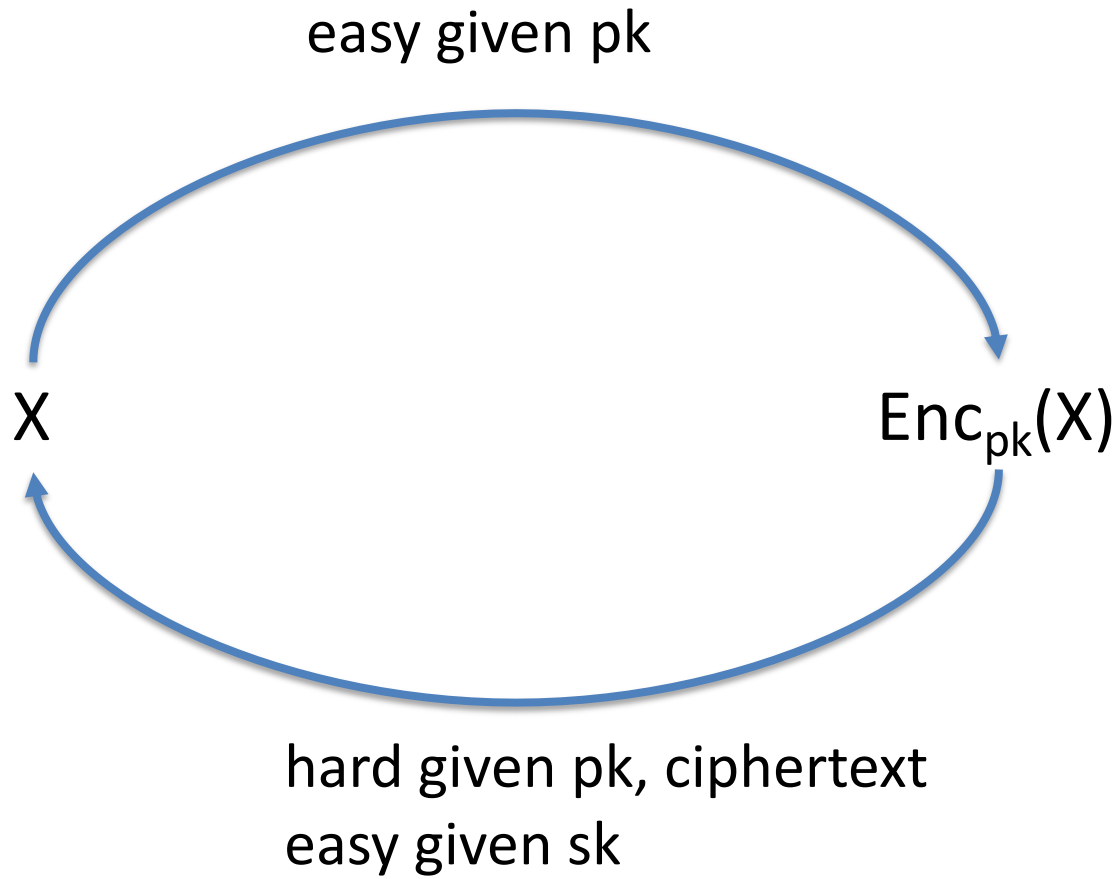
ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) } →

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

Bracket notation means contents encrypted

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

Security against passive adversaries rely on inability to invert asymmetric encryption



The RSA trapdoor function

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

RSA math

Let N be a positive number

Looking ahead: N = pq for large primes p,q

N will be called the modulus

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

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

RSA math

Let N be a positive number

Looking ahead: N = pq for large primes p,q

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

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

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

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

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

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

$$17 \bmod 15 = 2 \qquad 105 \bmod 15 = 0$$

Def. $a \bmod N = r \in \mathbf{Z}_N$

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

Operations work in natural way:

$$a \bullet b \bmod N \qquad a+b \bmod N$$

RSA math

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

$(\mathbf{Z}_N^* , \bullet)$ is a **group** where $\bullet$ denotes multiplication mod N

Group $(G,\bullet)$ is a set G and operator $\bullet$ that satisfy:

1. *Closure*: for all $a,b \in G$ it holds that $a \bullet b \in G$
2. *Associativity*: for all $a,b,c \in G$ it holds that $a \bullet (b \bullet c) = (a \bullet b) \bullet c$
3. *Identity*: Exists $I \in G$ s.t. for all $a \in G$ $a \bullet I = a$
4. *Inverses*: for $a \in G$ there exists $a^{-1} \in G$ s.t. $a \bullet a^{-1} = I$

Abelian group is additionally commutative:

for all $a,b \in G$ it holds that $a \bullet b = b \bullet a$

RSA math

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

$(\mathbf{Z}_N^* , \bullet)$ is a **group**

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

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

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

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

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

Some needed algorithms

Algorithm	**Running time (n = log N)**
Modular multiplication ab mod N	$O(n^2)$
Modular exponentation a^i mod N	$O(n^3)$
Modular inverse a^{-1} mod N	$O(n^2)$

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 mod N
Return X'
```

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

```
SqrAndMulExp(h,x)
b_k,…,b_0 = x
f = 1
For i = k down to 0 do
            f = f^2 mod N
            If b_i = 1 then
                        f = f•h mod N
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 mod N

    If b_i = 1 then

        f = f•h mod N

Return f

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

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

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

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

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

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

$b_0 = 1$ $f_0 = (h^4 \bullet h)^2 \bullet h = h^8 \bullet h^2 \bullet h$

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

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

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

First equality is
by Euler's Theorem:
$x^{\phi(N)} \bmod N = x \bmod N$

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

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

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

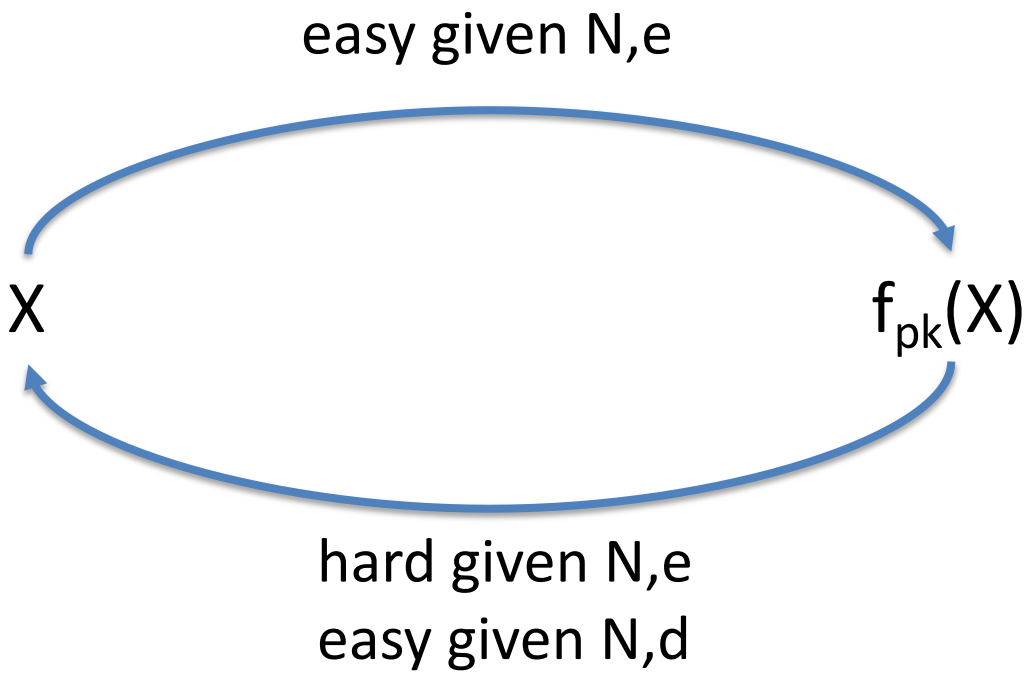
$e = 3$, $d = 3$ gives $ed \bmod 8 = 1$

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?

$\phi(N) = |\{1,\ldots,N-1\}| - |\{ip : 1 \le i \le q-1\}| - |\{iq : 1 \le i \le p-1\}|$

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

$= pq - p - q + 1$

$= (p-1)(q-1)$

The RSA trapdoor permutation

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

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

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

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

How to find suitable p,q prime?

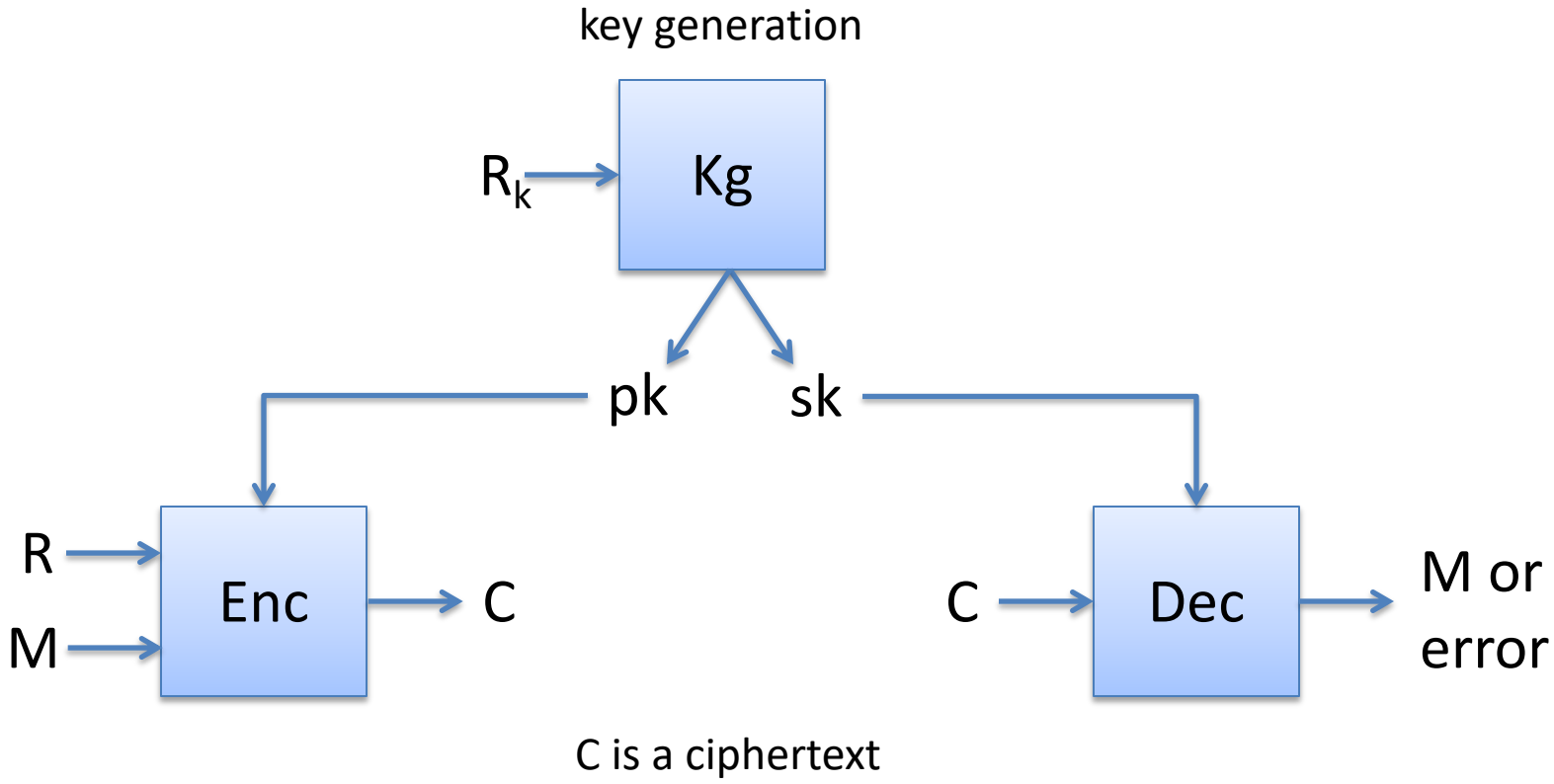
Choose random numbers and test primality (Miller-Rabin testing)

https://eprint.iacr.org/2018/749.pdf

Summary

- 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

Public-key encryption



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

Textbook RSA

- Why not just use $M^e \bmod N$ directly to encrypt?
- Lots of reasons:
  - It is deterministic!
  - 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 (we'll return to chosen-ciphertext attacks later on)
  - Small e attacks.
    - e = 3 and M smallish, $M^3 < N$, solve by taking 3rd root of M^3

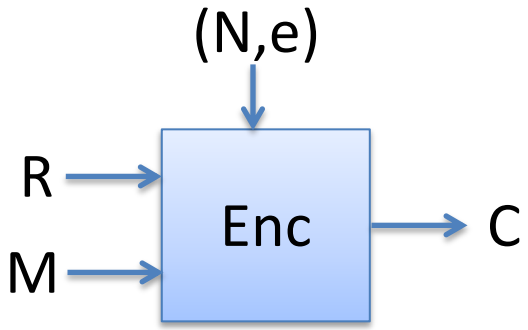
https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf

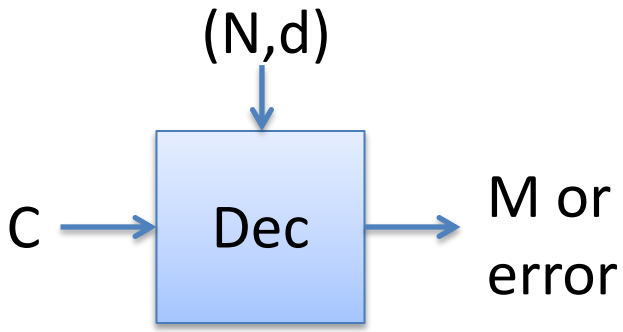
PKCS #1 RSA encryption

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)

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

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

  Return error

pad || 00 || M = w

Return M

TLS handshake for RSA transport

Client

Server

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_s , signature over it)

C →

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

PMS <- Dec(sk,C)

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

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

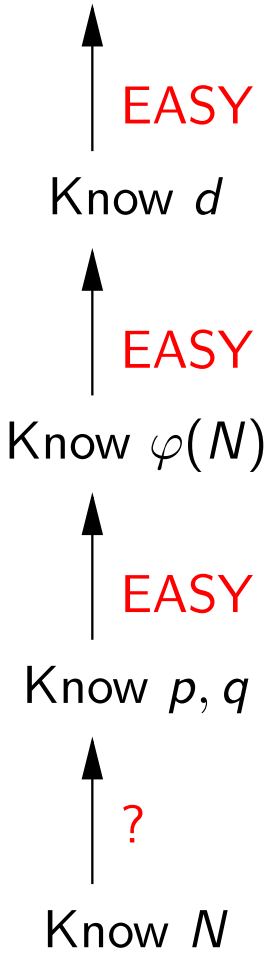
Bracket notation means contents encrypted

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

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

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-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

C_1

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

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

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
- RSA PKCS#1 v1.5 is insecure due to padding oracle attacks. Don’t use it in new systems.
  - Use OAEP instead