Gems of TCS

Public Key Cryptography II

Sasha Golovnev

April 15, 2021

RSA

Modular Arithmetic

$0 \le x, y, N < 2^n$ – n-bit long int

Easy Problems – $O(n^3)$

- Addition, Subtraction, Multiplication
- GCD – Greatest Common Divisor
- Modular Inverse
- Modular Exponentiation $x, y, N \rightarrow x^y \bmod N$
- Primality Test N is prime?

Modular Arithmetic

Easy Problems

- Addition, Subtraction, Multiplication
- GCD
- Modular Inverse
- Modular Exponentiation
- Primality Test

Hard Problems

$N = p \cdot q$

- Factorization
- eth root: $x^{1/e}$

$x, e \longrightarrow y = x^{1/e} \mod N$

$y^e = x \mod N$

Euler's Function

$$\forall N \in \mathbb{N},$$

x is inv. $\exists$ y s.t. $x \cdot y = 1$ (in Z_N)

$$\phi(N) = \#\text{ of invertible els in } Z_N$$
$$= |\{x \colon GCD(x, N) = 1\}|\,.$$

$N = p$ - prime $\quad \phi(N) = N - 1$

$N = p \cdot q$, p & q - primes $\quad \phi(N) = N - p - q + 1 =$

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

Euler's Theorem

Euler's Function

$$\forall N \in \mathbb{N},$$
$$\phi(N) = \#\text{ of invertible els in } Z_N$$
$$= |\{x\colon GCD(x, N) = 1\}|\,.$$

Euler's Theorem

$$\forall N \in \mathbb{N}, \forall x \in Z_N^*,$$
$$\boxed{x^{\phi(N)} = 1 \text{ in } Z_N\,.}$$

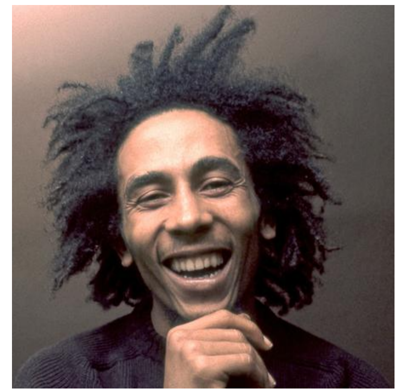
If N is prime, $\Rightarrow$ Fermat's thm $x^{p-1} = 1$ in $\mathbb{Z}_p$

Public Key Cryptography

$\boxed{pk}$ can be for encryption

posts

$c = Enc(pk, m)$



Alice, sk

used for decryption

$m = Dec(sk, c)$

Bob

RSA Cryptosystem

Alice generates

- $N = pq$ p, q are primes, 1024-bit long

RSA Cryptosystem

Alice generates

- $N = pq$
- $e \cdot d = 1 \mod \phi(N) = (p-1)(q-1)$

For example, for random e, she can compute
d s.t. $ed = 1 \mod \phi(n)$, because modular
inv. is EASY

RSA Cryptosystem

Alice generates

- $N = pq$
- $e \cdot d = 1 \mod \phi(N)$
- $\mathrm{pk} = (N, e)$ – can be used for encryption

RSA Cryptosystem

Alice generates

- $N = pq$
- $e \cdot d = 1 \mod \phi(N)$
- $\mathsf{pk} = (N, e)$
- $\mathsf{sk} = (N, d)$ — can be used for decryption

RSA Cryptosystem

Alice generates

- $N = pq$
- $e \cdot d = 1 \mod \phi(N)$
- $\text{pk} = (N, e)$
- $\text{sk} = (N, d)$

Encryption/Decryption

Mod exp is EASY

For a message $m \in \mathbb{Z}_N^*$:

$$\boxed{c} = \text{Enc}(pk, m) = \text{Enc}(N, e, m) = m^e \text{ in } Z_N^* .$$

RSA Cryptosystem

Alice generates

- $N = pq$
- $e \cdot d = 1 \mod \phi(N)$
- $\mathsf{pk} = (N, e)$
- $\mathsf{sk} = (N, d)$

Encryption/Decryption

For a message $m \in \mathbb{Z}_N^*$:

$$c = \mathsf{Enc}(pk, m) = \mathsf{Enc}(N, e, m) = m^e \text{ in } Z_N^* .$$

For a ciphertext $c \in \mathbb{Z}_N^*$:

$$m = \mathsf{Dec}(\underline{sk}, \underline{c}) = \mathsf{Dec}(\underline{N}, \underline{d}, c) = \boxed{c^d} \text{ in } Z_N^* .$$

FAST, CORRECT, SECURE

Modular exp is EASY $\Rightarrow$ RSA is FAST

CORRECT:

$$c = m^e$$

Alice computes $c^d = (m^e)^d = m^{ed} =$ [$ed = 1 \bmod \varphi(n) \Rightarrow ed = k \cdot \varphi(n) + 1$]

$$= m^{k \cdot \varphi(n) + 1} = \left(m^{\varphi(n)}\right)^k \cdot m =$$ [$m^{\varphi(n)} = 1 \bmod N$]

$$= 1^k \cdot m = m \quad \text{in } \mathbb{Z}_N$$

SECURE: Eve sees all communication between Alice & Bob, can she decrypt c?

$c = m^e \longrightarrow m$ — need to compute e-th root.

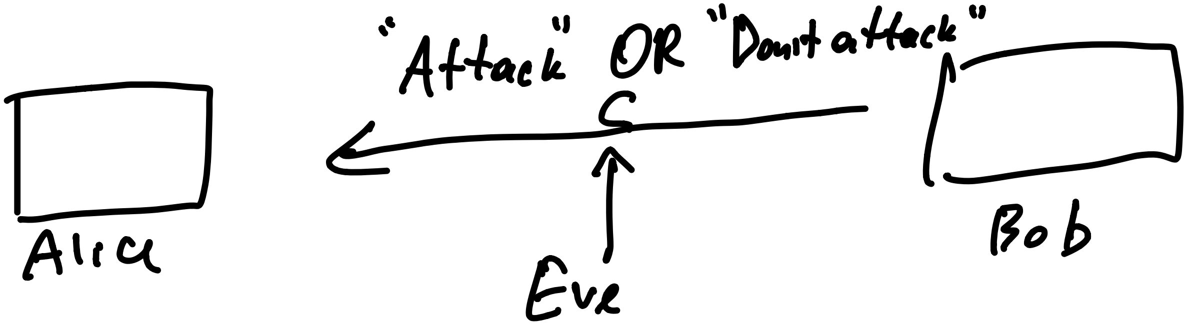
HARD

Ubiquitous RSA

- Online banking
- SSL/TLS
- Emails
- Secure file systems
- ...

Attacks on
(bad implementations of)
RSA

Textbook RSA is not Secure



Eve knows $pk = (N, e)$

Eve can compute $m =$ "Attack"
$c_1 = m^e \bmod N$

Eve can compute $m =$ "Don't attack"
$c_2 = m^e \bmod N$

Compares Bob's ciphertext c with c_1 & c_2

Solution: Enc must be randomized

$$m' = \big[\ \underbrace{\text{128 bits long}}_{m}\ \big|\ \text{256 random bits}\ \big]$$

$c = (m')^e$

Send this c to Alice.

Alice decodes, and sees m.

Enc must be randomized = same m should be mapped to many different ciphertexts.

$H(m)$ For hash function, one shouldn't be able to find collision:

m_1, m_2 $H(m_1) = H(m_2)$

FACTORING AND RSA

Factor $N = p \cdot q$

Assume we have poly-time Factoring alg.

This will break RSA

Alice $\longleftarrow$ Bob

N, d $\qquad$ N, e

Eve

N, e

Eve can factor $\boxed{N} = p \cdot q$.

Eve can compute $(p-1)\cdot(q-1) = \boxed{\varphi(N)}$

Given e and $\varphi(N) \Rightarrow$ compute d s.t. $ed = 1 \mod \varphi(N)$

Eve now has $sk = (N, e)$

Shor's alg:

Quantum alg that solves factoring in poly time.

We know quantum-secure crypto systems, we're (slowly) switching to those systems.

RSA with Prime Modulus

Why RSA uses $N = p \cdot q$, not $N = p$ - prime?

If N was prime, $\varphi(N) = N - 1$

Eve could compute $\varphi(N)$

She knows e, she can compute $d \cdot e \equiv 1 \mod \varphi(n)$

She learns $sk = (N, d)$

Small Difference

$N = p \cdot q$, p & q are 1024-bit long primes,
$|p-q| < 10^6$

Recall factoring N is sufficient RSA.
If Eve knows that $N = p \cdot q$ $|p-q| < 10^6$?

Wlog $p \leq \sqrt{N} \leq q$ $q - p \leq 10^6 \Rightarrow$

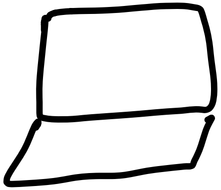
$\sqrt{N} - 10^6 \leq p \leq \sqrt{N}$

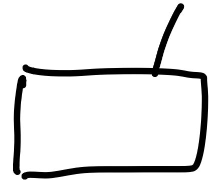
Eve brute forces 10^6 : $\sqrt{N} - 10^6$ to $\sqrt{N}$, and checks if one them divides N.

Solution: When generate $N = p \cdot q$, if $|p-q| < 10^{12}$, then regenerate N

Not Enough Randomness

user





router

Random p

Random q

$N = p \cdot q$

After router reset, router doesn't have enough rand. for p & q.

Many different routers will share p (but not q)

HTTPS

$$N_1 = p \cdot q_1$$

$$A \longleftrightarrow \text{Serv } A$$

$$N_2 = p \cdot q_2$$

$$B \longleftrightarrow \text{Serv } B$$

$$N_3 = p_2 \cdot q_2$$

$$C \longleftrightarrow \text{Serv } B$$

$$GCD(N_1, N_2) = p \Rightarrow$$

Factor N_1 & N_2

$$GCD(N_2, N_3) = q_2 \Rightarrow$$

Factor N_2 & N_3

In 2012, 0,4% of HTTPS could be decrypted this way

PKCS1

Public Key Cryptography Standard

16 bits	128 bits	
0000... 10	m	lots of randomness

this message is going Enc

User $\xrightarrow{m}$ Server

"Ok" / "Error"

the first 16 bits ≠ 00...010

Attack

$Enc(m)$

$Enc(m \cdot R_1)$

$Enc(m \cdot R_2)$ ← "Ok" / "Error"

⋮

$Enc(m \cdot R_{1000})$ ← "Ok" / "Error"

Baby version of this attack.

Pretend $N = 2^k$

$00\ldots\boxed{00}\quad m$

$\xrightarrow{Enc(2m)}$

$\xleftarrow{\text{"ok"}}$

I know $m_1 = 0$

$\xleftarrow{\text{"Error"}}$

I know $m_1 = 1$

$\xrightarrow{Enc(4m)}$

---

by $Enc(8m)$, I'll learn all bits of m one by one.

Problem: "Error" message reveals info about m.

Solution: Instead error, send random

Summary

- Encryption must be randomized!

"Attack" $\rightarrow c_1$

$\rightarrow c_2$

$\vdots$

$\rightarrow c_{2^{128}}$

Summary

- Encryption must be randomized!
- RSA is powerful and ubiquitous

Summary

- Encryption must be randomized!
- RSA is powerful and ubiquitous
- Simple, but needs to be implemented correctly

Summary

- Encryption must be randomized!
- RSA is powerful and ubiquitous
- Simple, but needs to be implemented correctly
- There are many great implementations

SKC $\xrightarrow{c = m \oplus R}$

$O(n)$ time

PKC $\xrightarrow{c = m^e}$

$O(n^3)$, $O(n^{2.5})$ time

Alice Use PKC Bob

$\xrightarrow{\text{long secret key } R}$

Use SKC

with this key