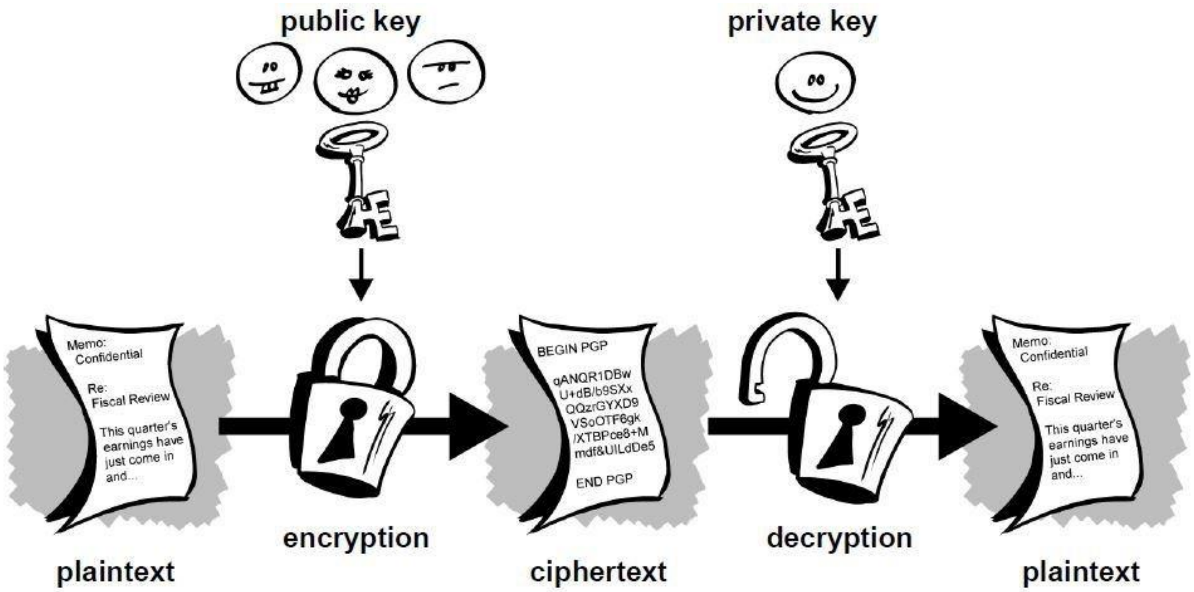
RSA Encryption and Signatures
By
Vipul Goyal



~~Fundamental lemma of powers?~~

If $a = b \pmod{N}$

Then $g^a = g^b \pmod{N}$?

NO!

Modulas changes to $\phi(N)$ in the exponent

(Correct) rule for powers

Euler's theorem: Let ϕ be the Euler's Phi function (formula for ϕ later)

If $a = b \pmod{\phi(N)}$
then $g^a = g^b \pmod{N}$

Remember: N becomes $\phi(N)$ in exponent

(Correct) rule for powers

Exercise: Prove $g^{\phi(N)} = 1 \pmod N$

Proof:

$\phi(N) = 0 \pmod{\phi(N)}$

Hence, $g^{\phi(N)} = g^0 = 1 \pmod N$

Also, $g^a = g^{a \bmod \phi(N)} \pmod N$

Even Faster Modular exponentiation:

To compute $g^a = g^{a \bmod \phi(N)} \pmod N$

Euler's Phi Function

Theorem: If p is a prime then $\phi(p) = (p-1)$. Easy to compute!

Theorem: If p,q are *distinct* primes then
$\phi(pq) = (p-1)(q-1)$

Say N = pq, can we compute $\phi(N)$?
Requires factoring!!

RSA: based on hardness of finding $\phi(N)$

Example: Faster Modular Exponentiation

Exercise: Compute $5^{237832} \pmod{13}$

Solution:

$\phi(13) = 12$ (since 13 is prime)

$237832 \bmod 12 = 4$

Hence, $5^{237832} \pmod{13} = 5^4 \pmod{13}$

Can compute $5^4 \pmod{13}$ by square and multiply or even directly since numbers are small

$5^4 \pmod{13} = 1$

RSA Basic Idea

To encrypt M, compute $C = M^E \pmod{N}$

- Here E, N is the public key
- To decrypt: need to take E-th root

Cool fact from Euler's theorem: If you know factoring of N, you can compute E-th root. How?

1) Compute $\phi(N) = (p-1)(q-1)$
2) Compute D such that E.D = 1 mod $\phi(N)$. In other words, D is the inverse of E. (can be done using Euclidean algorithm)
3) Compute

$$\begin{aligned} C^D &= M^{E.D} \pmod{N} \\ &= M^{E.D \bmod \phi(N)} \pmod{N} \\ &= M \pmod{N} \end{aligned}$$

RSA In Detail

Key Generation:

- Pick random distinct primes p, q. Compute $N = pq$.
- Choose E arbitrarily
- Compute D s.t. $E.D = 1 \bmod \phi(N)$
- $PK = (N, E)$, $SK = D$

Enc(PK, M)

- Given PK = (N, E), output ciphertext $C = M^E \pmod N$

Dec(SK, C)

- Given SK = D, output message $C^D \pmod N$

RSA Encryption Picture

pick $N = pq$

pick E

M, E, N

Enc

$C = M^E \pmod N$

Compute D s.t.

$E.D = 1 \bmod \phi(N)$

C, D

Dec

$M = C^D \pmod N$

Security of RSA

What if $N = p$ (rather than pq)?

- $\phi(N) = p-1$ can by publicly computed
- Given E, N, anyone can compute D s.t. $ED = 1 \bmod \phi(N)$
- Adversary can compute secret key and decrypt

But if $N = pq$, computing $\phi(N)$ seems hard.
In fact, as hard as factoring!

Security of RSA

Theorem: If $N = pq$, computing $\phi(N)$ as hard as factoring

Proof: Suppose someone can compute $\phi(N)$, will show that person can also easily factor N.

Given $N = pq$ and $\phi(N) = (p-1)(q-1)$, compute p, q as follows. Compute

- First compute $p+q = pq - (p-1)(q-1) + 1$. Call it N'.
- Since $N' = p+q$, we have $q = N' - p$
- Since $N = pq$, we have $N = p(N' - p)$
- Hence, $p^2 - N'p + N = 0$
- Solve quadratic eqn to find p
- Once we have p, easy to find $q = N' - p$

Is RSA Secure if Factoring is Hard?

We don’t know!!

Adversary cannot compute $\phi(N)$. But maybe there are other ways of decrypting?

Maybe other ways of taking E-th roots? Some weird formula no one has thought about?

Proving that breaking RSA is equivalent to factoring remains an open problem despite 40 years of research!

While we can’t prove it, we also don’t know how to break RSA

RSA Cannot be Used Directly

RSA Encryption is deterministic $C = M^E \pmod{N}$

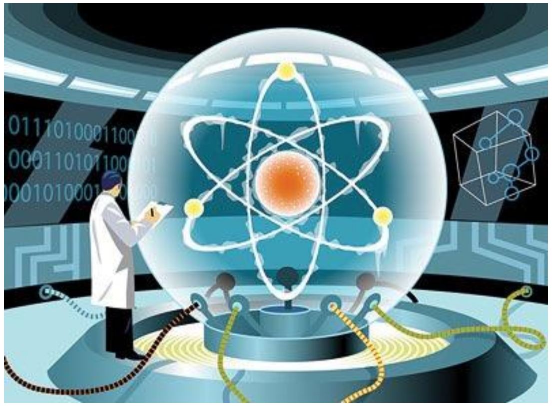
Recall: deterministic public key encryption is bad. Say I encrypt ATTACK on day 1 and day 3. Adversary knows it's the same message!

Randomized encryption: encrypting the same message twice gives two different ciphertexts (e.g. ElGamal encryption))

RSA-OAEP is Used Instead

- Textbook RSA: almost never used because of this problem
- A variant of textbook RSA called RSA-OAEP: randomizes the encryption algorithm.
- Variants of this are used all over the internet (e.g. in HTTPS protocol)

ElGamal, RSA, ……



Quantum computers
can factor efficiently

Quantum computers
can compute discrete log efficiently

Post-quantum crypto: a (very successful) branch of crypto developing security against quantum computers

- Lattice-based cryptography can’t be broken by quantum computers
- SHA-256, AES can’t be broken by quantum computers either

RSA Digital Signatures

Key Generation:

- Pick random distinct primes p, q. Compute $N = pq$.
- Choose E arbitrarily
- Compute D such that $E.D = 1 \bmod \phi(N)$
- VK = (N, E), SK = D

Sign(SK, M)

- Given SK = D, output signature $\sigma = M^D \pmod N$

Verify(VK, M, σ)

- Given VK = (N, E), Verify that M matches $\sigma^E \pmod N$

RSA Signature Security

- To compute $\sigma = M^D \pmod{N}$, need to know D
- But computing D is hard given (E, N) as we saw before!

Warning 1:

- Interchanging PK, SK to convert encryption to signatures works only for RSA.
- E and D are interchangeable. We just need $E.D = 1 \bmod \phi(N)$
- Doesn't work for other encryption schemes like ElGamal

Warning 2:

- RSA Signatures are not directly secure. Why?

RSA Signature Attack

Given signatures on two messages $\sigma_1 = M_1^D \pmod N$ and $\sigma_2 = M_2^D \pmod N$

- Compute $\sigma_1 . \sigma_2 = (M_1^D)(M_2^D) \pmod N = (M_1 M_2)^D \pmod N$
- This is a valid signature on message $M_3 = M_1 M_2$
- Thus, given signatures on two messages, easy to compute signature on a third message without the secret key just by multiplying them
- Doesn’t mean you can compute a signature on any desired message

Hence: RSA signatures are never used directly. Need to modify by first hashing the messages.

RSA Signatures with Hashes

Simple Idea: instead of signing M, sign H(M)

Key Generation (same):

- Pick random distinct primes p, q. Compute $N = pq$.
- Choose E arbitrarily
- Compute D such that $E.D = 1 \bmod \phi(N)$
- $VK = (N, E)$, $SK = D$

Sign(SK, M)

- Given M, compute H(M) first
- Given SK = D, output signature $\sigma = H(M)^D \pmod N$

Verify(VK, M, σ)

- Given M, first compute H(M)
- Given VK = (N, E), Verify that H(M) matches $\sigma^E \pmod N$

Security: Previous Attack Doesn't Work

Given signatures on two messages $\sigma_1 = H(M_1)^D \pmod N$ and $\sigma_2 = H(M_2)^D \pmod N$

- Say you compute $\sigma_1.\sigma_2 = (H(M_1).H(M_2))^D \pmod N$
- If this is a valid signature on message M_3 then it must be:

$$H(M_3) = H(M_1).H(M_2)$$

- But if the hash function H is random, probability that this relation is satisfied is very small!

Questions?