Cryptography
CS 555

Topic 24: Finding Prime Numbers, RSA

Recap

- Number Theory Basics
- *Abelian Groups*
- $\phi(pq) = (p-1)(q-1)$ for distinct primes p and q
- $\phi(N) = |\mathbb{Z}_N^*|$

$$[g^x \text{mod N}] = \left[g^{[x\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \text{mod N}\right]$$

RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p-1)(q-1)$

Step 3: …

Question: How do we accomplish step one?

Bertrand’s Postulate

Theorem 8.32. For any n > 1 the fraction of n-bit integers that are prime is at least ${}^{1}/_{3n}$.

GenerateRandomPrime(1^n)
For i=1 to $3n^2$:
  p’← $\{0,1\}^{n-1}$
  p← $1\|p'$
  if isPrime(p) **then**
    return p
return fail

Can we do this in polynomial time?

Bertrand's Postulate

Theorem 8.32. For any n > 1 the fraction of n-bit integers that are prime is at least ${}^{1}/_{3n}$.

GenerateRandomPrime(1^n)

For i=1 to $3n^2$:

  p' ← $\{0,1\}^{n-1}$

  p ← $1||p'$

  if isPrime(p) **then**

    return p

return fail

Assume for now that we can run isPrime(p). What are the odds that the algorithm fails?

On each iteration the probability that p is not a prime is $\left(1-\frac{1}{3n}\right)$

We fail if we pick a non-prime in all $3n^2$ iterations. The probability is

$$\left(1-\frac{1}{3n}\right)^{3n^2} = \left(\left(1-\frac{1}{3n}\right)^{3n}\right)^{n} \leq e^{-n}$$

isPrime(p): Miller-Rabin Test

- We can check for primality of p in polynomial time in $||p||$.

Theory: Deterministic algorithm to test for primality.

- See breakthrough paper "Primes is in P"

Practice: Miller-Rabin Test (randomized algorithm)

- **Guarantee 1:** If p is prime then the test outputs YES
- **Guarantee 2:** If p is not prime then the test outputs NO except with negligible probability.

https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf

The "Almost" Miller-Rabin Test

Input: Integer N and parameter 1^t

Output: "prime" or "composite"

for i=1 to t:

    a ← {1,…,N-1}

    if $a^{N-1} \neq \text{mod N}$ then return "composite"

Return "prime"

Claim: If N is prime then algorithm always outputs "prime"

Proof: For any a ∈ {1,…,N−1} we have $a^{N-1} = a^{\phi(N)} = 1 \; mod \; N$

The “Almost” Miller-Rabin Test

Input: Integer N and parameter 1^t

Output: “prime” or “composite”

for i=1 to t:

  a ← {1,…,N-1}

  if $a^{N-1} \neq 1 \bmod \mathrm{N}$ then return “composit

Return “prime”

Need a bit of extra work to handle Carmichael numbers.

Fact: If N is composite and not a Carmichael number then the algorithm outputs “composite” with probability

$$1 - 2^{-t}$$

Back to RSA Key-Generation

KeyGeneration(1^n)

Step 1: Pick two random n-bit primes p and q

Step 2: Let N=pq, $\phi(N) = (p-1)(q-1)$

Step 3: Pick e > 1 such that gcd(e, $\phi(N)$)=1

Step 4: Set d=[e^{-1} mod $\phi(N)$] (secret key)

Return: N, e, d

- How do we find d?
- **Answer:** Use extended gcd algorithm to find e^{-1}mod $\phi(N)$.

(Plain) RSA Encryption

- Public Key: PK=(N,e)
- Message $\mathrm{m} \in \mathbb{Z}_{\mathrm{N}}$

$$\mathbf{Enc}_{\mathbf{PK}}(\mathrm{m}) = [m^e \bmod \mathrm{N}]$$

- **Remark:** Encryption is efficient if we use the power mod algorithm.

(Plain) RSA Decryption

- Public Key: SK=(N,d)
- Ciphertext $c \in \mathbb{Z}_N$

$$\mathbf{Dec_{SK}(c)} = [c^d \bmod \mathrm{N}]$$

- **Remark 1:** Decryption is efficient if we use the power mod algorithm.
- **Remark 2:** Suppose that $\mathrm{m} \in \mathbb{Z}_N^*$ and let c=**Enc$_{PK}$(**m**)** $= [m^e \bmod \mathrm{N}]$

$$\begin{aligned}\mathbf{Dec_{SK}(c)} &= [(m^e)^d \bmod \mathrm{N}] = [m^{ed} \bmod \mathrm{N}] \\ &= \left[m^{[ed \bmod \boldsymbol{\phi}(\boldsymbol{N})]} \bmod \mathrm{N}\right] \\ &= [m^1 \bmod \mathrm{N}] = m\end{aligned}$$

RSA Decryption

- Public Key: SK=(N,d)
- Ciphertext $c \in \mathbb{Z}_N$

$$\textbf{Dec}_{\textbf{SK}}\textbf{(}c\textbf{)} = [c^d \bmod \mathrm{N}]$$

- **Remark 1:** Decryption is efficient if we use the power mod algorithm.
- **Remark 2:** Suppose that $\mathrm{m} \in \mathbb{Z}_N^*$ and let c=**Enc**$_{\textbf{PK}}$**(**m**)** $= [m^e \bmod \mathrm{N}]$ **then**

$$\textbf{Dec}_{\textbf{SK}}\textbf{(}c\textbf{)} = m$$

- **Remark 3:** Even if $\mathrm{m} \in \mathbb{Z}_N - \mathbb{Z}_N^*$ and let c=**Enc**$_{\textbf{PK}}$**(**m**)** $= [m^e \bmod \mathrm{N}]$ **then**

$$\textbf{Dec}_{\textbf{SK}}\textbf{(}c\textbf{)} = m$$

  - Use Chinese Remainder Theorem to show this

Factoring Assumption

Let **GenModulus**(1^n) be a randomized algorithm that outputs ($N=pq,p,q$) where p and q are n-bit primes (except with negligible probability **negl**(n)).

Experiment $FACTOR_{A,n}$

1. ($N=pq,p,q$) ← **GenModulus**(1^n)
2. Attacker A is given N as input
3. Attacker A outputs $p' > 1$ and $q' > 1$
4. Attacker A wins if $N=p'q'$.

Factoring Assumption

- Necessary for security of RSA.
- Not known to be sufficient.

Experiment $\text{FACTOR}_{A,n}$

1. $(N=pq,p,q) \leftarrow$ **GenModulus**(1^n)
2. Attacker A is given N as input
3. Attacker A outputs $p' > 1$ and $q' > 1$
4. Attacker A wins ($\text{FACTOR}_{A,n} = 1$) if and only if $N=p'q'$.

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}\ \Pr[\text{FACTOR}_{A,n} = 1] \leq \mu(n)$$

RSA-Assumption

RSA-Experiment: RSA-INV$_{A,n}$

1. **Run KeyGeneration**(1^n) **to obtain (N,e,d)**
2. **Pick uniform** $y \in \mathbb{Z}_N^*$
3. Attacker A is given N, e, y and outputs $x \in \mathbb{Z}_N^*$
4. Attacker wins (RSA-INV$_{A,n}$=1) if $x^e = y \bmod N$

$$\forall PPT\ A\ \exists \mu \text{ (negligible) s.t } \Pr[\text{RSA–INV}_{A,n} = 1] \leq \mu(n)$$

(Plain) RSA Discussion

- We have not introduced security models like CPA-Security or CCA-security for Public Key Cryptosystems

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- Plain RSA is also highly vulnerable to chosen-ciphertext attacks
  - Attacker intercepts ciphertext c of secret message m
  - Attacker generates ciphertext c’ for secret message 2m
  - Attacker asks for decryption of c’ to obtain 2m
  - Divide by 2 to recover original message m

(Plain) RSA Discussion

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- In a public key setting the attacker does have access to an encryption oracle

- Encrypted messages with low entropy are vulnerable to a brute-force attack

(Plain) RSA Discussion

- Plain RSA is also highly vulnerable to chosen-ciphertext attacks
  - Attacker intercepts ciphertext $c = [m^e \bmod \mathrm{N}]$
  - Attacker asks for decryption of $[c2^e \bmod \mathrm{N}]$ and receives 2m.
  - Divide by two to recover message
- As above example shows plain RSA is also highly vulnerable to ciphertext-tampering attacks
  - See homework questions ☺

Mathematica Demo

https://www.cs.purdue.edu/homes/jblocki/courses/555_Spring17/slides/Lecture24Demo.nb

Note: Online version of mathematica available at https://sandbox.open.wolframcloud.com (reduced functionality, but can be used to solve homework bonus problems)

Next Class

- Read Katz and Lindell 8.3, 11.5.1
- Discrete Log, DDH + Attacks on Plain RSA