CS641

Modern Cryptology

Lecture 12

RSA: Time Complexity of Key Generation

- Generating two primes p and q of size ℓ is done by randomly choosing a number of size ℓ and testing if it is prime.
  - Prime Number Theorem guarantees that a random number of size ℓ is prime with probability roughly $\frac{1}{\ell}$.
  - Testing if a number is prime can be done efficiently through any one of several methods.
- Computing d from e can be done via extended GCD algorithm, which is very efficient.
- Therefore, key generation is efficient.

RSA: Time Complexity of Encryption and Decryption

- Computing $m^e \pmod{n}$ using repeated squaring requires $O(\log e) = O(\log n)$ multiplications.
- Each multiplication is between two $\log n$ bit numbers and so requires $O(\log n \log\log n)$ operations.
- This results in overall time complexity of $O(\log^2 n \log\log n)$ which is approximately quadratic.
- Also, the constant inside O-notation is not small.
- Hence, encrypting a plaintext block, even though efficient, takes non-trivial time.
- In view of this, RSA is used primarily for exchanging small amount of information (like AES keys), and AES is used for bulk encryption.

RSA Cryptanalysis

- Since encryption key is public, Chosen Plaintext attack is trivially possible:
  - Ela does not need Anubha or Braj for this attack!
- Chose Ciphertext attack still requires access to Braj's decryption.

RSA Cryptanalysis: Computing Private Key

- Given e and n, how does one compute d?
- Relationship between them:

$$de = 1 \pmod{\phi(n)}.$$

- If $\phi(n)$ is known, d can be easily computed by running Extended GCD algorithm on e and $\phi(n)$.
- How does one compute $\phi(n)$?
- Computing $\phi(n)$ from n is equivalent to factoring n:
  - If $n = pq$ can be factored, then $\phi(n) = (p-1)(q-1)$ can be easily computed.
  - If $\phi(n)$ is computed, then $p + q = n + 1 - \phi(n)$ can be computed.
  - Factors p and q can then be recovered as roots of the quadratic equation $x^2 - (p+q)x + n$.

RSA Cryptanalysis: Computing Private Key

- Can one compute d without computing $\phi(n)$?
- It has been shown that computing d in general is equivalent to factoring n.
- The fastest known algorithm for factoring integers takes time $2^{\Omega((\log n)^{1/3}(\log\log n)^{2/3})}$.
- Therefore, with n of 1024 bits, it is hard to compute d.
- It is important to note that computing d for some special cases may still be possible.

RSA Cryptanalysis: Computing Private Key

- As observed earlier, if $m \notin Z_n^*$, n can be factored and d can be recovered.
- Let us estimate how likely that is:
  - If we choose m randomly from Z_n, then the probability that $m \notin Z_n^*$ equals $1 - \frac{|Z_n^*|}{n} = \frac{p+q-1}{n} = \Theta(1/n^{1/2})$ assuming $p, q = \Theta(n^{1/2})$.
  - Hence, average time taken to find one such m is much more that time taken to factor n!

RSA Cryptanalysis: Computing Plaintext

- Given $c = m^e \pmod{n}$, e, and n, how does one compute m?
- If d is known, m can be easily computed as $c^d \pmod{n}$.
  - However, computing d is hard as already observed.
- There appears no other way of computing m:
  - Until now, no general method is known to recover m without factoring n.
  - This does not rule out computing m in special cases.

RSA Cryptanalysis

- This makes RSA encryption very secure in general.
- However, one needs to avoid, when choosing the key, several special cases when the system becomes insecure.
- We discuss major such cases.

RSA Cryptanalysis: Prime Factors

- Number $n = pq$, with $p < q$, can be trivially factored using $O(p)$ divisions.
- So if p is small, n can be factored quickly.
- Similarly, if $q - p$ is small, then n can be factored quickly as follows:
  - Let $t = q - p$.
  - Since $p < \sqrt{n} < q$, at least one of the two primes is within $t/2$ distance from $\sqrt{n}$.
  - Starting from $\sqrt{n}$ and checking $t/2$ numbers on both sides of it factors n.
- Therefore, both p and q should be large but not close to each other.

RSA Cryptanalysis: Smooth Primes

Smooth Numbers and Primes

Composite number m is k-smooth if all prime factors of m are $\leq k$. Prime number p is k-smooth if $p - 1$ is k-smooth.

- Let $n = pq$ and suppose p is k-smooth but q is not.
- Let $T = (k!)^{\log n}$.
- Choose a random $a \in [1, n]$ and compute $\gcd(a^T - 1, n)$.

RSA Cryptanalysis: Smooth Primes

Lemma

$\gcd(a^T - 1, n) = p$ for most a's.

- Since p is k-smooth but q is not, $p-1$ divides $T = (k!)^{\log n}$ but $q-1$ does not.
- Suppose $a \in Z_n^*$.
- Then, $a^T = a^{r(p-1)} = 1 \pmod{p}$.
- On the other hand, $a^T \pmod{q} \neq 1$ whenever a is generator of F_q^*.
- Since most a's are generators of F_q^*, it follows that $a^T - 1$ is divisible by p but not by q.

RSA Cryptanalysis: Smooth Primes

- Hence, both p and q should not be k-smooth for small k.
- The best strategy is to choose p and q such that both $(p-1)/2$ and $(q-1)/2$ are prime numbers.
- Such primes are called Sophie Germain primes.
- Large number of Sophie Germain primes are found to exist:
  - It is conjectured that there exist $n/(\ln n)^2$ Sophie Germain primes in $[1, n]$.
  - So a random selection will yield such a prime quickly.