EE 720: An Introduction to Number Theory and Cryptography (Spring 2020)

Lecture 17 — March 23, 2020

Instructor: Saravanan Vijayakumaran *Scribe: Saravanan Vijayakumaran*

1 Lecture Plan

- Using the Chinese Remainder Theorem
- RSA Encryption

2 Chinese Remainder Theorem

- **Chinese Remainder Theorem:** Let $N = pq$ where p, q are integers greater than 1 which are relatively prime, i.e. $\gcd(p, q) = 1$. Then

  $$\mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q \text{ and } \mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*.$$

  Moreover, the function $f : \mathbb{Z}_N \mapsto \mathbb{Z}_p \times \mathbb{Z}_q$ defined by

  $$f(x) = (x \bmod p, x \bmod q)$$

  is an isomorphism from $\mathbb{Z}_N$ to $\mathbb{Z}_p \times \mathbb{Z}_q$, and the restriction of f to $\mathbb{Z}_N^*$ is an isomorphism from $\mathbb{Z}_N^*$ to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$.

- Usage
  - Compute $11^{53} \bmod 15$
  - Compute $18^{25} \bmod 35$

- How to go from $(x_p, x_q) = (x \bmod p, x \bmod q)$ to $x \bmod N$ where $\gcd(p, q) = 1$?
  - Compute X, Y such that $Xp + Yq = 1$.
  - Set $1_p := Yq \bmod N$ and $1_q := Xp \bmod N$.
  - Compute $x := x_p \cdot 1_p + x_q \cdot 1_q \bmod N$.

- Example: $p = 5, q = 7$ and $N = 35$. What does $(4, 3)$ correspond to?

- An extension of the Chinese remainder theorem says that if $m_1, m_2 \dots, m_l$ are pairwise relatively prime (i.e., $\gcd(m_i, m_j) = 1$ for all $i \neq j$) and $M = \Pi_{i=1}^{l} m_i$, then

  $$\mathbb{Z}_M \simeq \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_l} \text{ and } \mathbb{Z}_M^* \simeq \mathbb{Z}_{m_1}^* \times \mathbb{Z}_{m_2}^* \times \cdots \times \mathbb{Z}_{m_l}^*.$$

- Let $m_1, m_2, \ldots, m_l$ be pairwise relatively prime positive integers. Then the unique solution modulo $M = m_1 m_2 \cdots m_l$ of the system of congruences

$$\begin{aligned} x &= a_1 \bmod m_1 \\ x &= a_2 \bmod m_2 \\ &\vdots \\ x &= a_l \bmod m_l \end{aligned}$$

  is given by

$$x = (a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_l M_l y_l) \bmod M$$

  where $M_i = \frac{M}{m_i}$ and $M_i y_i = 1 \bmod m_i$.

- Example: Solve for x modulo 105 which satisfied the following congruences.

$$\begin{aligned} x &= 1 \bmod 3 \\ x &= 2 \bmod 5 \\ x &= 3 \bmod 7 \end{aligned}$$

3 RSA Encryption

- Given a composite integer N, the factoring problem is to find integers $p, q > 1$ such that $pq = N$.

- One can find factors of N by *trial division*, i.e. exhaustively checking if p divides N for $p = 2, 3, \ldots, \lfloor\sqrt{N}\rfloor$. But trial division has running time $\mathcal{O}\left(\sqrt{N} \cdot \texttt{polylog}(N)\right) = \mathcal{O}\left(2^{\|N\|/2} \cdot \|N\|^c\right)$ which is exponential in the input length $\|N\|$.

3.1 The Factoring Assumption

- Let `GenModulus` be a polynomial-time algorithm that, on input 1^n, outputs (N, p, q) where $N = pq$, and p and q are n-bit primes except with probability negligible in n.

- **The factoring experiment** $\texttt{Factor}_{\mathcal{A},\texttt{GenModulus}}(n)$:
  1. Run $\texttt{GenModulus}(1^n)$ to obtain (N, p, q).
  2. $\mathcal{A}$ is given N, and outputs $p', q' > 1$.
  3. The output of the experiment is 1 if $N = p'q'$, and 0 otherwise.

- We use p', q' in the above experiment because it is possible that `GenModulus` returns composite integers p, q albeit with negligible probability. In this case, we could find factors of N other than p and q.

- **Definition: Factoring is hard relative to** `GenModulus` if for all PPT algorithms $\mathcal{A}$ there exists a negligible function `negl` such that $\Pr[\texttt{Factor}_{\mathcal{A},\texttt{GenModulus}}(n) = 1] \leq \texttt{negl}(n)$.

- The **factoring assumption** states that there exists a `GenModulus` relative to which factoring is hard.

3.2 Plain RSA

- Let `GenRSA` be a PPT algorithm that on input 1^n, outputs a modulus N that is the product of two n-bit primes, along with integers $e, d > 1$ satisfying $ed = 1 \bmod \phi(N)$.
- If we chose $e > 1$ such that $\gcd(e, \phi(N)) = 1$, then the multiplicative inverse d of e in $\mathbb{Z}^*_{\phi(N)}$ will satisfy the required conditions.
- Define a public-key encryption scheme as follows:
  - `Gen`: On input 1^n run `GenRSA`(1^n) to obtain N, e, and d. The public key is $\langle N, e \rangle$ and the private key is $\langle N, d \rangle$.
  - `Enc`: On input a public key $pk = \langle N, e \rangle$ and message $m \in \mathbb{Z}^*_N$, compute the ciphertext $c = m^e \bmod N$.
  - `Dec`: On input a private key $sk = \langle N, d \rangle$ and ciphertext $c \in \mathbb{Z}^*_N$, output $\hat{m} = c^d \bmod N$.

4 References and Additional Reading

- Section 8.1.5 from Katz/Lindell
- Sections 8.2.3, 11.5.1 from Katz/Lindell