Discrete Mathematics
Lecture 5

Liangfeng Zhang
School of Information Science and Technology
ShanghaiTech University

Summary of Lecture 4

Public-key encryption: $\Pi = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}) + \mathcal{M}, |\mathcal{M}| > 1$

- **Correctness**: $\mathbf{Dec}(sk, \mathbf{Enc}(pk, m)) = m$ for any pk, sk, m
- **Security**: if sk is not known, it's difficult to learn m from pk, c

Plain RSA: $N = pq, \;\; \mathcal{M} = \{m: 0 \leq m < N, \gcd(m, N) = 1\}$

- $pk = (N, e), sk = (N, d); \gcd(e, \phi(N)) = 1; de \equiv 1 \pmod{\phi(N)}$
- $c = m^e \bmod N$
- $m = c^d \bmod N$

Implementation Issues: $p, q, N, \phi(N), m, c$ are all large

- Given n, how to choose n-bit primes p, q
- Given $(e, \phi(N))$, how to compute d
- Given pk, m, how to compute c

Addition

Bit Length of Integer: $\boxed{\ell(a) = \begin{cases} \lfloor \log_2(|a|) \rfloor + 1 & a \neq 0 \\ 1 & a = 0 \end{cases}}$

Binary Representation: a 0-1 sequence

- $a = (a_{k-1} \ldots a_1 a_0)_2 \Leftrightarrow a = a_{k-1} 2^{k-1} + \cdots + a_1 \cdot 2^1 + a_0 \cdot 2^0$

Algorithm for Addition:

- **Input**: $a = (a_{k-1} \cdots a_1 a_0)_2, b = (b_{k-1} \cdots b_1 b_0)_2$
- **Output**: $c = a + b = (c_k c_{k-1} \cdots c_1 c_0)_2$
  - $carry \leftarrow 0$
  - for $i \leftarrow 0$ to $k - 1$ do
    - $t \leftarrow a_i + b_i + carry$;
    - set c_i and $carry$ such that $t = 2 \cdot carry + c_i$
  - $c_k \leftarrow carry$
- **Complexity**: $O(k)$ bit operations

Subtraction

Algorithm for Subtraction:

- **Input**: $a = (a_{k-1} \cdots a_1 a_0)_2, b = (b_{k-1} \cdots b_1 b_0)_2, a \geq b$
- **Output**: $c = a - b = (c_{k-1} \cdots c_1 c_0)_2$
  - $carry \leftarrow 0$
  - for $i \leftarrow 0$ to $k - 1$ do
    - $t \leftarrow a_i - b_i + carry$;
    - set c_i and $carry$ such that $t = 2 \cdot carry + c_i$
- **Complexity**: $O(k)$ bit operations

Multiplication

Algorithm for Multiplication:

- **Input**: $a = (a_{k-1} \cdots a_0)_2, b = (b_{k-1} \cdots b_0)_2$
- **Output**: $c = ab = (c_{2k-1} \cdots c_0)_2$
  - $c \leftarrow 0; x \leftarrow a$
  - for $i \leftarrow 0$ to $k - 1$ do
    - if $b_i = 1$, then $c \leftarrow c + x$;
    - $x \leftarrow x + x$;
- **Complexity**: $O(k^2)$ bit operations

Division

Algorithm for Division:

- **Input**: $a = (a_{k-1} \cdots a_0)_2, b = (b_{l-1} \cdots b_0)_2, a \geq b, a_{k-1} = b_{l-1} = 1$
- **Output**: $q = (q_{k-l} \cdots q_0)_2$ and $r = (r_{l-1} \cdots r_0)_2$ s.t. $a = bq + r, 0 \leq r < b$
  - $(r_k r_{k-1} \cdots r_0)_2 \leftarrow (0 a_{k-1} \cdots a_0)_2$
  - for $i \leftarrow k - l$ down to 0 do
    - $q_i \leftarrow 2r_{i+l} + r_{i+l-1}$;
    - If $q_i \geq 2$, then $q_i \leftarrow 1$;
    - $(r_{i+l} \cdots r_i)_2 \leftarrow (r_{i+l} \cdots r_i)_2 - q_i \cdot b$;
    - while $r_{i+l} < 0$ do
      - $(r_{i+l} \cdots r_i)_2 \leftarrow (r_{i+l} \cdots r_i)_2 + b$;
      - $q_i \leftarrow q_i - 1$;
  - output $q = (q_{k-l} \cdots q_0)_2$ and $r = (r_{l-1} \cdots r_0)$;
- **Complexity**: $O((k - l + 1) \cdot l)$ bit operations

$$\begin{array}{r} 11 \\ 111 \overline{)10111} \\ \underline{111} \\ 100 \\ \underline{111} \\ 10 \end{array}$$

$10111 = (111) \times (11) + (10)$

Arithmetic Modulo n 算术模n

THEOREM: Let $a, b \in \{0,1,\ldots,n-1\}$. Then

- $(a \pm b) \bmod n$ can be computed in $O(\ell(n))$ bit operations
  - $\ell(a), \ell(b) \le \ell(n)$
    - $a \pm b$ are computable in $O(\ell(n))$ bit operations
  - $0 \le |a+b|, |a-b| < 2n$
    - $(a \pm b) \bmod n$ are computable in $O((\ell(2n) - \ell(n) + 1)\ell(n)) = O(\ell(n))$ bit operations
- $(ab) \bmod n$ can be computed in $O(\ell(n)^2)$ bit operations
  - $\ell(a), \ell(b) \le \ell(n)$
    - ab is computable in $O(\ell(n)^2)$ bit operations
  - $0 \le |ab| < n^2$
    - $(ab) \bmod n$ is computable in $O\big((\ell(n^2) - \ell(n) + 1)\ell(n)\big) = O(\ell(n)^2)$ bit operations.

Arithmetic Modulo n

Modulo exponentiation: For $0 \leq a < n, e \in \mathbb{N}$, $a^e \bmod n = ?$

- Complexity? How to compute efficiently?

EXAMPLE: modulo exponentiation

- $m =$1437339113920498981637906207424471163445460406448981415203760376263650078098
  99615665793895112104794373551079787727363529151277801402630305742433442340983358
  787394193855033926469913603762712163723160462115649025
- $e =$46310011625494823943446873944318243690297367227688331207962573871391818800156
  61440418125399478543429257625536255388418199849246329730346646442802201832756472
  3810228367576715525319623371983456905064392494176785
- $N =$2452466449002782119765176635730880184670267876783327597434144517150616008300
  38587216952208399332071549103626827191679864079776723243005600592035631246561218
  465817904100131859299619933817012149335034875870551067
- $\phi(N) =$2452466449002782119765176635730880184670267876783327597434144517150616008
  30038587216952208399332071549102628322861627039184220494270313938703906283392288
  487724394251766892786817697178343799758481228648091667216
- $m^e \bmod N = ?$

Square-and-Multiply

ALGORITHM: compute $a^e \bmod n$ in polynomial time

- **Input:** $a \in \{0,1,\ldots,n-1\}$; $e = (e_{k-1} \cdots e_0)_2$ $\quad //k = \ell(e)$
  - $e = e_{k-1} \cdot 2^{k-1} + \cdots + e_1 \cdot 2^1 + e_0 \cdot 2^0$
- **Output:** $a^e \bmod n$
  - **Square**: this step requires $O(k)$ multiplications modulo n
    - $x_0 = a$
    - $x_1 = (x_0^2 \bmod n) = (a^2 \bmod n)$
    - $x_2 = (x_1^2 \bmod n) = (a^{2^2} \bmod n)$
    - …
    - $x_{k-1} = (x_{k-2}^2 \bmod n) = (a^{2^{k-1}} \bmod n)$
  - **Multiply**: this step requires $O(k)$ multiplications modulo n
    - $(a^e \bmod n) = (x_0^{e_0} \cdot x_1^{e_1} \cdots x_{k-1}^{e_{k-1}} \bmod n)$
- **Complexity**: $O(k)$ multiplications modulo n

$O(k) \cdot (|n|^2)$

Square-and-Multiply

EXAMPLE: Compute $2^{123} \bmod 35$ using square-and-multiply.

- **Input**: $a = 2; n = 35;\ e = 123 = (1\,1\,1\,1\,0\,1\,1)_2;\ k = 7$
- **Square:** $k - 1$ multiplications modulo n will be done
  - $x_0 = a = 2;$
  - $x_1 = x_0^2 \bmod n = 4$
  - $x_2 = x_1^2 \bmod n = 16$
  - $x_3 = x_2^2 \bmod n = 11$
  - $x_4 = x_3^2 \bmod n = 16$
  - $x_5 = x_4^2 \bmod n = 11$
  - $x_6 = x_5^2 \bmod n = 16$
- **Multiply:** at most $k - 1$ multiplications modulo n will be done
  - $a^e = x_0 x_1 x_3 x_4 x_5 x_6 = 2 \times 4 \times 11 \times 16 \times 11 \times 16 \equiv 8 \pmod{35}$
  - $(2^{123} \bmod 35) = 8$

$\varphi(35) = 4 \times 6 = 24$

$\gcd(2, 35) = 1$

$2^{123} \equiv 2^{123 \% 24} \equiv 2^3 \equiv 8$

Euclidean Algorithm (EA)

gcd(a,b)=gcd(b,a%b)

ALGORITHM: compute $\gcd(a, b)$

- **Input**: a, b $(a \geq b > 0)$
- **Output**: $d = \gcd(a, b)$
  - $r_0 = a; r_1 = b;$
  - $r_0 = r_1 q_1 + r_2$ $(0 < r_2 < r_1)$
  - $\vdots$
  - $r_{i-1} = r_i q_i + r_{i+1}$ $(0 < r_{i+1} < r_i)$
  - $\vdots$
  - $r_{k-2} = r_{k-1} q_{k-1} + r_k$ $(0 < r_k < r_{k-1})$
  - $r_{k-1} = r_k q_k$
  - output r_k

$a = 12345, b = 123$		
i	r_i	q_i
0	12345	
1	123	100
2	45	2
3	33	1
4	12	2
5	9	1
6	3	3
7	0	

(12345,123)

Correctness: $d = \gcd(r_0, r_1) = \cdots = \gcd(r_{k-1}, r_k) = r_k$ =(123,45)

=(45,33)

=(33,12)=(12,9)=(9,3)=(3,0)=3

Extended Euclidean Algorithm (EEA)

ALGORITHM: compute $d = \gcd(a, b)$, s, t such that $as + bt = d$

- **Input**: a, b $(a \geq b > 0)$
- **Output**: $d = \gcd(a, b)$, integers s, t such that $d = as + bt$
  - $r_0 = a; r_1 = b;$ $\binom{s_0}{t_0} = \binom{1}{0}; \binom{s_1}{t_1} = \binom{0}{1};$
  - $r_0 = r_1 q_1 + r_2 \;\; (0 < r_2 < r_1);$ $\binom{s_2}{t_2} = \binom{s_0}{t_0} - q_1 \binom{s_1}{t_1}$
  - $\vdots$
  - $r_{i-1} = r_i q_i + r_{i+1} \;\; (0 < r_{i+1} < r_i);$ $\binom{s_{i+1}}{t_{i+1}} = \binom{s_{i-1}}{t_{i-1}} - q_i \binom{s_i}{t_i}$
  - $\vdots$
  - $r_{k-2} = r_{k-1} q_{k-1} + r_k \; (0 < r_k < r_{k-1});$ $\binom{s_k}{t_k} = \binom{s_{k-2}}{t_{k-2}} - q_{k-1} \binom{s_{k-1}}{t_{k-1}}$
  - $r_{k-1} = r_k q_k$
  - output r_k, s_k, t_k

EEA

Correctness: We have that $r_i = as_i + bt_i$ for $i = 0,1,2,\ldots,k$

- $r_0 = a = (a,b)\binom{s_0}{t_0}$; $r_1 = b = (a,b)\binom{s_1}{t_1}$;
- $r_2 = r_0 - q_1 r_1 = (a,b)\binom{s_0}{t_0} - q_1 \cdot (a,b)\binom{s_1}{t_1} = (a,b)\binom{s_2}{t_2}$;
- $\vdots$
- $r_k = r_{k-2} - q_{k-1} r_{k-1} = (a,b)\binom{s_{k-2}}{t_{k-2}} - q_{k-1} \cdot (a,b)\binom{s_{k-1}}{t_{k-1}} = (a,b)\binom{s_k}{t_k}$

EXAMPLE: Execution of the EEA on input $a = 12345, b = 123$

i	r_i	q_i	s_i	t_i
0	12345		1	0
1	123	100	0	1
2	45	2	1	-100
3	33	1	-2	201
4	12	2	3	-301
5	9	1	-8	803
6	3	3	11	-1104
7	0			

Complexity

THEOREM: Let $\alpha = \frac{1}{2}(1+\sqrt{5})$. Then $k \leq \ln b / \ln \alpha + 1$ in EA.

- $k = 1$: $k \leq \ln b / \ln \alpha + 1$
- $k > 1$: we show that $r_{k-i} \geq \alpha^i$ for $i = 0,1,\dots,k-1$
  - $i = 0$: $r_k \geq 1 = \alpha^0$
  - $i = 1$: $r_{k-1} > r_k \Rightarrow r_{k-1} \geq r_k + 1 \geq 2 \geq \alpha^1$
  - Suppose that $r_{k-i} \geq \alpha^i$ for $i \leq j$
    - $$\begin{aligned} r_{k-(j+1)} &= r_{k-j} q_{k-j} + r_{k-(j-1)} \\ &\geq \alpha^j + \alpha^{j-1} \\ &= \alpha^{j-1}(\alpha + 1) \\ &= \alpha^{j+1} \end{aligned}$$
- $b = r_1 \geq \alpha^{k-1} \Rightarrow k \leq \ln b \,/ \ln \alpha + 1$

Complexity of EA and EEA: $O(\ell(a)\ell(b))$ bit operations

Prime Number Theorem

DEFINITION: For $x \in \mathbb{R}^+$, $\pi(x) = \sum_{p \leq x} 1$: # of primes $\leq x$

THEOREM: $\lim\limits_{x \to \infty} \pi(x)/(x/\ln x) = 1$

- Conjectured by Legendre and Gauss
- Chebyshev: if the limit exists, then it is equal to 1
- Rosser and Schoenfeld:
  - $\pi(x) > \frac{x}{\ln x}\left(1 + \frac{1}{2\ln x}\right)$ when $x \geq 59$
  - $\pi(x) < \frac{x}{\ln x}\left(1 + \frac{3}{2\ln x}\right)$ when $x > 1$

NOTATION: $\mathbb{P}$- the set of all primes; $\mathbb{P}_n = \{p \in \mathbb{P}: 2^{n-1} \leq p < 2^n\}$.

THEOREM: $|\mathbb{P}_n| \geq \frac{2^n}{n \ln 2}\left(\frac{1}{2} + O\left(\frac{1}{n}\right)\right)$ when $n \to \infty$.

Number of n-bit Primes

EXAMPLE: The number of n-bit primes for $n \in \{10, \ldots, 25\}$.

n	$\lvert\mathbb{P}_n\rvert$	$2^{n-1}/n\ln 2$	n	$\lvert\mathbb{P}_n\rvert$	$2^{n-1}/n\ln 2$
10	75	73.8	18	10749	10505.4
11	137	134.3	19	20390	19904.9
12	255	246.2	20	38635	37819.4
13	464	454.6	21	73586	72036.9
14	872	844.2	22	140336	137525.0
15	1612	1575.8	23	268216	263091.4
16	3030	2954.6	24	513708	504258.5
17	5709	5561.7	25	985818	968176.3

Prime Number Generation

Basic Idea: randomly choose n-bit integers until a prime found.

- The number of n-bit integers is 2^{n-1}
- $|\mathbb{P}_n| \geq \frac{2^n}{n \ln 2}\left(\frac{1}{2} + O\left(\frac{1}{n}\right)\right)$ when $n \to \infty$
- The probability that a prime is chosen in every trial is equal to

$$\alpha_n = \frac{1}{n \ln 2}\left(1 + O\left(\frac{1}{n}\right)\right), n \to \infty$$

- In $\alpha_n^{-1} = \frac{n \ln 2}{1+O\left(\frac{1}{n}\right)} \leq 2n \ln 2$ trials, we get a prime.

Efficient Algorithms: An algorithm is considered as efficient if its (expected) running time is a polynomial in the bit length of its input. //a.k.a. (expected) polynomial-time algorithm

EXAMPLE: Choosing an n-bit prime can be done efficiently.

- The expected # of trials is $\leq 2n \ln 2$, a polynomial in n (input length)
- Determine if an n-bit integer is prime can be done efficiently