Some Crypto Math

Primes in Computer Security

- Exponentiation mod a prime is easy, but the reverse is hard
- Easy: Compute $y = g^x \bmod p$
- Hard: Given y, g and p: Solve for x ("discrete logarithm")

- Multiplying two primes is easy, but the reverse is hard
- Easy: Compute $n = pq$
- Hard: Given n: Solve for p and q ("factoring")

- These lead to cryptography that's easy to use, but hard to break

Euclidian Algorithm

The Euclidian Algorithm is a way of finding the gcd(a,b) without needing to factor a and b.

- Assume a>b
- First find q_1 and r_1 such as $a=b*q_1+r_1$
- Then find q_2 and r_2 such as $b = q_2*r_1+r_2$
- Find q's and r's using $r_{i-2}=q_i*r_{i-1}+r_i$ for i>2 until $r_i=0$.
- Then $\gcd(a,b)=r_{i-1}$

- The Euclidian algorithm runs in $O(\log^3(a))$; so it is very fast

Euclidian Algorithm

Example:

- Find gcd(1547, 560)
- 1547 = 2*560+427
- 560= 1*427 + 133
- 427 = 3*133 + 28
- 133 = 4*28 + 21
- 21 = 1*21 + 7
- 21 = 3*7 + 0

Thus gcd(1547,560) = 7

Extended Euclidian GCD Algorithm

- In several cryptographic algorithms you want to find an inverse a^{-1} such as $a*a^{-1}=1 \bmod n$
- We use the fact that
  - if d = gcd(a,b), where a>b, then there exists integer u and v such that d = u*a + v*b.
  - finding u and v can be done in $O(\log^3 a)$
- Then we use an extended Euclidian algorithm to find $a^{-1} \bmod n$ under the condition gcd(a, n) = 1

Let Inverse(a,n) = a^{-1}

Extended Euclidian GCD Algorithm

The algorithm: Inverse(a,n) is given by:

- $g_0=n$ $u_0=1$ $v_0=0$
- $g_1=a$ $u_1=0$ $v_1=1$
- let
  - $y = g_{i-1} \text{ div } g_i$
  - $g_{i+1} = g_{i-1} - y*g_i = g_{i-1} \bmod g_i$
  - $u_{i+1} = u_{i-1} - y*u_i$
  - $v_{i+1} = v_{i-1} - y*v_i$
- when $g_i=0$ then inverse(a,n) = v_{i-1}

Extended Euclidian GCD Algorithm

- Example: Find inverse of 3 mod 460

i	y	g	u	v
0	-	460	1	0
1	-	3	0	1
2	153	1	1	-153
3	3	0	-3	460

  - So, 3^{-1} mod 460 = -153 mod 460 = 307 mod 460

Euler phi-function

Euler φ function

$\varphi(n) = |\{0 \le b < n \mid \gcd(b,n) = 1\}|$, i.e. the number of positive integers b less than n that are relatively prime to n

How to compute $\varphi(n)$:

- If p is prime than $\varphi(p) = p-1$
- if gcd(m,n) = 1, then $\varphi(m \times n) = \varphi(m) \times \varphi(n)$,
- $\varphi(p^\alpha)=p^\alpha - p^{\alpha-1}$

Algorithms to find Inverses

Algorithms to find **Inverse(a,n)**

1. Search 1,...,n-1 until an a^{-1} is found with $a.a^{-1}$ mod n
2. if $\varphi(n)$ is known, then from Euler's Generalization
   - $a^{-1} = a^{\varphi(n)-1} \bmod n$
3. Otherwise use Extended Euclid's algorithm for inverse

Chinese Remainder Theorem (CRT)

- Motivation: According to D.Wells, the following problem was posed by Sun Tsu Suan-Ching (4th century AD):

There are certain things whose number is unknown. Repeatedly divided by 3, the remainder is 2; by 5 the remainder is 3; and by 7 the remainder is 2. What will be the number?

- Application
  - We want to solve a system of congruences to different moduli
  - Compute c^d mod n faster by computing mod p and mod q then combining results with the CRT

Chinese Remainder Theorem (CRT)

- The system is
  - $x = a_1 \bmod m_1$
  - $x = a_2 \bmod m_2$
  - ....
  - $x = a_r \bmod m_r$
- Assume that $\gcd(m_i, m_j) = 1$ for $i \neq j$.
  - Then the system has a unique solution modulo $M=m_1 m_2 \dots m_r$
  - In particular when $m_1, m_2, \dots, m_r$ are distinct primes

Chinese Remainder Theorem

The Chinese remainder theorem provides a way of solving an equation mod n, where n=p*q and p and q are prime, solving equations mod p and mod q

- Let $b_1 = q^{-1} \bmod p$ and $b_2 = p^{-1} \bmod q$
- If $a = a_1 b_1 q + a_2 b_2 p$ we have that
  - $a = a_1(b_1 q) + a_2(b_2 p) = a_1 \bmod p$
  - $a = a_1(b_1 q) + a_2(b_2 p) = a_2 \bmod q$
- So if I know a_1 and a_2 I know a

Example

- $p = 5$, $q = 7$, $n = 35$
- Weights: $b_1 = 3$ (i.e. $b_1 \times q \bmod p=1$), $b_2 = 3$
- Suppose <u>$x \bmod 5 = 2$</u> and <u>$x \bmod 7 = 1$</u>
- Then $x \bmod 35$

$$
\begin{aligned}
x \bmod 35 &= [(x \bmod p)\, b_1\, q + (x \bmod q)\, b_2\, p] \bmod 35 \\
&= [2 \times 3 \times 7 + 1 \times 3 \times 5] \bmod 35 \\
&= [42 + 15] \bmod 35 \\
&= 57 \bmod 35 \\
&= 22
\end{aligned}
$$

- Check: 22 mod 5 = 2, 22 mod 7 = 1

Complexity of ops on n-bit numbers

- Addition: O(n)
- Multiplication:
  - Schoolbook method: $O(n^2)$
  - Karatsuba-Ofman: $O(3n^{\log\_2(3)})=O(n^{1.585})$ (larger than 320–640 bits)
  - Schönhagen-Strassen: O(n log n log log n)
    (in practice for numbers larger than 2^{15} to 2^{17} bits)
- Mod multiplication: $O(n^2)$
- Mod exponentiation: avg 1.5n Mod multiplications

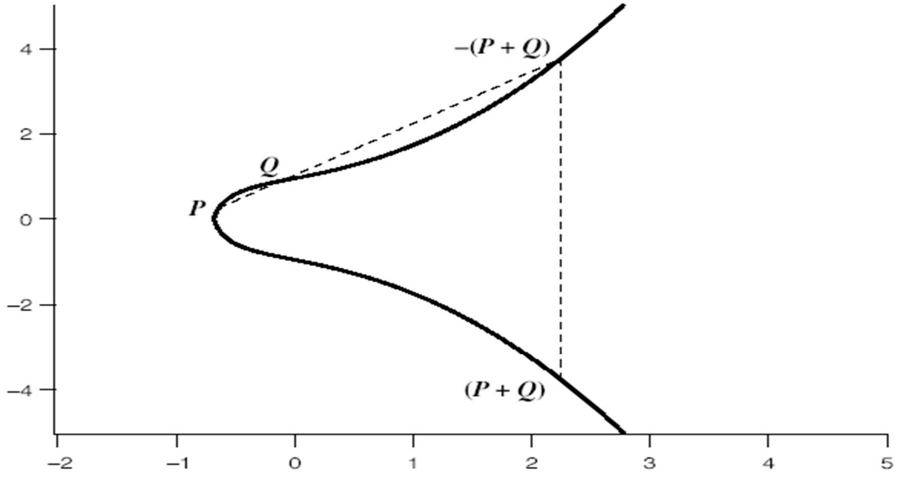
Elliptic Curve Cryptography

- majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller bit sizes

Real Elliptic Curves

- an elliptic curve is defined by an equation in two variables x & y, with coefficients
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define zero point O
- An addition operation for elliptic curves:
  - geometrically sum Q+R of two point Q and R is the reflection of intersection R

Real Elliptic Curve Example ("addition of points")



(b) $y^2 = x^3 + x + 1$

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - use integers modulo a prime
    - best in software
  - binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best in hardware
    - tricky: in choice and implementation patents

Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply (ECC repeated addition is analog of modulo exponentiation)
- need "hard" problem equiv to discrete log
  - `Q=kP`, where Q,P belong to a prime curve
  - is "easy" to compute Q given k,P
  - but "hard" to find k given Q,P
  - known as the elliptic curve logarithm problem
- Certicom example: $E_{23}(9,17)$
- U.S. National Security Agency: in its Suite B set of recommended algorithms ECC is included and allowed for protecting information classified up to top secret with 384-bit keys