Computational Number Theory Part III

Isomorphisms, Pairings, CRT, etc.

Isomorphisms

- Abelian groups G and H are isomorphic if they have same underlying algebraic structure
- Isomorphism just a different, but equivalent way to represent elements of a group

- Let $(G, \bullet)$ and $(H, *)$ be abelian groups. A function f: G—>H is said an isomorphism from G to H, if:
  - f is a bijective function (one-to-one mapping)
  - For all $g_1, g_2 \in G$, we have: $f(g_1 \bullet g_2) = f(g_1) * f(g_2)$
- If such an f exists, groups G, H are said to be isomorphic, i.e., $G \simeq H$

Isomorphisms

- If $G \simeq H$, and G is finite, H must be finite too
- $|G| = |H|$ too
- If $f: G \rightarrow H$ exists, $f^{-1}: H \rightarrow G$ exists too (either may/may not be efficiently computable)

- General idea: group homomorphisms
- Isomorphism one possible mapping (bijective)
- Others:
  - f is surjective (many-to-one): epimorphism
  - f is injective (one-to-one, but preserves distinctness): monomorphism

Direct Product

- Given two groups, $(G,\bullet)$, $(H,*)$, the direct product of G and H is defined as: $(G \times H, \oplus)$

- The elements of group $(G \times H, \oplus)$ are *ordered pairs* (g,h), where $g \in G$, and $h \in H$
- If $|G| = n$, $|H| = m$, $|G \times H| = mn$
- $(g,h) \oplus (g',h') = (g \bullet g', h * h')$

Is this Useful?

- Certainly

- Pairing-based crypto: entire area of research within public-key crypto
- Many hardness assumptions based on group mappings or “pairings”
- Lots of primitives/protocols built around these assumptions
- Math relatively more involved
- Efficiency? Not as efficient as, say, RSA, ElGamal, etc., but other advantages[1]…

1: Don’t need an established PKI, or standardized certificates, for one

Group Pairings

- Broadly speaking, 3 kinds of pairings

- Type I pairings: "Symmetric":
  - $e: G \times H \rightarrow G_T$; $G = H$
  - e is a *bilinear map*[1]

- Type II pairings: "Asymmetric", with efficiently computable isomorphism:
  - $e: G \times H \rightarrow G_T$; $G \neq H$, but ψ exists, such that $g = \psi(h)$; $g \in G$, $h \in H$
  - ψ is an *efficiently computable* isomorphism

1: Why e? Why not f? By convention. Why G_T? Again convention!

Group Pairings

- Type III pairings: "Asymmetric", with no efficiency computable isomorphism:
  - $e: G \times H \rightarrow G_T$; $G \neq H$, no *efficiently computable* ψ exists, such that $g=\psi(h)$; $g \in G$, $h \in H$

- Pairing between 2 groups: e is a *Bilinear map*
- Pairings between k groups: k-linear, or *multi-linear map*

Pairing Assumption (DLIN)

- Decision Linear Diffie-Hellman Assumption (DLIN)

- Let G, H be two groups, with G=H. The DLIN assumption holds in G and H if, given (u,v,h,u^a,v^b,h^c), it is computationally infeasible to determine if $c = a+b$, or $c \leftarrow Z_p$, where $|G|=|H|=p$

- $| \Pr[A(u,v,h,u^a,v^b,h^{a+b}) = 1 : u,v,h \leftarrow G;\ a,b \leftarrow Zp]$

  $- \Pr[A(u,v,h,u^a,v^b,\eta) = 1 : u,v,h,\eta \leftarrow G,\ a,b \leftarrow Zp]$

  $| \leq 1/2 + \mathrm{negl}(\varepsilon)$

Pairing Assumption (qSDH)

- Let G, H be groups or order p. Let g∈G, and h∈H be generators of G and H respectively. The q-SDH assumption holds in G and H, if, given the elements $(g, h, h^x, h^{x^2}, h^{x^3},\ldots,h^{x^q})$, it is computational infeasible for an algorithm* to output any pair $(c, g^{1/(x+c)})$, where $x,c \leftarrow Z_p$

- $| \Pr[A(g, h, h^x, h^{x^2}, h^{x^3},\ldots,h^{x^q}) = (c, g^{1/(x+c)})] | \geq \varepsilon$

- Defined over symmetric/asymmetric groups

*: Math assumptions are usually defined in terms of a general algorithm, not "adversary". Its only when we build a crypto protocol on top of the assumption, that adversary comes into picture

Pairing Assumptions

- Other pairing assumptions: Symmetric Diffie Hellman (SDH), subgroup assumptions, inner-product assumptions, and few more

- Boondoggle…? Nope
- Families of protocols built on pairing-based assumptions:
  - Identity-based encryption (IBE)
  - Attribute-based cryptosystems (ABE/ABS)
  - Non-interactive zero-knowledge proofs (NIZK)[1], and more…

1: There exist interactive, non-pairing-based "cut-and-choose" ZKPs too, but those are communication-intensive, and hence inefficient

Chinese Remainder Theorem

- Let $N = pq$, where $p,q > 1$, and p,q are co-prime. Then:

$$Z_N \simeq Z_p \times Z_q, \text{ and } Z_N^* \simeq Z_p^* \times Z_q^*$$

- Let f be a function that maps Z_N to $Z_p \times Z_q$
- f maps $x \in \{0,\ldots,N-1\}$ to pairs (x_p,x_q), where $x_p \in \{0,\ldots,p-1\}$ and $x_q \in \{0,\ldots,q-1\}$, defined as:

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

Chinese Remainder Theorem

- f is an isomorphism from Z_N to $Z_p \times Z_q$
- If f* is a function that maps Z_N^* to $Z_p^* \times Z_q^*$, then f* is also an isomorphism from Z_N^* to $Z_p^* \times Z_q^*$
- Modular exponentiation very computationally expensive operation; CRT can considerably speed up implementations

Chinese Remainder Theorem

- Example: consider group $Z_{15}^* = \{1,2,4,7,8,11,13,14\}$, $|Z_{15}^*| = 8$
- CRT stipulates $Z_{15}^* \simeq Z_5^* \times Z_3^*$

- Check:
  - $Z_5^* = \{1,2,3,4\}$, $Z_3^* = \{1,2\}$
  - Needs to be a one-to-one mapping between Z_{15}^* and $Z_5^* \times Z_3^*$
  - $1 \leftrightarrow (1,1)$, $2 \leftrightarrow (2,2)$, $4 \leftrightarrow (4,1)$, $7 \leftrightarrow (2,1)$, $8 \leftrightarrow (3,2)$, $11 \leftrightarrow (1,2)$, $13 \leftrightarrow (3,1)$, $14 \leftrightarrow (4,2)$

We leave verifying $Z_N \simeq Z_p \times Z_q$ with f(x) as a exercise

CRT Tricks

- Example 1: Compute 14•13 mod 15. Assume Z_{15}^* is a group

- We know 14↔(4,2), and 13↔(3,1)
- We also know $Z_{15}^* \simeq Z_5^* \times Z_3^*$
- (4,2) • (3,1) = ([4 • 3 mod 5], [2 • 1 mod 3])
- = (2,2)
- = 2, since 2↔(2,2)
- So, 14•15 mod 15 = 2

CRT Tricks

- Example 2: Compute 11^{53} mod 15. Assume Z_{15}^* is a group

- We know $11 \leftrightarrow (1,2)$, and $Z_{15}^* \simeq Z_5^* \times Z_3^*$
- = $([1^{53} \bmod 5],[(-1)^{53} \bmod 3])$ (since $2 \equiv (-1 \bmod 3)$)
- = $(1, [-1 \bmod 3])$
- = $(1,2)$
- $11 \leftrightarrow (1,2)$
- So, 11^{53} mod 15 = 11

CRT Tricks

- Example 3: Compute $[18^{25} \bmod 35]$. Assume Z_{35}^{*} is a group

- We know $18 \leftrightarrow (3,4)$, and $Z_{35}^{*} \simeq Z_5^{*} \times Z_7^{*}$
- $((3,4)^{25} \bmod 35) = ([3^{25} \bmod 5], [4^{25} \bmod 7])$
- $= ([3^{25 \bmod 4} \bmod 5], [4^{25 \bmod 6} \bmod 7])$ (refer to numTheory II slide 7)
- $= ([3], [4])$
- $18 \leftrightarrow (3,4)$
- So, $18^{25} \bmod 35 = 18$

In an implementation, this is much, much faster than the long way

Factoring

- Integer factoring:
- Given composite number N, such that N = pq, find primes p,q > 1

- Solution:
- Heuristic search (trial-and-error): $\forall\ p \in [2, .., \lfloor\sqrt{N}\rfloor]$, check if p divides N
- Complexity: $O(\sqrt{N}\ \mathrm{polylog}(N))$, but marginally better ones also exist
- No polynomial-time algorithm known[1]

1: When p,q are primes, that is. Of course, if one of them is even, it is easy to factor N in polynomial time

Prime Generation

- Ok, but how do we generate large primes?

- Bertrand's postulate: For any $n>1$, $\exists$ at least one prime p, s.t., $n < p < 2n$

- Miller-Rabin procedure for primality testing
  - Fairly simple process
  - Works over Z_n^* groups
  - Uses property that $|Z_n^*| = n-1$, if n is prime

FLT Primality Test

- Let Z_n^* be a group. If n is prime, $|Z_n^*| = n-1$
- So, for any $a \in Z_n^*$, $a^{n-1} \bmod n = 1$ (group property)

- Attempt 1: Fermat's little theorem

for i = 1 to t /*t reasonably big integer*/

Choose a random (but uniformly chosen) $a \in \{1, \ldots, n-1\}$

if $a^{n-1} \bmod n \neq 1$, return "n is composite"

return "n is prime"

FLT Test

- Sounds simplistic, does it work?

- Somewhat…
  - Let an a, such that $a^{n-1} \bmod n \neq 1$ be a *witness* (of being composite/non-prime)
  - Known result: i*f there exists a witness*, then 1/2 of elements in Z_n^* are witnesses that n is composite, for a given n
  - So, if n is composite, at least $|Z_n^*|/2$ witnesses exist
  - Probability that no witness found in t iterations is $1/2^t$

FLT Test

- Problem?

- There are infinitely many composite numbers that don't have *any* witnesses :-(
  - Carmichael numbers!*

- Attempt 2: (Weak) Miller Rabin test
  - Given n, compute r, u, such that $(n-1) = 2^r u$; where $r \geq 1$, u is odd, $n > 2$
  - In attempt 1, if $a^{2^r u} \bmod n \neq -1$, return "n is composite", for $a \in \{1,\ldots,n-1\}$

*: Satisfy certain primality tests, despite not being prime. First such number is 561

Miller-Rabin Test

- Attempt 3: Miller-Rabin test
  - Instead of checking if $a^{2^r u} \bmod n \neq -1$, for a single r, check the entire sequence from i = {1, …,r-1}, for n > 2, $(n-1) = 2^r u$
  - In attempt 2, if $((a^u \bmod n \neq 1)$ AND $(a^{2^1 u} \bmod n \neq -1)$, AND $(a^{2^2 u} \bmod n \neq -1)$, AND $(a^{2^3 u} \bmod n \neq -1)$, AND ….., $(a^{2^r u} \bmod n \neq -1))$, return "n is composite", for n > 2

- $a \in \{1,…,n-1\}$ is a *strong witness*, else *strong liar*

Miller-Rabin Test

- Composite integer, n is a *prime power*, if $n = p^r$, for some prime p, and $r \geq 1$

- Known result: If n is an odd number that is not a prime power, at least 1/2 of elements in Z_n^* are strong witnesses that n is composite

Miller-Rabin Test

- Final Attempt:

if n is even, return "n is composite"

if n is a prime power[1], return "n is composite"

find $r \geq 1$ and odd u, such that $(n-1) = 2^r u$

for i = 1 to t /*t reasonably big integer*/

Choose a random (but uniformly chosen) element $a \in \{1,\ldots,n-1\}$

if ($a^u \bmod n \neq 1$) and if ($a^{2^i u} \bmod n \neq -1$), $\forall$ $i \in \{1,\ldots,r-1\}$), return "n is composite"

return "n is prime"

1: Textbook gives slightly more general notion of perfect power; every prime power is also a perfect power

Factoring Assumption

- Factoring Experiment, $\text{Fact}_{A,\text{GenModulus}}(n)$:

GenModulus (1^n) —> (N,p,q), such that N=pq; p,q are n-bit primes

Poly-time algorithm A: A(N) —> p′,q′

If {p,q} = {p′,q′}, output 1, else 0

- Factoring is hard w.r.t. GenModulus, if, for all PPT algorithms A, there exists a negligible function, negl, s.t.:

$$\Pr[\text{Fact}_{A,\text{GenModulus}}(n) = 1] \leq \text{negl}(n)$$