AM 106/206: Applied Algebra **Madhu Sudan**[1]

Lecture Notes 6

September 20, 2017

1 Cyclic Groups

- Reading: Gallian Chapter 4.
- **Def:** A group G is *cyclic* if $G = \langle g \rangle$ for some $g \in G$. Such an element g is called a *generator* of G.
- **Examples:**
  1. $\mathbb{Z}$?
  2. $\mathbb{Z}_n$?
  3. S_3?
  4. $\mathbb{Z}_{12}^*$?
  5. $\mathbb{Z}_{13}^*$?
- **Fact:** for every prime p, $\mathbb{Z}_p^*$ is cyclic. More generally, $\mathbb{Z}_n^*$ is cyclic if and only if $n = 4$ or n is of the form p^k or $2p^k$ for an odd prime p and $k \in \mathbb{N}$.
  (We will not prove this fact but you may use it throughout the course.)
- **Thm 4.1 (Classification of cyclic groups):** Let $G = \langle g \rangle$ be a cyclic group.
  1. If g has infinite order, then $\ldots, g^{-2}, g^{-1}, g^0 = e, g^1, g^2, \ldots$ are all distinct (and $g^i \cdot g^j = g^{i+j}$).
  2. If g has finite order n, then:
     - $g^0, g^1, \ldots, g^{n-1}$ are all of the distinct elements of G.
     - For an arbitrary integer k, $g^k = g^{k \bmod n}$ (so $g^i \cdot g^j = g^{(i+j) \bmod n}$).
- **Example:** Arithmetic in $\mathbb{Z}_{13}^*$
- **Proof of Thm 4.1, Item 2:**

[1]These notes are copied mostly verbatim from the lecture notes from the Fall 2010 offering, authored by Prof. Salil Vadhan. I will attempt to update them, but apologies if some references to old dates and contents remain.

- **Corollary:** $|\langle g \rangle| = |g|$.
- **Thms 4.2–4.3 (Subgroups of Cyclic Groups):** Let $G = \langle g \rangle$ be a cyclic group of order n. Then:

  1. Every subgroup of G is cyclic.
  2. For every integer k, $\langle g^k \rangle = \langle g^{\gcd(k,n)} \rangle$.
  3. If $d|n$, then $\langle g^d \rangle$ is cyclic of order n/d.

  In particular, the subgroups of G are in one-to-one correspondence with the divisors d of n.

  Proof: in Gallian.
- **Example:** subgroups of $G = \langle g \rangle$ of order 12, $\mathbb{Z}_{12}$, and $\mathbb{Z}_{13}^*$.

- **Corollary:**

  1. If $G = \langle g \rangle$ is cyclic of order n, then g^k generates G iff $\gcd(k, n) = 1$.
  2. k generates $\mathbb{Z}_n$ iff $\gcd(k, n) = 1$.

2 Computation in Cyclic Groups

- Computations in a finite cyclic group $G = \langle g \rangle$ are easy *if* we can "access" the exponents.
- For what follows, let $G = \langle g \rangle$ a cyclic group of known order q with a known generator g in which group elements can be represented by bitstrings of length $n = O(\log_2 q)$, and where multiplication (given $a, b \in G$, compute $ab \in G$) and inversion (given $a \in G$, compute $a^{-1} \in G$) can be done in efficiently (in time poly(n)).
  - **Example:** $G = \mathbb{Z}_p^*$ for an n-bit prime p.
- **Exponentiation in G:** given $x \in \mathbb{Z}_q$, compute $g^x \in G$.
  - Can be done with $O(\log q) = O(n)$ multiplications in G. How?

  So going from an exponent x to the corresponding element of G is an "easy" problem. However, going from an element of G to the corresponding exponent x, as in the following problem, seems to be much harder.
- **Discrete Logarithm Problem in G:** given $a \in G$ (selected uniformly at random), compute the (unique) $x \in \mathbb{Z}_q$ such that $g^x = a$.
  - Naive algorithm: $O(q \log q) = O(n \cdot 2^n)$ multiplications in G.
  - PS3: $O(\sqrt{q} \log q) = O(2^{n/2} \cdot n)$ multiplications in G.

- Best known for $G = \mathbb{Z}_p^*$: time $2^{O(n^{1/3}\log^{2/3} n)}$.
- Best known for $G =$ "elliptic curve group": time $O(2^{n/2})$.

- $\Rightarrow$ exponentiation seems to be a "one-way function," easy in forward direction, hard in reverse direction.
  $\Rightarrow$ useful for cryptography (build algorithms that are easy to use but hard to break)

3 Cryptography from Cyclic Groups

One of the most basic goals in cryptography is to enable two parties (e.g. you and an online merchant) to communicate securely over an insecure communication channel (e.g. send your credit card number over the Internet without an eavesdropping hacker being able to learn it). Traditionally, it was always assumed that the two honest parties would need a means to securely share a "secret key" in advance, which they then would use to encrypt their messages. In 1976, Diffie and Hellman published a landmark paper, which showed how secure communication on an insecure channel could be done even between parties that hadn't shared any secrets in advance. Their construction relied on cyclic groups, and was based on the (conjectured) gap in complexity between exponentiation and discrete logarithms. Specifically, they gave the following protocol for establishing a shared secret key on an insecure communication channel.

- **Diffie–Hellman Key Exchange Protocol** for parties A (=Alice) and B (=Bob) to generate a shared secret over an insecure communication channel.
  1. A chooses random $x \in \mathbb{Z}_q$, computes $a = g^x$ and sends it to B.
  2. B chooses random $y \in \mathbb{Z}_q$, computes $b = g^y$ and sends it to A.
  3. Both parties compute the shared key $k = g^{xy} = b^x = a^y$.

- Security of Diffie–Hellman
  - Eavesdropping adversary E (=Eve) is given $a = g^x$ and $b = g^y$ and needs to compute g^{xy}. It is not obvious how to do this without computing discrete logarithms.
  - Usually we actually want the adversary to not be able to learn *anything* about the shared secret k; that is, it should "look like" a uniformly random element of $\mathbb{Z}_q$ to the adversary. This idea is formalized by the following assumption (which is a stronger assumption than just assuming that discrete logarithms are hard).
  - **Decisional Diffie–Hellman (DDH) Assumption:** No efficient (e.g. poly(n)-time) algorithm E can distinguish (g^x, g^y, g^{xy}) from (g^x, g^y, g^z) for uniformly random and independent $x, y, z \in \mathbb{Z}_q$. That is,

    $$|\Pr[E(g^x, g^y, g^{xy}) = 1] - \Pr[E(g^x, g^y, g^z) = 1]| \leq \varepsilon(n),$$

    for some "negligible" function $\varepsilon(n) \to 0$.
  - On PS3, you will show that the DDH false if $q = |G|$ has small factors, so we usually take $|G|$ prime. For example, we take G to be the subgroup of order q in $\mathbb{Z}_p^*$ where $p = 2q + 1$ and p, q are both large primes (e.g. around 1000 bits long).

We can also use the above ideas to construct a *public-key encryption scheme*. This replaces the key-exchange interaction above with A just generating a *public key* and *secret key*, placing the public key in a public directory, and keeping the secret key to herself. Now, when anyone (such as B) wants to send her a message, they can encrypt it using her public key. Decryption, however, will require the secret key, which only A possesses.

- **El Gamal Public-Key Encryption Scheme**
  1. A chooses random $x \in \mathbb{Z}_q$, publishes $a = g^x$ as her *public key*, and keeps x as her *secret key*.
  2. To send a message $m \in G$, B chooses random $y \in \mathbb{Z}_q$, sends $b = g^y$ and $c = m \cdot k = m \cdot g^{xy}$.
  3. After receiving (b, c), A can recover m.
- Security of El Gamal
  - If DDH is true, then encryptions of any two messages $m_0, m_1 \in G$ are indistinguishable to an efficient adversary E: An encryption of m_0 is $(g^x, g^y, m_0 \cdot g^{xy})$, which is indistinguishable from $(g^x, g^y, m_0 \cdot g^z)$, which has the same distribution as (g^x, g^y, g^w), where x, y, w, z are all uniformly random and independent elements of G. And similarly for m_1.
  - **Key fact:** Multiplying a fixed group element (m_0) by a uniformly random group element (g^z) gives a uniformly random group element (g^w).
  - No partial information about the message leaks! Even if the adversary knows you're going to send one of two messages (e.g. "buy" or "sell"), it cannot distinguish their encryptions.
- For more on cryptography, see Katz & Lindell "Introduction to Modern Cryptography" or take CS 220r (or MIT 6.875J).