Cryptography

Lecture 19

Announcements

- HW7 due 4/22/20

Agenda

- More Number Theory!

Chinese Remainder Theorem

Going from $(a, b) \in Z_p \times Z_q$ to $x \in Z_N$

Find the unique $x \bmod N$ such that

$$x \equiv a \bmod p$$
$$x \equiv b \bmod q$$

Recall since $\gcd(p, q) = 1$ we can write

$$Xp + Yq = 1$$

Note that

$$Xp \equiv 0 \bmod p$$
$$Xp \equiv 1 \bmod q$$

Whereas

$$Yq \equiv 1 \bmod p$$
$$Yq \equiv 0 \bmod p$$

Going from $(a, b) \in Z_p \times Z_q$ to $x \in Z_N$

Find the unique $x \; mod \; N$ such that

$$x \equiv a \; mod \; p$$
$$x \equiv b \; mod \; q$$

Claim:

$$b \cdot Xp + a \cdot Yq \equiv a \; mod \; p$$
$$b \cdot Xp + a \cdot Yq \equiv b \; mod \; q$$

Therefore, $x \equiv b \cdot Xp + a \cdot Yq \; mod \; N$

Modular Exponentiation

Modular Exponentiation

Is the following algorithm efficient (i.e. poly-time)?

ModExp(a, m, N) //computes $a^m \ mod\ N$

Set $temp := 1$

For $i = 1$ to m

Set $temp := (temp \cdot a) mod\ N$

return $temp$;

No—the run time is $O(m)$. m can be on the order of N. This means that the runtime is on the order of $O(N)$, while to be efficient it must be on the order of $O(\log N)$.

Modular Exponentiation

We can obtain an efficient algorithm via “repeated squaring.”

ModExp(a, m, N) //computes $a^m \; mod \; N$, where $m = m_{n-1}m_{n-2} \cdots m_1 m_0$ are the bits of m.

Set $s \coloneqq a$

Set $temp \coloneqq 1$

For $i = 0$ to $n - 1$

  If $m_i = 1$

    Set $temp \coloneqq (temp \cdot s) mod \; N$

  Set $s \coloneqq s^2 \; mod \; N$

return $temp$;

This is clearly efficient since the loop runs for n iterations, where $n = \log_2 m$.

Modular Exponentiation

Why does it work?

$$m = \sum_{i=0}^{n-1} m_i \cdot 2^i$$

Consider $a^m = a^{\sum_{i=0}^{n-1} m_i \cdot 2^i} = \prod_{i=0}^{n-1} a^{m_i \cdot 2^i}$.

In the efficient algorithm:

s values are precomputations of a^{2^i}, for $i = 0\ to\ n - 1$ (this is the "repeated squaring" part since $a^{2^i} = (a^{2^{i-1}})^2$).

If $m_i = 1$, we multiply in the corresponding s-value.

If $m_i = 0$, then $a^{m_i \cdot 2^i} = a^0 = 1$ and so we skip the multiplication step.

Getting Back to Z_p^*

Group $Z_p^* = \{1, \ldots, p-1\}$ operation: multiplication modulo p.

Order of a finite group is the number of elements in the group.

Order of Z_p^* is $p-1$.

Fermat's Little Theorem

Theorem: For prime p, integer a:

$$a^p \equiv a \; mod \; p.$$

Corollary: For prime p and a such that $(a, p) = 1$:

$$a^{p-1} \equiv 1 \; mod \; p$$

Generalized Theorem

Theorem: Let G be a finite group with $m = |G|$, the order of the group. Then for any element $g \in G, g^m = 1$.

Corollary of Fermat's Little Theorem is a special case of the above when G is the multiplicative group $Z^*{}_p$ and p is prime.

Multiplicative Groups Mod N

- What about multiplicative groups modulo N, where N is composite?
- Which numbers $\{1, \ldots, N-1\}$ have multiplicative inverses $mod\ N$?
  - a such that $\gcd(a, N) = 1$ has multiplicative inverse by Extended Euclidean Algorithm.
  - a such that $\gcd(a, N) > 1$ does not, since $\gcd(a, N)$ is the smallest positive integer that can be written in the form $Xa + YN$ for integer X, Y.
- Define $Z_N^* \coloneqq \{a \in \{1, \ldots, N-1\} |\ \gcd(a, N) = 1\}$.
- Z_N^* is an abelian, multiplicative group.
  - Why does closure hold?

Order of Multiplicative Groups Mod N

- What is the order of Z_N^*?
- This has a name. The order of Z_N^* is the quantity $\phi(N)$, where ϕ is known as the Euler totient function or Euler phi function.
- Assume $N = p \cdot q$, where p, q are distinct primes.
  - $\phi(N) = N - p - q + 1 = p \cdot q - p - 1 + 1 = (p-1)(q-1)$.
  - Why?

Order of Multiplicative Groups Mod N

General Formula:

Theorem: Let $N = \prod_i p_i^{e_i}$ where the $\{p_i\}$ are distinct primes and $e_i \geq 1$. Then

$$\phi(N) = \prod_i p_i^{e_i-1}\,(p_i - 1)\,.$$

Another Special Case of Generalized Theorem

Corollary of generalized theorem:

For a such that $\gcd(a, N) = 1$:

$$a^{\phi(N)} \equiv 1 \; mod \; N.$$

Another Useful Theorem

Theorem: Let G be a finite group with $m = |G| > 1$. Then for any $g \in G$ and any integer x, we have

$$g^x = g^{x \bmod m}.$$

Proof: We write $x = a \cdot m + b$, where a is an integer and $b \equiv x \bmod m$.

- $g^x = g^{a \cdot m + b} = (g^m)^a \cdot g^b$
- By "generalized theorem" we have that
$$(g^m)^a \cdot g^b = 1^a \cdot g^b = g^b = g^{x \bmod m}.$$

An Example:

Compute $3^{25} \mod 35$ by hand.

$$\phi(35) = \phi(5 \cdot 7) = (5-1)(7-1) = 24$$
$$3^{25} \equiv 3^{25 \mod 24} \mod 35 \equiv 3^{1} \mod 35$$
$$\equiv 3 \mod 35.$$

Toolbox for Cryptographic Multiplicative Groups

Can be done efficiently	No efficient algorithm believed to exist
Modular multiplication	Factoring
Finding multiplicative inverses (extended Euclidean algorithm)	RSA problem
Modular exponentiation (via repeated squaring)	Discrete logarithm problem
	Diffie Hellman problems

We have seen the efficient algorithms in the left column.
We will now start talking about the "hard problems" in the right column.