Cryptography

Lecture 20
Announcements

• HW7 due on Monday, 4/22
• Sign up for EC
• Instructor OH will be held tomorrow (Thursday) 9-10am instead of Friday.
Agenda

• More Number Theory!
• Hard Problems
Multiplicative Groups Mod N

• What about multiplicative groups modulo \( N \), where \( N \) is composite?

• Which numbers \( \{1, \ldots, N - 1\} \) have multiplicative inverses \( \text{mod } N \)?
  
  – \( a \) 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 \( Z_N^* := \{a \in \{1, \ldots, N - 1\} \mid \gcd(a, N) = 1\} \).

• \( Z_N^* \) is an abelian, multiplicative group.
  
  – Why does closure hold?
Order of Multiplicative Groups Mod N

• What is the order of $\mathbb{Z}_N^*$?
• This has a name. The order of $\mathbb{Z}_N^*$ is the quantity $\phi(N)$, where $\phi$ 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

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

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

- $g^x = g^{a \cdot m + b} = (g^m)^a \cdot g^b$
- By “generalized theorem” we have that
  $$(g^m)^a \cdot g^b = 1^a \cdot g^b = g^b = g^{x \mod m}.$$
An Example:

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

\[
\phi(35) = \phi(5 \cdot 7) = (5 - 1)(7 - 1) = 24
\]
\[
3^{25} \equiv 3^{25 \mod 24} \mod 35 \equiv 3^1 \mod 35
\]
\[
\equiv 3 \mod 35.
\]
Background for RSA

Recall the fact that

\[ a^m \equiv a^{m \mod \phi(N)} \mod N. \]

For \( e \in \mathbb{Z}_{\phi(N)}^* \), let \( f_e : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^* \) be defined as \( f_e(x) := x^e \mod N \).

**Theorem:** \( f_e(x) \) is a permutation.

**Proof:** To prove the theorem, we show that \( f_e(x) \) is invertible.

Let \( d \) be the multiplicative inverse of \( e \mod \phi(N) \).

Then for \( y \in \mathbb{Z}_N^* \), \( f_d(y) := y^d \mod N \) is the inverse of \( f_e \).

To see this, we show that \( f_d(f_e(x)) = x \).

\[
f_d(f_e(x)) = (x^e)^d \mod N = x^{e \cdot d} \mod N = x^{e \cdot d \mod \phi(N)} \mod N = x \mod N.
\]

**Note:** Given \( d \), it is easy to compute the inverse of \( f_e \).

However, we saw in the homework that given only \( e, N \), it is hard to find \( d \), since finding \( d \) implies that we can factor \( N = p \cdot q \).

This will be important for cryptographic applications.
Toolbox for Cryptographic Multiplicative Groups

<table>
<thead>
<tr>
<th>Can be done efficiently</th>
<th>No efficient algorithm believed to exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular multiplication</td>
<td>Factoring</td>
</tr>
<tr>
<td>Finding multiplicative inverses (extended Euclidean algorithm)</td>
<td>RSA problem</td>
</tr>
<tr>
<td>Modular exponentiation (via repeated squaring)</td>
<td>Discrete logarithm problem</td>
</tr>
<tr>
<td></td>
<td>Diffie Hellman problems</td>
</tr>
</tbody>
</table>

We have seen the efficient algorithms in the left column. We will now start talking about the “hard problems” in the right column.
The Factoring Assumption

The factoring experiment $Factor_{A,Gen}(n)$:
1. Run $Gen(1^n)$ to obtain $(N, p, q)$, where $p, q$ are random primes of length $n$ bits and $N = p \cdot q$.
2. $A$ is given $N$, and outputs $p', q' > 1$.
3. The output of the experiment is defined to be 1 if $p' \cdot q' = N$, and 0 otherwise.

Definition: Factoring is hard relative to $Gen$ if for all ppt algorithms $A$ there exists a negligible function $neg$ such that

$$\Pr[Factor_{A,Gen}(n) = 1] \leq neg(n).$$
How does $Gen$ work?

1. Pick random $n$-bit numbers $p, q$
2. Check if they are prime
3. If yes, return $(N, p, q)$. If not, go back to step 1.

Why does this work?

- **Prime number theorem:** Primes are dense!
  - A random $n$-bit number is a prime with non-negligible probability.
  - *Bertrand’s postulate:* For any $n > 1$, the fraction of $n$-bit integers that are prime is at least $1/3n$.

- **Can efficiently test whether a number is prime or composite:**
  - If $p$ is prime, then the Miller-Rabin test always outputs “prime.” If $p$ is composite, the algorithm outputs “composite” except with negligible probability.
Miller-Rabin Primality Test

ALGORITHM 8.44
The Miller-Rabin primality test

Input: Integer \( N > 2 \) and parameter \( 1^t \)
Output: A decision as to whether \( N \) is prime or composite

if \( N \) is even, return “composite”
if \( N \) is a perfect power, return “composite”
compute \( r \geq 1 \) and \( u \) odd such that \( N - 1 = 2^r u \)
for \( j = 1 \) to \( t \):
    \( a \leftarrow \{1, \ldots, N - 1\} \)
    if \( a^u \not\equiv \pm 1 \mod N \) and \( a^{2^i u} \not\equiv -1 \mod N \) for \( i \in \{1, \ldots, r - 1\} \)
        return “composite”
return “prime”

Why does it work?
First, note that \( a^{2^i u} = \sqrt{a^{2^i u + 1}} \), and that if \( p \) is prime then \( \sqrt{1} \mod p \equiv \pm 1 \).

• If \( N \) is prime: By Fermat’s Little Theorem, \( a^{N-1} \equiv a^{2^r u} \equiv 1 \mod N \).
  • Case 1: One of \( a^{2^i u} \equiv -1 \mod N \).
  • Case 2: None of \( a^{2^i u} \equiv -1 \mod N \). Then by the facts above, all of \( a^{2^i u} \equiv 1 \mod N \). In particular, \( a^{2^i u} \equiv 1 \mod N \). So by facts, \( a^u \equiv \sqrt{a^{2^i u}} \equiv \pm 1 \mod N \).

• If \( N \) is composite: At least half of \( a \in \mathbb{Z}_N^* \) will satisfy \( a^u \not\equiv \pm 1 \mod N \) and \( a^{2^i u} \not\equiv -1 \mod N \) for \( i \in \{1, \ldots, r - 1\} \).
The RSA Assumption

The RSA experiment $RSA - inv_{A,Gen}(n)$:
1. Run $Gen(1^n)$ to obtain $(N, e, d)$, where $\gcd(e, \phi(N)) = 1$ and $e \cdot d \equiv 1 \mod \phi(N)$.
2. Choose a uniform $y \in Z^*_N$.
3. $A$ is given $(N, e, y)$, and outputs $x \in Z^*_N$.
4. The output of the experiment is defined to be 1 if $x^e = y \mod N$, and 0 otherwise.

Definition: The RSA problem is hard relative to $Gen$ if for all ppt algorithms $A$ there exists a negligible function $neg$ such that

$$\Pr[RSA - inv_{A,Gen}(n) = 1] \leq neg(n).$$
Relationship between RSA and Factoring

Known:
- If an attacker can break factoring, then an attacker can break RSA.
  - Given $p, q$ such that $p \cdot q = N$, can find $\phi(N)$ and $d$, the multiplicative inverse of $e \mod \phi(N)$.
- If an attacker can find $\phi(N)$, can break factoring.
- If an attacker can find $d$ such that $e \cdot d \equiv 1 \mod \phi(N)$, can break factoring.

Not Known:
- Can every efficient attacker who breaks RSA also break factoring?

Due to the above, we have that the RSA assumption is a stronger assumption than the factoring assumption.