Introduction to Cryptology

Lecture 20

Announcements

- HW9 due today
- HW10 posted, due on Thursday 4/30
- HW7, HW8 grades are now up on Canvas.

Agenda

- More Number Theory!
  - Our focus today will be on computational complexity: Which problems in multiplicative groups are “easy” and which are “hard”?

Loose Ends from Last Time

Recall that we saw last time that

$$a^m \equiv a^{m \bmod \phi(N)} \bmod N.$$

For $e \in Z^*{}_N$, let $f_e : Z_N^* \to Z_N^*$ be defined as $f_e(x) \coloneqq x^e \bmod N$.

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 $e \bmod \phi(N)$.
Then for $y \in Z_N^*$, $f_d(y) \coloneqq y^d \bmod N$ is the inverse of f_e.

To see this, we show that $f_d(f_e(x)) = x$.
$f_d(f_e(x)) = (x^e)^d \bmod N = x^{e \cdot d} \bmod N = x^{e \cdot d \bmod \phi(N)} \bmod N = x^1 \bmod N = x \bmod N$.

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.

Modular Exponentiation

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

```
ModExp($a, m, N$) //computes $a^m \; mod \; N$
    Set $temp \coloneqq 1$
    For $i = 1$ to $m$
        Set $temp \coloneqq (temp \cdot a) mod \; N$
    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.

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.

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

$$\Pr\left[Factor_{A,Gen}(n) = 1\right] \leq neg(n).$$

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.

The RSA Assumption

The RSA experiment $RSA - inv_{A,Gen}(n)$:

1. Run $Gen(1^n)$ to obtain (N, e, d), where $\gcd\big(e, \phi(N)\big) = 1$ 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

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

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.

Cyclic Groups

For a finite group G of order m and $g \in G$, consider:

$$\langle g \rangle = \{g^0, g^1, \ldots, g^{m-1}\}$$

$\langle g \rangle$ always forms a cyclic subgroup of G.

However, it is possible that there are repeats in the above list.

Thus $\langle g \rangle$ may be a subgroup of order smaller than m.

If $\langle g \rangle = G$, then we say that G is a cyclic group and that g is a generator of G.

Examples

Consider $Z^*{}_{13}$:

2 is a generator of $Z^*{}_{13}$:

2^0	1
2^1	2
2^2	4
2^3	8
2^4	$16 \rightarrow 3$
2^5	6
2^6	12
2^7	$24 \rightarrow 11$
2^8	$22 \rightarrow 9$
2^9	$18 \rightarrow 5$
2^{10}	10
2^{11}	$20 \rightarrow 7$
2^{12}	$14 \rightarrow 1$

3 is not a generator of $Z^*{}_{13}$:

3^0	1
3^1	3
3^2	9
3^3	$27 \rightarrow 1$
3^4	3
3^5	9
3^6	$27 \rightarrow 1$
3^7	3
3^8	9
3^9	$27 \rightarrow 1$
3^{10}	3
3^{11}	9
3^{12}	$27 \rightarrow 1$

Definitions and Theorems

Definition: Let G be a finite group and $g \in G$. The order of g is the smallest positive integer i such that $g^i = 1$.

> Ex: Consider Z_{13}^*. The order of 2 is 12. The order of 3 is 3.

Proposition 1: Let G be a finite group and $g \in G$ an element of order i. Then for any integer x, we have $g^x = g^{x \bmod i}$.

Proposition 2: Let G be a finite group and $g \in G$ an element of order i. Then $g^x = g^y$ iff $x \equiv y \bmod i$.

More Theorems

Proposition 3: Let G be a finite group of order m and $g \in G$ an element of order i. Then $i\,|m$.

Proof:

- We know by the generalized theorem of last class that $g^m = 1 = g^0$.
- By Proposition 1, we have that $g^m = g^{m \bmod i} = g^0$.
- By the $\leftarrow$ direction of Proposition 2, we have that $0 \equiv m \bmod i$.
- By definition of modulus, this means that $i|m$.

Corollary: if G is a group of prime order p, then G is cyclic and all elements of G except the identity are generators of G.

Why does this follow from Proposition 3?

Theorem: If p is prime then ${Z^*}_p$ is a cyclic group of order $p - 1$.

Prime-Order Cyclic Groups

Consider ${Z^*}_p$, where p is a strong prime.

- Strong prime: $p = 2q + 1$, where q is also prime.
- Recall that ${Z^*}_p$ is a cyclic group of order $p - 1 = 2q$.

The subgroup of quadratic residues in ${Z^*}_p$ is a cyclic group of prime order q.

Example of Prime-Order Cyclic Group

Consider Z^*_{11}.

Note that 11 is a strong prime, since $11 = 2 \cdot 5 + 1$.

$g = 2$ is a generator of Z^*_{11}:

2^0	1
2^1	2
2^2	4
2^3	8
2^4	$16 \to 5$
2^5	10
2^6	$20 \to 9$
2^7	$18 \to 7$
2^8	$14 \to 3$
2^9	6

The even powers of g are the "quadratic residues" (i.e. the perfect squares). Exactly half the elements of Z^*_p are quadratic residues.

Note that the even powers of g form a cyclic subgroup of order $\frac{p-1}{2} = q$.

Verify:

- closure (Multiplication translates into addition in the exponent. Addition of two even numbers mod $p - 2$ gives an even number mod $p - 1$, since for prime $p > 3$, $p - 1$ is even.)
- Cyclic –any element is a generator. E.g. it is easy to see that all even powers of g can be generated by g^2.