Chapter 5: RSA and Factorization

Math 495, Fall 2008

Hope College

October 20, 2008

RSA

- Let p and q be distinct (odd) primes. Let $n = pq$.
- We have $\phi(n) = (p-1)(q-1)$.
- $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$.
- $\mathcal{K} = \{(n, p, q, a, b) : ab \equiv 1 \pmod{\phi(n)}\}$.
- For $x \in \mathcal{P}$ and $y \in \mathcal{C}$, define

$$e_K(x) = x^b \mod n,$$

  and

$$d_K(y) = y^a \mod n.$$

- Public key: n and b.
- Private information: p, q, a.

RSA

- $e_K(x) = x^b \mod n$ where $d_K(y) = y^a \mod n$, $ab \equiv 1 \pmod{\phi(n)}$.
- We need to show that decryption "works,", i.e. that for all x, $d_K(e_K(x)) = x$. This amounts to showing that

$$(x^b)^a \equiv x \pmod{n} \qquad \text{for all } x \in \mathbb{Z}_n.$$

- If $x \in \mathbb{Z}_n^*$, then, mod n,

$$(x^b)^a \equiv x^{ab} \equiv x^{\phi(n)t+1} \equiv (x^{\phi(n)})^t x \equiv 1^t x \equiv x.$$

- If $x \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$ and $x \not\equiv 0$, then x has either p or q, but not both, as a factor. Suppose $x = p^i r$, where r is $p \nmid r$ and $q \nmid r$. Then, mod n,

$$((p^i r)^b)^a \equiv (p^i r)^{ab} \equiv p^{iab} r^{ab} \equiv p^{i(\phi(n)t+1)} r \equiv p^{i(p-1)(q-1)t} p^i r \equiv p^i r.$$

- RSA is believed secure for large primes p and q.
- $e_K(x) = x^b \mod n$ is believed to be a one-way function.
- The trapdoor is the factorization of n as pq.
- If someone knows p and q, they can compute $\phi(n) = (p-1)(q-1)$, and thereby compute a using the extended Euclidean algorithm.

Example of RSA

- Suppose $n = 98069$ and $b = 36119$.
- If the plaintext is $x = 76111$, then

$$e_K(x) = 76111^{36119} \mod 98069 = 91332.$$

- With additional information $n = 281 \cdot 349$, Bob can compute $\phi(n) = 280 \cdot 348 = 97440$, and then compute

$$36119^{-1} \mod 97440 = 839.$$

Then

$$d_K(91332) = 91332^{839} \mod 98069 = 76111.$$

Implementation

- The primes p and q must be chosen large enough so that factoring n is computationally infeasible. For safety, p and q are typically primes that require 512 bits to represent them in binary. We will discuss how to find large primes and test their primality.
- Let n be a k-bit integer. RSA requires modular addition and subtraction mod n ($O(k)$), modular multiplication mod n ($O(k^2)$), and modular inversion mod n (($O(k^3)$)).
- Computing $x^c \mod n$ can be done using $c - 1$ modular multiplications, but this is very inefficient if c is large.
- Instead, we use the SQUARE AND MULTIPLY ALGORITHM, which runs in time $O(k^2 \log c)$.

Repeated Squaring

- The implementation of repeated squaring to compute $x^c \mod n$ is discussed in Algorithm 5.5 of the book.
- Intuitively, we express c in binary as $c_{\ell-1}c_{\ell-2}\cdots c_1c_0$, then compute $x^c \mod n$ by computing

$$x^{c_0}(x^{c_1}(x^{c_2}(\cdots(x^{c_{\ell-1}})^2\cdots)^2)^2)^2.$$

- For example, to compute $3^{57} \mod 7$, we write $57 = 111001_2$. Then

$$3^{57} = 3^{32}3^{16}3^{8}3^{1} = 3(((3(3(3)^2)^2)^2)^2)^2.$$

From this, we can see that $3^{57} \mod 7 = 6$.

RSA Implementation and Parameter Generation

- Choose two large primes p and q. Algorithms for doing this will be discussed in the next section.
- Set $n = pq$ and $\phi(n) = (p-1)(q-1)$. This can be done in time $O((\log n)^2)$.
- Choose a random b with $\gcd(b, \phi(n)) = 1$, and compute $a = b^{-1} \pmod{\phi(n)}$. This can be done in time $O((\log n)^2)$ using the EXTENDED EUCLIDEAN ALGORITHM.
- RSA encryption and decryption using the SQUARE AND MULTIPLY ALGORITHM each take time $O((\log n)^3)$.