Computer Science 418

Efficiency and Security of RSA and ElGamal, Semantic Security

Mike Jacobson

Department of Computer Science
University of Calgary

Week 10

Outline

Efficiency of RSA

Set-up (need only be done once):

- Prime generation uses a pseudo-random number generator (PRNG), followed by a probable primality test (like the Fermat test, more in PMAT 529).
- Generating e again requires a PRNG and one gcd calculation (EA) – or just pick you favourite e.
- Computing n and $\phi(n)$ is negligible.
- Computing d requires finding a modular inverse (EEA)

Encryption and Decryption: modular exponentiation (like Diffie-Hellman).

Security of RSA

Resides in the presumed difficulty of the *Integer Factorization Problem:*

- Given an integer N, find a non-trivial factor of N.

Attacks on RSA

The following approaches break RSA:

Factoring n
$\Downarrow$ $\phi(n) = (p-1)(q-1)$ $\Uparrow$ Assignment 4
Finding $\phi(n)$
$\Downarrow$ Proceed as designer $\Uparrow$ See note below
Finding the private key d

Note 1

There is an efficient algorithm that given any multiple of $\phi(n)$ finds $\phi(n)$ with high probability. Note that $ed - 1$ is such a multiple.

Attacks on RSA, cont.

All three approaches (prev. slide) are computationally equivalent:

- if one can be achieved, any of the other two one can be achieved with very little computational overhead.
- i.e., there are *three* trapdoors here: d, $\phi(n)$, and $\{p, q\}$

There is no proof that RSA is secure!

- no proof that factoring is hard
- not proven that other methods to compute M given C, e, n do not exist, which do not rely on factoring (i.e., not known whether breaking RSA is *equivalent* to factoring n)

Nevertheless, we need to design RSA systems such that $n = pq$ cannot be factored easily.

Factoring Record

The fastest known factoring algorithm is again the Number Field Sieve (slightly different from the DLP NFS, but invented first). Run time:

$$\exp\left(c(\log n)^{1/3}(\log\log n)^{2/3}\right) = n^{c(\log n/\log\log n)^{2/3}}$$

with

$$c = \sqrt[3]{\frac{64}{9}} = 1.92\ldots$$

Current RSA modulus factoring record: RSA200 (200 digits, 663 bits), Bahr, Boehm, Franke and Kleinjung, May 9, 2005.

Choice of RSA Parameters

Requirements for p and q:

1. Probable primes with high probability (say 2^{-100}) — use a good probabilistic primality test.
2. Large: at least $2^{1536} \approx 10^{463}$ (so n is 3072 bits)
3. Not too close together; $|p - q| > 2^{128}$ for $p, q \approx 2^{1536}$
4. $p-1$, $q-1$, $p+1$, $q+1$ must all have a large prime factor (see p. 150 of the *Handbook of Applied Cryptography*). Eg. pick $p = 2p' + 1$ to be a Sophie Germain prime so that $(p+1)/4 = (p'+1)/2$ is prime or has a large prime factor; same for q.
5. p/q should not be near the ratio of two small (relatively prime) integers a/b (say $a, b \le 100$).

Choice of RSA Parameters, cont.

Requirement for e:

- For efficiency reasons, e is often chosen small; a popular choice is $e = 2^{16} + 1 = 65537$ (great for binary exponentiation, only two '1' bits).
- Beware of really small e for some applications; see Assignment 4.
- In practice, can use $e = 3$, but *only when* RSA is used in conjunction with a secure padding mechanism (eg. OAEP — next week!)

Requirement for d:

- $d > n^{0.292}$ (Boneh & Durfee 2000).

Advantages of RSA

Advantages:

1. Seems to be secure.
2. Key size is "relatively" small — two 463 digit numbers — although other PKC's have smaller keys (eg. elliptic curve systems).
3. No message expansion — ciphertexts and plaintexts have the same length.
4. Can be used as a signature scheme (covered later).

Disadvantages of RSA

Disadvantages:

1. Very slow compared with DES, AES, and other symmetric key cryptosystems. Decryption is also slower than elliptic curve based systems.
2. Finding keys is fairly expensive.
3. Security is unproven
4. "Textbook" version (what we've been discussing!) leaks information and is vulnerable to active attacks (later).

Probabilistic Encryption

One disadvantage of deterministic PKCs is that identical messages always encrypt to the same ciphertext (like block ciphers in ECB mode).

- particularly problematic if the message space is small (*e.g.* electronic yes/no vote)

Probabilistic or *randomized encryption* utilizes randomness to attain a provable, stronger level of security.

As a result, every message can have many possible encryptions, so a small message space is no longer a problem.

- leads to the notion of *semantic* security.

The ElGamal PKC

Randomized, security based on DLP (alternative to RSA which was based on IFP)

Set-up: the designer produces her public and private keys as follows:

1. Selects a large prime p and a primitive root g of p
2. Computes $y = g^x \pmod{p}$ where $0 < x < p - 1$.

Public key: $\{p, g, y\}$
Private key: $\{x\}$

ElGamal Encryption

Messages for the designer are integers M, $0 < M < p$ (so $M \in \mathbb{Z}_p^*$).

To send M encrypted, proceed as follows:

1. Select a random $k \in \mathbb{Z}$, $0 < k < p$.
2. Compute and send (C_1, C_2) where

$$\begin{aligned} C_1 &\equiv g^k \pmod{p}, \quad 0 < C_1 < p, \\ C_2 &\equiv My^k \pmod{p}, \quad 0 < C_2 < p \ . \end{aligned}$$

ElGamal Decryption

To decrypt (C_1, C_2), the designer computes

$$\begin{aligned} C_2 C_1^{p-1-x} &\equiv (My^k)(C_1^{p-1-x}) \\ &\equiv (Mg^{xk})(g^{k(p-1-x)}) \\ &\equiv Mg^{xk+k(p-1)-kx} \\ &\equiv M(g^{p-1})^k \\ &\equiv M \pmod{p} \ . \end{aligned}$$

Think of C_1 as a "clue" that can be used to remove the "mask" y^k in C_2, thus "unmasking" the encrypted message M.

Summary of ElGamal

As with DH key establishment, the security of this system relies on the presumed difficulty of the DLP, but it is unknown whether there are other ways of breaking ElGamal.

Disadvantages:

- Message expansion by a factor of 2 (ciphertext is twice as long as the plaintext).
- Twice as much computational work for encrypting as RSA:
  - two exponentiations (and one multiplication), as opposed to one exponentiation only for RSA.
- A new random number k must be generated for each message.

Advantages: different security assumption, works in other settings (eg. elliptic curves)

Polynomial Security

Definition 1 (Polynomial security, IND-CPA security)

A PKC is said to be *polynomially secure* or *IND-CPA secure* if no passive adversary can in expected polynomial time select two plaintexts M_1 and M_2 and then correctly distinguish between encryptions of M_1 and M_2 with probability significantly greater than $1/2$.

IND-CPA: indistinguishability under chosen plaintext attacks.

Semantic Security

Definition 2 (Semantic security)

A PKC is said to be *semantically secure* if for all probability distributions over the message space, anything that can be computed by a passive adversary in expected polynomial time about the plaintext given the ciphertext can also be computed in expected polynomial time without the ciphertext.

Intuitively, semantic security is a weaker version of perfect security

- an adversary with polynomially-bounded computational resources (as opposed to infinite resources in perfect security) can learn nothing about the plaintext from the ciphertext.

Equivalance

Theorem 1

A PKC is semantically secure if and only if it is polynomially secure.

Although El Gamal is randomized, it is *not* semantically secure as presented here (next week).

We will soon look at a PKC that is semantically secure assuming that a certain number theoretic problem (not DLP or IFP) is hard. But first, we need a bit more number theory.