CPSC/PMAT 669

Public-Key Cryptography and RSA

Mike Jacobson

Department of Computer Science
University of Calgary

Topic 5

Outline

Public-Key Cryptography

Whitfield Diffe and Martin Hellman, "New Directions in Cryptography", 1976.

- Note that Diffie and Hellman did not describe a specific means of *implementing* a public-key cryptosystem.
- They merely described how one could be used to achieve security, authentication, (and indirectly, integrity and non-repudiation).

Also secretly discovered in 1970 as "non-secret encryption" by Clifford Cocks and James H. Ellis of CESG (Communications-Electronics Security Group, part of the the UK Government's Government Communications Headquarters(GCHQ))

- disclosed in 1987; see `http://jya.com/ellisdoc.htm`.

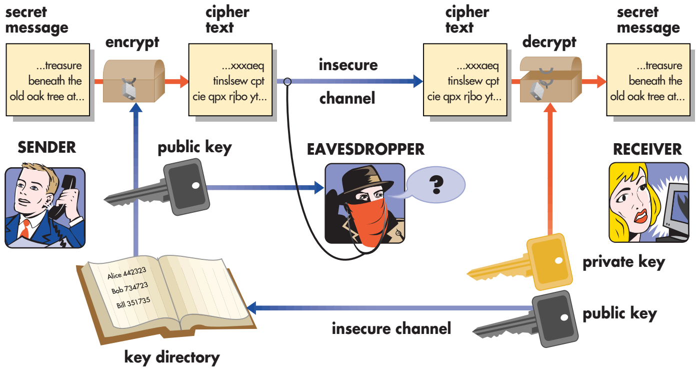
Idea of Public-Key Cryptography

Every user has *two* keys

- encryption key is public (so everyone can encrypt messages)
- decryption key is only known to the receiver

Deducing the decryption key from the encryption key should be computationally infeasible.

Diagram of a Public-Key Cryptosystem



Trap-door One-Way Functions

Definition 1 (Trap-door one-way function)

A function f that satisfies the following properties:

1. *Ease of Computation:* $f(x)$ is easy to compute for any x.
2. *Computation Resistance with Trap-door:* Given $y = f(x)$ it is computationally infeasible to determine x *unless* certain special information used in the design of f is known.
   - When this *trap-door* k is known, there exists a function g which is easy to compute such that $x = g(k, y)$.

Key to designing public-key cryptosystems: decryption key acts as a trap door for the encryption function.

Public-Key Cryptosystem

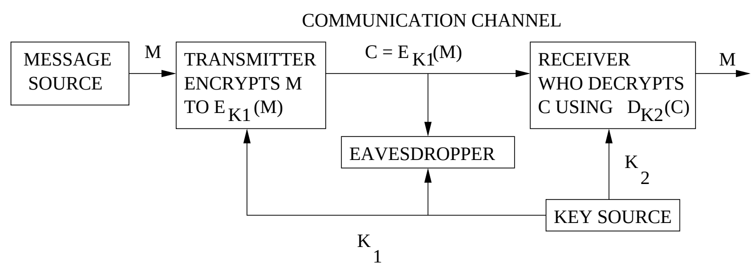
Definition 2 (Public Key Cryptosystem (PKC))

A PKC consists of a plaintext space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, a *public key* space $\mathcal{K}$, and encryption functions $E_{K_1} : \mathcal{M} \to \mathcal{C}$, indexed by public keys $K_1 \in \mathcal{K}$, with the following properties:

1. Every encryption function E_{K_1} has a left inverse D_{K_2}, where K_2 is the *private* key corresponding to the public key K_1.
2. $E_{K_1}(M)$ and $D_{K_2}(C)$ are easy to compute when K_1 and K_2 are known.
3. $D_{K_2}(E_{K_1}(M)) = M$ for all $M \in \mathcal{M}$.
4. Given K_1, E_{K_1}, and $C = E_{K_1}(M)$, it is computationally infeasible to find M or K_2.

Properties 2, 3, 4 describe E_{K_1} as a trapdoor one-way function.

Schematic of a Public-Key Cryptosystem



Note 1

In a public-key cryptosystem (PKC), it is *not* necessary for the key channel to be secure.

Properties of a PKC

Unlike conventional cryptosystems, messages encrypted using public key cryptosystems contain sufficient information to uniquely determine the plaintext and the key (given enough ciphertext, resources etc)

- The entropy contained in these systems is *zero*.
- This is the exact opposite of a perfectly secret system like the one-time pad.

Security in a public key cryptosystem lies solely in the computational cost of computing the plaintext and/or private key from the ciphertext (computional security).

Hybrid Encryption

All PKC's in use today are much slower (by a factor of 1000-1500 or so) than conventional systems like AES, so they are generally not used for bulk encryption. Most common uses:

- Encryption and transmission of keys for conventional cryptosystems (*hybrid* encryption)
- Authentication and non-repudiation via digital signatures (later).

RSA Motivation

In 1978, Ron Rivest, Adi Shamir and Len Adleman came up with the first actual realization of a PKC, called RSA after their initials.

This requires more number theory!

Linear Diophantine Equations

Solve the *linear Diophantine equation*

$$ax + by = 1$$

given $a, b \in \mathbb{Z}$, $b > 0$, and $\gcd(a, b) = 1$.

- If $\gcd(a, b) \neq 1$, there is no solution.
  - In general, an equation of the form $ax + by = c$ has a solution if and only if $\gcd(a, b)$ divides c.
- If $b < 0$, use $-b$ and solve for $(x, -y)$.

Diophantine equations are named after Diophantus, a Greek mathematician who lived around 300-200 BCE.

Euclidean Algorithm

Repeated division with remainder.

Given $a, b \in \mathbb{Z}$, $b > 0$, and $\gcd(a, b) = 1$:

$$\begin{aligned}
a &= bq_0 + r_0 & q_0 &= \lfloor a/b \rfloor, 0 < r_0 < b \\
b &= r_0 q_1 + r_1 & q_1 &= \lfloor b/r_0 \rfloor, 0 < r_1 < r_0 \\
r_0 &= r_1 q_2 + r_2 & q_2 &= \lfloor r_0/r_1 \rfloor, 0 < r_2 < r_1 \\
&\vdots \\
r_{n-3} &= r_{n-2} q_{n-1} + r_{n-1} & r_{n-1} &= \gcd(a, b) \\
r_{n-2} &= r_{n-1} q_n + r_n & r_n &= 0
\end{aligned}$$

Termination

Notice that the sequence of remainders (the r_i) is strictly decreasing

- thus, the sequence is finite (algorithm terminates).

Theorem 1 (Lamé, 1844)

$n < 5 \log_{10} \min(a, b)$.

More exactly, Lamé's Theorem states

$$n \le \log_\tau(\min(a, b) + 1)$$

where $\tau = (1 + \sqrt{5})/2$ is the golden ratio.

Extended Euclidean Algorithm

Let $A_{-2} = 0$, $A_{-1} = 1$, $B_{-2} = 1$, $B_{-1} = 0$ and

$$A_k = q_k A_{k-1} + A_{k-2}, \qquad B_k = q_k B_{k-1} + B_{k-2}$$

for $k = 0, 1, \ldots$.

We have $A_n = a$ and $B_n = b$ (n from above), and

$$A_k B_{k-1} - B_k A_{k-1} = (-1)^{k-1} \ .$$

Putting $k = n$ yields

$$\begin{aligned}
A_n B_{n-1} - B_n A_{n-1} &= (-1)^{n-1} \\
a(-1)^{n-1} B_{n-1} + b(-1)^n A_{n-1} &= 1 \ .
\end{aligned}$$

Thus, a solution of $ax + by = 1$ is given by

$$x = (-1)^{n-1} B_{n-1}, \quad y = (-1)^n A_{n-1} \ .$$

Modular Inverses

Recall that $\mathbb{Z}_m^* = \{a \in \mathbb{Z}_m \mid \gcd(a, m) = 1\}$ is the set of integers between 1 and m that are coprime to m.

$\mathbb{Z}_m^*$ consists of exactly those integers that have *modular inverses*:

- for every $a \in \mathbb{Z}_m^*$, there exists $x \in \mathbb{Z}_m^*$ such that $ax \equiv 1 \pmod{m}$.

Computing Modular Inverses

Given $a \in \mathbb{Z}_m^*$, solve the linear congruence $ax \equiv 1 \pmod{m}$ for $x \in \mathbb{Z}_m^*$.

- We want x such that

$$m \mid ax - 1 \implies ax - 1 = ym \implies ax - my = 1 .$$

- Can be solved using the Extended Euclidean Algorithm.
- We only need to compute the B_i because we only need x, not y.

Example 3

For $a \equiv 95x \equiv 1 \pmod{317}$, we obtain $x == \equiv -10 \pmod{317}$, so $x \equiv 307 \pmod{317}$ is the modular inverse of 95.

The RSA Cryptosystem

Named after Ron Rivest, Adi Shamir, and Len Adleman, 1978.

Initially, NSA pressured these guys to keep their invention secret.

Both encryption and decryption are modular exponentiations (same modulus, different exponents):

- Encryption: $C \equiv M^e \pmod{n}$
- Decryption: $M \equiv C^d \pmod{n}$

RSA Setup

The designer

1. Selects two distinct large primes p and q (each around $2^{1536} \approx 10^{463}$)
2. Computes $n = pq$ and $\phi(n) = (p-1)(q-1)$.
3. Selects a random integer $e \in \mathbb{Z}_{\phi(n)}^*$ (so $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$).
4. Solves the linear congruence

$$de \equiv 1 \pmod{\phi(n)}$$

   for $d \in \mathbb{Z}_{\phi(n)}^*$.
5. Keeps d secret and makes n and e public:
   - the public key is $K_1 = \{e, n\}$
   - the private key is $K_2 = \{d\}$ (or $\{d, p, q\}$, discussed later).

RSA Encryption and Decryption

Encryption: Messages for the designer are integers in $\mathbb{Z}_n^*$

- if a message exceeds n, block it into less-than-n size blocks

To send M encrypted, compute and send

$$C \equiv M^e \pmod{n} \text{ where } 0 < C < n .$$

Decryption: To decrypt C, the designer computes

$$M \equiv C^d \pmod{n} \text{ where } 0 < M < n .$$

Why this Works

We have
$$C^d \equiv (M^e)^d \equiv M^{ed} \pmod{n},$$

Since d is chosen such that $ed \equiv 1 \pmod{\phi(n)}$ we have
$$ed = k\phi(n) + 1 \text{ for some } k \in \mathbb{Z},$$

and
$$M^{ed} \equiv M^{k\phi(n)+1} \equiv MM^{k\phi(n)} \equiv M(M^{\phi(n)})^k \pmod{n} .$$

Euler's Theorem states that $a^{\phi(n)} \equiv 1 \pmod{n}$, so we have
$$C^d \equiv M(M^{\phi(n)})^k \equiv M(1)^k \equiv M \pmod{n} .$$

What if $\gcd(M, n) \neq 1$?

We have assumed that $\gcd(M, n) = 1$ in the description of RSA and for applying Euler's Theorem. Is this a problem?

- Can prove that encryption/decryption still work (Assignment 2!).
- The probability that $\gcd(M, n) \neq 1$ is $1/p + 1/q$, i.e., *very* small.
- Note that since $n = pq$ and $M < n$, $\gcd(M, n) \in \{1, p, q\}$, and thus in these extremely rare cases we would likely find a factor of n.
- Paranoid users can guarantee that $\gcd(M, n) = 1$ by simply taking messages in blocks such that $M < p, q$ (twice as slow).

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).
- 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 2

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: RSA-768 (232 digits, 768 bits), Thorsten Kleinjung et. al., December 12, 2009.

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 $a, b \leq 100$).

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 2.
- In practice, can use $e = 3$, but *only when* RSA is used in conjunction with a secure padding mechanism (eg. OAEP — coming soon)

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).