IV054 RSA Cryptosystem

The most important public-key cryptosystem is the RSA cryptosystem on which one can also illustrate a variety of important ideas of modern public-key cryptography.

For example we will discuss various possible attacks on the RSA cryptosystem and various other problems related to security of the RSA cryptosystems.

A special attention will be given to the problem of factorization of integers that play such an important role for security of RSA.

Several factorization methods will be presented and discuss. In doing that we will illustrate modern distributed techniques to factorize very large integers.

IV054 DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

Basic idea: prime multiplication is very easy, integer factorization seems to be unfeasible.

Design of RSA cryptosystems

1. Choose two large (512 - 1024 bits) primes p,q and denote

$$n = pq,\ \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\mathrm{mod}\, \phi(n))$$

Public key: n (modulus), e (encryption algorithm)

Trapdoor information: p, q, d (decryption algorithm)

Plaintext w

Encryption: cryptotext $c = w^e \bmod n$

Decryption: plaintext $w = c^d \bmod n$

Details: A plaintext is first encoded as a word over the alphabet {0, 1,…,9}, then divided into blocks of length i -1, where $10^{i-1} < n < 10^i$. Each block is taken as an integer and decrypted using modular exponentiation.

DESIGN and USE of RSA CRYPTOSYSTEM

Example of the design and of the use of RSA cryptosystems.

- By choosing $p = 41, q = 61$ we get $n = 2501$, $\phi(n) = 2400$
- By choosing $d = 2087$ we get $e = 23$
- By choosing $d = 2069$ we get $e=29$
- By choosing other values of d we get other values of e.

Let us choose the first pair of encryption/decryption exponents (e=23 and d=2087).

Plaintext: KARLSRUHE

Encoding: 100017111817200704

Since $10^3 < n < 10^4$, the numerical plaintext is divided into blocks of 3 digits $\Rightarrow$ 6 plaintext integers are obtained

Encryption: 100, 017, 111, 817, 200, 704

$100^{23} \bmod 2501$, $17^{23} \bmod 2501$, $111^{23} \bmod 2501$

$817^{23} \bmod 2501$, $200^{23} \bmod 2501$, $704^{23} \bmod 2501$

provides cryptotexts: 2306, 1893, 621, 1380, 490, 313

Decryption:

$2306^{2087} \bmod 2501 = 100$, $1893^{2087} \bmod 2501 = 17$

$621^{2087} \bmod 2501 = 111$, $1380^{2087} \bmod 2501 = 817$

$490^{2087} \bmod 2501 = 200$, $313^{2087} \bmod 2501 = 704$

Correctness of RSA

Let $c = w^e \bmod n$ be the cryptotext for a plaintext w, in the cryptosystem with

$$n = pq,\ ed \equiv 1 \pmod{\phi(n)},\ \gcd(d, \phi(n)) = 1$$

In such a case

$$w \equiv c^d \bmod n$$

and, if the decryption is unique, $w = c^d \bmod n$.

Proof Since $ed \equiv 1 \pmod{\phi(n)}$, there exist a $j \in \mathbb{N}$ such that $ed = j\phi(n) + 1$.

- Case 1. Neither p nor q divides w.

In such a case gcd(n, w) = 1 and by the Euler's Totien Theorem we get that

$$c^d = w^{ed} = w^{j\phi(n)+1} \equiv w \pmod{n}$$

- Case 2. Exactly one of p,q divides w - say p.

In such a case $w^{ed} \equiv w \pmod{p}$ and by Fermat's Little theorem $w^{q-1} \equiv 1 \pmod{q}$

$$\Rightarrow w^{q-1} \equiv 1 \pmod{q} \Rightarrow w^{\phi(n)} \equiv 1 \pmod{q}$$
$$\Rightarrow w^{j\phi(n)} \equiv 1 \pmod{q}$$
$$\Rightarrow w^{ed} \equiv w \pmod{q}$$

Therefore: $w \equiv w^{ed} \equiv c^d \pmod{n}$

- Case 3 Both p,q divide w.

This cannot happen because, by our assumption, $w < n$.

RSA challenge

One of the first description of RSA was in the paper.

Martin Gardner: Mathematical games, Scientific American, 1977

and in this paper RSA inventors presented the following challenge.

Decrypt the cryptotext:

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

Encrypted using the RSA cryptosystem with

n: 114 381 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 830 597 123 513 958 705 058 989 075 147 599 290 026 879 543 541.

and with *e* = 9007

The problem was solved in 1994 by first factorizing *n* into one 64-bit prime and one 65-bit prime, and then computing the plaintext

THE MAGIC WORDS ARE SQUEMISH OSSIFRAGE

How to design a good RSA cryptosystem

1. How to choose large primes *p,q*?

Choose randomly a large integer *p*, and verify, using a randomized algorithm, whether *p* is prime. If not, check *p* + 2, *p* + 4,…

From the Prime Number Theorem if follows that there are approximately

$$\frac{2^d}{\log 2^d} - \frac{2^{d-1}}{\log 2^{d-1}}$$

d bit primes. (A probability that a 512-bit number is prime is 0.00562.)

2. What kind of relations should be between *p* and *q*?

2.1 Difference |*p*-*q*| should be neither too small not too large.

2.2 gcd(*p*-1, *q*-1) should not be large.

2.3 Both *p*-1 and *q*-1 should contain large prime factors.

2.4 Quite ideal case: *q, p* should be safe primes - such that also (*p*–1)/2 and (*q*-1)/2 are primes (83,107,10^{100} – 166517 are examples of safe primes).

3. How to choose *e* and *d*?

3.1 Neither *d* nor *e* should be small.

3.2 *d* should not be smaller than $n^{1/4}$. (For $d < n^{1/4}$ a polynomial time algorithm is known to determine *d*.

DESIGN OF GOOD RSA CRYPTOSYSTEMS

Claim 1. Difference $|p-q|$ should not be small.

Indeed, if $|p - q|$ is small, and $p > q$, then $(p + q)/2$ is only slightly larger than $\sqrt{n}$ because

$$\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$$

In addition $\frac{(p+q)^2}{4} - n$ is a square, say y^2.

In order to factor n it is then enough to test $x > \sqrt{n}$ until such x is found that $x^2 - n$ is a square, say y^2. In such a case

$p + q = 2x$, $p - q = 2y$ and therefore $p = x + y$, $q = x - y$.

Claim 2. gcd(p-1, q-1) should not be large.

Indeed, in the opposite case s = lcm(p-1, q-1) is much smaller than $\phi(n)$. If

$$d'e \equiv 1 \bmod s,$$

then, for some integer k,

$$c^{d'} \equiv w^{ed'} \equiv w^{ks+1} \equiv w \bmod n$$

since $p - 1|s$, $q - 1|s$ and therefore $w^{k1s} \equiv 1 \bmod p$ and $w^{ks+1} \equiv w \bmod q$. Hence, d' can serve as a decryption exponent.

Moreover, in such a case s can be obtained by testing.

Question Is there enough primes (to choose again and again new ones)?

No problem, the number of primes of length 512 bit or less exceeds 10^{150}.

How important is factorization for breaking RSA?

1. If integer factorization is feasible, then RSA is breakable.

2. There is no proof that factorization is needed to break RSA.

3. If a method of breaking RSA would provide an effective way to get a trapdoor information, then factorization could be done effectively.

Theorem Any algorithm to compute $\phi(n)$ can be used to factor integers with the same complexity.

Theorem Any algorithm for computing d can be converted into a break randomized algorithm for factoring integers with the same complexity.

4. There are setups in which RSA can be broken without factoring modulus n.

Example An agency chooses p, q and computes a modulus $n = pq$ that is publicized and common to all users U_1, U_2 and also encryption exponents e_1, e_2,… are publicized. Each user U_i gets his decryption exponent d_i.

In such a setting any user is able to find in deterministic quadratic time another user's decryption exponent.

Security of RSA

None of the numerous attempts to develop attacks on RSA has turned out to be successful.

There are various results showing that it is impossible to obtain even only partial information about the plaintext from the cryptotext produces by the RSA cryptosystem.

We will show that were the following two functions, computationally polynomially equivalent, be efficiently computable, then the RSA cryptosystem with the encryption (decryption) algorithm e_k (d_k) would be breakable.

$parity_{ek}(c)$ = the least significant bit of such an w that $e_k(w) = c$;

$$half_{ek}(c) = 0 \text{ if } 0 \le w < \frac{n}{2} \text{ and } half(c) = 1 \text{ if } \frac{n}{2} \le w \le n-1.$$

We show two important properties of the functions *half* and *parity*.

1. Polynomial time computational equivalence of the functions *half* and *parity* follows from the following identities

$$half_{ek}(c) = parity_{ek}(c \times e_k(2) \bmod n)$$

$$parity_{ek}(c) = half_{ek}(c \times e_k(\tfrac{1}{2}) \bmod n)$$

and the multiplicative rule $e_k(w_1)e_k(w_2) = e_k(w_1w_2)$.

2. There is an efficient algorithm to determine plaintexts w from the cryptotexts c obtained by RSA-decryption provided efficiently computable function *half* can be used as the oracle:

BREAKING RSA USING AN ORACLE

Algorithm:

for $i = 0$ to $\lfloor \lg n \rfloor$ **do**
    $c_i \leftarrow half(c)$; $c \leftarrow (c \times e_k(2)) \bmod n$
$l \leftarrow 0$; $u \leftarrow n$
for $i = 0$ to $\lfloor \lg n \rfloor$ **do**
    $m \leftarrow (l + u) / 2$;
    if $c_i = 1$ **then** $l \leftarrow m$ **else** $u \leftarrow m$;
$w \leftarrow [u]$

Indeed, in the first cycle

$$c_i = half\left(c \times \left(e_k(2)\right)^i\right) = half\left(e_k\left(2^i w\right)\right),$$

is computed for $0 \le i \le \lg n$.

In the second part of the algorithm binary search is used to determine interval in which w lies. For example, we have that

$$half\left(e_k(w)\right) = 0 \equiv w \in \left[0, \tfrac{n}{2}\right)$$
$$half\left(e_k(2w)\right) = 0 \equiv w \in \left[0, \tfrac{n}{4}\right) \cup \left[\tfrac{n}{2}, \tfrac{3n}{4}\right)$$
$$half\left(e_k(4w)\right) = 0 \equiv w \in$$

Security of RSA

There are many results for RSA showing that certain parts are as hard as whole. For example any feasible algorithm to determine the last bit of the plaintext can be converted into a feasible algorithm to determine the whole plaintext.

Example Assume that we have an algorithm ***H*** to determine whether a plaintext x designed in RSA with public key e, n is smaller than $n / 2$ if the cryptotext y is given.

We construct an algorithm ***A*** to determine in which of the intervals $(jn/8, (j+1)n/8)$, $0 \leq j \leq 7$ the plaintext lies.

Basic idea H is used to decide whether the plaintexts for cryptotexts $x^e \bmod n$, $2^e x^e \bmod n$, $4^e x^e \bmod n$ are smaller than $n / 2$.

Answers

yes, yes, yes $0 < x < n/8$	no, yes, yes $n/2 < x < 5n/8$
yes, yes, no $n/8 < x < n/4$	no, yes, no $5n/8 < x < 3n/4$
yes, no, yes $n/4 < x < 3n/8$	no, no, yes $3n/4 < x < 7n/8$
yes, no, no $3n/8 < x < n/2$	no, no, no $7n/8 < x < n$

RSA with a composite "to be a prime"

Let us explore what happens if some integer p used, as a prime, to design a RSA is actually not a prime.

Let $n = pq$ where q be a prime, but $p = p_1p_2$, where p_1, p_2 are primes. In such a case

$$\phi(n) = (p_1 - 1)(p_2 - 1)(q - 1)$$

but assume that the RSA-designer works with $\phi_1(n) = (p - 1)(q - 1)$

Let u = lcm(p_1 - 1, p_2 - 1, q -1) and let gcd(w, n) = 1. In such a case

$$w^{p_1-1} \equiv 1 \pmod{p_1}, \quad w^{p_2-1} \equiv 1 \pmod{p_2}, \quad w^{q-1} \equiv 1 \pmod{q}$$

and as a consequence

$$w^u \equiv 1 \pmod{n}$$

In such a case u divides $\phi(n)$ and let us assume that also u divides $\phi_1(n)$.

Then

$$w^{\phi_1(n)+1} \equiv w \pmod{n}.$$

So if $e_d \equiv 1 \bmod \phi_1(n)$, then encryption and decryption work as if p were prime.

Example $p = 91 = 7 \cdot 13$, $q = 41$, $n = 3731$, $\phi_1(n) = 3600$, $\phi(n) = 2880$, lcm(6, 12, 40) = 120, $120 | \phi_1(n)$.

If gcd(d, $\phi_1(n)$) = 1, then gcd(d, $\phi(n)$) = 1 $\Rightarrow$ one can compute e using $\phi_1(n)$.
However, if u does not divide $\phi_1(n)$, then the cryptosystem does not work properly.

IV054 Two users should not use the same modulus

Otherwise, users, say *A* and *B*, would be able to decrypt messages of each other using the following method.

Decryption: *B* computes

$$f = \gcd(e_B d_B - 1, e_A), m = \frac{e_B d_B - 1}{f}$$

Since

$$e_B d_B - 1 = k\phi(n) \text{ for some } k$$

it holds:

$$\gcd(e_A, \phi(n)) = 1 \Rightarrow \gcd(f, \phi(n)) = 1$$

and therefore

$$m \text{ is a multiple of } \phi(n).$$

m and e_A have no common divisor and therefore there exist integers u, v such that

$$um + ve_A = 1$$

Since m is a multiple of $\phi(n)$ we have

$$ve_A = 1 - um \equiv 1 \bmod \phi(n)$$

and since $e_A d_A \equiv 1 \bmod \phi(n)$ we have

$$(v - d_A)e_A \equiv 0 \bmod \phi(n)$$

and therefore

$$v \equiv d_A \bmod \phi(n)$$

is a decryption exponent of *A*. Indeed, for a cryptotext c:

$$c^v \equiv w^{e_A v} \equiv w^{e_A d_A + c\phi(n)} \equiv w \bmod n.$$

Private-key versus public-key cryptography

- The prime advantage of public-key cryptography is increased security - the private keys do not ever need to be transmitted or revealed to anyone.

- Public key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.

- Example RSA and DES are usually combined as follows
  1. The message is encrypted with a random DES key
  2. DES-key is encrypted with RSA
  3. DES-encrypted message and RSA-encrypted DES-key are sent.

This protocol is called RSA digital envelope.

- In software (hardware) DES is generally about 100 (1000) times faster than RSA.

If *n* users communicate with secrete-key cryptography, they need *n* (*n* - *1*) / *2* keys. In the case they use public key cryptography *2n* keys are sufficient.

Public-key cryptography allows spontaneous communication.

Private-key versus public-key cryptography

• If RSA is used for digital signature then the public key is usually much smaller than private key => verification is faster.

• An RSA signature is superior to a handwritten signature because it attests both to the contents of the message as well as to the identity of the signer.

As long as a secure hash function is used there is no way to take someone's signature from one document and attach it to another, or to after the signed message in any way.

The slightest change in a signed message will cause the digital signature verification process to fail.

• Digital signature are the exact tool necessary to convert even the most important paper based documents to digital form and to have them only in the digital form.