Cryptocomplexity II

Kryptokomplexität II

Sommersemester 2024

Chapter 4: Rabin's Public-Key Cryptosystem

Dozent: Prof. Dr. J. Rothe

Rabin's Public-Key Cryptosystem

- In 1979, Michael O. Rabin developed a public-key cryptosystem whose security is based on the difficulty of computing square roots modulo some integer n.
- His cryptosystem is provably secure against chosen-plaintext attacks, assuming that the factoring problem is computationally intractable, i.e., assuming that it is hard to find the prime factors of $n = pq$ by a randomized algorithm with nonnegligible probability.
- However, it is insecure against chosen-ciphertext attacks.

Rabin's Public-Key Cryptosystem

Step	Alice	Erich	Bob
1			chooses two large random primes, p and q with $p \equiv q \equiv 3 \bmod 4$ and $p \neq q$, keeps them secret, and computes his public key $$n = pq$$
2		$\Leftarrow n$	
3	encrypts the message m by $$c = m^2 \bmod n$$		
4		$c \Rightarrow$	
5			decrypts c by computing $$m = \sqrt{c} \bmod n$$

Rabin's Public-Key Cryptosystem

1. **Key Generation.** Bob randomly chooses two large distinct prime numbers p and q, which satisfy $p \equiv q \equiv 3 \bmod 4$.
   - The pair (p, q) is his private key.
   - He then computes the module $n = pq$, his public key.

2. **Communication.** Bob's public key n is now known to Alice.

3. **Encryption.** Given the public key n, Alice computes her ciphertext c by squaring her message m modulo n, i.e., the encryption function $E_n : \mathbb{Z}_n^* \rightarrow \mathbb{Z}_n^*$ is defined by

$$E_n(m) = c = m^2 \bmod n.$$

4. **Communication.** Alice sends the ciphertext c to Bob.

Rabin's Public-Key Cryptosystem

5. **Decryption.** The decryption function is given by

$$D_{(p,q)}(c) \quad = \quad \sqrt{c} \bmod n. \tag{1}$$

- It is not clear yet how the private key (p, q) is used for decryption.
- Note that, in general, computing square roots modulo some integer with unknown prime factors is considered to be a hard problem.
- However, since Bob knows the prime factors p and q of n, he can make use of the fact that determining m by (1) is equivalent to solving the following two congruences for the values m_p and m_q:

$$(m_p)^2 \quad \equiv \quad c \bmod p; \tag{2}$$

$$(m_q)^2 \quad \equiv \quad c \bmod q. \tag{3}$$

Rabin's Public-Key Cryptosystem

- By Euler's criterion, Bob can efficiently decide
  - whether or not c is a quadratic residue modulo p, and also
  - whether or not c is a quadratic residue modulo q.
- However, Euler's criterion does not actually find these square roots.
- Fortunately, using the assumption that $p \equiv q \equiv 3 \bmod 4$, Bob can apply our lemma (on slide 41, Chapter 3): If p is a prime number with $p \equiv 3 \bmod 4$, then every $\alpha \in \mathrm{QR}_p$ has the two square roots

$$\pm\alpha^{(p+1)/4} \bmod p.$$

So, he first computes

$$m_p = c^{(p+1)/4} \bmod p \quad \text{and} \quad m_q = c^{(q+1)/4} \bmod q.$$

Rabin's Public-Key Cryptosystem

- Note that c must be a square root modulo p, provided that c is a valid ciphertext, i.e., provided that c was created by proper encryption of some message.

- Again by Euler's criterion, c is a quadratic residue modulo p if and only if $c^{(p-1)/2} \equiv 1 \bmod p$. Hence,

$$(\pm m_p)^2 \equiv \left(\pm c^{(p+1)/4}\right)^2 \equiv c^{(p+1)/2} \equiv c^{(p-1)/2} c \bmod p \equiv c \bmod p,$$

  which proves (2).

- Thus, $\pm m_p$ are the two square roots of c modulo p.

- Analogously, $\pm m_q$ are the two square roots of c modulo q, which proves (3).

Rabin's Public-Key Cryptosystem

- Then, using the Chinese Remainder Theorem, Bob determines the four square roots of c modulo n.
- To this end, he first uses the extended Euclidean Algorithm to compute integer coefficients z_p and z_q such that

$$z_p p + z_q q = 1.$$

- Finally, applying the Chinese Remainder Theorem, he computes

$$s = (z_p p m_q + z_q q m_p) \bmod n \quad \text{and} \quad t = (z_p p m_q - z_q q m_p) \bmod n.$$

  It can be checked that $\pm s$ and $\pm t$ are the four square roots of c modulo n.

- Which one yields the "right" plaintext, is not immediately clear.

Rabin's Public-Key Cryptosystem

Remark:

1.
- Note that encryption in Rabin's system is *not injective*.
- That is, since n is the product of two prime numbers, every ciphertext c has four square roots modulo n.
- Thus, Rabin's system has the disadvantage that decryption recovers not only the original plaintext, but also three other square roots of c that hopefully are "sufficiently meaningless" so as to be eliminated.
- One way for Bob to tell the "right" decryption apart from these three "wrong" decryptions is to give the plaintext a special structure identifying the original plaintext. For example, one might repeat one specified block of plaintext, e.g., attach to m the last 64 bits of m.
- However, the proof that breaking the Rabin system is "computationally equivalent" to the factoring problem is then no longer valid.

Rabin's Public-Key Cryptosystem

Remark:

2
- Rabin's system also works for prime factors that are not so-called *Blum numbers*, i.e., not of the form $p \equiv q \equiv 3 \bmod 4$.
- However, the usage of Blum numbers simplifies the analysis of this system.
- For example, if $p \equiv 1 \bmod 4$, then there is no known *deterministic* polynomial-time algorithm for computing the square roots modulo p, which is needed for efficient decryption, even though there is an efficient randomized *Las Vegas algorithm* for this problem.
- Finally, note that in Rabin's system it would also be possible to use $\mathbb{Z}_n$ instead of $\mathbb{Z}_n^*$ as the message and ciphertext space.

Rabin's Public-Key Cryptosystem: Example

Example (Rabin's public-key cryptosystem)

Suppose that Bob chooses the prime numbers $p = 43$ and $q = 47$.

Note that $43 \equiv 47 \equiv 3 \bmod 4$.

He then computes the Rabin modulus $n = pq = 2021$.

To encrypt the message $m = 741$, Alice computes

$$c = 741^2 = 549081 \equiv 1390 \bmod 2021$$

and sends $c = 1390$ to Bob.

Rabin's Public-Key Cryptosystem: Example

Example (Rabin's public-key cryptosystem: continued)

To decrypt the ciphertext c, Bob first determines the following values:

$$\begin{aligned} m_p &= 1390^{(43+1)/4} = 1390^{11} \equiv 10 \bmod 43; \\ m_q &= 1390^{(47+1)/4} = 1390^{12} \equiv 36 \bmod 47, \end{aligned}$$

using fast exponentiation ("square-and-multiply").

Now, using the extended Euclidean Algorithm, he computes the integer coefficients $z_p = -12$ and $z_q = 11$ satisfying

$$z_p p + z_q q = -12 \cdot 43 + 11 \cdot 47 = 1.$$

Rabin's Public-Key Cryptosystem: Example

Example (Rabin's public-key cryptosystem: continued)

Finally, by the Chinese Remainder Theorem, he computes

$$\begin{aligned} s &= z_p p m_q + z_q q m_p = -12 \cdot 43 \cdot 36 + 11 \cdot 47 \cdot 10 \equiv 741 \bmod 2021; \\ t &= z_p p m_q - z_q q m_p = -12 \cdot 43 \cdot 36 - 11 \cdot 47 \cdot 10 \equiv 506 \bmod 2021. \end{aligned}$$

As can easily be checked, the four plaintexts that are encrypted to the same ciphertext $c = 1390$ are $\pm s$ and $\pm t$, i.e., 741, 1280, 506, and 1515.

Security of Rabin's Public-Key Cryptosystem

- Suppose Erich is able to factor the Rabin module n. He thus obtains Bob's private key and can decipher any message sent to Bob.
- That is, breaking the Rabin system is computationally no harder than solving the factoring problem.
- Conversely, we show that factoring large integers is no harder than breaking the Rabin system, so these are equally hard problems.
- Thus, Rabin's cryptosystem has a proof of security that is based on the assumption that factoring is computationally intractable.
- In this regard, Rabin's system is superiour to other public-key systems such as RSA or ElGamal.

The Problem of Breaking Rabin's System

- This result is proven by a polynomial-time *randomized (Las Vegas) Turing reduction* from the factoring problem to the (functional) problem of breaking Rabin's system.
- Informally stated, a *Las Vegas algorithm* is a randomized algorithm that never gives a wrong answer, although it might happen that it doesn't give any answer at all, i.e., it has "zero-sided error" (ZPP).
- *Monte Carlo algorithms* are randomized algorithms with "one-sided error" $(\mathrm{RP}$ and $\mathrm{coRP})$.
- There are also "two-sided error" randomized algorithms (BPP).

The Problem of Breaking Rabin's System

- Recall that the set of quadratic residues modulo n is denoted by

$$\mathrm{QR}_n = \{x^2 \bmod n \,\big|\, x \in \mathbb{Z}_n^*\}.$$

Definition

Define the (functional) *problem of breaking Rabin*, denoted by BREAK-RABIN as follows: Given $\langle n, c \rangle$, where

- n is the product of two (unknown) prime numbers in $3 + 4\mathbb{Z}$ and
- $c \in \mathrm{QR}_n$,

compute some $m \in \mathbb{Z}_n^*$ such that

$$c = m^2 \bmod n.$$

RP and coRP: Yes- and No-biased Monte Carlo Algorithms

- A *no-biased Monte Carlo algorithm* for a decision problem A is a randomized polynomial-time algorithm that:
  - always gives reliable "yes" answers, but
  - possibly incorrect "no" answers.

  They accept problems in the complexity class RP.

- *Random polynomial time* (denoted by RP) is the complexity class of all decision problems A for which there is a randomized polynomial-time algorithm M such that for each input x,
  - $x \in A \Longrightarrow \Pr(M \text{ accepts } x) \geq 1/2$;
  - $x \notin A \Longrightarrow \Pr(M \text{ accepts } x) = 0$.

- A *yes-biased Monte Carlo algorithm* for A is a no-biased Monte Carlo algorithm for the complement of A.

  They accept problems in the complexity class $\mathrm{coRP} = \{\overline{A} \mid A \in \mathrm{RP}\}$.

ZPP: Las Vegas Algorithms

- By repeated trials, the error probability of a yes- or no-biased algorithm can be made arbitrarily small, from $1/2$ to $2^{-|x|}$.
- In addition, there are *Las Vegas algorithms*, randomized algorithms that never lie (but may give no answer at all): $\mathrm{ZPP} = \mathrm{RP} \cap \mathrm{coRP}$.
- *Zero-error probabilistic polynomial time* (denoted by ZPP) is the complexity class of all decision problems A for which there is a randomized polynomial-time algorithm M with three types of final states (s_a accepts, s_r rejects, and $s_?$ for "don't know") such that for each input x,
  - $x \in A \implies (\Pr(M \text{ accepts } x) \geq 1/2 \text{ and } \Pr(M \text{ rejects } x) = 0)$;
  - $x \notin A \implies (\Pr(M \text{ rejects } x) \geq 1/2 \text{ and } \Pr(M \text{ accepts } x) = 0)$.

The Problem of Breaking Rabin's System

Theorem

There is a polynomial-time Las Vegas algorithm RANDOM-FACTOR *that, given any integer $n = pq$ with $p \equiv q \equiv 3 \bmod 4$, uses its function oracle* BREAK-RABIN *to find the prime factors of n with probability at least $1/2$.*

Proof: Let $n = pq$ be the Rabin modulus to be factored, where

$$p \equiv q \equiv 3 \bmod 4.$$

Consider the algorithm RANDOM-FACTOR with oracle BREAK-RABIN on the next slide.

Algorithm Random-Factor with Oracle break-rabin

Random-Factor$^{\text{BREAK-RABIN}}(n)$ {

(* *Rabin module* $n = pq$ *with* $p \equiv q \equiv 3 \bmod 4$ *for distinct primes* p *and* q *)

Randomly choose a number $x \in \mathbb{Z}_n^*$ under the uniform distribution;

$c := x^2 \bmod n$;

$m := \text{BREAK-RABIN}(\langle n, c \rangle)$;

(* *query the oracle about* $\langle n, c \rangle$ *to obtain an* m *with* $c = m^2 \bmod n$ *)

`if` $(m \equiv \pm x \bmod n)$ `return` "failure" and halt;

`else`

$p := \mathtt{gcd}(m - x, n)$;

$q := n/p$;

`return` "p and q are the prime factors of n" and halt;

}

Figure: Factoring a Rabin module using an oracle to break Rabin's system

The Problem of Breaking Rabin's System

On input n, $\textsc{Random-Factor}$ with oracle $\textsc{break-rabin}$ randomly picks an element $x \in \mathbb{Z}_n^*$ and squares it modulo n to obtain

$$c \in \mathrm{QR}_n.$$

Then, the algorithm queries its oracle $\textsc{break-rabin}$ about the pair $\langle n, c \rangle$ and obtains the answer m, which is one of the square roots of c modulo n.

The two square roots m and x of c modulo n need not be identical.

However, m and x must satisfy either one of the following two cases.

The Problem of Breaking Rabin's System

Case 1: $\mathbf{m} \equiv \pm\mathbf{x} \bmod \mathbf{n}$.

Then, we have either $m = x$ or $m + x = n$.

Thus, $\texttt{gcd}(m - x, n)$ is either n or 1.

In both cases, the algorithm does not find a prime factor of n and returns "failure."

Case 2: $\mathbf{m} \equiv \pm\alpha\mathbf{x} \bmod \mathbf{n}$, where α is a nontrivial square root of $\mathbf{1} \bmod \mathbf{n}$. In this case,

$$m^2 \equiv x^2 \bmod n \quad \text{and} \quad m \not\equiv \pm x \bmod n.$$

Thus, $\texttt{gcd}(m - x, n)$ is either p or q, which yields the factorization of n.

The Problem of Breaking Rabin's System

To estimate the success probability of RANDOM-FACTOR, let x be any element randomly chosen in $\mathbb{Z}_n^*$ under the uniform distribution.

Let α be a nontrivial square root of 1 mod n.

Consider the set

$$R_x = \{\pm x \bmod n\} \cup \{\pm \alpha x \bmod n\}.$$

Squaring any element r of R_x yields the same $c = r^2 = x^2 \bmod n$.

In particular, the oracle answer

$$m = \text{BREAK-RABIN}(\langle n, c \rangle)$$

is an element of R_x, and is independent of which of the four elements of R_x in fact was chosen to yield c.

The Problem of Breaking Rabin's System

In Case 2 above, we noted that the algorithm finds the prime factors of n if and only if $m \equiv \pm\alpha x \bmod n$.

For fixed m, the probability that an $x \in R_x$ with $m \equiv \pm\alpha x \bmod n$ was chosen is $1/2$.

Hence, the success probability of RANDOM-FACTOR is $1/2$. ❑

Remark: The success probability of RANDOM-FACTOR can be amplified so as to be arbitrarily close to one.

Corollary

Assuming that large integers cannot be factored by an efficient randomized algorithm with nonnegligible probability of success, Rabin's cryptosystem is secure against chosen-plaintext attacks.

Security of Rabin's Public-Key Cryptosystem

Corollary

Rabin's cryptosystem is insecure against chosen-ciphertext attacks.

Proof: The scenario of a chosen-ciphertext attack is that a cryptanalyst has temporary access to the decryption device.

Thus, choosing some ciphertext c at will, he learns the corresponding plaintext m.

This can be seen as having an efficient algorithm (as opposed to a hypothetical oracle) for computing BREAK-RABIN.

By the previous theorem, the attacker can take advantage of this fact as follows.

Security of Rabin's Public-Key Cryptosystem

He chooses some plaintext x at random, computes

$$c = x^2 \bmod n,$$

and decrypts c to obtain a square root m of c modulo n.

As in the proof of the previous theorem, he can factor the Rabin modulus n with high probability, and obtains the private key. ❑

Example (factoring by breaking Rabin's system)

Let $n = 23 \cdot 7 = 161$ be the given Rabin modulus.

Suppose Erich does not know the prime factors 7 and 23.

However, he has the oracle BREAK-RABIN (or an efficient algorithm for computing it) and can thus determine square roots modulo 161.

Security of Rabin's Public-Key Cryptosystem

Example (factoring by breaking Rabin's system: continued)

Using the algorithm RANDOM-FACTOR, Erich randomly picks $x = 13$; note that $\gcd(161, 13) = 1$, so $13 \in \mathbb{Z}_{161}^{*}$.

He then computes $c = 13^2 \bmod 161 = 8$.

The four square roots of 8 mod 161 are $R_{13} = \{13, 36, 125, 148\}$.

Let m be the oracle answer for the query $\langle 161, 8 \rangle$, i.e.,

$$m = \text{BREAK-RABIN}(\langle 161, 8 \rangle).$$

For each possible answer $m \in R_{13}$, we determine $\gcd(m - x, n)$.

Security of Rabin's Public-Key Cryptosystem

Example (factoring by breaking Rabin's system: continued)

If $m = 13$ then $\mathtt{gcd}(m - x, n) = \mathtt{gcd}(0, 161) = 161$.

And if $m = 148$ then $\mathtt{gcd}(m - x, n) = \mathtt{gcd}(135, 161) = 1$.

In both cases, Random-Factor fails to find the prime factors of 161.

But if $m = 36$ then $\mathtt{gcd}(m - x, n) = \mathtt{gcd}(23, 161) = 23$,

and if $m = 125$ then $\mathtt{gcd}(m - x, n) = \mathtt{gcd}(112, 161) = 7$.

In these two cases, Random-Factor succeeds and provides Erich with the prime factors of 161.

Thus, Erich has a fifty percent chance of factoring n.