Integer factoring and related cryptosystems

Diego F. Aranha

Institute of Computing
UNICAMP

Public key cryptography

Definition

Asymmetric cryptographic schemes are cryptosystems that do not require a secure channel for transmitting the encryption key. Thus, their security is based on the computational infeasibility of deriving the private key from the public key.

Civil history:

- Notion of public key cryptography (1976).
- RSA encryption and digital signatures (1977).

Military history:

- "Nonsecret cryptography" first discovered by Ellis (1970).
- RSA analogue proposed by Cocks (1973).

Important: Impossible to obtain perfect secrecy. Why?

Public key cryptography

Definition

Asymmetric cryptographic schemes are cryptosystems that do not require a secure channel for transmitting the encryption key. Thus, their security is based on the computational infeasibility of deriving the private key from the public key.

Civil history:

- Notion of public key cryptography (1976).
- RSA encryption and digital signatures (1977).

Military history:

- "Nonsecret cryptography" first discovered by Ellis (1970).
- RSA analogue proposed by Cocks (1973).

Important: Attacker can enumerate plaintexts!

Public key cryptography

Conclusion: We need to rely on computational security.

Public key cryptography

Conclusion: Encryption function should be easy to compute but hard to invert.

Definition

A function computable in polynomial time whose inverse can only be computed in superpolynomial time is called a *one-way function*.

Public key cryptography

Conclusion: We need to rely on computational security.

Definition

A function computable in polynomial time whose inverse can only be computed in superpolynomial time is called a *one-way function*.

Current state:

- One-way function candidates are known.
- No candidate was proven one-way.

Public key cryptography

Conclusion: We need to rely on computational security.

Definition

A function computable in polynomial time whose inverse can only be computed in superpolynomial time is called a *one-way function*.

Current state:

- One-way function candidates are known.
- No candidate was proven one-way.
- Such a proof implies $P \neq NP$.

Public key cryptography

Problem: But how to decrypt in polynomial time?

Public key cryptography

Problem: Encryption function becomes easy to invert with special information!

Definition

A function computable in polynomial time with inverses computable in polynomial time using secret information is called a *trapdoor one-way function*.

Example: Let $n = pq$ with $p, q \in \mathbb{Z}$.
Then $f : \mathbb{Z}_n \to \mathbb{Z}_n$ such that $f(x) = x^b \bmod n$ is a one-way candidate when $gcd(b, \phi(n)) = 1$. We have that $f^{-1}(x) = x^a \bmod n$ for a certain value of a.

RSA (Rivest, Shamir, Adleman, 1977)

Key generation:

1. Generate primes p and q with $k/2$ bits.
2. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
3. Select e such that $gcd(e, \phi(N)) = 1$. (small prime?)
4. Compute d such that $d = e^{-1} \bmod \phi(N)$.
5. $\mathcal{M} = \mathcal{C} = \mathbb{Z}_N$.
6. $K = (N, p, q, d, e)$.
7. Public key is (e, N), private key is (d, N, p, q).

Encryption: Compute $Enc_K(x) = x^e \bmod N$.

Decryption: Compute $Dec_K(y) = y^d \bmod N$.

Hardness assumptions

Integer factoring

Let $\mathcal{A}$ a PPT algorithm and $Factor_{\mathcal{A},GenModulus}(n)$ the execution of an experiment with $\mathcal{A}$ at security level n:

1. Modulus $(N, p, q) \leftarrow GenModulus(1^n)$ is generated.
2. $\mathcal{A}$ receives N and outputs $p', q' > 1$.
3. The experiment output is 1 if $p' \cdot q' = N$, and 0 otherwise.

The *integer factorization problem* is hard if for every $\mathcal{A}$, the advantage of winning the game above is negligible as a function of n:

$$\Pr[Factor_{\mathcal{A},GenModulus}(n) = 1] \leq \delta(n).$$

Hardness assumptions

RSA problem

Let $\mathcal{A}$ a PPT algorithm and $RSA\text{-}inv_{\mathcal{A},GenRSA}(n)$ the execution of an experiment with $\mathcal{A}$ at security level n:

1. Modulus $(N, d, e) \leftarrow GenRSA(1^n)$ is generated.
2. Choose $y \leftarrow \mathbb{Z}_N^*$.
3. $\mathcal{A}$ receives N, e, y and outputs $x \in \mathbb{Z}_N^*$.
4. The output experiment is 1 if $x^e = y \bmod N$, and 0 otherwise.

The *RSA problem* is hard if for every $\mathcal{A}$, the advantage of winning the game above is negligible as a function of n:

$$\Pr[RSA\text{-}inv_{\mathcal{A},GenRSA}(n) = 1] \leq \delta(n).$$

Important: Reducing RSA to integer factoring is easy, but reducing integer factoring to RSA only for *generic* algorithms.

Integer factoring

Many algorithms:

- Sieve of Erathostenes
- Trival division
- Quadratic sieve
- Elliptic curve method (ECM)
- NFS – Number Field Sieve
- Pollard ρ and $p-1$ algorithms
- Williams' $p+1$ algorithm
- Continuous fractions
- Shor's quantum algorithm

Integer factoring

A direct attack against RSA consists in factoring $N = pq$.

Important: For composite n, n has a prime factor $p \leq \lfloor \sqrt{n} \rfloor$.

Integer factoring

A direct attack against RSA consists in factoring $N = pq$.

Important: For composite n, n has a prime factor $p \leq \lfloor \sqrt{n} \rfloor$.

Trial division and the Sieve of Erathostenes work for $n \approx 10^{12}$, but we need more sophisticated techniques for larger n.

Objective

Find a non-trivial factor n_1 of composite n. Thus, we can write $n = n_1 n_2$, with $1 < n_1, n_2 < n$, test the primality of n_1, n_2 and continue factoring if needed.

Pollard $p - 1$ (1974)

Algorithm

Input: Integer n to factor, upper bound B for smallest factor.

1. $a \leftarrow 2$
2. For $j \leftarrow 2$ to B, do $a \leftarrow a^j \bmod n$
3. $d \leftarrow gcd(a - 1, n)$
4. If $1 < d < n$, return d. Otherwise, return FAIL.

Pollard $p-1$ (1974)

Algorithm

Input: Integer n to factor, upper bound B for smallest factor.

1. $a \leftarrow 2$
2. For $j \leftarrow 2$ to B, do $a \leftarrow a^j \bmod n$
3. $d \leftarrow gcd(a-1, n)$
4. If $1 < d < n$, return d. Otherwise, return FAIL.

Proof: Suppose p a prime factor of n and $q \leq B$ for all prime powers that satisfy $q|(p-1)$. Hence, we have that $(p-1)|B!$.

At the end of the loop, $a \equiv 2^{B!} \pmod{n}$ and, since $p|n$, we have that $a \equiv 2^{B!} \pmod{p}$. Because $2^{p-1} \equiv 1 \pmod{p}$, it is true that $a \equiv 1 \pmod{p}$ and $p|(a-1)$. Since $p|n$, there is $d = gcd(a-1, n)$ such that $p|d$.

Pollard $p-1$ (1974)

Complexity:

- Algorithm is polynomial in the size of B.
- For small B, low chance of success.
- For large $B \approx \sqrt{n}$, degenerates in trial division.

The method works when n has a prime factor p such that $p-1$ only has small factors. Defenses:

- Find prime p_1 such that $p = 2p_1 + 1$ is also prime.
- Find prime q_1 such that $q = 2q_1 + 1$ is also prime.
- The modulus $N = pq$ will be resistant against Pollard $p-1$.

Pollard ρ (Lenstra, 1980)

Let p the smallest prime factor of n. Suppose that there are two integers $x, x' \in \mathbb{Z}_n$ such that $x \neq x'$ e $x \equiv x' \pmod{p}$. Then $p \leq gcd(x - x', n) < n$.

Idea: Define a random set $X \subseteq \mathbb{Z}_n$ and compute $gcd(x - x', n)$ for all pairs $x, x' \in X$. The method works iff (x mod p) generates at least a collision.

By the Birthday Paradox, we need $|X| \approx 1.17\sqrt{p}$ for 50% chance of collision. Hence, we need to compute at least $C_{|X|,2} > p/2$ functions $gcd(\cdot)$ to find a factor of n.

Pollard ρ (Lenstra, 1980)

Now suppose $f(x) = x^2 + a$, with small a and $f(x)$ mod p behaves like a random function. Let $x_1 \in \mathbb{Z}_n$ and consider the set $X = \{x_1, x_2 \ldots, x_m\}$ generated by $x_i = f(x_{i-1})$ mod n, $1 < i < m$. Thus, X will be a random subset with m elements of $\mathbb{Z}_n$.

Suppose $x_i \equiv x_j \pmod{p}, i < j$. We have that $f(x_i) \equiv f(x_j) \pmod{p}$. Then x_{i+1} mod $p = f(x_i)$ mod p and $x_{j+1} = f(x_j)$ mod p.

Thus, $x_{i+1} \equiv x_{j+1} \pmod{p}$, or even that $x_{i'} \equiv x_{j'} \pmod{p}$ if $j' > i' \geq i$ and $j' - i' \equiv 0 \pmod{(j - i)}$.

Now, build a graph G with vertices in the set $\mathbb{Z}_p$, and edges between (x_i mod p) and (x_{i+1} mod p). There is a pair x_i, x_j with $i < j$ such that $x_i \equiv x_j \pmod{p}$. Thus, G looks like a ρ. Let's find collisions in this graph with $j = 2i$.

Pollard ρ (Lenstra, 1980)

Algorithm

1. $x \leftarrow x_1$
2. $x' \leftarrow f(x) \bmod n$
3. $p \leftarrow gcd(x - x', n)$
4. While $p = 1$, do:
   - $x \leftarrow f(x) \bmod n$
   - $x' \leftarrow f(x') \bmod n, x' \leftarrow f(x') \bmod n$
   - $p \leftarrow gcd(x - x', n)$
5. If $p = n$, return FAIL.Otherwise, return p.

Invariant: In the i-th iteration of the algorithm, $x = x_i$ e $x' = x_{2i}$.

Important: Expected $\sqrt{p}$ iterations to find p.

Integer factoring

In practice, the performance of the best integer factoring algorithms is measured with money prizes.

Progress so far:

- 2003: RSA-512, RSA-576
- 2007: Factoring $2^{1039} - 1$
- 2009: RSA-768
- (2018?): RSA-1024

Other attacks

- Leaking $\phi(N)$:

  We have that $\phi(N) = (p-1)(q-1)$. Because $q = N/p$, $p^2 - (N - \phi(N) + 1)p + N = 0$.
- Leaking the private key:

  If the value d is published, it is possible to factor N.
  That means, that one should not only pick a new pair (d, e) but also new N.
- Short decryption exponent (but resistant against exhaustive search).

  Possible to compute d when $3d < N^{1/4}$ and $q < p < 2q$.
  This means that d should have at least $1/4$ of the length of N and p, q should not be too close.

Other attacks

Key generation I

1. Seed a random number generator with seed S.
2. Generated p, q determined from S.

Key generation II

1. Seed a random number generator with seed S_1.
2. Generate p determined by S_1.
3. Seed a random number generator with seed S_2.
4. Generate q determined by S_2.

Important: What is more secure?

Other attacks

Key generation I

1. Seed a random number generator with seed S.
2. Generated p, q determined from S.

Key generation II

1. Seed a random number generator with seed S_1.
2. Generate p determined by S_1.
3. Seed a random number generator with seed S_2.
4. Generate q determined by S_2.

Important: What is more secure?
Solution: The second algorithm is vulnerable against bad S_1. To find shared factors, compute $gcd(\cdot)$ of collected moduli.

Implementation issues

Problem: How to securely encrypt small messages?

Implementation issues

Problem: How to securely encrypt small messages?
Solution: Add random padding!

Standardized versions:

1. **PKCS #1 v1.5**: for m with up to $|N| - 11$ *bytes*, compute:

$$c = (00000000 \parallel 00000010 \parallel r \parallel 00000000 \parallel m)^e \bmod N$$

   Important: r has at least 64 *bits* and cannot contain null bytes. No formal security assessment for specific construction and vulnerable against chosen ciphertext attacks.

2. **PKCS #1 v2.1 (OAEP)**: preferable to use *Optimal Asymmetric Encryption Padding* with security reduction.

Modular square roots

Theorem

Let p odd prime. The equation $y^2 \equiv a \pmod{p}$ has no or two solutions when $\left(\frac{a}{p}\right) = -1$ or $\left(\frac{a}{p}\right) = 1$, respectively.

Let's generalize this result to powers of p.

Modular square roots

Theorem

Let p odd prime. The equation $y^2 \equiv a \pmod{p}$ has no or two solutions when $\left(\frac{a}{p}\right) = -1$ or $\left(\frac{a}{p}\right) = 1$, respectively.

Let's generalize this result to powers of p.

Theorem

Let p odd prime, $e > 0 \in \mathbb{Z}$, $gcd(a, p) = 1$. The congruence $y^2 \equiv a \pmod{p^e}$ has no solutions if $\left(\frac{a}{p}\right) = -1$ and two solutions (modulo p^e) if $\left(\frac{a}{p}\right) = 1$.

Modular square roots

Theorem

Suppose $n > 1 \in \mathbb{Z}$ odd with factorization $n = \prod_{i=1}^{\ell} {p_i}^{e_i}$ such that $gcd(a, n) = 1$. The conguence $y^2 \equiv a \pmod{n}$ has 2^ℓ solutions modulo n if $\left(\frac{a}{p_i}\right) = 1$ for all $i \in \{1, \ldots, \ell\}$, or none otherwise.

Modular square roots

Theorem

Suppose $n > 1 \in \mathbb{Z}$ odd with factorization $n = \prod_{i=1}^{\ell} {p_i}^{e_i}$ such that $gcd(a, n) = 1$. The conguence $y^2 \equiv a \pmod{n}$ has 2^ℓ solutions modulo n if $\left(\frac{a}{p_i}\right) = 1$ for all $i \in \{1, \ldots, \ell\}$, or none otherwise.

Proof: The equation only has two solutions modulo n iff it only has solutions modulo each power $p_i^{e_i}$. Thus, the solutions modulo n can be found by the Chinese Remainder Theorem for the system of equations $y \equiv b_i \pmod{p_i^{e_i}}, 1 \leq i \leq \ell$, with b_i one of the solutions of $y^2 \equiv a \pmod{p_i^{e_i}}$. Since we have two choices for each b_i, the equation modulo n has 2^ℓ solutions.

Rabin cryptosystem

Security equivalent to integer factoring.

Key generation:

1. Generate primes p and q with $k/2$ bits and $p, q \equiv 3 \pmod{4}$.
2. Compute $N = pq$ and select $\mathcal{M} = \mathcal{C} = \mathbb{Z}_N^*$.
3. $K = (N, p, q)$.
4. The public key is N and the private key is (N, p, q).

Encryption: $Enc_K(x) = x^2 \bmod N$.

Decryption: $Dec_K(y) = \sqrt{y} \bmod N$.

Paillier cryptosystem (1999)

Key generation:

1. Generate primes p and q with $k/2$ bits such that $gcd(pq, \phi(N)) = 1$.
2. Compute $N = pq$, $\phi(N) = (p-1)(q-1)$, $\mu = \phi(N)^{-1} \bmod N$ and $g = N + 1$.
3. Choose $\mathcal{M} = \mathbb{Z}_N, \mathcal{C} = \mathbb{Z}_N^2$.
4. The public key is (g, N) and the private key is (λ, μ).

Encryption: $Enc_K(x) = g^m r^N \bmod N^2$, for random $r \in \mathbb{Z}_N^*$

Decryption: $Dec_K(y) = L(c^\lambda \bmod N^2) \cdot \mu \bmod N$, for $L(u) = (u-1)/N$.

Important: Cryptosystem with *homomorphic* property of $Dec_K(Enc_K(m_1, r_1) \cdot Enc_K(m_2, r_2)) = m_1 + m_2$.