The RSA problem

- Let $N = pq$ with p and q distinct, odd primes
- $\mathbb{Z}_N^* =$ *invertiable* elements under multiplication modulo N
  - The order of $\mathbb{Z}_N^*$ is $\phi(N) = (p-1)\cdot(q-1)$
  - $\phi(N)$ is *easy* to compute if p, q are known
  - $\phi(N)$ is *hard* to compute if p, q are not known
    - Equivalent (believed) to factoring N
- Fix e with $\gcd(e, \phi(N)) = 1$
  - Raising to the e-th power is a permutation of $\mathbb{Z}_N^*$
- If $ed \equiv 1 \bmod \phi(N)$, raising to the d-th power is the *inverse* of raising to the e-th power
  - I.e., $(x^e)^d \equiv x \bmod N$
  - x^d is the e-th root of x modulo N

公、私钥产生：① 随机选择两个不同大质数 p, q，计算 $N = pq$

② $\varphi(N) = \varphi(p)\varphi(q) = (p-1)(q-1)$

③ 选择 $e < \varphi(N)$，有 $\gcd(e, \varphi(N)) = 1$，并求得 d，有 $ed \equiv 1 \pmod{\varphi(N)}$

④ 销毁 p, q，输出

public key: (e, N)

private key: (d, N)

加密：$C = M^e \bmod N$

解密：$M' = C^d \bmod N$

证明：即证 $M^{ed} \equiv M \pmod N$，已知 $ed \equiv 1 \pmod{\varphi(N)}$，即 $ed = k\varphi(N) + 1$，即证

$$M^{k\varphi(N)+1} \equiv M \pmod N$$

① 若 $\gcd(M, N) = 1$，则由欧拉定理：$M^{\varphi(N)} \equiv 1 \pmod N$，则

$$M^{k\varphi(N)+1} = M \cdot (M^{\varphi(N)})^k \equiv M \pmod N$$

② 若 $\gcd(M, N) \neq 1$，则 $M = kp$ 或 $M = kq$，且 $M < N$.

假设 $M = xp$，则 $x < q$. 由于 q 是质数，则由欧拉定理

$$M^{\varphi(q)} \equiv 1 \pmod q$$

则

$$M^{k\varphi(N)} = M^{k(p-1)(q-1)} = (M^{\varphi(q)})^{k(p-1)} \equiv 1 \pmod q$$

即

$$\begin{aligned} M^{k\varphi(N)+1} &= M(1 + uq) \\ &= M + uqM \\ &= M + uqxp \\ &= M + uxN \end{aligned}$$

即

$$M^{k\varphi(N)+1} \equiv M \pmod N$$

以上是 plain RSA，还不能直接用.

$C_1 \equiv M_1^e \bmod N$
$C_2 \equiv M_2^e \bmod N$
$\Rightarrow C_1 C_2 \equiv (M_1 M_2)^e \bmod N$ 可伪造出新的对.

Hardness of factoring N

- If p, q are known:
  - $\Rightarrow \phi(N)$ can be computed
  - $\Rightarrow d = e^{-1} \bmod \phi(N)$ can be computed
  - $\Rightarrow$ possible to compute e-th roots modulo N
- If p, q are *not* known:
  - $\Rightarrow$ computing $\phi(N)$ is as hard as factoring N
  - $\Rightarrow$ computing d is as hard as factoring N

  Q: Given d and e, can we factor N?
- Very useful for *public-key* cryptography

知道 $p,q \Rightarrow N \Rightarrow \phi(N)$

知道 $\phi(N) \Rightarrow p,q$ 因为 $\phi(N)=(p-1)(q-1)=pq-(p+q)+1$，所以 $\begin{cases} p+q=N-\phi(N)+1 \\ pq=N \end{cases}$

知道 $d \Rightarrow p,q$

以下4个问题的困难度是一样的.

① Given N, compute its factors p, q

② Given N, compute $\varphi(N)=(p-1)(q-1)$

③ Given N, e, compute d (satisfying $ed \equiv 1 \pmod{\varphi(N)}$)

④ Given N, find any $x \not\equiv \pm 1 \pmod N$ such that $x^2 \equiv 1 \pmod N$

①→②：能分解 N 得 p,q，易得 $\varphi(N)=(p-1)(q-1)$

②→③：能算出 $\varphi(N)$，由扩展欧几里得算法易得 d.

③→④：子程序构建

④→①：子程序构建

Square roots of unit modulo N

满足 $x^2 \equiv 1 \pmod N$ 的 x 有4个解（$x \in \mathbb{Z}_N^*$）. 因为 $\begin{cases} x \equiv \pm 1 \pmod p \\ x \equiv \pm 1 \pmod q \end{cases}$

其中有两个是 trivial 的，即 $x \equiv \pm 1 \pmod N$，而另外两个是 non-trivial 的，即 $x \not\equiv \pm 1 \pmod N$

证④→①：构建如下：

The reduction is rather simple. Suppose NTSRU is an algorithm that on input N returns a non-trivial square root of unity modulo N. Then we can factor N with the following algorithm:

```
FACTOR(N):
  x := NTSRU(N)
  return gcd(N, x + 1) and gcd(N, x − 1)
```

计求得 x 满足 $x^2 \equiv 1 \pmod N$ 但 $x \not\equiv \pm 1 \pmod N$

计返回 $p=\gcd(N, x+1)$ 和 $q=\gcd(N, x-1)$

$x^2 \equiv 1 \pmod N \Rightarrow N \mid x^2-1 \Rightarrow pq \mid (x+1)(x-1)$

$x \not\equiv 1 \pmod N \Longrightarrow pq \nmid (x-1)$

$x \not\equiv -1 \pmod N \Longrightarrow pq \nmid (x+1)$

因为 p, q 都是质数，故 $(x+1)(x-1)$ 含有因子 p 和 q.

但又由于 $(x+1)$ 和 $(x-1)$ 都不含因子 p 和 q，所以 $(x+1)$ 和 $(x-1)$ 恰自含 p, q 其一，即

$\begin{cases} x+1=kp \\ x-1=lq \end{cases}$ 或 $\begin{cases} x+1=lq \\ x-1=kp \end{cases} \Rightarrow \{\gcd(N, x+1), \gcd(N, x-1)\} = \{p, q\}$

证③->④：构建如下：

Suppose we have an algorithm FIND_D that on input (N, e) returns the corresponding exponent d. Then consider the following algorithm which uses FIND_D as a subroutine:

```
SRU(N):
  choose e as a random n-bit prime
  d := FIND_D(N, e)
  write ed − 1 = 2^s r, with r odd
  // i.e., factor out as many 2s as possible
  w ← Z_N
  if gcd(w, N) ≠ 1: // w ∉ Z*_N
     use gcd(w, N) to factor N = pq
     compute a nontrivial square root of unity using p & q
  x := w^r % N
  if x ≡_N 1 then return 1
  for i = 0 to s:
     if x^2 ≡_N 1 then return x
     x := x^2 % N
```

这个算法的返回值都是 square roots of unit.

for-loop 肯定能停止，因为最后一个值

$w^{2^s r} = w^{ed-1} \equiv w^{(ed-1) \bmod \varphi(N)} \equiv w^{1-1} \equiv 1 \pmod N$

令 $k = ed-1$，则 $\phi(N) \mid k$（因为 $ed \equiv 1 \pmod{\varphi(N)}$）

由欧拉定理知：$\exists x \in \mathbb{Z}_N^*$，$x^{\phi(N)} \equiv 1 \pmod N \Rightarrow x^k \equiv 1 \pmod N$

令 $k = 2^r \cdot u$，u 为奇数，$r \geq 1$.

重复随机取 $x \in \mathbb{Z}_N^*$ 并计算 $x^u, x^{2u}, \dots x^{2^r \cdot u} \bmod N$

取最大的 i 有 $x^{2^i u} \bmod N \neq 1$，并令 $y = i$，是满足 $y^2 \equiv 1 \pmod N$ 的

定义 $Bad \overset{def}{=} \{x \mid x^{2^i u} \equiv \pm 1 \bmod N\}$

若想令 y 为 non-trivial 的解，则随机取的 $x \notin Bad$

取 x 是一个随机算法，需要保证事件 $x \notin Bad$ 的概率较大.

需证明：Bad 是 $\mathbb{Z}_N^*$ 的严格子群

i) $Bad \neq \phi$ $(1 \in Bad)$

ii) 若 $x, x' \in Bad$，则 $(x \cdot x')^{2^i \cdot u} = x^{2^i u} \cdot x'^{2^i \cdot u} \equiv \pm 1 \bmod N$（乘法闭包）

故 Bad 是 $\mathbb{Z}_N^*$ 的严格子群

所以就会有 $|Bad| \leq \frac{|\mathbb{Z}_N^*|}{2}$，$x \notin Bad$ 概率就能 $\geq \frac{1}{2}$，也即取到 x 为 non-trivial 的概率 $\geq \frac{1}{2}$.

重复以上流程 n 次，最终能输出一个 non-trivial 的 x，有概率 $1 - 2^{-n}$.

The RSA assumption (formal)

- GenRSA: on input 1^n, outputs (N, e, d) with $N = pq$ a product of two distinct n-bit primes, with $ed = 1 \bmod \phi(N)$
- Experiment RSA-inv$_{A,GenRSA}(n)$:
  - Compute $(N, e, d) \leftarrow GenRSA(1^n)$
  - Choose uniform $y \in \mathbb{Z}_N^*$
  - Run $A(N, e, y)$ to get x
  - Experiment evaluates to 1 if $x^e = y \bmod N$
- The *RSA problem* is *hard* relative to GenRSA if for all PPT algorithms A,

$$\Pr[\text{RSA-inv}_{A,GenRSA}(n) = 1] < negl(n)$$

- Informally: given N, e, and uniform element $y \in \mathbb{Z}_N^*$, compute the e-th root of y
- *RSA assumption*: this is a hard problem!

Implementing GenRSA

- One way to implement GenRSA:
  - Generate uniform n-bit primes p, q
  - Set $N := pq$
  - Choose arbitrary e with $\gcd(e, \phi(N)) = 1$
  - Compute $d := e^{-1} \bmod \phi(N)$
  - Output (N, e, d)
- Choice of e?
  - Does *not* seem to affect hardness of the *RSA problem*
  - $e = 3$ or $e = 2^{16} + 1$ for *efficient* exponentiation

RSA and factoring

- If factoring moduli output by GenRSA is easy, then the *RSA problem* is easy relative to GenRSA
  - Factoring is easy $\Rightarrow$ RSA problem is easy
- Hardness of the *RSA problem* is not known to be implied by hardness of factoring
  - Possible factoring is hard but *RSA problem* is easy
  - Possible both are hard but *RSA problem* is "easier"
  - Currently, RSA is believed to be *as hard as factoring*

RSA问题的困难与分解困难不一定有关.
但认为RSA问题与分解同样困难.

Trapdoor functions

- **Definition 10.1** (*Trapdoor functions*) A *trapdoor function collection* is a collection $\mathcal{F}$ of finite functions such that every $f \in \mathcal{F}$ is a one-to-one function from some set S_f to a set T_f. The following properties are requried.
  - Efficient generation, computation and inversion
    There is a PPT algorithm G that on input 1^n outputs a pair (f, f^{-1}), where these are two $poly(n)$ size strings that describe the functions f, f^{-1}
  - Efficient sampling There is a PPT algorithm that given f can output a random element of S_f
  - One-wayness The function f is hard to invert without knowing the *invertion key*. For all PPT A there is a negligible function ϵ s.t.

$$\Pr_{(f,f^{-1}) \leftarrow_R G(1^n),\ x \leftarrow_R S_f}[A(1^n, f, f(x)) = x] < \epsilon(n)$$

RSA trapdoor function

- **Keys**: choose P, Q as random primes of length n, $N = P \cdot Q$. Choose e at random from $\{1, \ldots, \phi(N) - 1\}$ with $\gcd(e, \phi(N)) = 1$
  Forward **Key**: N, e
  Backward **Key**: d with $ed \equiv 1 \bmod \phi(N)$
  Function: $RSA_{N,e}(X) = X^e \pmod N$
  Inverse: If $Y = RSA_{N,e}(X) = X^e \bmod N$, then $Y^d \bmod N = X$.

- **RSA Assumption**: the RSA function is indeed a *trapdoor function*
  - This is stronger than the assumption that factoring is hard

Rabin's trapdoor function

- Assume that *factoring* random *Blum integers* is hard. A *Blum integer* is a number $n = pq$ where $p, q \equiv 3 \pmod 4$.

- Define $\mathcal{B}_n := \{P \in [1 \ldots 2^n] : P \text{ prime and } P \equiv 3 \bmod 4\}$

  The Factoring Axiom For every PPT algorithm A there is a negligible function ϵ s.t.

$$\Pr_{P,Q \leftarrow_R \mathcal{B}_n}[A(P \cdot Q) = \{P, Q\}] < \epsilon(n)$$

- **Keys**: choose P, Q as random primes of length n with $P, Q \equiv 3 \bmod 4$, $N = P \cdot Q$.
  Forward **Key**: N
  Backward **Key**: P, Q
  Function: $Y = RABIN_N(X) = X^2 \bmod N$, which is a permutation on QR_N, where QR_N denotes the set of quadratic residues modulo N

  Inverse: Compute $A = Y \bmod P$ and $B = Y \bmod Q$. Since $P, Q \equiv 3 \bmod 4$, let $P = 4t + 3$ and $Q = 4t' + 3$.

  Compute $X_1 = A^{t+1} \bmod P$ and $X_2 = B^{t'+1} \bmod Q$. Using CRT, we find X.

  We know that $X = S^2 \bmod P$, then

$$X_1 = (X^2)^{t+1} = S^{4(t+1)} = S^{P-1+2} = S^2 = X \bmod P.$$

  Similarly, $X_2 = S^2 = X \bmod Q$.

仅给Y，N，反向求出正确的X的概率只有$\frac{1}{4}$

Inverse: given Y, find X

step 1: 求出 $A = Y \bmod P$，$B = Y \bmod Q$

step 2: X有四个解 $X_1 = A^{t+1} \bmod P$，$X_2 = B^{t'+1} \bmod Q$

step 3: 对 $\begin{cases} X \equiv A^{t+1} \bmod P \\ X \equiv B^{t'+1} \bmod Q \end{cases}$ 使用中国剩余定理.

Lemma 10.2 Let X, Y be such that $X \not\equiv \pm Y \pmod{N}$ but $X^2 \equiv Y^2 \pmod{N}$. Then $\gcd(X - Y, N) \notin \{1, N\}$.

Proof. easy.

Theorem 10.3 (*One-wayness of Rabin's function*)
Rabin's function is a *trapdoor function* under the factoring axiom.

Proof. By contradiction. (see blackboard)

proof Lemma 10.2:
因为 $X \not\equiv \pm Y \pmod{N}$，故 $N \nmid (X-Y)$，$N \nmid (X+Y)$
所以 $\gcd(X-Y, N) \neq N$.
因为 $X^2 \equiv Y^2 \pmod{N}$，故有 $N \mid (X-Y)(X+Y)$
如果 $\gcd(X-Y, N) = 1$，那么 $N \mid (X+Y)$，是矛盾的，
所以 $\gcd(X-Y, N) \neq 1$
(更进一步说，$\gcd(X-Y, N) = p$ 或 q)

proof of Theorem 10.3
假设存在一个 Rabin's function 的 invertor A，有成功概率为 ε.
随机 uniform 选取 X，令 $X' = A(X^2)$
则至少有概率 $\frac{1}{2}$ 存在 X' 满足 $X'^2 \equiv X^2 \pmod{N}$ 但 $X' \neq \pm X \pmod{N}$
这个概率是因为 non-trivial 的解有 2 个而总共有 4 个解.