RSA (Rivest–Shamir–Adleman) public key cryptosystem:

- Key generation:
  - Pick two large prime numbers $p, q \in \mathbb{P}$.
  - Let $n = p \cdot q$.
  - Pick $e \in \{1, \ldots, \varphi(n)-1\}$ so, that $\gcd(e, \varphi(n)) = 1$.
  - Let $d = e^{-1} \pmod{\varphi(n)}$.

  Public key: (n, e). Secret key (n, d).
- Message space: $\{0, \ldots, n-1\}$.
- Encryption: $E_{(n,e)}(m) = m^e \bmod n$.
- Decryption: $D_{(n,d)}(c) = c^d \bmod n$.

Note that $\varphi(n) = (p-1) \cdot (q-1)$.

Example: let's pick $p = 43$, $q = 47$. Then $n = 2021$ and $\varphi(n) = 1932$.

Pick $e = 19$. Then $d = 1627$.

Let us encrypt the message $m = 503$. Then $503^{19} \bmod 2021 = 1233$.

Decryption: $1233^{1627} \bmod 2021 = 503$.

Exponentiation by repeated squaring:

$$a^b = \begin{cases} 1, & \text{if b=0} \\ (a^2)^{\frac{b}{2}}, & \text{if } b \text{ is even} \\ a \cdot (a^2)^{\frac{b-1}{2}}, & \text{if } b \text{ is odd} \end{cases}$$

Computation of a^b requires up to $2 \log b$ multiplications.

When doing RSA encryption or decryption, all computations are done in $\mathbb{Z}_n$.

Hence the maximum length of intermediate results is twice the length of n.

Decryption can be done faster:

During key generation, compute $p' = p^{-1} \pmod{q}$ and $q' = q^{-1} \pmod{p}$. Store $p, (p \cdot p'), q, (q \cdot q')$ in the secret key.

To decrypt c, compute $m_p = c^d \bmod p$ and $m_q = c^d \bmod q$.

Use chinese remainder theorem to find $c^d \bmod n$:

$$c^d \bmod n = ((q \cdot q') \cdot m_p + (p \cdot p') \cdot m_q) \bmod n \ .$$

Computing $c^d \bmod p$ is four times faster than computing $c^d \bmod n$.

Indeed, the numbers involved are only half as long and multiplication has quadratic complexity.

Hence the method is twice as fast than the previous one.

Typical sizes of the key:

- RSA — n is usually taken 1024–4096 bits long. p and q have then half that length.
- The suggested lengths of a key in a knapsack cryptosystem was about 300, the elements of the superincreasing knapsack should be from 2^{300} to 2^{600}.
  - The key is much longer than for RSA.
  - But the operations are faster.
- Modern symmetric cryptosystems use 128–256-bit keys.

As asymmetric cryptosystems usually work much slower, the following "hybrid" method is usually used to encrypt a plaintext x with a public key k_p.

Let a symmetric cryptosystem be fixed. It may be a block cipher with a fixed mode of operation.

1. Generate a new key k_s of the symmetric cryptosystem.
2. Let $y = E_{k_s}^{\text{symm}}(x)$.
3. Let $k' = E_{k_p}^{\text{asymm}}(k_s)$.
4. The cryptotext is (k', y).

The asymmetric cryptosystem is usually also a block cipher. The "message" k_s is usually short enough to fit into a single block.

Why does RSA "work"?

Theorem (Euler). If $\gcd(a, n) = 1$ then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Proof. $a \in \mathbb{Z}_n^*$. $|\mathbb{Z}_n^*| = \varphi(n)$.

Corollary (Fermat little theorem). If $p \in \mathbb{P}$ does not divide a then $a^{p-1} \equiv 1 \pmod{p}$.

Let (n, e) and (n, d) be the RSA public and secret keys.

We have $ed \equiv 1 \pmod{\varphi(n)}$. I.e. $ed = k \cdot \varphi(n) + 1$ for some $k \in \mathbb{N}$.

Let $m \in \{0, \ldots, m-1\}$ be a message. If $\gcd(m, n) = 1$ then

$$D_{(n,d)}(E_{(n,e)}(m)) = (m^e)^d = m^{ed} = m^{k\varphi(n)+1} = \\ m \cdot (m^{\varphi(n)})^k = m \cdot 1^k = m \pmod{n}$$

If $m = 0$ then also $(m^e)^d = 0$.

If $m \neq 0$ and $\gcd(m, n) > 1$ then $\gcd(m, n) \in \{p, q\}$. In this case we have managed to factor n.

To generate a RSA key, we need to generate two large primes. How?

Theorem (Chebyshev). Let $\pi(n) = |\mathbb{P} \cap \{1, \ldots, n\}|$. Then

$$\pi(n) \geqslant \frac{n}{\ln n} \qquad \text{and} \qquad \lim_{n \to \infty} \frac{\pi(n) \cdot \ln n}{n} = 1 \ .$$

$\ln 2^{512} \approx 355$. I.e. about every 355th 512-bit number is prime.

I.e. about every 177th 512-bit odd number is a prime.

A viable strategy to generate a prime is to generate random odd numbers and test their primality.

Primality testing is doable in polynomial time (in length of the number). But the degree of the polynomial is high.

Fortunately, there exist efficient Monte-Carlo algorithms.

A probabilistic algorithm $\mathcal{A}$ for testing whether a bit string x belongs to some set $P \subseteq \{0,1\}^*$ is a Monte-Carlo algorithm if $\exists \varepsilon > 0 \; \forall x \in \{0,1\}^*$:

- If $x \in P$ then $\Pr[\mathcal{A}(x) = \mathsf{true}] = 1$.
- If $x \notin P$ then $\Pr[\mathcal{A}(x) = \mathsf{false}] \geqslant \varepsilon$.

If $x \notin P$ and we execute $\mathcal{A}(x)$ n times then the probability of getting true all times is at most $(1-\varepsilon)^n$.

Fermat little theorem said:

$$a^{p-1} \equiv 1 \pmod{p} \text{ if } \gcd(a, p) = 1$$

So, is the following a Monte-Carlo algorithm for testing the primality of odd n?

1. Generate a random $w \in \{1, \ldots, n-1\}$, such that $\gcd(w, n) = 1$.
   - If $\gcd(w, n) > 1$ then n is definitely composite. Return "no".
2. If $w^{n-1} \equiv 1 \pmod{n}$ then return "yes" else return "no".

If $n \in \mathbb{P}$ then the algorithm always returns "yes".

Let $w \in \mathbb{Z}_n^*$.

n is a pseudoprime to base w if $w^{n-1} \equiv 1 \pmod{n}$.

In this case w is a witness for the primality of n. Otherwise it is a witness for the compositeness of n.

Lemma. If n has witnesses for compositeness then at least half of the elements of $\mathbb{Z}_n^*$ are witnesses for the compositeness of n.

Proof. Let $W_p \subset \mathbb{Z}_n^*$ be the set of witnesses for the primality of n. Let $w \in \mathbb{Z}_n^* \backslash W_p$. We have $w^{n-1} \not\equiv 1 \pmod{n}$.

Consider the set

$$W_c = \{w \cdot w_p \,|\, w_p \in W_p\} \ .$$

Then $|W_c| = |W_p|$. We have

$$(w \cdot w_p)^{n-1} = w^{n-1} \cdot w_p^{n-1} \equiv w^{n-1} \cdot 1 \not\equiv 1 \pmod{n}$$

Hence all the elements of W_c witness the compositeness of n. □

Hence for a $n \in \mathbb{N}$ there are three possible cases:

1. n is prime and all elements of $\mathbb{Z}_n^*$ witness that;
2. n is composite, but all elements of $\mathbb{Z}_n^*$ witness for the primality of n;
3. n is composite and at least half of the elements of $\mathbb{Z}_n^*$ witness that.

If the second case were impossible then the presented algorithm would be a Monte-Carlo algorithm for primality.

Unfortunately, there exist composite numbers n, such that $w^{n-1} \equiv 1 \pmod{n}$ for all $w \in \mathbb{Z}_n^*$.

They are called Carmichael numbers. There are infinitely many of them, the smallest is 561.

Still, the presented test is suitable if numbers from a trusted source are tested.

Let $p \in \mathbb{P}$ and $a \in \mathbb{Z}_p^*$. Then a is a quadratic residue (*ruut-jääk*) if there exists $b \in \mathbb{Z}_p^*$ such that $b^2 \equiv a \pmod{p}$.

Half of the elements of $\mathbb{Z}_p^*$ are quadratic residues and half are non-residues.

The Legendre symbol $\left(\frac{a}{p}\right)$ for $p > 2$ is defined by

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p \text{ divides } a \\ 1, & \text{if } a \text{ is a quadratic residue modulo } p \\ -1, & \text{if } a \text{ is a quadratic non-residue modulo } p. \end{cases}$$

The Legendre symbol satisfies

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

and hence also

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \ .$$

In the following, we do not prove some number-theoretic claims. The proofs are given in the Number Theory course (MTPM.01.009).

Generalization: let $n = p_1^{i_1} \cdots p_k^{i_k}$ where $p_1, \ldots, p_k \in \mathbb{P} \backslash \{2\}$.
Let $a \in \mathbb{Z}$. The Jacobi symbol $\left(\frac{a}{n}\right)$ is defined by

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{i_1} \cdots \left(\frac{a}{p_k}\right)^{i_k}.$$

It satisfies the following:

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right) \qquad \left(\frac{1}{n}\right) = 1 \qquad \left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$$

$$\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}} \qquad \left(\frac{m}{n}\right) = (-1)^{\frac{(m-1)(n-1)}{4}} \left(\frac{n}{m}\right)$$

if $m, n \geqslant 3$ are odd numbers.

The second row of identities once more:

$$\left(\frac{2}{n}\right) = \begin{cases} 1, & \text{if } n \bmod 8 \in \{1, 7\} \\ -1, & \text{if } n \bmod 8 \in \{3, 5\} \end{cases}$$

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right), & \text{if } m \bmod 4 = n \bmod 4 = 3 \\ \left(\frac{n}{m}\right), & \text{otherwise.} \end{cases}$$

These identities allow us to compute $\left(\frac{a}{n}\right)$ without factoring n.

The algorithm resembles Euclid's algorithm.

Solovay-Strassen primality test (for an odd n):

1. Generate a random $w \in \{1, \ldots, n-1\}$.
2. Let $a = w^{\frac{n-1}{2}} \bmod n$. Let $b = \left(\frac{w}{n}\right)$.
3. If $a \equiv b \pmod{n}$ then return "yes", otherwise "no".

We'll show that this primality test is a Monte-Carlo algorithm with $\varepsilon \geqslant \frac{1}{2}$.

If w witnesses n's primality according to S-S's test then it also witnesses n's primality according to Fermat's test.

Indeed if $w^{\frac{n-1}{2}} \equiv \left(\frac{w}{n}\right) \pmod{n}$ and $w \in \mathbb{Z}_n^*$ then

$$w^{n-1} = (w^{\frac{n-1}{2}})^2 \equiv \left(\frac{w}{n}\right)^2 = \begin{cases} 1^2 \\ (-1)^2 \end{cases} = 1 \pmod{n}$$

Theorem. If n is odd composite then at least half of the elements of $\mathbb{Z}_n^*$ witness the compositeness of n.

Proof. First we show that there exists a $w \in \mathbb{Z}_n^*$ witnessing the compositeness of n. There are two cases:

1. case. $n = p_1 \cdots p_k$ where $p_1, \ldots, p_k \in \mathbb{P}$ are all different. Let u be a quadratic non-residue modulo p_1. Let w satisfy

$$w \equiv u \pmod{p_1} \qquad w \equiv 1 \pmod{p_2 \cdots p_k} \ .$$

(use CRT to find such $w \in \mathbb{Z}_n^*$). Then

$$\left(\frac{w}{n}\right) = \left(\frac{w}{p_1}\right) \cdots \left(\frac{w}{p_k}\right) = \left(\frac{u}{p_1}\right)\left(\frac{1}{p_2}\right) \cdots \left(\frac{1}{p_k}\right) = -1 \ .$$

To get $w^{\frac{n-1}{2}} \equiv -1 \pmod{n}$ we need $w^{\frac{n-1}{2}} \equiv -1 \pmod{p_i}$ for $1 \leqslant i \leqslant k$.

But $w^{\frac{n-1}{2}} \equiv 1 \pmod{p_i}$ for $2 \leqslant i \leqslant k$.

2. case. There exists a $p \in \mathbb{P}$ and $k \geqslant 2$, such that p^k divides n (and p^{k+1} does not divide n).

Let $w = 1 + n/p$. Then $w \equiv 1 \pmod{q}$ for any $q \in \mathbb{P}$ dividing n.

$$\left(\frac{w}{n}\right) = \prod_{q \in \mathbb{P}} \left(\frac{w}{q}\right)^{\operatorname{ind}_q n} = \prod_{q \in \mathbb{P}} 1^{\operatorname{ind}_q n} = 1$$

To get $w^{\frac{n-1}{2}} \equiv 1 \pmod{n}$ we need $w^{\frac{n-1}{2}} \equiv 1 \pmod{q^{\operatorname{ind}_q n}}$ for all $q \in \mathbb{P}$ dividing n.

Consider $w^{\frac{n-1}{2}} \bmod p^k$. The order of the group $\mathbb{Z}^*_{p^k}$ is $p^{k-1}(p-1)$, hence the order of w in $\mathbb{Z}^*_{p^k}$ can be $(p-1)$ or p^i or $p^i(p-1)$ for some $i \in \{1, \ldots, k-1\}$. We show that

$$w^p \equiv 1 \pmod{p^k}$$

hence the order of w in $\mathbb{Z}^*_{p^k}$ is p. As p does not divide $\frac{n-1}{2}$, we cannot have $w^{\frac{n-1}{2}} \equiv 1 \pmod{p^k}$.

$$w^p = (1 + lp^{k-1})^p = \sum_{i=0}^{p} \binom{p}{i} (lp^{k-1})^i =$$

$$1 + p \cdot l \cdot p^{k-1} + \sum_{i=2}^{p} \binom{p}{i} l^i p^k \cdot p^{(k-1)(i-1)-1} =$$

$$1 + p^k \left(l + \sum_{i=2}^{p} \binom{p}{i} l^i p^{(k-1)(i-1)+1} \right) \equiv 1 \pmod{p^k}$$

We have found a witness $w \in \mathbb{Z}_n^*$ for the compositeness of n. I.e. $w^{\frac{n-1}{2}} \not\equiv \left(\frac{w}{n}\right) \pmod{n}$. We continue as before:

Let $W_p \subset \mathbb{Z}_n^*$ be the set of witnesses for the primality of n.

Consider the set

$$W_c = \{w \cdot w_p \,|\, w_p \in W_p\} \ .$$

Then $|W_c| = |W_p|$. We have

$$(w \cdot w_p)^{\frac{n-1}{2}} = w^{\frac{n-1}{2}} \cdot {w_p}^{\frac{n-1}{2}} \equiv w^{\frac{n-1}{2}} \cdot \left(\frac{w_p}{n}\right) \not\equiv$$

$$\left(\frac{w}{n}\right)\left(\frac{w_p}{n}\right) = \left(\frac{w \cdot w_p}{n}\right) \pmod{n}$$

Hence all the elements of W_c witness the compositeness of n. $\square$

Let x be a random k-bit odd integer. Consider the following events:

- $\mathbf{A} \equiv x$ is composite;
- $\mathbf{B} \equiv$ the S-S test returns "x is prime" m times in a row.

We have determined that $\Pr[\mathbf{B}|\mathbf{A}] \leqslant 2^{-m}$.

Our confidence after running the S-S test m times is better reflected by the probability $\Pr[\mathbf{A}|\mathbf{B}]$.

The complementary event $\overline{\mathbf{A}}$ denotes that x is prime.

$$\Pr[\overline{\mathbf{A}}] = \frac{\pi(2^k) - \pi(2^{k-1})}{2^{k-1}} \approx \frac{1}{2^{k-1}}\left(\frac{2^k}{k\ln 2} - \frac{2^{k-1}}{(k-1)\ln 2}\right) = \frac{k-2}{k(k-1)\ln 2} \approx \frac{1}{k\ln 2}$$

$$\Pr[\mathbf{A}|\mathbf{B}] = \frac{\Pr[\mathbf{A}]\Pr[\mathbf{B}|\mathbf{A}]}{\Pr[\mathbf{B}]} = \frac{\Pr[\mathbf{A}]\Pr[\mathbf{B}|\mathbf{A}]}{\Pr[\mathbf{B}|\mathbf{A}]\Pr[\mathbf{A}] + \Pr[\mathbf{B}|\overline{\mathbf{A}}]\Pr[\overline{\mathbf{A}}]} = \frac{\Pr[\mathbf{A}]}{\Pr[\mathbf{A}] + \frac{\Pr[\mathbf{B}|\overline{\mathbf{A}}]\Pr[\overline{\mathbf{A}}]}{\Pr[\mathbf{B}|\mathbf{A}]}} \lesssim \frac{1 - \frac{1}{k\ln 2}}{1 - \frac{1}{k\ln 2} + 2^m\frac{1}{k\ln 2}} = \frac{k\ln 2 - 1}{k\ln 2 - 1 + 2^m}$$

I.e. $\Pr[\mathbf{A}|\mathbf{B}]$ may be somewhat larger than $\Pr[\mathbf{B}|\mathbf{A}]$.

The idea of the Miller-Rabin primality test (for an odd n) is to find a square root of 1 modulo n.

If n is prime then $\sqrt{1} \pmod{n}$ has two values — ± 1. We call these values the trivial square roots of 1.

If n is not prime then $\sqrt{1} \pmod{n}$ has more values.

If we have found $x \notin \{1, -1\}$, such that $x^2 \equiv 1 \pmod{n}$ then n must be composite.

Let s and r be defined by $n - 1 = 2^s \cdot r$, where r is odd.

Miller-Rabin primality test for $n = 2^s r + 1$ where $s \geqslant 1$ and r is odd:

1. Generate a random $w \in \{1, \ldots, n-1\}$.
2. Compute the values $u_i = w^{2^i r} \bmod n$ for $0 \leqslant i \leqslant s-1$.
   - Compute $u_0 = w^r \bmod n$ and $u_{i+1} = u_i^2 \bmod n$.
3. If one of the following holds
   - $u_0 \equiv 1 \pmod{n}$;
   - for some i, $u_i \equiv -1 \pmod{n}$;

   then return "yes" else return "no".

We'll show that this primality test is a Monte-Carlo algorithm with $\varepsilon \geqslant \frac{3}{4}$.

What is going on?

Define also $u_s = w^{n-1} \bmod n$.

For all $i \in \{1, \ldots, s\}$, u_{i-1} is one of the square roots of u_i.

If $u_s \not\equiv 1$ then n is composite. Then also none of the u_i-s can be ± 1. Assume now that $u_s = 1$.

If $u_i \equiv -1$ then the elements $u_{i+1}, \ldots, u_s$ must all be 1. Then we have found -1 as the square root of 1.

If $u_0 = 1$ then we have only found 1 as the square root of 1.

Otherwise $u_{i-1} \neq \pm 1$, $u_i = 1$ for some $i \in \{1, \ldots, s\}$. Then we have found a nontrivial square root of 1.

Lemma. Let G be a cyclic group, $|G| = m$. Then $x^k = 1$ has exactly $\gcd(k, m)$ solutions in G.

Proof. Let g be a generator of G. Then g^j is a solution to $x^k = 1$ iff $m \mid jk$. Among $\{0, \ldots, m-1\}$, there are exactly $\gcd(k, m)$ such values for j. $\square$

Lemma. Let $p \in \mathbb{P} \backslash \{2\}$, let $p-1 = 2^s \cdot r$ where r is odd. Consider the equation $x^{2^u t} = -1$ where t is odd. In $\mathbb{Z}_p$, it has the following number of solutions:

- 0, if $u \geqslant s$;
- $2^u \gcd(r,t)$ if $u < s$.

Proof. Let g be a generator of $\mathbb{Z}_p^*$. We are looking for the number of j-s in $\{0, \ldots, p-2\}$ satisfying $(g^j)^{2^u t} = g^{\frac{p-1}{2}}$ or

$$j \cdot 2^u \cdot t \equiv 2^{s-1} \cdot r \pmod{2^s r} \ .$$

If $u \geqslant s$ then we divide the equation and the modulus by 2^{s-1} and obtain

$$j \cdot 2^{u-s+1} \cdot t \equiv r \pmod{2r}$$

here the left hand side is even, but the right hand side is odd. As the modulus is also even, there can be no solutions.

Otherwise let $d = \gcd(r, t)$, then $\gcd(2^u t, 2^{s-1} r) = 2^u d$. Divide everything by $2^u d$ and get

$$j \cdot (t/d) \equiv 2^{s-1-u} \cdot (r/d) \pmod{2^{s-u}(r/d)} \ .$$

This has a unique solution j modulo $2^{s-u}(r/d)$ (because $(t/d) \perp 2^{s-u}(r/d)$). Hence it has $2^u d$ solutions modulo $2^s r$. $\square$

Theorem. Let n be an odd composite number. Then at most one quarter of elements $w \in \{1, \ldots, n-1\}$ witness the primality of n according to M-R test.

Proof. Let $n - 1 = 2^s r$ with r odd. If w witnesses the primality of n then $u_s \equiv 1 \pmod{n}$.

Consider the following three cases:

1. $p^2 \mid n$ for some odd prime p;
2. $n = pq$ for distinct primes p and q;
3. $n = p_1 \cdots p_k$ for $k \geqslant 3$ distinct primes.

1. case. Let w witness the primality of n, then $u_s = w^{n-1} \equiv 1 \pmod{n}$. Then also $w^{n-1} \equiv 1 \pmod{p^2}$.

Consider the equation $w^{n-1} = 1$. In the cyclic group $\mathbb{Z}_{p^2}^*$ it has exactly $d = \gcd(n-1, p(p-1))$ solutions.

As $p \perp n-1$, we must have $d \leqslant p-1$. Modulo n, the number of solutions is at most $(n/p^2) \cdot d$. The fraction of the solutions is at most

$$\frac{n}{p^2} \cdot \frac{d}{n-1} \leqslant \left(\frac{n-1}{n} \cdot \frac{p^2}{p^2-1}\right) \cdot \frac{n}{p^2} \cdot \frac{p-1}{n-1} = \frac{p-1}{p^2-1} = \frac{1}{p+1} \leqslant \frac{1}{4}$$

because p is an odd prime.

2. case. Let $p-1 = 2^{s'} r'$ and $q-1 = 2^{s''} r''$ with r', r'' odd. Assume w.l.o.g. that $s' \leqslant s''$.

If w is a witness for the primality of n then one of the following holds:

1. $w^r \equiv 1 \pmod{p}$ and $w^r \equiv 1 \pmod{q}$.
2. $w^{2^i r} \equiv -1 \pmod{p}$ and $w^{2^i r} \equiv -1 \pmod{q}$ for some $i \in \{0, \ldots, s-1\}$.

Indeed, using the chinese remainder theorem we get $\ldots \equiv 1 \pmod{n}$ in the first case and $\ldots \equiv -1 \pmod{n}$ in the second.

These cases have the following number of solutions:

1. $\gcd(r, p-1) \cdot \gcd(r, q-1) = \gcd(r, r') \cdot \gcd(r, r'') \leqslant r'r''$.
2. $2^i \gcd(r, r') \cdot 2^i \gcd(r, r'') \leqslant 4^i r'r''$ (for each i where $i < \min(s', s'') = s'$).

The total is at most $r'r'' + \sum_{i=0}^{s'-1} 4^i r'r'' = r'r''(1 + \frac{4^{s'}-1}{3})$.

Their fraction is

$$\frac{r'r''(1 + \frac{4^{s'}-1}{3})}{n-1} \leqslant \frac{r'r''(1 + \frac{4^{s'}-1}{3})}{(p-1)(q-1)} = \frac{r'r''(1 + \frac{4^{s'}-1}{3})}{2^{s'+s''}r'r''} = \frac{1}{2^{s'+s''}} \cdot \frac{4^{s'}+2}{3} \ .$$

Either $s' < s''$ or $s' = s''$. If $s' < s''$ then

$$\frac{1}{2^{s'+s''}} \cdot \frac{4^{s'}+2}{3} \leqslant \frac{1}{2^{2s'+1}} \cdot \frac{4^{s'}+2}{3} =$$

$$\frac{1}{2 \cdot 3} + \frac{1}{2^{2s'+1}} \cdot \frac{2}{3} \leqslant \frac{1}{6} + \frac{1}{12} = \frac{1}{4} \ .$$

If $s' = s''$ then either $\gcd(r, r') < r'$ or $\gcd(r, r'') < r''$. Indeed, assume the contrary, i.e. $r' \mid r$ and $r'' \mid r$. Then

$$n - 1 = 2^s r = pq - 1 = (p-1)q + (q-1) = \\ 2^{s'} r' q + (q-1) \equiv q - 1 \pmod{r'} \ .$$

We also have $2^s r \equiv 0 \pmod{r'}$, hence $q - 1 \equiv 0 \pmod{r'}$, i.e. $r' \mid (q-1)$.

We got $r' \mid 2^{s''} r''$. As $r' \perp 2^{s''}$, we must have $r' \mid r''$.

Analogously we can get $r'' \mid r'$. I.e. $r' = r''$ and $p = q$. This contradicts our premises.

We have $\gcd(r, r') \cdot \gcd(r, r'') < r'r''$. As all prime factors are odd, we can estimate $\gcd(r, r') \cdot \gcd(r, r'') \leqslant \frac{r'r''}{3}$.

Our two cases for w witnessing the primality of n have at most

$$\frac{r'r''}{3} + \sum_{i=0}^{s'-1} 4^i \frac{r'r''}{3} = r'r'' \frac{4^{s'} + 2}{9}$$

solutions. Their fraction is at most

$$\frac{r'r'' \frac{4^{s'}+2}{9}}{2^{2s'} r'r''} = \frac{1}{9} + \frac{2}{9 \cdot 2^{2s'}} \leqslant \frac{1}{9} + \frac{1}{18} < \frac{1}{4} \ .$$

3. case. Let $p_j - 1 = 2^{s_j} r_j$ where r_j is odd. Assume w.l.o.g. that $s_1 = \min(s_1, \ldots, s_k)$.

Denote $R = r_1 \cdots r_k$.

If w witnesses the primality of n then one of the following holds:

1. $w^r \equiv 1 \pmod{p_j}$ for $1 \leqslant j \leqslant k$.
2. $w^{2^i r} \equiv -1 \pmod{p_j}$ for all $j \in \{1, \ldots, k\}$ and for some $i \in \{0, \ldots, s_1 - 1\}$.

These two cases have at most the following number of solutions:

1. $\prod_{j=1}^{k} \gcd(r, p_j - 1) = \prod_{j=1}^{k} \gcd(r, r_j) \leqslant\leqslant \prod_{j=1}^{k} r_j \leqslant R.$
2. $\prod_{j=1}^{k} 2^i \gcd(r, r_j) \leqslant 2^{ki} R$ (for each $i \in \{0, \ldots, s_1 - 1\}$).

And the total is at most

$$R + \sum_{i=0}^{s_1 - 1} 2^{ki} R = R\left(1 + \frac{2^{k s_1} - 1}{2^k - 1}\right) \ .$$

And the fraction is at most

$$\frac{R\left(1+\frac{2^{ks_1}-1}{2^k-1}\right)}{n-1} \leqslant \frac{R\left(1+\frac{2^{ks_1}-1}{2^k-1}\right)}{\prod_{j=1}^{k}(p_j-1)} = \frac{R\left(1+\frac{2^{ks_1}-1}{2^k-1}\right)}{2^{s_1+\ldots+s_k}R} =$$

$$2^{-s_1-\ldots-s_k}\left(1+\frac{2^{ks_1}-1}{2^k-1}\right) \leqslant 2^{-ks_1}\left(\frac{2^k-2}{2^k-1}+\frac{2^{ks_1}}{2^k-1}\right) \leqslant$$

$$2^{-k}\frac{2^k-2}{2^k-1}+\frac{1}{2^k-1} = \frac{2-2^{1-k}}{2^k-1} = 2^{1-k} \leqslant 2^{1-3} = \frac{1}{4} \ .$$

□

The M-R test requires only half as many trials as the S-S test to achieve the same level of confidence.

How could we try to break RSA? Given (n, e), we could try to do one of the following:

1. factor n;
2. find $\varphi(n)$;
3. find $d = e^{-1} \pmod{\varphi(n)}$;
4. devise a method that, given $m^e \bmod n$ produces m.

1–3 are equivalent. 4 is not known to be equivalent to factoring (but we'll come back to it).

4 is the RSA problem — given n, e, $m^e \bmod n$, find m.

If we could find $\varphi(n)$ then we could factor n. We'd have the system of equations

$$\begin{cases} pq = n \\ (p-1)(q-1) = \varphi(n) \end{cases}$$

From which $p + q = n + 1 - \varphi(n)$. Then p and q are the solutions of the following quadratic equation (over $\mathbb{R}$):

$$x^2 - (n + 1 - \varphi(n))x + n = 0 \ .$$

Given n, e and d, we can factor n as follows.

We have $ed = k\varphi(n) + 1$ for some $k \in \mathbb{Z}$. We try to find a non-trivial square root of 1 modulo n.

If $x^2 \equiv 1 \pmod{n}$, but $x \not\equiv \pm 1 \pmod{n}$ then
$0 \equiv x^2 - 1 = (x+1)(x-1) \pmod{n}$.

We have $n \mid (x+1)(x-1)$, but $x \in \{2, \ldots, n-2\}$. Hence neither $(x+1)$ nor $(x-1)$ is a multiple of n.

We must have $(x+1) = k'p$ and $(x-1) = k''q$ (or vice versa). We'll find p and q by computing $\gcd(x+1, n)$ and $\gcd(x-1, n)$.

Let $ed - 1 = 2^s r$ where r is odd.

Pick a random $w \in \{1, \ldots, n-1\}$. If $\gcd(w, n) \neq 1$ then we have factored n.

Otherwise let $u_i = w^{2^i r} \bmod n$. $(0 \leqslant i \leqslant s)$ If there exists an i, such that $u_i \equiv 1$ and $u_{i-1} \not\equiv \pm 1 \pmod{n}$ then we have found a non-trivial square root of 1.

What is the fraction of w-s that give us a non-trivial square root of 1?

Note that we certainly have $u_s \equiv 1 \pmod{n}$.

If w does not give a non-trivial square root of n then one of the following holds:

- $w^r \equiv 1 \pmod{p}$ and $w^r \equiv 1 \pmod{q}$.
- $w^{2^i r} \equiv -1 \pmod{p}$ and $w^{2^i r} \equiv -1 \pmod{q}$ for some $i \in \{0, \ldots, s-1\}$.

We have already counted that the fraction of such w-s is at most $1/4$.

Hence the fraction of w-s giving us a non-trivial square root of n is at least $3/4$.

Let us review some factoring algorithms.

Trial division — to factor n, we try to divide it with all prime numbers not larger than $\sqrt{n}$. Feasible only if n is small.

In the following let n (the number to factor) be odd. Our task is to find a non-trivial factor of n.

Pollard's $p-1$ algorithm: Let $B \in \mathbb{N}$ be a parameter ("upper bound").

1. Let $a_i = 2^{i!} \bmod n$ for $1 \leqslant i \leqslant B$.
   - Compute: $a_1 = 2$ and $a_i = a_{i-1}^i \bmod n$.
2. Let $d_i = \gcd(a_i - 1, n)$.
3. If $1 < d_i < n$ for some d_i then return d_i else fail.

The method works if n has a prime factor p, such that $(p-1)$ has only small factors.

An integer with only small prime factors is called smooth.

If $q \mid (p-1)$ and q is a prime power then $q \leqslant B$ must hold.

In this case $(p-1) \mid B!$. We have $a_B \equiv 2^{B!} \pmod{n}$, hence also $a_B \equiv 2^{B!} \pmod{p}$.

This, together with $2^{p-1} \equiv 1 \pmod{p}$ and $(p-1) \mid B!$ gives us $a_B \equiv 1 \pmod{p}$ and $p \mid (a_B - 1)$.

Together with $p \mid n$ we get $p \mid \gcd(a_B - 1, n)$. This gcd is a non-trivial factor of n.

To protect the RSA modulus $n = pq$ from factoring with the Pollard's $p - 1$ algorithm we must ensure that $p - 1$ and $q - 1$ have large factors.

A common way to do this is to let p' and q' be prime numbers of appropriate size and define $p = 2p' + 1$, $q = 2q' + 1$.

A prime p is safe is $\frac{p-1}{2}$ is also prime.

Dixon's algorithm attempts to find numbers $x, y \in \mathbb{Z}_n$, such that $x^2 = y^2$, but $x \neq \pm y$. Then $\gcd(x+y, n)$ and $\gcd(x-y, n)$ are non-trivial factors of n.

1. Somehow fix a set $\mathcal{B} \subset \mathbb{P}$ of "small" primes.
2. Search for elements $x \geqslant \sqrt{n}$, such that all prime factors of $x^2 \bmod n$ are in $\mathcal{B}$. Let $\mathrm{d}(x, p)$ be the degree of $p \in \mathcal{B}$ in the prime factorization of $x^2 \bmod n$.
3. Choose elements $x_1, \ldots, x_k$, such that $s_i = \sum_{i=1}^{k} \mathrm{d}(x_i, p)$ is even for all $p \in \mathcal{B}$.
4. We have $(x_1 \cdots x_k)^2 \equiv \left(\prod_{p \in \mathcal{B}} p^{\frac{s_i}{2}} \right)^2 \pmod{n}$. Hopefully these two numbers under $(\cdot)^2$ are neither equal nor negations of each other.

The set $\mathcal{B}$ will simply contain the first d primes for some d. There is a trade-off:

- If d is small then we need less different x-s to come up with a set $\{x_1, \ldots, x_k\}$ whose product is a square, but $x^2 \bmod n$ has all prime factors in $\mathcal{B}$ for less x-s.
- If d is large then a larger fraction of $x^2 \bmod n$-s has all prime factors in $\mathcal{B}$ but we need more different x-s to come up with the set $\{x_1, \ldots, x_k\}$.

The search for x-s will typically just consider the numbers $j + \lfloor \sqrt{n} \rfloor$, where $j \in \mathbb{N}$.

A variant of the Dixon's algorithm is the quadratic sieve.

An RSA modulus $n = pq$ will be factored using this method if $p \approx q$.

In this case $\frac{p+q}{2}$ is just a little bit larger than $\sqrt{n}$, so it will be considered.

Also, $\left(\frac{p+q}{2}\right)^2 \bmod n = \frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$ which is a small number (hence it is likely that all its factors are in $\mathcal{B}$) and moreover a perfect square.

The set of x_i-s can then be a single number — $\frac{p+q}{2}$.

To thwart this attack, the lengths of p and q should differ by a few bits.

We may try to solve the RSA problem (find m from n, e, $c = m^e \bmod n$) as follows (cycling attack):

- Compute $c^e \bmod n$, $c^{e^2} \bmod n$, $c^{e^3} \bmod n$, $c^{e^4} \bmod n$ etc. until $c^{e^k} \equiv c \pmod{n}$ for some k.
  - Compute: $c^{e^i} \bmod n = \left(c^{e^{i-1}}\right)^e \bmod n$.
- Then $c^{e^{k-1}}$ is a suitable m.

Generalization of the attack: find the smallest k, such that $\gcd(c^{e^k} - c, n) > 1$. If this gcd is n then we have solved the RSA problem, otherwise we have factored n.

The second case supposedly appears much more frequently, hence this attack is equivalent to factoring.

But factoring is no harder than solving the RSA-problem using a generic algorithm.

A generic algorithm $\mathcal{A}$ accesses an oracle $\mathcal{O}$ to perform the operations in $\mathbb{Z}_n$.

The internal state of $\mathcal{O}$ is a list L of elements of $\mathbb{Z}_n$ (initially $[c]$).

$\mathcal{A}$ can give $\mathcal{O}$ commands $\langle i, j, \circ \rangle$, where $\circ$ is an arithmetic operation. $\mathcal{O}$ then computes $L_i \circ L_j$ and appends it to L.

$\mathcal{A}$ can also ask $\mathcal{O}$ whether L_i and L_j are equal.

Finally, $\mathcal{A}$ must output an index i, such that $L_i^e = c$.

Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. $f(x)$ contains partial information about x.

Is it possible to extracting some non-trivial partial information about m from n, e, $c = m^e \bmod n$?

I.e. given n, e, c, compute $f(m)$.

We show that for some interesting f-s, extracting partial information is as hard as finding the message m.

We consider functions

$$parity(m) = m \bmod 2$$

$$half_n(m) = \left\lfloor \frac{m}{\lceil n/2 \rceil} \right\rfloor \quad .$$

Suppose that there exists an efficient algorithm $\mathcal{O}$, such that $\mathcal{O}(c, n, e) = half_n(m)$, where $m^e \equiv c \pmod{n}$.

We can compute m from c, n, e as follows:

- Let $b_i = \mathcal{O}(2^{ie} c \bmod n, n, e)$.
  - Compute: 2^e, $2^{0e} c = c$ and $2^{ie} c = (2^{(i-1)e}) c \cdot 2^e$.
    - ... everything modulo n
- Return

$$n \cdot \sum_{i=0}^{\lfloor \log n \rfloor} \frac{b_i}{2^{i+1}} \quad .$$

- By querying $\mathcal{O}(c, n, e)$ we'll know whether $0 \leqslant m < n/2$ or $n/2 \leqslant m < n$.
- Note that $2^e c = (2m)^e$. By querying $\mathcal{O}(2^e c, n, e)$ we'll know whether $2m \bmod n$ is smaller or larger than $n/2$. I.e. we'll know whether

$$(0 \leqslant m < n/4 \vee n/2 \leqslant m < 3n/4) \text{ or } (n/4 \leqslant m < n/2 \vee 3n/4 \leqslant m < n) \ .$$

  The answer to the first query picks the left or the right side of $\vee$.

We have now fixed a quarter of the interval $[0, n)$ where m must lay.

- The query $\mathcal{O}(2^{2e}c, n, e) = \mathcal{O}((4m)^e, n, e)$ allows us to fix an 8th of the interval $[0, n)$ where m must lay.
- The query $\mathcal{O}(2^{3e}c, n, e) = \mathcal{O}((8m)^e, n, e)$ allows us to fix a 16th of the interval $[0, n)$ where m must lay.
- etc.

In $\log n$ queries the length of the permissible interval will be at most 1.

Suppose that there exists an efficient algorithm $\mathcal{Q}$, such that $\mathcal{Q}(c, n, e) = parity(m)$, where $m^e \equiv c \pmod{n}$.

Then we can implement $\mathcal{O}$: $\mathcal{O}(c, n, e) = \mathcal{Q}(2^e c, n, e)$.

I.e. to compute the size of m, we consider the parity of $2m \bmod n$.

If $m < n/2$ then $2m < n$ and $2m \bmod n = 2m$ which is even.

If $m \geqslant n/2$ then $2m \geqslant n$ and $2m \bmod n = 2n - m$ which is odd.

Other tests of whether the plaintext has a particular shape can be used to factor n.

For example, the RSA Laboratories' Public Key Cryptography Standard #1 v. 1.5 specified that to encrypt a message M, it has to be padded as follows:

$$00|02||PS||00||M,$$

where PS is a random sequence of at least 8 non-zero bytes.

In 1998 it was shown how to use a subroutine for checking the PKCS #1 v. 1.5 conformance of the plaintext to factor n.

Let $n = pq$ be the product of two large primes. We saw that if we knew a non-trivial square root of 1 modulo n then we could factor n.

In general, if we know x and y, such that $x \not\equiv \pm y \pmod{n}$, but $x^2 \equiv y^2 \pmod{n}$ then we can factor n.

Indeed, then $x^2 - y^2 = (x+y)(x-y) \equiv 0 \pmod{n}$, implying $n \mid (x+y)(x-y)$, but n divides neither $x+y$ nor $x-y$. We have $\gcd(n, x+y) = p$ and $\gcd(n, x-y) = q$ (or vice versa).

If $n = pq$ and $m \in \mathbb{Z}_n^*$ is a quadratic residue modulo m then m has exactly four different square roots (modulo n). If one of them is x_1 then the other three are the solutions of

$$\begin{cases} x_2 \equiv x_1 \pmod{p} \\ x_2 \equiv -x_1 \pmod{q} \end{cases} \qquad \begin{cases} x_3 \equiv -x_1 \pmod{p} \\ x_3 \equiv x_1 \pmod{q} \end{cases}$$

$$\begin{cases} x_4 \equiv -x_1 \pmod{p} \\ x_4 \equiv -x_1 \pmod{q} \end{cases}$$

Here $x_1 \equiv -x_4 \pmod{n}$ and $x_2 \equiv -x_3 \pmod{n}$.

Consider the family of functions $\{f_n\}_{n \in \mathbb{N}}$ defined by

$$f_n(x) = x^2 \bmod n$$

and the distribution of n is such that $n = pq$ for randomly chosen primes p and q of certain (large) size.

f is a family of one-way functions unless factoring is easy.

Assume that there exists an efficient algorithm $\mathcal{A}$, such that

$$\big(\mathcal{A}(n,m)\big)^2 \equiv m \pmod{n} \ .$$

If we want to factorize some $n = pq$ then we

- Randomly generate $x \in \mathbb{Z}_n$.
  - Check whether $\gcd(x,n) > 1 \ldots$
- Let $y = \mathcal{A}(n, x^2 \bmod n)$.
- If $x \not\equiv \pm y \pmod{n}$ then
  - let $d = \gcd(n, x+y)$,
  - return $(d, n/d)$

  else fail.

Success probability is 50% because $\mathcal{A}$ does not know x.

Rabin's cryptosystem: let $n = pq$ where $p, q \in \mathbb{P}$ and $p \equiv q \equiv 3 \pmod 4$. Public key: n. Private key: (p, q).

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

Decryption: $D_{(p,q)}(c) = \sqrt{c} \pmod n$.

Decryption is not unique. The message m is assumed to contain enough redundancy, such that one of the four possible values of $\sqrt{c}$ can be selected.

Later schemes overcome this non-uniqueness.

How to take square roots modulo n?

Compute $\sqrt{c} \pmod{p}$ and $\sqrt{c} \pmod{q}$ and use the Chinese Remainder Theorem.

How to take square roots modulo p?

$c^{\frac{p+1}{4}}$ is one of the square roots of c modulo p. $\frac{p+1}{4} \in \mathbb{Z}$ because $p \equiv 3 \pmod{4}$.

Indeed,

$$\left(c^{\frac{p+1}{4}}\right)^2 = c^{\frac{p+1}{2}} = c \cdot c^{\frac{p-1}{2}} \equiv c \cdot \left(\frac{c}{p}\right) = c \pmod{p} \ .$$

And the other square root modulo p is $-c^{\frac{p+1}{4}}$.

If $p \equiv 1 \pmod{4}$ then finding square roots modulo p is feasible, but more complex.

Rabin's cryptosystem is provably secure against chosen-plaintext attacks.

- As long as $D_{(p,q)}$ randomly chooses one of four possibilites.

But it is insecure against chosen-ciphertext attacks.

- As long as $D_{(p,q)}$ randomly chooses one of four possibilites.

If redundancy is taken into account, $D_{(p,q)}(x^2 \bmod n)$ returns x with overwhelming probability.

- Other three values of $\sqrt{x^2}$ are not valid plaintexts.