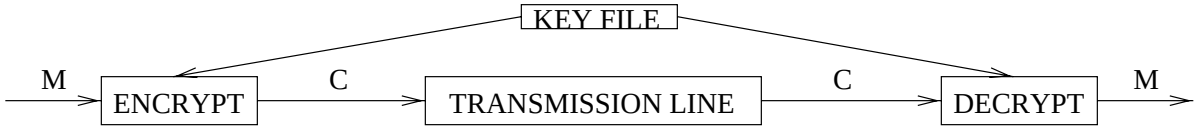
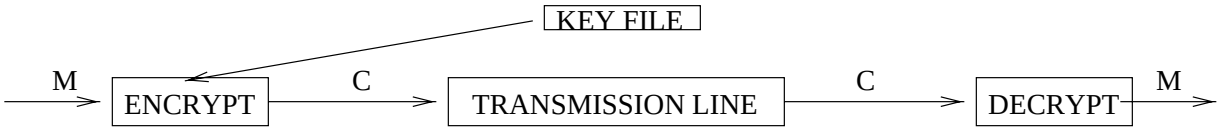
PUBLIC KEY CRYPTOGRAPHY

Public vs Nonpublic Unlike Private key cryptography, there is no need to share keys. Instead, there is a public "phone number" available to any potential user and a private key.





TRAPDOOR

Public Key Cryptography (PKC) is based on the idea of a **trapdoor** function $f : X \to Y$, i.e.,

- f is one-to-one,
- f is easy to compute,
- f is public,
- f^{-1} is difficult to compute,
- f^{-1} becomes easy to compute if a trapdoor is known.

Thus, although in conventional cryptography the prior exchange of keys is necessary, this is not so in public key cryptography.

The idea of PKC was first proposed by Diffie and Hellman in 1976. Here are some important PKCs that we will study.

- RSA
- Rabin
- Merkle-Hellman
- McEliece
- ElGamal
- Elliptic Curve

RSA CRYPTOSYSTEM

n,p,q: Define $n = pq$ where p and q are large primes.

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

M: M is the number representing the message to be encrypted.

C: C is the number representing the "Cypher-text" (i.e., the encrypted text).

Public Information: n, e.

Private Information: d.

PRIMES

An integer $n > 1$ is prime if 1 and n are its only divisors.

Euclid: There are infinitely many primes.
If $p_1 < p_2 < \cdots < p_n$ are the first n primes then any prime divisor of the integer $1 + p_1 p_2 \cdots p_n$ must be larger than p_n.

The number $\pi(n)$ of primes $\leq n$ is asymptotically equal to $\frac{n}{\ln n}$. More generally,

Dirichlet-Hadamard-de la Valleé Poussin: If $\gcd(a, b) = 1$ then the number $\pi_{a,b}(n)$ of primes $p \leq n$ of the form $p = ak + b$ is asymptotically equal to $\frac{1}{\phi(a)} \frac{n}{\ln n}$.

Bertrand's Postulate For any integer there is always a prime between $n + 1$ and $2n$. A beautiful elementary proof is due to Erdös.

Open problem of Hardy and Wright: Is there a prime between n^2 and $(n + 1)^2$?

Interesting Problems with Primes

Ulam's Problem: Start with 1 and write consecutive integers in a counterclockwise spiral!

100	99	98	97	96	95	94	93	92	91
65	64	63	62	61	60	59	58	57	90
66	37	36	35	34	33	32	31	56	89
67	38	17	16	15	14	13	30	55	88
68	39	18	**5**	4	3	12	29	54	87
69	40	19	6	1	2	11	28	53	86
70	41	20	7	8	9	10	27	52	85
71	42	21	22	23	24	25	26	51	84
72	43	44	45	46	47	48	49	50	83
73	74	75	76	77	78	79	80	81	82

Primes seem to line up in diagonals. Can you prove or disprove this? Do experiments!

What is ratio of primes of the form n^2+n+17? How about n^2+n+41? Difficult problems!

No single variable polynomial with integer coefficients can generate all the primes! (Related to Hilbert's tenth problem.)

EULER'S TOTIENT FUNCTION

$\phi(n)$ is the number of non-negative integers less than n which are relatively prime to n.

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	0	10	4	19	18
2	1	11	10	20	8
3	2	12	4	21	12
4	2	13	12	22	10
5	4	14	6	23	22
6	2	15	8	24	8
7	6	16	8	25	20
8	4	17	16	26	12
9	4	18	6	27	18

Some Important Values of $\phi(n)$:

n	$\phi(n) =$	Conditions
p	$p-1$	p prime
p^n	$p^n - p^{n-1}$	p prime
$s \cdot t$	$\phi(s) \cdot \phi(t)$	$\gcd(s,t) = 1$
$p \cdot q$	$(p-1) \cdot (q-1)$	p, q prime

NUMBER THEORY

Example 1: It is easy to generate e such that $\gcd(e, \phi(n)) = 1$, since

$$|\{e < n : \gcd(e, \phi(n)) = 1\}| = \phi(\phi(n))$$

Example 2: $p = 101, q = 113, n = 11413$. Then $\phi(n) = (p-1)(q-1) = 11200 = 2^6 5^2 7$. So any integer not divisible by $2, 5, 7$ can be used as a public key. We can choose $e = 3533$. Using the Euclidean algorithm we easily compute $e^{-1} \bmod 11200 = 6597$.

Example 3: $p = 5, q = 7, n = 35$. Can choose $e = 11$. Let the message be $M = 12$. How do we compute $12^{11} \bmod 35$?

We will review several concepts from Number Theory.

HOW IT WORKS

RSA Encryption: $M \to E(M) := M^e \equiv C \bmod n$

RSA Decryption: $C \to D(C) := C^d \equiv M \bmod n$.

When and Why it Works: Recall that $\phi(n) = (p-1)(q-1)$. For RSA to work $M < n$, $\gcd(e,(p-1)(q-1)) = 1$, p and q are prime and $de \equiv 1(\bmod (p-1)(q-1))$.

RSA works because: $C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{1+k(p-1)(q-1)}(\bmod n)$

Assume that $\gcd(M,q) = \gcd(M,p) = 1$. Then by Fermat's Little Theorem:

$$C^d \equiv M(M^{p-1})^{k(q-1)} \equiv M(1)^{k(p-1)} \equiv M(\bmod p)$$
$$C^d \equiv M(M^{q-1})^{k(p-1)} \equiv M(1)^{k(q-1)} \equiv M(\bmod q)$$

Therefore $C^d \equiv M(\bmod n)$.

Representations of Numbers

Representations in base b.

$$\begin{aligned} n \bmod b &= a_k b^k + a_{k-1} b^{k-1} + \cdots + a_0 \bmod b \\ &= a_0 \\ \lfloor \tfrac{n}{b} \rfloor \bmod b &= a_k b^{k-1} + a_{k-1} b^{k-2} + \cdots + a_1 \\ &= a_1 \\ &\vdots \\ \lfloor \tfrac{n}{b^i} \rfloor \bmod b &= a_i \end{aligned}$$

Using this fact we can write an algorithm for changing the representation of a number into any base.

Procedure *base b expansion* (n, b)
$q := n$
$k := 0$
while $q \neq 0$
    $a_k := q \bmod b$
    $q := \lfloor \frac{q}{b} \rfloor$
    $k := k + 1$
endwhile
return $(a_{k-1} \cdots a_1 a_0)_b$

OPERATIONS ON NUMBERS

Addition of two k-bit numbers can be done in time $O(k)$.

$$\begin{array}{r} 010110101 \\ 11010010 \\ \hline 110000111 \end{array}$$

Multiplication of two k-bit numbers can be done in time $O(k^2)$.

$$\begin{array}{r} 1011 \\ 110 \\ \hline 0000 \\ 1011 \\ 1011 \\ \hline 100010 \end{array}$$

Both are well-known algorithms. Of course there are "faster" algorithms (see Knuth's: "Art of Computer Programming").

Exponentiation of two k-bit numbers can be done in time $O(k^3)$.

Example: $p = 5, q = 7, n = 35$.
Can choose $e = 11$. Let the message be $M = 12$. To compute $12^{11} \bmod 35$.
First write $(11)_{10} = (1011)_2$. Then calculate

$$\begin{aligned} M^{11} &= M^{1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0} \\ &= (M^{1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0})^2 M \\ &= ((M^{1\cdot 2^1 + 0\cdot 2^0})^2 M)^2 M \\ &= ((M^2)^2 M)^2 M \end{aligned}$$

The formal algorithm is as follows: Compute the binary representation of $e = \sum_{i=0}^{k-1} e_i 2^i$, where $k = \lceil \log_2 \rceil$ and perform the following algorithm:

Procedure *exponentiation* (x, e, n)
$z := 1$
for $i = k - 1$ **downto** 0 **do**
    $z := z^2 \bmod n$
    if $e_i = 1$ **then** $z := z \cdot x \bmod n$
return $x^e \bmod n$

TIMING ATTACKS ON RSA

This is similar to a burglar observing how long it takes for someone to turn the dial of a safe. It is applicable to other cryptosystems as well.

A cryptanalyst can compute a private key by keeping track of how long it takes the computer to decipher messages. The exponent is computed bit-by-bit starting with the low-end bit.

For a given ciphertext it is possible to time how long it takes to perform modular exponentiation. We can therefore determine unknown bits by exploiting timing differences in responses. (This attack was implemented by Koeher in 1996.)

The problem is eliminated by using any of the following remedies: (a) constant exponentiation time, (b) random delay, or (c) blinding by multiplying the ciphertext with random number prior to exponentiation.

EUCLIDEAN ALGORITHM

Finding the gcd(a, b) without the factorization of a and b uses the Euclidean Algorithm. Without loss of generality assume $a > b$.

Lemma: Let $a = bq + r$ where a, b, q and $r < b$ are integers. Then gcd$(a, b) =$ gcd(b, r).

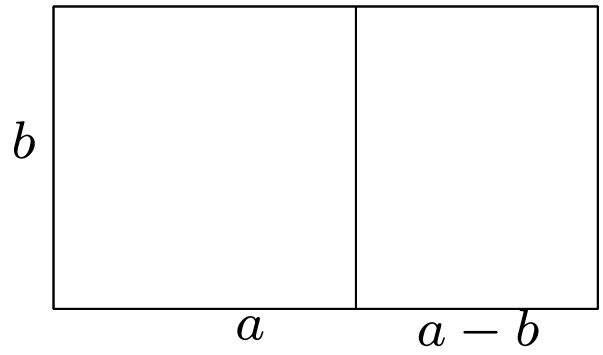
Proof: Given $a = bq + r$ where $a, b, , q$ and $r < b$ are integers. Let d be any number such that $d|a$ and $d|b$. Then it follows that $d|(a - bq)$. Since $(a - bq) = r$ then $d|r$. Thus any divisor of a and b also divides r. This implies gcd$(a, b) =$ gcd(b, r). Since $r = a - bq$ we have $r = a$ mod b. **We iterate** $(r_0 = a, r_1 = b, r_2 = r, q_1 = q)$:

$$
\begin{array}{ll}
r_0 = q_1 r_1 + r_2 & 0 < r_2 < r_1 \\
r_1 = q_2 r_2 + r_3 & 0 < r_3 < r_2 \\
\cdots & \\
r_i = q_{i+1} r_{i+1} + r_{i+2} & 0 < r_{i+2} < r_{i+1}
\end{array}
$$

We define gcd$(x, 0) = x$. Note that the sequence $r_0 > r_1 > \cdots > r_n$ is decreasing. Hence there exists a term r_n such that $r_{n+1} = 0$ and gcd$(r_n, 0) = r_n$. Therefore gcd$(a, b) = r_n$.

Let f_n be the n-th Fibonacci number. Recall: $f_0 = f_1 = 1, f_n = f_{n-1} + f_{n-2}$. Solving this difference equation (by guessing that $f_n = R^n$, for some R) we obtain $f_n = ((1+\sqrt{5})/2)^n$.

Beauty and the Golden Mean:
The rectangle is "aesthetically most pleasing" when cutting a square the remaining portion is congruent to the original rectangle! (Construction of the Parthenon uses this principle!)



The big rectangle has dimensions $a \times b$.
The small rectangle has dimensions $b \times (a-b)$.
Congruence means they are similar, i.e.,

$$R := \frac{a}{b} = \frac{b}{a-b}, \; R = \frac{1}{1-R}.$$

Solving for R we obtain $R^2 = R+1$ and hence $R = (1+\sqrt{5})/2$.

We show by induction $r_{n-i} \geq f_i$. Initial step $i = 0$ is easy. And

$$\begin{aligned} r_{n-(i+1)} &= q_{n-i} r_{n-i} + r_{n+1-i} \\ &\geq r_{n-i} + r_{n+1-i} \\ &\geq f_i + f_{i-1} = f_{i+1} \end{aligned}$$

It follows that $a = r_0 \geq f_n$ and $n = O(\log a)$.

Procedure gcd(a, b; positive integers)
$x := a$;
$y := b$;
while $y \neq 0$ **do**
  $r := x$ mod y
  $x := y$
$y := r$
end while
return x

Theorem: If a and b are positive integers, then there exists integers s and t such that $\gcd(a, b) = sa + bt$. Moreover, s, t can be computed in time logarithmic in the input.

Example: $1 = \gcd(50, 21)$

$$\begin{array}{lcl} 50 \bmod 21 \equiv 8 & \Leftrightarrow & 8 = 50 - (2)21 \\ 21 \bmod 8 \equiv 5 & \Leftrightarrow & 5 = 21 - (2)8 \\ 8 \bmod 5 \equiv 3 & \Leftrightarrow & 3 = 8 - (1)5 \\ 5 \bmod 3 \equiv 2 & \Leftrightarrow & 2 = 5 - (1)3 \\ 3 \bmod 2 \equiv 1 & \Leftrightarrow & 1 = 3 - (1)2 \end{array}$$

Reversing the steps we have

$$\begin{array}{l} 8 = 50 - (2)21 \\ 5 = 21 - (2)8 \\ 5 = 21 - (2)\,(50 - (2)21) \\ 5 = (5)21 - (2)50 \\ 3 = 8 - (1)5 \\ 3 = (50 - (2)21) - (1)\,((5)21 - (2)50) \\ 3 = -(7)21 + (3)50 \\ 2 = 5 - (1)3 \\ 2 = ((5)21 - (2)50) - (1)\,(-(7)21 + (3)50) \\ 2 = (12)21 - (5)50 \\ 1 = 3 - (1)2 \\ 1 = (-(7)21 + (3)50) - (1)\,((12)21 - (5)50) \\ 1 = -(19)21 + (8)50 \end{array}$$

This can be used to compute modular inverses, e.g. $50^{-1} \bmod 21 = 8$.

FACTORING ATTACKS

The encrypted message can be decrypted if the decryption key is known.

One approach to attacking RSA is to try to factor n.

If that were possible then one could compute p, q such that $n = pq$.

Since e is public we can solve the linear congruence

$$ex \equiv 1 \bmod (p-1)(q-1)$$

to compute the inverse of e modulo n, which is equal to the decryption exponent d.

However, factoring is not an easy problem to solve!

CHINESE REMAINDER THEOREM

Find a number x that leaves a remainder of 1 when divided by 3, 2 when divided by 5 and 3 when divided by 7. Means: find x such that $x \equiv 1 \bmod 3$, $x \equiv 2 \bmod 5$ and $x \equiv 3 \bmod 7$. The solution to this problem is: $x \equiv 52 \bmod 105$. How is this solution found?

Theorem: Let $m_1, m_2, \ldots, m_n$ be pairwise relatively prime positive integers. The system:

$$\begin{aligned} x &\equiv a_1 \bmod m_1 \\ x &\equiv a_2 \bmod m_2 \\ &\vdots \\ x &\equiv a_n \bmod m_n \end{aligned}$$

has a unique solution modulo $m = m_1 m_2 \ldots m_n$.

Proof: Let $m = m_1 m_2 \cdots m_n$ and $M_k = \frac{m}{m_k}$. For each value M_k find its inverse y_k modulo m_k (i.e., $M_k y_k \equiv 1 \bmod m_k$). Then $x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_n M_n y_n$. This completes the proof.

For $n = pq$, the mapping $Z_n^* \to Z_p^* \times Z_q^* : x \bmod n \to (x \bmod p, x \bmod q)$ is one-to-one. Since, $|Z_n^*| = \phi(n) = \phi(p) \cdot \phi(q) = |Z_p^*| \cdot |Z_q^*|$ it is also onto.

The Chinese remainder theorem provides for solving congruences with composite modulus by inverting the above mapping. Here is how it works.

Suppose we have a pair $(a_1, a_2) \in Z_p^* \times Z_q^*$. Consider $b_1 = q^{-1} \bmod p$ and $b_2 = p^{-1} \bmod q$. Put $a = a_1 b_1 q + a_2 b_2 p$ and observe that

$$\begin{aligned} a &= a_1 b_1 q + a_2 b_2 p \equiv a_1 \bmod p \\ a &= a_1 b_1 q + a_2 b_2 p \equiv a_2 \bmod q \end{aligned}$$

Example: Solve $x \equiv 5 \bmod 7, x \equiv 6 \bmod 11$. We compute $7^{-1} \bmod 11 = 8$ and $11^{-1} \bmod 7 = 2$. So $a = 5 \cdot 2 \cdot 11 + 6 \cdot 8 \cdot 7 = 446 \equiv 61 \bmod 77$ is the solution of the two congruences simultaneously.

Z_n: the set of integers $0 \leq a < n$ is an additive group modulo n.

Z_n^*: the set of integers $0 \leq a < n$ which are prime to n is a multiplicative group modulo n.

Example: Group tables of $(Z_6, +)$ and $(Z_6^*, \cdot)$:

$(Z_6, +)$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

$(Z_6^*, \cdot)$	1	5
1	1	5
5	5	1

Fermat's Little Theorem: If p is prime and $p \not| a$ then $a^{p-1} \equiv 1 \bmod p$.

a	$a^6 \bmod 7$
2	$2^6 = 64 \equiv 1 \bmod 7$
3	$3^6 = 729 \equiv 1 \bmod 7$
4	$4^6 = 4,096 \equiv 1 \bmod 7$
5	$5^6 = 15,6251 \equiv 1 \bmod 7$

Proof: Let a be such that $p \not| a$. List all the elements of Z_p^*.

$$\begin{array}{rcccc} & x_1 & x_2 & \cdots & x_{p-1} \\ a: & a \cdot x_1 & a \cdot x_2 & \cdots & a \cdot x_{p-1} \end{array}$$

$$\begin{aligned} x_1 x_2 \cdots x_{p-1} &= (ax_1)(ax_2)\cdots(ax_{p-1}) \\ &= a^{p-1}(x_1 x_2 \cdots x_{p-1}) \end{aligned}$$

The group Z_p^* is cyclic, in the sense that there is a generator g such that $Z_p^* = \{g^0, g^1, \ldots, g^{p-1}\}$.

This fact was first proved by Gauss who also proved something more general:

For all m, Z_m^* is cyclic if and only if m is of the form $1, 2, 4, p^k, 2p^k$, where p is an odd prime and k is a positive integer.

The order of an element $a \in Z_p^*$ is the smallest $i \neq 1$ such that $a^i = 1$.

Lagrange's Theorem: If H is a subgroup of the group G then $|H|$ divides $|G|$.

Proof: Define the equivalence relation on elements of the group G:

$$a \approx b \Leftrightarrow ab^{-1} \in H$$

The equivalence classes are easily shown to be the cosets $Ha = \{ha : h \in H\}$. They all have the same size, namely $|H|$. It follows that $|H|$ divides $|G|$.

For a generator g of Z_p^*, the element g^k has order $\frac{p-1}{\gcd(p-1,k)}$. Moreover

$$g^k \text{ generates } Z_p^* \Leftrightarrow \gcd(p-1, k) = 1.$$

The group Z_p^* has $\phi(p-1)$ generators.

Euler's Theorem: If a is an integer which is prime to n, then $a^{\phi(n)} \equiv 1 \bmod n$.

Totient Function Attack on RSA

Is there an "efficient" algorithm which given n (a product of two primes) as input will compute $\phi(n)$?

Assume such an algorithm $n \rightarrow \phi(n)$ exists!

We can prove $n + 1 - \phi(n) = p + q$. This is because

$$\begin{aligned} \phi(n) &= (p-1)(q-1) \\ &= pq - p - q + 1 \\ &= n - p - q + 1 \end{aligned}$$

It follows that $\phi(n) = (p-1)(n/p-1) = n - n/p - p + 1$ and consequently

$$p^2 - (n + 1 - \phi(n))p + n = 0$$

By solving this quadratic we obtain

$$p = \frac{n + 1 - \phi(n) \pm \sqrt{(n + 1 - \phi(n))^2 - 4n}}{2}$$

Thus, assuming an "efficient" algorithm $n \rightarrow \phi(n)$ exists, there is an "efficient" algorithm for factoring n.

CHOSEN CIPHERTEXT ATTACK

RSA is vulnerable to chosen ciphertext attacks as the following argument indicates.

The following attack indicates that RSA is not good for signing!

Assume a cryptanalyst listening to communication reads a ciphertext C and wants to recover M such that $M = C^d \bmod n$.

Cryptanalyst chooses a random $r < n$ and uses the public key to compute

$$x = r^e \bmod n, y = xC \bmod n, t = r^{-1} \bmod n$$

and gets y signed with the private key d, i.e. $u = y^d \bmod n$. Cryptanalyst can now compute

$$tu \equiv r^{-1}(xC)^d \equiv r^{-1}x^d C^d \equiv r^{-1}rM \equiv M \bmod n$$

A remedy we will discuss later is to use hashing.

RSA-BIT ATTACKS

Assume $n = pq$ and n is odd.

RSA bit attacks "target" specific bits of RSA output, e.g. the least significant bit.

Knowledge of the least significant bit of an RSA encrypted message is equivalent to "locating" the message in a certain subinterval of $[0, n]$. More precisely,

We define: $Parity(M^e \bmod n)$ = low order bit of M, i.e. 0 if M is even, and 1, otherwise.

$Half(M^e \bmod n) = 0$ if $0 \leq M < n/2$, and 1, if $n/2 \leq M < n$, i.e., $Half(M^e \bmod n) = 0 \Leftrightarrow 2M < n$. Hence,

$$\begin{aligned} Half(M^e \bmod n) &= Parity((2M)^e \bmod n) \\ Parity(M^e \bmod n) &= Half((M/2)^e \bmod n) \end{aligned}$$

Then we have:

$$
\begin{aligned}
Half(M^e \bmod n) = 0 \quad &\Leftrightarrow M \in [0, \tfrac{n}{2}) \\
Half((2M)^e \bmod n) = 0 \quad &\Leftrightarrow M \in [0, \tfrac{n}{4}) \cup [\tfrac{n}{2}, \tfrac{3n}{4}) \\
Half((4M)^e \bmod n) = 0 \quad &\Leftrightarrow M \in [0, \tfrac{n}{8}) \cup [\tfrac{n}{4}, \tfrac{3n}{8}) \\
&\quad \cup [\tfrac{n}{2}, \tfrac{5n}{8}) \cup [\tfrac{3n}{4}, \tfrac{7n}{8})
\end{aligned}
$$

By using binary search we can locate precisely the value of M.

This means if there is an efficient algorithm for computing the low order RSA-bit (i.e., the $Parity$ function) then there is an efficient algorithm for computing the original message.

Thus the low order RSA bit is as secure as RSA.

QUADRATIC RESIDUES

$a^{p-1} - 1 \equiv (a^{\frac{p-1}{2}} - 1)(a^{\frac{p-1}{2}} + 1) \equiv 0 \bmod p$. We define the **Legendre symbol** by

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a^{\frac{p-1}{2}} \equiv +1 \bmod p \ \& \ a \not\equiv 0 \bmod p \\ -1 & \text{if } a^{\frac{p-1}{2}} \equiv -1 \bmod p \ \& \ a \not\equiv 0 \bmod p \\ 0 & \text{if } a \equiv 0 \bmod p \end{cases}$$

This implies that

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

More generally, given the prime factorization $n = p_1^{r_1} p_2^{r_2} \cdots p_k^{r_k}$ we define the **Jacobi Symbol**

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{r_1} \left(\frac{a}{p_2}\right)^{r_2} \cdots \left(\frac{a}{p_k}\right)^{r_k}$$

Example 1: If you know the factorization of $9975 = 3 \cdot 5^2 \cdot 7 \cdot 19$ then we compute:

$$\begin{aligned} \left(\tfrac{6278}{9975}\right) &= \left(\tfrac{6278}{3}\right) \cdot \left(\tfrac{6278}{5}\right)^2 \cdot \left(\tfrac{6278}{7}\right) \cdot \left(\tfrac{6278}{19}\right) \\ &= \left(\tfrac{2}{3}\right) \cdot \left(\tfrac{3}{5}\right)^2 \cdot \left(\tfrac{6}{7}\right) \cdot \left(\tfrac{8}{19}\right) \\ &= (-1) \cdot (-1)^2 \cdot (-1) \cdot (-1) = -1 \end{aligned}$$

Computing the Jacobi symbol does not require the factorization of n. A "Euclidean style" algorithm will be discussed in the sequel. Assume m, n are odd. Then we have the following properties.

1. $a \equiv b \bmod n \Rightarrow \left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$
2a. $\left(\frac{2}{n}\right) = 1$ if $n \equiv \pm 1 \bmod 8$
2b. $\left(\frac{2}{n}\right) = -1$ if $n \equiv \pm 3 \bmod 8$
2c. $\left(\frac{2^k t}{n}\right) = \left(\frac{2}{n}\right)^k \cdot \left(\frac{t}{n}\right)$ for t odd
3. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right)$
4a. $\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right)$ if $a \equiv b \equiv 3 \bmod 4$
4b. $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)$ otherwise

Example 2: If you do not know the factorization of 383 then we use the following algorithm to compute:

$$\begin{aligned}\left(\tfrac{219}{383}\right) &= -\left(\tfrac{383}{219}\right) = -\left(\tfrac{164}{219}\right) = -\left(\tfrac{2}{219}\right)^2 \cdot \left(\tfrac{41}{219}\right) \\ &= -\left(\tfrac{41}{219}\right) = -\left(\tfrac{219}{41}\right) = -\left(\tfrac{14}{41}\right) \\ &= -\left(\tfrac{2}{41}\right) \cdot \left(\tfrac{7}{41}\right) = -\left(\tfrac{7}{41}\right) \\ &= -\left(\tfrac{41}{7}\right) = -\left(\tfrac{-1}{7}\right) = 1\end{aligned}$$

Definition: a is a quadratic residue modulo p (denoted $a \in QR_p$) if and only if $\exists b \in Z_p^*(a \equiv b^2 \bmod p)$.

Important: For $a \in Z_p^*$, $a \in QR_p \Leftrightarrow \left(\frac{a}{p}\right) = 1$.

$\Rightarrow$: Assume $a \equiv b^2 \bmod p$, for $b \in Z_p^*$. Then $a^{\frac{p-1}{2}} \equiv (b^2)^{\frac{p-1}{2}} \equiv b^{p-1} \equiv 1 \bmod p$.

$\Leftarrow$: Take a generator g of Z_p^*. It follows that $a = g^i$, for some i. Assume $\left(\frac{a}{p}\right) = 1$. Therefore $a^{\frac{p-1}{2}} \equiv 1 \bmod p$ and hence $a^{\frac{p-1}{2}} \equiv g^{\frac{i(p-1)}{2}} \equiv 1 \bmod p$. It follows $(p-1)|\frac{i(p-1)}{2}$. Hence i is even and $a \in QR_p$.

This has significant applications for primality testing.

Test for Pseudo-Primes

Input: n
Pick random $a \in [1, n]$
Compute $g := \gcd(a, n)$
If $g \neq 1$ then n is composite
If $g = 1$ then compute $e := a^{n-1} \bmod n$
If $e \neq 1$ then n fails the test
If $e = 1$ then n passes the test

What is the probability that a composite number passes the test? It seems that by repeating the test you increase your chances for a correct answer!

Unfortunately there are composite numbers that pass the test for all "bases" to which they are relatively prime (these are known as Carmichael numbers, and there are infinitely many of them!)

PROBABILISTIC PRIMALITY TESTS

No polynomial time algorithm is known for primality testing. The problem is known to be in $NP \cap co - NP$.

In probabilistic primality tests we construct a sequence $\{P_n \subseteq Z_n^* : n \geq 1\}$ of sets such that (a) $P_n = \emptyset$, if n is prime, (b) it is easy to check membership in P_n, and (c) Pr[$x \in Z_n^* : x \notin P_n$] $\leq c$, for some constant $c < 1$ independent of n.

Input: n
1. Choose random $1 \leq a \leq n$
2a. if $a \notin P_n$ **Output** PRIME
2b. if $a \in P_n$ **Output** COMPOSITE

Theorem: If algorithm outputs COMPOSITE then n is indeed composite. Moreover, if n is composite then Pr[Output is PRIME] $\leq c$.

Since $c < 1$, we can reduce the error by iterating the test a sufficiently large number of times.

SOLOVAY-STRASSEN TEST

Input: n (odd)

1. Choose random $1 \leq a \leq n$

2a. if $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \bmod n$ **Output** PRIME

2b. if $\left(\frac{a}{n}\right) \not\equiv a^{\frac{n-1}{2}} \bmod n$ **Output** COMPOSITE

Theorem: (Solovay-Strassen) If algorithm outputs COMPOSITE then n is indeed composite.

Let $P_n = \{a \in Z_n^* : \left(\frac{a}{n}\right) \not\equiv a^{\frac{n-1}{2}} \bmod n\}$. We prove $\Pr[Z_n^* \backslash P_n] \leq \frac{1}{2}$ Indeed, first of all observe the above set is a subgroup of Z_n^*. By Langrange's Theorem in group theory the claim will follow if we prove that it is a proper subgroup, i.e., $\{a \in Z_n^* : \left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \bmod n\} \neq Z_n^*$. W.l.o.g. assume $n = pq$. Take $u \notin QR_p$. By Chinese remainder theorem there is an $a \in Z_n^*$ such that $a \equiv u \bmod p, a \equiv 1 \bmod q$. Hence, $\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{a}{q}\right) = (-1) \cdot (+1) = -1$. Since, $a^{\frac{n-1}{2}} \equiv 1^{\frac{n-1}{2}} \equiv 1 \bmod q$, it follows that $\left(\frac{a}{n}\right) \not\equiv a^{\frac{n-1}{2}} \bmod n$.

MILLER-RABIN TEST

Input: n (odd)

1. Choose random $1 \leq a \leq n$
   Write $n-1 = 2^k m$, with m odd

2a. **If** $a^m \equiv 1 \bmod n$ or $\exists l < k (a^{2^l m} \equiv -1 \bmod n)$
then Output PRIME

2b. **else Output** COMPOSITE

Theorem: (Miller-Rabin) If algorithm outputs COMPOSITE then n is indeed composite.

Assume n is prime. Then $a^{n-1} \equiv a^{2^k m} \equiv 1 \bmod n$, i.e., $a^{2^k m} - 1 \equiv (a^{2^{k-1} m} - 1)(a^{2^{k-1} m} + 1) \equiv 0 \bmod n$. Hence, either $a^{2^{k-1} m} \equiv 1 \bmod n$ or $a^{2^{k-1} m} \equiv -1 \bmod n$. Continuing we obtain that $a^m \equiv 1 \bmod n$ or $\exists l < k (a^{2^l m} \equiv -1 \bmod n)$, i.e., the test will output n is PRIME.

We also state without proof.

Theorem: If n is composite then

$$\Pr[\text{Output is PRIME}] \leq 1/4.$$

GENERATING RSA PRIMES (Heuristic)

1. Choose a k-bit odd integer p at random.

2. Test divide p by all small primes, i.e., less than or equal some small prime.

3. If p passes the above test then apply the Miller-Rabin test for r different "bases".

4. If p passes all these tests then it is prime with high probability $\geq 1 - 4^{-r}$.

5. If p is not prime then change p to $p+2$ and go to step 1.

CONGRUENCES

Congruences are like equations but with the equality sign $=$ replaced by the congruence sign $\equiv$. A linear congruence has the form

$$a \cdot x \equiv b \bmod n \tag{1}$$

where x is the unknown variable.

Congruence (1) has a solution iff $\gcd(a, n)|b$. If x_0 is one solution then any other solution is $x = x_0 + \frac{in}{\gcd(a,n)}$, where $0 \leq i < \gcd(a, n)$.

Higher degree congruences can also be solved, e.g.,

$$x^m \equiv 1 \bmod n \tag{2}$$

Similarly we can determine exactly when

$$x^m \equiv -1 \bmod n \tag{3}$$

has a solution. E.g., to solve (2) and (3) take "discrete logarithms" of both sides and reduce to linear congruences.

DECRYPTION EXPONENT ATTACK

Assume we have an efficient algorithm A, which given the encryption exponent e of RSA as input it outputs the decryption exponent of RSA. We use this to give an efficient Las Vegas algorithm for factoring n.

1. Choose a random integer $1 \leq w \leq n-1$ and compute $z := \gcd(w, n)$. If $1 < z < n$ then you have a prime factor. Quit and report success. $\Pr[\gcd(w, n) = 1] = \frac{\phi(n)}{n} = (1 - \frac{1}{p})(1 - \frac{1}{q})$.

2. Compute $d := A(e)$, which satisfies $ed \equiv 1 \bmod \phi(n)$. Write $ed - 1 = 2^s r$ where r is odd and compute $v := w^r \bmod \phi(n)$. Clearly, $2^s r \equiv 0 \bmod \phi(n)$, which implies $v^{2^s} \equiv w^{2^s r} \equiv w^0 \equiv 1 \bmod n$. It follows that

$$(v^{2^{s-1}}+1)(v^{2^{s-2}}+1)\cdots(v+1)(v-1) \equiv 0 \bmod n$$

3. This gives rise to the following test:

if $v \equiv 1 \bmod n$ **quit** (failure)
while $v \not\equiv 1 \bmod n$ **do**
$v_0 := v$
$v \equiv v^2 \bmod n$
if $v_0 \equiv -1 \bmod n$ **then** quit (failure)
else compute $\gcd(v_0 + 1, n)$ **(success)**

4. If successful, at the end of the while loop we find a value v_0 such that $v_0^2 \equiv 1 \bmod n, \not\equiv 1 \bmod n$. If $v_0 \equiv -1 \bmod n$ then the algorithm fails. Otherwise we have that v_0 satisfies

$$v_0^2 \equiv 1 \bmod n, \not\equiv 1 \bmod n, v_0 \not\equiv -1 \bmod n,$$

which of course can be used to factor n.

Theorem: Pr[success] $\geq \frac{1}{2}$.

Proof: The algorithm may fail in one of the following two ways:

$$(1) \quad w^r \equiv 1 \bmod n$$
$$(2)_t \quad w^{2^t r} \equiv -1 \bmod n, 0 \leq t \leq s-1,$$

which gives rise to $s + 1$ congruences. Any solution of the system leads to failure.

Write $p-1 = 2^i p_1, q-1 = 2^j q_1$. Both p_1, q_1 are odd. Therefore we have that

$$2^{i+j} p_1 q_1 = \phi(n) | ed - 1 = 2^s r.$$

which implies $i + j \leq s$ and $p_1 q_1 | r$.

We know that $x^m \equiv -1 \bmod n$ has a solution $\Leftrightarrow$ $\nu_2(m) < \nu_2(p-1), \nu_2(q-1)$. We can count the solutions by reducing to a linear congruence $m \operatorname{index}(x) \equiv \operatorname{index}(-1) \bmod \phi(n)$. The same applies to congruences of the form $x^m \equiv 1 \bmod n$ which always have solutions.

$$\#\text{ solutions} \begin{cases} (1) & \gcd(r, p-1) \gcd(r, q-1) \\ (2)_t & \gcd(2^t r, p-1) \gcd(2^t r, q-1), \\ & \text{if } t < \min\{i, j\} \\ (2)_t & 0, \\ & \text{otherwise} \end{cases}$$

Observe that

$$\begin{aligned} &\gcd(r, p-1) = p_1 \\ &\gcd(r, q-1) = q_1 \\ &\gcd(2^t r, p-1) = 2^{\min\{t,i\}} p_1 \\ &\gcd(2^t r, q-1) = 2^{\min\{t,j\}} q_1 \end{aligned}$$

Without loss of generality assume $i \leq j$. By the above we have

$$\begin{aligned} n \cdot \Pr[\text{ failure}] &\leq p_1 q_1 + \\ &\quad p_1 q_1 (1 + 2^2 + 2^4 + \cdots + 2^{2(i-1)}) \\ &= p_1 q_1 (1 + (2^{2i} - 1)/3) \\ &= p_1 q_1 (2/3 + 2^{2i}/3) \\ &\leq \frac{2 p_1 q_1}{3} + \frac{p_1 q_1 2^{i+j}}{3} \\ &= \frac{2 p_1 q_1}{3} + \frac{\phi(n)}{3} \\ &= \frac{2 p_1 q_1}{3} + \frac{n}{3} \end{aligned}$$

It follows that

$$\begin{aligned} \Pr[\text{ failure}] &\leq \frac{p_1 q_1}{n} \frac{2}{3} + \frac{1}{3} \\ &\leq \frac{1}{4} \cdot \frac{2}{3} + \frac{1}{3} \\ &= \frac{1}{2} \end{aligned}$$

This proves the theorem.