RSA Basics

– RSA = Rivest, Shamir and Adleman, 3 proposers, MIT, 1978.

– RSA is a **public key** system, i.e. a public key is used for encrypting. But a private key must be known for deciphering. So anyone, essentially, can encrypt. But only cognoscenti can decrypt.

– Plaintext is encrypted in blocks.

– Each block is upper-bounded by a numeric value n. This means that block size is $\leq \log_2(n)$. (Why? $n = 7$ gives binary value 111. $\log_2 7 < 3$ since $\log_2 8 = 3$.)

– In practice, block size is 2^k bits with $2^k < n \leq 2^{k+1}$.

– Plaintext block: M

– Ciphertext block: C

– $C = M^e \bmod n$

– $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

– Both sender and receiver know the value of n.

– Sender knows the value of e.

– Receiver knows the value of d.

– So:

Public key = $\{e, n\}$

Private key = $\{d, n\}$

– Note on right hand term of expression for M above:

$(a^x)^y = a^{xy}$

Cf. $(3^2)^2 = 9^2 = 81$

or $3^4 = 3.3.3.3 = 81$

– Aim: choose e, d (and n) in such a way that nothing (e.g. ciphertext C characteristics; or public key values e and n) will "give the game away".

– Numbers which are prime or relatively prime are a very good basis for such values.

– A "one-way function": a function which is easy to calculate but difficult to compute inverse function. E.g. power function. 3^7 is not too hard to determine; but the 7th root of a number is not easy to determine. Second example: multiplying prime numbers together. Both examples are used in RSA.

– Also known as a "trapdoor function": determining inverse is difficult.

– Knowledge of $\{e, n\}$ is more or less useless for inferring value of d. As we will see, both have prime factors, p and q. Knowledge of p and q would be an excellent start to cryptanalysis of such a public key system (we think!). (Have a quick look at the next section to see how e and d are constructed from prime numbers p and q.)

We could factorize n to get p and q, we suppose.

We know e because it is public.

So d could be derived fairly easily.

Factorize n? Say n is about 200 decimal digits. Then it would take about 30 million years to do the factorizing!

Example pp. 278-279 of Singh (1999): Scientific American (Martin Gardner) published a challenge in 1997 to factor a number N with 129 digits. Solution found by team of 600 people in 1994. In solution p and q were 64- and 65-digit primes.

RSA: the algorithm

– Select two primes, p and q. E.g. $p = 3, q = 17$.

– Form product, $n = pq = 51$.

– And: $(p-1)(q-1) = 32$. ;1;1;128;128;1;0x;1;1;128;128;1;0x – Choose number e between 3 and $(p-1)(q-1)$, relatively prime to the latter. E.g. $e = 7$.

– Next choose d such that $ed \bmod ((p-1)(q-1)) = 1$.

I.e. $ed = k(p-1)(q-1)+1$, k integer.

Here: $7d \bmod 32 = 1$

giving $d = 23$.

Show this: $7 \,.\, 23 = 161$. $161/32 = 5$ + remainder 1. Consequently we see that $161 \bmod 32 = 1$.

– Next define the public key: $(e, n) = (7, 51)$.

– Say, message is $M = 2$.

– Ciphertext, $C = M^e \bmod n = 2^7 \bmod 51 = 128 \bmod 51 = 26$.

– Decipherment requires knowledge of (d, n). I.e. $(23, 51)$.

– Then: $M = C^d \bmod n = 26^{23} \bmod 51$.

But how do we solve this?!

– One thing we can do is factorize as follows:

$26^{23} \bmod 51 = (26^1.26^2.26^4.26^{16}) \bmod 51$

We know that $ab \bmod n = (a \bmod n \,.\, b \bmod n) \bmod n$.

So this implies:

$(26^1.26^2.26^4.26^{16}) \bmod 51 = 26 \bmod 51 \,.\, 676 \bmod 51 \,.\, 26^4 \bmod 51 \,.\, 26^{16} \bmod 51) \bmod 51$

(We've left some parts unresolved for the moment. Now we'll use: $676 = 51 \,.\, 13$ + remainder 13)

$= (26.13.((13.13) \bmod 51).(26^{16} \bmod 51)) \bmod 51$

(Now we'll use: $13 \,.\, 13 = 169$. $169 \bmod 51 = 16$.)

$= (26.13.16.((16.16.16.16) \bmod 51)) \bmod 51$

$= (26.13.16.((16.16) \bmod 51).((16.16) \bmod 51)) \bmod 51$

(Now use: $16 \,.\, 16 = 256$. $256 \bmod 51 = 1$.)

$= (26.13.16.1.1) \bmod 51$

(Just multiply this out. $26 \,.\, 13 \,.\, 16 = 5408$. We find that this equals $51 \,.\, 106$ with remainder 2.)

$= 2$.

This is of course our input message, $M = 2$.

We certainly need a better way to solve such an equation. We will write a little program which will do this for us.

Algorithm for finding private key, d

– We want to solve the following for d: $ed \bmod \phi = 1$

– By definition, $\phi = (p-1)(q-1)$

– Take $e = 7, \phi = 32$. We are using the figures used in earlier example.

– Define $r(0) = \phi = 32$, and $r(1) = e = 7$

– Solving means that we seek: $e.d = k.\phi + 1$ for constant k, i.e., $-k.\phi + e.d = 1$ or using our new notation, Const. . $r(0) + d.r(1) = 1$.

– First phase of the algorithm is to determine a sequence of remainder terms based on modulo $r(.)$ terms (see the following) until we get $r(k) = 1$ for some integer k:

$r(0) = a(1)r(1) + r(2)$

$r(1) = a(2)r(2) + r(3)$

$r(2) = a(3)r(3) + r(4)$

etc.

– This iterative scheme allows us, by back-substitution, to determine $r(k) = u.r(0)+v.r(1)$ which is the solution we are looking for.

– The form of this solution is right: $r(k) = 1$ following the recurrence above, $\phi = r(0)$ by definition.

– $r(1) = e$ again by definition

– This leads to the coefficient v being what we are looking for.

– Let's proceed:

$32 = a(1).7 + r(2) \Longrightarrow 32 = 4.7 + 4$

$r(1) = a(2).r(2) + r(3) \Longrightarrow 7 = a(2).4 + r(3) \Longrightarrow 7 = 1.4 + 3$

$r(2) = a(3).r(3) + r(4) \Longrightarrow 4 = a(3).3 + r(4) \Longrightarrow 4 = 1.3 + 1$

So $r(k) = 1$ and $k = 4$.

Example of back-substitution.
Take [1] $r(2) = a(3).r(3) + r(4) \Longrightarrow r(4) = r(2) - a(3).r(3)$.
Now $r(1) = a(2).r(2) + r(3) \Longrightarrow r(3) = r(1) - a(2).r(2)$
Back to [1]:
$r(4) = r(2) - a(3).r(1) + a(2).a(3).r(2) \Longrightarrow r(4) = r(2) - 1.7 + 1.1.r(2)$
Now $r(0) = a(1).r(1) + r(2) \Longrightarrow r(2) = r(0) - a(1).r(1) \Longrightarrow r(2) = 32 - 4.7 \Longrightarrow r(2) = 4$
So, $r(4) = r(0) - a(1).r(1) - a(3).r(1) + a(2).a(3)[r(0) - a(1).r(1)] \Longrightarrow$
$r(4) = r(0)[1 + a(2).a(3)] + r(1)[-a(1) - a(3) - a(1).a(2).a(3)] \Longrightarrow$
$r(4) = r(0)[1 + 1.1] + r(1)[-4 - 1 - 1.1.4] \Longrightarrow$
$r(4) = r(0).2 + r(1).(-9)$
So now we finally have our value, -9. Are we right? We have $r(0) = 32, r(1) = 7$, and $r(4) = 1$. So we are right.

But a positive number is needed (unless we want to deal with very small fractions). We are working in an arithmetic modulo ϕ, i.e. modulo 32. We know that $-9 \equiv 23 \bmod 32$ which is what we are looking for. This was the value used in the example previously..

Example 2 of RSA encryption

– M = "MAKE CONTACT IMMEDIATELY"
– Say, A = 01, B = 02, C = 03, D = 04, etc.
– M = 13 01 11 05 03 15 14 20 01 ...
– Use blocks of 4 digits:
$M_1 = 1301$
$M_2 = 1105$
etc.
– Take $p = 47, q = 59$.
– So $n = pq = 2773$ and $(p-1)(q-1) = 2668$.
– Value of e must be chosen between 3 and 2668.
– Let's take it as $e = 17$.
– Now, for d, $ed \bmod (p-1)(q-1) = 17d \bmod 2668 = 1$.
(Explanation: $17d = k.2668 + 1$ for some integer k. Find $d = 157$.)
– Encipherment of $M_1 = (1301)^{17} \bmod 2773 = x$.
– Decipherment of $x = (x)^{157} \bmod 2773$.
Latter has to equal our message block, 1301.

RSA Example 3

– 2 primes, p, q. $p = 7, q = 17$.
– $n = pq$: 7 x 17 $= 119 = n$.
– $\phi(n) = (p-1)(q-1) = 6 \times 16 = 96$.
– e relatively prime to $\phi(n)$, less than $\phi(n)$, greater than 3. Take $e = 5$.
– d s.t. $ed = 1 \bmod 96$ and $d < 96$. Find $d = 77$ (because $77 \times 5 = 385 = 4 \times 96 + 14$).
– Public key $= (e, n) = (5, 119)$.
Private key $= (d, n) = (77, 119)$.
– Plaintext input message: $M = 19$.
– Ciphertext, $C = M^e (\text{mod } n) = 19^5 (\text{mod } 119) = 2476099 \bmod 119 = 66$.
– Decryption: $M = C^d (\text{mod } n) = 66^{77} (\text{mod } 119) = 19$.
– Best to have a little program to solve these equations! We'll look now at an algorithm.

Review
DES, IDEAL etc. use symmetric keys
I.e. same key for E and for D
With public key systems, you create K_E and K_D
K_E is public, available to anyone
K_D is private, available to you only
Key length is short, e.g. 56
compared to key lengths in public systems

Euler's totient function, $\phi(n)$

We have already seen how we use $n = pq$ in the RSA algorithm, and furthermore $\phi = (p-1)(q-1)$. This function ϕ is interesting in its own right. It will lead us also to Euler's theorem, to be used below.

Euler's totient function, $\phi(n)$: the number of positive integers less than n, and relatively prime to n. Let's see:

n	1	2	3	4	5	6	7	8	9	10	11
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10

(See Stallings, chapter 7.) For a prime number, p, $\phi(p) = p - 1$.
Now suppose we have two prime numbers, p and q, then for $n = pq$, $\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$.
Euler's theorem: from every a and n that are relatively prime, $a^{\phi(n)} \equiv 1 \bmod n$.

Example 1: $a = 3$, $n = 10$. Then: $\phi(10) = 4$. We have: $3^4 = 81$. We can easily see that Euler's theorem is true: $81 \equiv 1 \bmod 10$.

Example 2: $a = 2$, $n = 11$. Then: $\phi(11) = 10$. We have: $2^{10} = 1024$. We see that: $1024 \equiv 1 \bmod 11$.

Remark: If n is prime, then $\phi(n) = n - 1$, and so we have Fermat's theorem (also known as Fermat's Little Theorem).

Solving expressions like $26^{23} \bmod 51$

Or $M = 66^{77} \bmod 119$
We will consider two different ideas, and then proceed to get them implemented as an easily-programmable algorithm.

1. $[(a \bmod n) \times (b \bmod n)] \bmod n = (ab) \bmod n$
   Check this:
   $17 \times 13 \bmod 5 = 221 \bmod 5$. Have: $221/5 = 44+$ rem. 1 which gives an answer of 1.
   Alternative: $17 \bmod 5 = 2; 13 \bmod 5 = 3; 2 \times 3 \bmod 5 = 1$.
2. Taking this idea further:
   $x^{16} = x \times x \times \ldots \times x$, resulting in $16 - 1 = 15$ multiplications.
   Alternative: $x^{16} : x \times x = x^2 \longrightarrow x^4 \longrightarrow x^8 \longrightarrow x^{16}$, resulting in 4 multiplications.
   We will now proceed to operationalize these two ideas.

Suppose we are trying to determine a^m with a and m positive integers.
Express m as a binary number: $m = b_k b_{k-1} \ldots b_0$ where each b is a binary digit.
Example: $17 = 1\ 0\ 0\ 0\ 1$ implying $b_4 = 1, b_3 = 0, b_2 = 0, b_1 = 0, b_0 = 1$.
Rewriting: $m = \sum_{i=0}^{k} b_i 2^i$ with $b_i \in \{0, 1\}$.

Therefore: $a^m = a^{\sum_{i=0}^{k} b_i 2^i} = \Pi_{i=0}^{k} a^{b_i 2^i}$
(Read: product over $i = 0, 2, \ldots k$ of a to the power of $b_i 2^i$.)
For the example of a^{17}:
$a^{1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0} = a^{1\times 2^4} \times a^{0\times 2^3} \times a^{0\times 2^2} \times a^{0\times 2^1} \times a^{1\times 2^0}$.
So, now, finally:
$a^m \bmod n = [\Pi_{i=0}^{k} a^{b_i 2^i}] \bmod n = (\Pi_{i=0}^{k} [a^{b_i 2^i} \bmod n]) \bmod n$.
Here we are generalizing: $(ab) \bmod n = [(a \bmod n)(b \bmod n)] \bmod n$.

Algorithm:

$c \rightarrow 0; d \rightarrow 1;$
for $i \rightarrow k$ downto 0{
$c \rightarrow 2 \times c;$
$d \rightarrow (d \times d) \bmod n;$
if $b_i = 1$ then {
$c \rightarrow c + 1;$
$d \rightarrow (d \times a) \bmod n;$
}
}
return d

Note that variable c is not output from the above.
Example: $n = 561, a = 7, m = 560$.
$a^m \bmod n \Longleftrightarrow 7^{560} \bmod 561$.
$m = 560 \Longrightarrow b_k b_{k-1} \ldots b_0 = 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0$. We find $k = 9$.

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	$2.0+1$ $=1$	2.1 $=2$	2.2 $=4$	2.4 $=8$	$2.8+1$ $=17$	$2.17+1$ $=35$	2.35 $=70$	2.70 $=140$	2.140 $=280$	2.280 $=560$
d	7	49	157	526	160	241	298	166	67	1

Answer: 1.
How did we get:
$d = 7$? Case of $b_i = 1$ so $d \leftarrow (d \times a) \bmod n = 1 \times 7 \bmod 561 = 7$.
$d = 49$? Case of $b_i \neq 1$ so $d \leftarrow (d \times d) \bmod n = (7 \times 7) \bmod 561 = 49$.
$d = 157$? Case of $b_i \neq 1$ so $d \leftarrow (d \times d) \bmod n = (49 \times 49) \bmod 561 = 2401 \bmod 561$.
Have $2401/561 = 4 +$ rem 157.

Another example (start only): $66^{77} \bmod 119$ (which, as we have said, $= 19$).
$77 = 1\ \ 0\ 0\ 1\ 1\ 0\ 1$ so that $k = 6$.

i	6	5	4	3	2	1	0
b_i	1	0	0	1	1	0	1
c	1	2	4	9	19	38	77
d	66						19

Generating large prime numbers

– Sieve of Eratosthenes.
– Pick a number, initially 2.
– Let 2 = first prime number.
– Divisible by a prime number found so far? No. Keep 2.
– Add 1 to number.

– Divisible by 2? No. Keep 3.
– Add 1.
– Divisible by 2, 3? Yes for 2. Reject.
– Etc. Not too practical for large primes.
– Note: there are "weak" and "strong" prime numbers for RSA.

– Alternative: we choose a number and then test for primality.
– Following algorithm like in previous section: we are working on the number $(n-1)$ represented by bits $b_k b_{k-1} \dots b_1 b_0$. The algorithm is to calculate $a^{n-1} \bmod n$.
– We know from Fermat's theorem that $a^{n-1} \equiv 1 \bmod n$ if n is prime.
– So if we solve $a^{n-1} \bmod n$ and find the result to be not equal to 1, then n is not prime.
– The algorithm we looked at above will do this test for us.
– So we choose an odd number n, and a randomly integer $a < n$, and carry out this test.
– Algorithm returns either "not prime" or "possibly prime".
– Let the probability of the above procedure saying that n is "possibly prime" and in fact is not prime be less than 0.5.
– So, repeatedly invoke the algorithm, using different values of a.
– If we get a "possibly prime" outcome s times in succession, then the probability that n is prime is at least $1 - 2^{-s}$.
– We take lots of values of a, and each test strengthens our case. There remains a small chance of a non-prime slipping through.
– This completes the overview of this probabilistic algorithm for testing whether or not we have a prime number.
– It is an example of a stochastic algorithm for primality testing.
– We take values for a and increase the evidence (and our confidence) in favour of the number being prime.
– We will never know for certain with this procedure if the number is prime.
– But we also know that the probability is arbitrarily low that it is non-prime.

El Gamal encryption

– Based on discrete logarithm problem (which we will not discuss).
– Unlike RSA, El Gamal encryption has public parameters which are shared between a number of users. They are called domain parameters.

- p, a large prime, around 1024 bits,
- $p-1$ is divisible by another prime, q, of around 160 bits,
- g such that $g^{(p-1)/q} \bmod p \neq 1$.

Then, one creates

- private key, x, an integer,
- public key, $h = g^x \bmod p$.

– Note: in RSA two large primes are needed. In El Gamal, a random number is needed, plus a modular exponentiation.
– To encrypt a message, m (also in mod p)

- generate a random ephemeral key, k,
- set $c_1 = g^k$
- set $c_2 = mh^k$
- which gives ciphertext $c = (c_1, c_2)$.

– Each message has a different ephemeral key, so encrypting the same message twice will produce different ciphertexts.

– To decrypt a ciphertext $c = (c_1, c_2)$:

$c_2/c_1^x = mh^k/g^{xk} = mg^{xk}/g^{xk} = m$ all with modulus p.

– Example: $p = 809, q = 101, g = 3$

– Aside: recall that Fermat's Little Theorem states that if p is prime, and a is a positive integer not divisible by p, then it holds: $a^{p-1} \equiv 1 \bmod p$, and hence $3^{808} \equiv 1 \bmod p$.

– For public and private keys, choose: $x = 68$, and $h = g^x \bmod p = 3^{68} \bmod 809 = 65$. (Exercise: check this.)

– Encrypt a message, $m = 100$.

– Generate a random ephemeral key, $k = 89$.

– $c_1 = g^k \bmod p = 3^{89} \bmod 809 = 345$. (Exercise: check this.)

– $c_2 = mh^k \bmod p = 100(65^{89} \bmod 809) \bmod p = 100 \times 709 \bmod p = 517$. (Exercise: check this.)

– Ciphertext, $c = (345, 517)$.

– Decryption: $c_2/c_1^x = 517/345^{68} = 517/709$. (Exercise: check this.)

– Here we look a bit stuck. Let's see... what is $709^{-1} \bmod 809$? We see that we need the multiplicative inverse of 709 modulo 809.

– What is $2^{-1} \bmod 5$? Answer is 3, since $2 \times 3 = 6 \equiv 1 \bmod 5$.

– Here is how we solve this. We first determine $gcd(709, 809)$ using the Euclid algorithm. Do this. We solve for $gcd(809, 709)$:

$809 = 1 \times 709 + 100$ $\quad gcd(709, 100)$

$709 = 7 \times 100 + 9$ $\quad gcd(100, 9)$

$100 = 11 \times 9 + 1$

So $gcd(809, 709) = 1$ (hence these numbers are relatively prime).

Now we seek the linear combination $d = ua + vb$ for $d = gcd(a, b)$.

We back-substitute from the Euclid algorithm in order to do this:

$1 = 100 - 11 \times 9$

$= \mathbf{809} - 1 \times \mathbf{709} - 11(\mathbf{709} - 7 \times 100)$

$= \mathbf{809} - 1 \times \mathbf{709} - 11 \times \mathbf{709} + 77(\mathbf{809} - 1 \times \mathbf{709})$

$= \mathbf{809} - 12 \times \mathbf{709} + 77 \times \mathbf{809} - 77 \times \mathbf{709}$

$= 78 \times \mathbf{809} - 89 \times \mathbf{709}$

(Excercise: check this.)

From this, we read off the multiplicative inverse of 709 with modulus 809, i.e. $709^{-1} \bmod 809 = -89$.

We need a positive integer, so knowing that $-1 \times 809 = 720 - 89$ we find the solution as 720.

Good, now back to solving 517/709 mod 809.

We have $517/709 \bmod 809 = 517 \times 709^{-1} \bmod 809 = [(517 \bmod 809)(709^{-1} \bmod 809)] \bmod 809 = 517 \times 720 \bmod 809 = 100$. QED.

– Another version of El Gamal uses elliptical functions. (We won't deal with these.)

– Rabin encryption: based on the difficulty of extracting square root modulo $n = pq$.

Authentication

1. Message confidentiality

– Disclosure

– Traffic analysis

2. Message authentication

– Masquerade

– Content modification

– Sequence modification

– Timing modification

3. Digital signature
– Repudiation

Encryption ensures against eavesdroppers
Authentication ensures against imposters
Integrity ensures against modified messages

Authentication functions

– MAC for **integrity**, i.e. demonstrably no tampering with message.
– Message **encryption**: public, private.
– Message authentication code (MAC): cryptographic checksum, appended to message.
– Hash function: a message is mapped onto a fixed length code.

MAC: need not be encrypted itself. It can be used to compare, using shared secret key, the message as transmitted at the sender end, and as received at the recipient end.

MAC $= f_k(M)$ where M is message, k is secret key, f is check function. Transmitted then is: $M \|$ MAC.

MAC is therefore separate from the issue of encryption.

Why MAC and not just encryption?

– E.g. message in the clear, one recipient (among many recipients) is responsible for authentication.
– Heavy load, encryption, one recipient responsible for authentication.
– Authentication of software, which itself doesn't have to be encrypted.

Example of MAC, based on DES, referred to as Data Authentication Algorithm.
– CBC (cipher block chaining) mode of DES, with initialization of zero.
– 64-bit blocks: D_1, D_2, ... , D_N. Last: padding with zeros if necessary.
– E = DES encryption algorithm, secret key, K.
– DAC, data authentication code:
$O_1 = E_K\ (D_1)$
$O_2 = E_K\ (D_2 \oplus O_1)$
$O_3 = E_K\ (D_3 \oplus O_2)$
...
$O_N = E_K\ (D_N \oplus O_{N-1})$
– Finally, DAC returns O_N or leftmost M bits of this block, with $16 \leq M \leq 64$.
– I.e. take final block as the MAC, after an optional post-processing stage and truncation.

Alternative based on hash functions: hash code can be appended to message and then all encrypted.

Simple hash function:
$c_i = b_{i1} \oplus b_{i2} \ldots \oplus b_{im}$
c_i is ith bit of block.
b is block, $\oplus$ is XOR.

	Fixed length	public function	key
D.S.	–	–	Y
MAC	Y	Y	Y
Hash	Y	Y	–

Informally, we could say that D.S. is to check human violation, MAC is to check "technical" and human violation, and Hash is to check "technical" violation.

Hash functions

– Basic properties include:
– 1. Ease of computation.
– 2. Compression. $h(x)$ is of fixed length.
– 3. Pre-image resistance. Given y, it is difficult to find x s.t. $h(x) = y$.
– 4. Weak collision resistence. Difficult to find x and x', $x \neq x'$ s.t. $h(x) = h(x')$.
– 5. Strong collision resistence. Infeasible to find...

– Points 1 to 4 lead to one-way hash function. Point 5 leads to a collision-resistent hash function.
– Output of hash function: hash value, message digest, checksum.
– Examples: SHA-1 (Secure Hash Algorithm 1), MD4, MD5. SHA-1 processes 512-bit blocks and generates 160-bit hash values.
– 80 rounds of processing on five 32-bit inputs. Combinations of AND, OR, NOT and XOR operations used.
– See Stallings.
– Rivest's MD5, Message Digest Algorithm.
– Arbitrary length message is mapped into 128-bit hash ("fingerprint of message digest").
– Seek:
(1) "Impossible" to find 2 messages which produce same result.
(2) "Impossible" to generate a message that will produce a predefined hash result.
– Estimated that the number of operations required to generate two messages that produce the same MD is 2^{64}.
– Complexity of finding a message with a predefined MD is 2^{128}.

Message integrity with hash functions, and birthday attacks

– Hash function: take variable input and map into set of output values of fixed length.
– Hash function is publicized.
– Integrity of the message can be checked by use of its hash value.
– Hash function should be a one-way function.
– Hash function must be collision-free. I.e. for two messages M and M', we cannot have h(M) = h(M'). A "birthday attack" on message integrity makes use of the birthday paradox.
– Birthday attack: it is probable enough that an alternative message produced the same hash code.
– Say we have 23 people at a party. Then the probability that two or more will have the same birthday is greater than 50%. How come? For 365 days in the year, we would expect that two people having the same birthday would be a fairly improbable event.
– We have stated the problem as "two or more people" having the same birthday. So let us put the problem into reverse gear for the moment, and ask what is the probability of two or more people do *not* share the same birthday.
– We proceed as follows. Let r = 23. Let n = 365.
– We take one of the people (on the list of r-values), and cross this person's birthday off the n-list. What is the probability that we do *not* find an already-selected birthday?
– For first person, we have no already-selected birthday, so this probability must be 1.
– For second person, probability is (n – 1)/n.
– For third person, probability is (n – 2)/n.
– Etc. Putting this all together (remember: a joint event means that the individual probabilities are multiplied):

P(no birthdays the same) = $\frac{n}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \ldots \cdot \frac{n-r+1}{n} = \frac{n!}{(n-r)!n^r}$

– For n = 365 and r = 23, this probability is 0.493.
– Therefore the probability that two or more people share the same birthday is 1.0 − 0.493 = 0.507.

– Translate this to the area of hash functions: for a given length of hash code, n, and a given number of attempts at cracking it, r, we find a particular probability, p.

– For the probability of cracking the hash code to be less than 10^{-6} after 2^{55} attempts, then we must have $n > 2^{128}$. I.e., hash code length, n, must be at least 128 bits.

– We can use DES as a hash function.

– Message, M, is enciphered with DES and a secret key.

– Result is added to M, modulo 2.

– Key for subsequent blocks could be defined from the hash function. If the hash function is based on a specified number of bits, then these can be read off (e.g., from the left of the final outcome).

Key distribution – and Diffie-Hellman

– Key distribution, communiction: why important?

– Problem of getting the top secret symmetric key between sender and receiver.

– Points to need for public keys, i.e. not top secret.

– Key distribution problem tackled by Diffie, Hellman and Merkle.

– Little story of Alice sending message to Bob, with malevolent Eve trying to intercept.

> Message integrity – was message okay?
>
> Entity authentication – was "Alice" really Alice?
>
> Message authentication – both together

– Principle:
A $\longrightarrow$ B: $E_A(M)$
B $\longrightarrow$ A: $E_B\ E_A(M)$
A decrypts using her own key leaving: $E_B(M)$
A $\longrightarrow$ B: $E_B(M)$
Now B knows his key and can decrypt.

– Alternatively:
$E_A(M)$
$E_B\ E_A(M)$
$D_A\ E_B\ E_A(M)$
$D_B\ D_A\ E_B\ E_A(M)$

But the problem is: is this algorithmically possible? What class of functions allow us to do other than the normal "last in, first out" calculation?

One-way functions are needed. (Reminiscent of mixing a number of paint colours. Easy to mix, practically impossible to separate afterwards.) Modular arithmetic is rich in one-way functions. One-way function can provide a solution to the "last in, first out" dilemma.

Use function like: $y^x (\mathrm{mod} p)$ where y and p are in the public domain – or at least not really critical.

Example where $y = 7$, and $p = 11$.

	A	**B**
Step 1	Secret key chosen, e.g. 3	Secret key chosen, e.g. 6
Step 2	Encrypt: $7^3 (\mathrm{mod} 11)$	Encrypt: $7^6 (\mathrm{mod} 11)$
	$= 343 \bmod 11 = 2$	$= 117649 \bmod 11 = 4$
Step 3	A sends 2 to B	B sends 4 to A
Step 4	Works out: $4^3 (\mathrm{mod} 11)$	Works out: $2^6 (\mathrm{mod} 11)$
	$= 64 \bmod 11 = 9$	$= 64 \bmod 11 = 9$

Hence they have ended up with one and the same key.
One could say that a secret key has been established by public discussion.
Diffie-Hellman-Merkle key exchange protocol.

Alice		Bob
a	$g^a \longrightarrow$	g^a
g^b	$\longleftarrow g^b$	b

– Ephemeral "secrets", a, b.
– Alice can compute $K = (g^b)^a$ since she knows a and was sent g^b by Bob.
– Bob can also compute $K = (g^a)^b$ since he knows b and was sent g^a by Alice.
– Attacker Eve can see g^a, g^b and needs to recover $K = g^{ab}$. Not easy!

– Another example: $p = 2147483659$, $g = 2$.

Alice		Bob
$a = 12345$		$b = 654323$
$A = g^a = 428647416$	given to Bob	A
B	Given to Alice by Bob	$B = g^b = 450904856$

Secret key: $A^b = 428647416^{654323} \bmod p$
$= B^a = 450904856^{12345} \bmod p$
= same key.

– Usually p is around 2^{1024}. Elliptic curves allow for use of $p = 2^{160}$.
– Diffie-Helman key agreement protocol: as seen.
– Diffie-Helman key transport protocol: one party uses a long-term key of the form g^a instead of ephemeral "secret".

– Problem of authenticating sender. When someone says he is Bob, how do you know it is not really Eve?

Alice		Eve		Bob
a	$\longrightarrow$	g^a		
g^m	$\longleftarrow$	m		
g^{am}		g^{am}		
		n	$\longrightarrow$	g^n
		g^b	$\longleftarrow$	b
		g^{bn}		g^{bn}

– So Alice agrees a key with Eve, thinking it is Bob.
– Bob agrees a key with Eve, thinking it is Alice.
– Eve examines communications as they pass through her. No detection of this.
– Solution: digital signature.

Digital signature

– To authenticate the sender.
– Let the sender encrypt a message using his/her private key.
(Not a good idea in principle, since anyone can decrypt this using the sender's known – axiomatically – public key.)
– Then decrypt using the sender's public key.
We know therefore that it could only have come from them.
– This is a public key system in reverse!
– Used e.g. for authentication of software.

Left → right	Encryption: You use: my public key	Decryption: I use: my private key
Left ← right	Signature verification: I use: your public key	Signing: You use: your private key

– We could use RSA for both data and the digital signature:
– Before A sends message, a digital signature is added which is encrypted with A's private key, S_A. (S = secret.)
– Then encrypted with public key of B, P_B.
– Sent.
– B then decrypts using S_B.
– Authenticates using A's public key, P_A.
– Overall: P_A S_B P_B S_A (M)

– Hash function may precede use of RSA to make for compact set of data.

Public key infrastructure, PKI

– Set of protocols, services and standards supporting applications of public key cryptography. Relatively recent use of term.
– Services include: **key registration** – issuing a new certificate for a public key; **certificate revocation** – cancelling a previously issued certificate; **key selection** – obtaining a party's public key; **trust evaluation** – determining whether a certificate is valid, and what operations it authorizes.
– There is no single pervasive PKI today. Often various "root" or "top level" certificate authorities in a global PKI.
– PKI standardization efforts are underway by various governments and standards bodies.

- Distribution of public keys
— Public announcement (cf. popular use of PGP)
— Publicly available directory (trusted organisation)
— Public key authority
— Public key certificates

- Distribution of private keys
— With or without confidentiality and authentication

See Stallings, section 6.3
See RSA "Frequently Asked Questions About Today's Cryptography", especially "Public Key Issues",
http://www.rsa.com/rsalabs/faq/4-1-3.html

– Key pair: anyone who wishes to sign messages or to receive encrypted messages must have a key pair.
– Companies may use one or more pairs for encryption – maybe keys stored under key escrow to safeguard the key in event of loss, maybe a single key non-escrowed for authentication.

– Key escrow.

– Trusted third parties.

... keep copies of private keys, to aid in key recovery.

– User can generate own key pair, or company officer. Kerberos is a secret-key authentication system, which uses a central server to generate keys.

– Kerberos

– Authentication system for use in open distributed computing environment.

– From MIT Project Athena.

– DES-based.

– After key pair generation, a user must register the public key with a central administration called a Certifying Authority, CA. The CA returns a certificate attesting to the validity of the user's public key.

– In RSA, private key (modulus, exponent) is unique. But the exponent could be shared by a group of users. Some values could be used as "public exponents" so that public key operations – encryption and signature verification – are relatively fast compared to private key operations – decryption and signing.

Security issues in relation to PKI agencies

1. Every key should have an expiration date to guard against long-term cryptanalytic attack.
2. Appropriate key size can be defined in relation to expiration date.
3. Signature verification should include check for expiration date.
4. An expired key may need to be retained for a period in order to decrypt still outstanding encrypted messages.
5. If a key is lost, the CA revokes the relevant certificate. A certificate revocation list, CRL, is kept up to date.
6. If a private key is compromised, the relevant CAs must be notified for revocation of the relevant certificates. Then a new key pair must be generated, and a new certificate obtained.
7. A private key should be stored encrypted under a password. Preferably on disk which is not network-accessible.
8. Similarly private keys at CAs have to allow for illegal acts by their employees, etc. Several people must use their data keys to access e.g. a system called SafeKeyper from BBN.
9. Passing public key: in the clear, but one should verify the key (e.g., by computing a hash of the key and verifying by some independent communication).

Certificates

– Certificates: digital documents attesting to the binding of a public key to an individual or other entity. In some cases, there may be a chain of certificates, each one certifying the previous one.

– In simplest form, certificates contain a public key and a name. Also: expiration date, name of CA, digital signature of the certificate issuer. Most widely used format for certificates is X.509.

– Certificates typically used to generate confidence in the legitimacy of the public key. They are a type of digital signature which protect keys from forgery, false representation, or alteration. Verification can be performed with greater or lesser rigour.

– Could associate one or more certificates with every signed message. Alternative is a hierarchical certificate chain, with one certificate testifying to the authenticity of the previous certificate. A top level certifying authority is trusted without a certificate from any other certifying authority.

– Examples of CAs: Baltimore Technologies, RSA Security, VeriSign. Or a CA protocol is found in Apple's Mac System 7.5.

– Preventing attacks on CAs: extensive cryptanalytic attack is preempted by regular key change; if an old key is broken, then time-stamping becomes important; impersonation is avoided by personal indentification, notarization, etc.; bribery or other CA employee illegality is limited by more than one employee being required to issue a certificate.

X.509 (ITU) certification authority

– Part of set of standards for directory services.

– Such services include also: user names, network addresses, etc.

– Repository of public key certificates. Each certificate: public key of user, lodged by and signed with the private key of a trusted certificate authority.

– Alternative authentication protocols are supported also.

– X.509 CA is used in SSL, SET and elsewhere. (On SSL and SET, see below.)

– In X.500, Distinguished Names can be defined. E.g., country or state names, company or trading name. (ITU X.400 standards relate to email. X.500 standards relate to directory services – naming and name resolution conventions, yellow page services, etc.)

– See Stallings, section 11.2.

Standards authorities

– (Not just related to public keys, of course.)
– ANSI
– ITU (CCITT)
– ISO
– IEEE
– IETF
– Industry consortia
– Government bodies, agencies

Zero-knowledge techniques

– In authentication, to prove your identity, you say something about yourself. E.g. password. But now your identity is known. What if the partner in the exchange exploited this knowledge to pretend that he/she was you?

– Zero knowledge techniques authenticate you without giving secret information away.

– Use techniques similar to RSA. But, for digital signatures, much more efficient techniques than RSA are used.

– Identification sequence is established which, for the correct person, must lead to a right conclusion for that person.

– See pp. 181-190 of van der Lubbe for a good description.

Fair cryptosystem

– How to cater for criminal use of cryptography?
– Force surrender of keys is threat to person's privacy.
– Fair cryptosystem: authorized body can decrypt.
– How?
– Deposits parts of key with trusted body.
– Clipper, capstone.

Two encryption systems, resp. for telephone and computer communication, adhering to American Escrowed Encryption Standard, adopted in 1994.

System of key escrow: a pre-installed chip holds half your private key, and the other half is lodged with a government agency.

Considered to have failed.

Internet and web security issues

– Have a look at the Internet protocols page at
http://www.rad.com/networks/1997/nettut/protocols.html
– Note: (i) OSI 7-layer model – Application, Presentation, Session, Transport, Network, Link and Physical; Internet protocol suite – in layers 5, 6 and 6, Telnet, ftp, SMTP, SNMP, HTTP, and others; in layer 4, tcp, UDP; in layer 3, IP.
– Internet security consists of three distinct services: (i) access security, (ii) transaction security, and (iii) authentication.

– Access security includes passwords and the use of proxy servers.
– What is a proxy server? It is a specialized http server. Used in conjunction with corporate firewall. Firewall implies no direct http access to outside world. Proxy is an indirect connection, in that everything is chanelled through it. A proxy can cache documents efficiently from the outside world.

– Transaction security includes SSL, Secure Socket Layer.
– Authentication includes password, digital signature, or biometric technique (fingerprint, iris).
– SSL runs on top of a transport protocol, but under application protocols such as http and ftp. Operates between browser and a server.
– SSL protects the link, not the documents or files transferred. It authenticates the server, and optionally the browser.
– Typically uses port 443. HTTP usually uses port 80. If SSL is in use, often the URL begins with https:// rather than http://
– SSL makes use of RSA as its public-key cipher, X.509 certificates for authentication, and DES or some other private key system for its symmetric cipher.
– TLS (Transport Layer Security) is a protocol like, and based on, SSL. WTLS (Wireless TLS) is used in WAP (Wireless Application Protocol), and is therefore best for low-bandwidth bearer networks.
– SSH (Secure Shell) permits secure remote access across a network from one computer to another. Effectively a secure telnet, rlogin or rsh.

– SET, Secure Electronic Transfer.
– Designed for credit card use over Internet. Developed jointly by Visa and Mastercard. Specification is open.
– Secure communication, X.509 certificates, privacy.
– Partners: cardholder, merchant, issuer. Also: certificate authority, payment gateway and acquirer (latter: financial institution processing authorization and payment).
– Uses DES for bulk data encryption, and RSA for signatures and public-key encryption of data encryption keys and bankcard numbers.

– S/MIME (Secure / Multipurpose Internet Mail Extensions) adds digital signatures and encryption to MIME messages. MIME is an email standard: (structured) header and (unstructured) body of message are specified. MIME allows for support of multimedia files (image, audio, video, etc.)

– PGP is due to Phil Zimmermann. Used, e.g., in email. Multiple platform. Address: www.pgp.com, www.pgpi.com

– Lots of difficulties due to (i) RSA holding licences, (ii) US Govt's objections to export of anything strongly encrypted, and even more so the technologies behind strong encryption.

– So put onto a newsgroup in 1991.

– Symmetric encryption is fast, IDEA even more than DES.

– Symmetric key is a problem, in practice.

– RSA looks after key distribution well, but is computationally more costly.

– PGP combines these: RSA is used for the key, and IDEA for the message.