Cryptography

pieces from work by Gordon Royle

The set-up

Cryptography is the mathematics of devising secure communication systems, whereas *cryptanalysis* is the mathematics of breaking such systems.

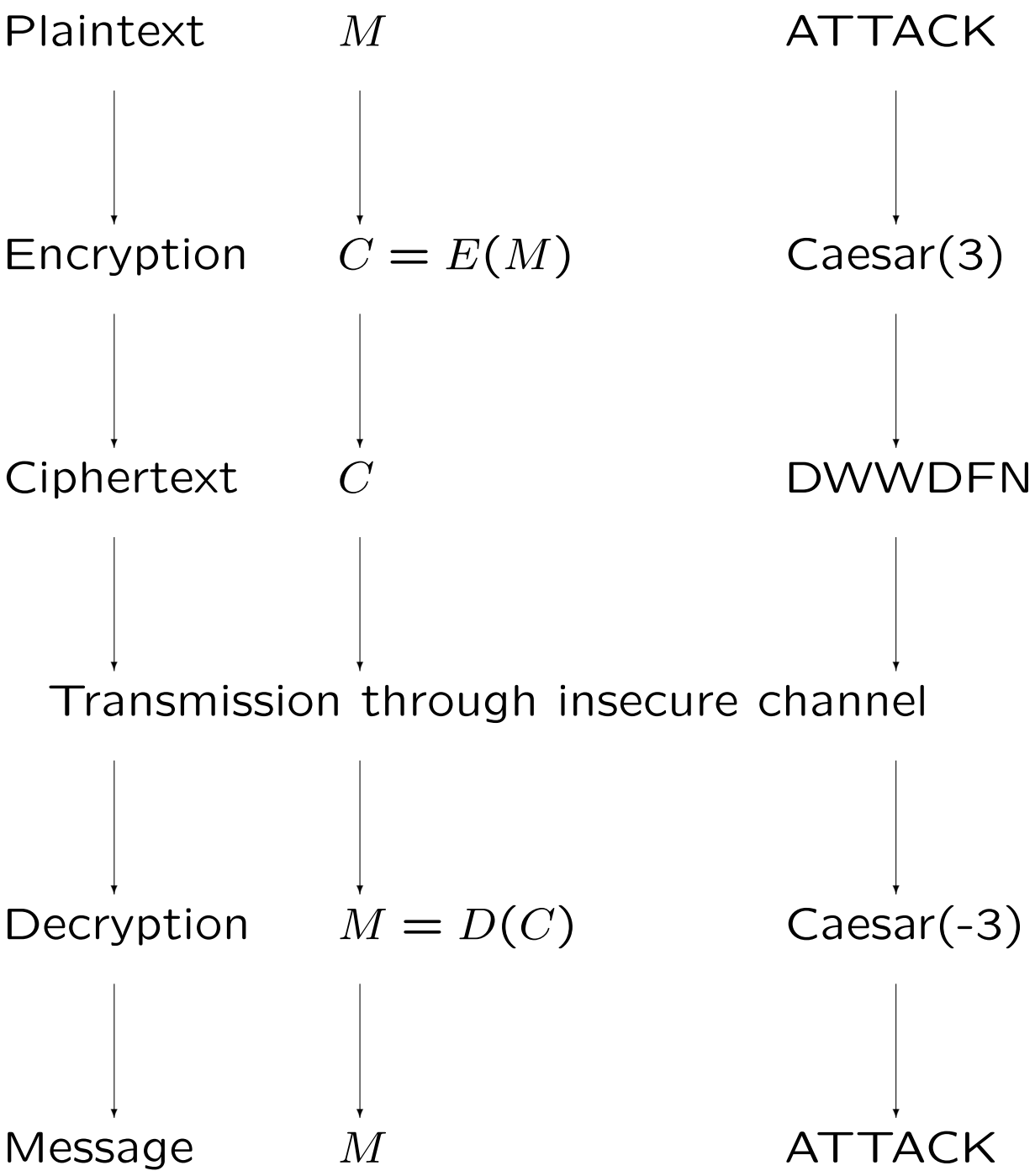
We suppose that Alice employs some encryption system E, and encrypts the message to form the *ciphertext* C

$$C = E(M)$$

On receipt of the ciphertext, Bob then *decrypts* the message recovering the plaintext.

$$M = D(C) = D(E(M))$$

Cryptography



Some attacks

Eavesdropping An attacker may learn something from the message passing from Alice to Bob.

Impersonation An attacker may wish to impersonate someone — for example a student at Dartmouth University sent forged electronic mail messages to his entire classs claiming that an exam was cancelled.

Alteration An attacker may intercept a message in transit, and try to change it to read something different.

It is widely held that it is necessary to be an excellent cryptanalyst in order to be a good cryptographer.

The key space

26 for shift cipher.

One way to increase the key space is to use a *key phrase* instead. Suppose that the key-phrase is a word, say `FIZZLEBOP`. For better subsitution cipher write down a copy of the alphabet

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Underneath, write down the key phrase with repeated letters deleted and then the remainder of the alphabet.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ

FIZLEBOPACDGHJKMNQRSTUVWXY
```

Exponentiation

Compute 11^{16} (it is 45949729863572161) and then take this value modulo n.

However we can compute as follows

$$11^2 \equiv 11 \times 11 \equiv 121 \pmod{233}$$

$$11^3 \equiv 11 \times 121 \equiv 166 \pmod{233}$$

$$11^4 \equiv 11 \times 166 \equiv 195 \pmod{233}$$

and so on ...

This naive algorithm takes 16 operations to compute the 16th power of a number.

Repeated squaring

We can be much more efficient if we observe that

$$x^{16} = (x^8)^2$$

Therefore we can proceed by

$$11^2 \equiv 11 \times 11 \equiv 121 \pmod{233}$$

$$11^4 \equiv 121 \times 121 \equiv 195 \pmod{233}$$

$$11^8 \equiv 195 \times 195 \equiv 46 \pmod{233}$$

$$11^{16} \equiv 46 \times 46 \equiv 19 \pmod{233}$$

therefore taking only 4 operations to compute the value.

Naturally it will only be on rare occasions where the exponent is an exact power of 2.

Nevertheless it is easy to extend the idea behind the repeated squaring to accommodate any exponent.

Exponentiation

Suppose we have to compute $g^a \pmod{n}$. Then let

$$(b_k, b_{k-1}, b_{k-2}, \ldots, b_0)$$

be the binary expansion of the exponent a.

Then start with the value $d = 1$ and read the binary expansion of a from high-order bit to low-order bit.

At each stage

$$d \leftarrow d^2 \pmod{n}$$

and if the bit is a 1, perform an additional operation

$$d \leftarrow d \times g \pmod{n}$$

After reading each bit the variable d will hold the required value.

Example

We will compute $11^{25} \pmod{233}$.

The binary expansion of 25 is 11001 so the procedure is as follows:

Step	Bit	Square	Multiply
1	1	1	11
2	1	121	166
3	0	62	
4	0	116	
5	1	175	61

Therefore we can compute $g^a \pmod{n}$ in only $\lg a$ steps, each of which involves at most two modular multiplications.

Euclid's algorithm

Euclid's algorithm for finding the greatest common divisor of two integers is probably the world's earliest surviving algorithm.

Then we can express a in the following manner

$$a = qb + r$$

where q is the *quotient* on dividing a by b and $r < b$ is the *remainder*.

Now $d = \gcd(a, b)$ must divide b and r, and moreover it is easy to see that any number that divides b and r must also divide a.

Therefore

$$d = \gcd(a, b) = \gcd(b, r)$$

Euclid's algorithm continued

This yields a trivial recursive function:

function gcd(a, b)
if $b = 0$ **return** a
else return gcd$(b, a \bmod b)$

An example is trying to compute gcd$(450, 42)$.

Call	a	$=$	$q.b$	$+$	r
gcd$(450, 42)$	450	$=$	10.42	$+$	30
gcd$(42, 30)$	42	$=$	1.30	$+$	12
gcd$(30, 12)$	30	$=$	2.12	$+$	6
gcd$(12, 6)$	12	$=$	2.6	$+$	0
gcd$(6, 0)$					

and so gcd$(450, 42) = 6$.

Now, in addition to simply computing the greatest common divisor d of a and b, we will also need to express d as a linear combination of a and b — that is, we need to find x and y such that

$$d = xa + yb$$

Extended Euclid formally

Let us create an extended version of the function gcd(a, b) that returns a triple (d, x, y) such that $d = \gcd(a, b) = xa + yb$.

if $b = 0$
return $(a, 1, 0)$

The recursive case requires a thought:

$$a = qb + r$$

and we know (from the recursive solution) that

$$d = x'b + y'r$$

then we have

$$d = x'b + y'(a - qb) = y'a + (x' - qy')b$$

Therefore

$(d, x', y') \leftarrow \gcd(b, a \bmod b)$
$(d, x, y) \leftarrow (d, y', x' - y'\lfloor a/b \rfloor)$
return(d, x, y)

The RSA cryptosystem

Devised by Rivest, Shamir, Adleman - 1978

1. Select two large prime numbers p and q (100 or more digits).

2. Compute the product $n = pq$ and the value $\phi(n) = (p-1)(q-1)$.

3. Choose a small odd integer e that is relatively prime to $\phi(n)$.

4. Use Euclid's extended algorithm to solve the equation $ed \equiv 1 \pmod{\phi(n)}$.

5. The public key is (n, e), which can be distributed, and the private key is d.

Using the cryptosystem

Suppose you wish to send Bob the message m. Then you look up Bob's public key (n, e) and compute

$$c = m^e \pmod{n}$$

When Bob receives the message c he computes

$$c^d \pmod{n}$$

using his secret key d.

So Bob computes

$$c^d = m^{ed} = m^{1+k\phi(n)} = m(m^{\phi(n)})^k$$

Now, some elementary number theory tells us that

$$x^{\phi(n)} \equiv 1 \pmod{n}$$

for any x and so we get the result that

$$m^{ed} \equiv m \pmod{n}.$$

Breaking RSA

The security of RSA depends (probably) on the difficulty of factoring the integer n. If you could somehow take n and compute the factorization $n = pq$ then you could compute $\phi(n)$ and hence the decryption key d.

These developments in the last decades have thrust the rather austere and esoteric branch of mathematics known as number theory into the forefront of applied mathematics and computer science, as people seek to develop algorithms to factor numbers or to prove that it is difficult.

It is interesting to note that the problem of factoring numbers is NOT known to be NP-hard.

RSA-129

In 1977, Rivest, Shamir and Adleman wrote a paper in *Scientific American* in which they published the following 129 digit number (this is about 425 bits).

```
11438 16257 57888 86766 92357 79976 14661
20102 18296 72124 23625 62561 84293 57069
35245 73389 78305 97123 56395 87050 58989
07514 75992 90026 87954 3541
```

This was an example of a possible modulus for use in RSA and they promised to pay $100 to the first person to factor this number — they expected it not to be factored in their lifetime.

However a team from Bellcore and MIT devised a factoring algorithm that could be performed in many hundreds of small parts.

Squeamish Ossifrage

Date: Wed, 27 Apr 94 22:03:30 PDT
To: Fun People
Subject: R.S.A. 129 falls

Using volunteers on the Internet, who downloaded portions of the problem using `ftp` and ran them on otherwise idle machines, an international effort using more than 1600 machines for 8 months managed to factor the number:

The two factors are

```
34905 29510 84765 09491 47849 61990 38981
33417 76463 84933 87843 99082 0577
```

and

```
32769 13299 32667 09549 96198 81908 34461
41317 76429 67992 94253 97982 88533
```

Decoding the phrase

```
THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE
```

RSA cracked!!

This feat was widely reported in the popular press - in particular one issue of the *New York Times* had the factorization printed across the entire front page.

One unfortunate side-effect of popular coverage was that the reports often mutated from "RSA-129 has been cracked" into "RSA has been cracked".

Factoring RSA-129 just means that *one particular key* has been cracked. Of course if you were unlucky enough to using that particular value of n as your public key, then you would have to change.

In general however, it means that if you have a 425-bit modulus, then you can expect it to take 1600 machines about 8 months to crack. Simply changing your modulus to a 1024-bit modulus makes a factorization attack completely infeasible (at the moment).