INT202
Complexity of Algorithms

Number Theory and Cryptography

2020-2021

Modular Exponentiation

$$3^{94} \pmod{17}$$

Any number can be represented as the sum of distinct powers of two.

$$94 = 64 + 16 + 8 + 4 + 2$$

How do you pick them?

94 =	1	0	1	1	1	1	0
	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1*64	0*32	1*16	1*8	1*4	1*2	0*1

Modular Exponentiation

$$3^{94} (\mathrm{mod}\ 17)$$

Any number can be represented as the sum of distinct powers of two.

$$94 = 64 + 16 + 8 + 4 + 2$$

$\mathrm{mod}\ 17$

$$3^2 \equiv 9$$

$$3^4 \equiv 81 \equiv 13 \equiv -4$$

$$3^8 \equiv (3^4)^2 \equiv 16 \equiv -1$$

$$3^{16} \equiv (3^8)^2 \equiv (-1)^2 \equiv 1$$

$$3^{64} \equiv (3^{16})^4 \equiv (1)^4 \equiv 1$$

Use the smallest numbers whether they are positive or negative

Modular Exponentiation

$$3^{94} \pmod{17}$$

Any number can be represented as the sum of distinct powers of two.

$\mod 17$

$$\begin{aligned} 3^{94} &\equiv 3^{64+16+8+4+2} \\ &\equiv 3^{64}3^{16}3^{8}3^{4}3^{2} \\ &\equiv (1)(1)(-1)(-4)(9) \\ &\equiv 36 \\ &\equiv 2 \end{aligned}$$

Modular Exponentiation

Theorem [Fermat's Little Theorem] : Let p be prime, and let x be an integer such that $x \mod p \neq 0$. Then

$$x^{p-1} \equiv 1 \mod p$$

Fermat's little theorem is a fundamental theorem in elementary number theory, which helps compute powers of integers modulo prime numbers.

- Example ($p = 5$):
  - $1^4 \mod 5 = 1$
  - $2^4 \mod 5 = 16 \mod 5 = 1$
  - $3^4 \mod 5 = 81 \mod 5 = 1$
  - $4^4 \mod 5 = 256 \mod 5 = 1$

Modular Exponentiation

Theorem [Fermat's Little Theorem] : Let p be prime, and let x be an integer such that $x \mod p \neq 0$. Then

$$x^{p-1} \equiv 1 \mod p$$

Corollary

Let p be a prime. For each nonzero residue x of Z_p, the multiplicative inverse of x is $x^{p-2} \bmod p$

Proof

$$x(x^{p-2} \bmod p) \bmod p = xx^{p-2} \bmod p = x^{p-1} \bmod p = 1$$

$$x^{-1} \equiv x^{p-2} \pmod{p}$$

Euler's function

- The multiplicative group for Z_n, denoted with Z^*_n, is the subset of elements of Z_n relatively prime with n
- The totient function of n, denoted with $\phi(n)$, is the size of Z^*_n
- Example

  $Z^*_{10} = \{ 1, 3, 7, 9 \}$ $\quad\quad$ $\phi(10) = 4$
- If p is prime, we have

  $Z^*_p = \{1, 2, \ldots, (p-1)\}$ $\quad\quad$ $\phi(p) = p - 1$

The general formula to compute φ(n)

Let the prime factorisation of n is given by $n=p_1^{e1}*\ldots*p_n^{en}$, then $\varphi(n) = n*(1-1/p_1)*\ldots*(1-1/p_n)$.

Example:

$15 = 3*5$, $\varphi(15) = 15*(1-1/3)*(1-1/5)=15*(2/3)*(4/5)=8$

$4=2^2$, $\varphi(4) = 4*(1-1/2) =2$

Euler's function

- The multiplicative group for Z_n, denoted with Z^*_n, is the subset of elements of Z_n relatively prime with n
- The totient function of n, denoted with $\phi(n)$, is the size of Z^*_n
- Example

  $Z^*_{10} = \{ 1, 3, 7, 9 \}$ $\quad\quad$ $\phi(10) = 4$
- If p is prime, we have

  $Z^*_p = \{1, 2, \ldots, (p-1)\}$ $\quad\quad$ $\phi(p) = p - 1$

Theorem [Euler's Theorem] : Let n be a positive integer, and let x be an integer such that $\gcd(x, n) = 1$. Then

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

- Example $(n = 10)$

  $3^{\phi(10)} \bmod 10 = 3^4 \bmod 10 = 81 \bmod 10 = 1$

  $7^{\phi(10)} \bmod 10 = 7^4 \bmod 10 = 2401 \bmod 10 = 1$

  $9^{\phi(10)} \bmod 10 = 9^4 \bmod 10 = 6561 \bmod 10 = 1$

Primality Testing

Let $n > 0$ be an integer. How to find out if n is prime? This is a central question in cryptographic computations.

Can we use Fermat's Little Theorem? No because of the existence of **Carmichael** numbers that have the property $x^{n-1} \equiv 1 \mod n$ for all $1 \leq x \leq n-1$, but n is composite. For example, 561 is Carmichael number (561=3*11*17).

This can be done using for example probabilistic methods: We can use a **witness function** witness(x ,n) (wherein x in a random number $1 \leq x \leq n-1$), which gives a definite answer when n is composite and an answer with some error probalility q when n is prime (as there are only 255 Carmichael numbers for integers from 1 to 10^8)

Cryptographic communications

Throughout history there has often been the need (or desire) to *securely* transmit information through *insecure* channels. Such applications include communications for business reasons and military purposes, and most recently, transactions through the Internet.

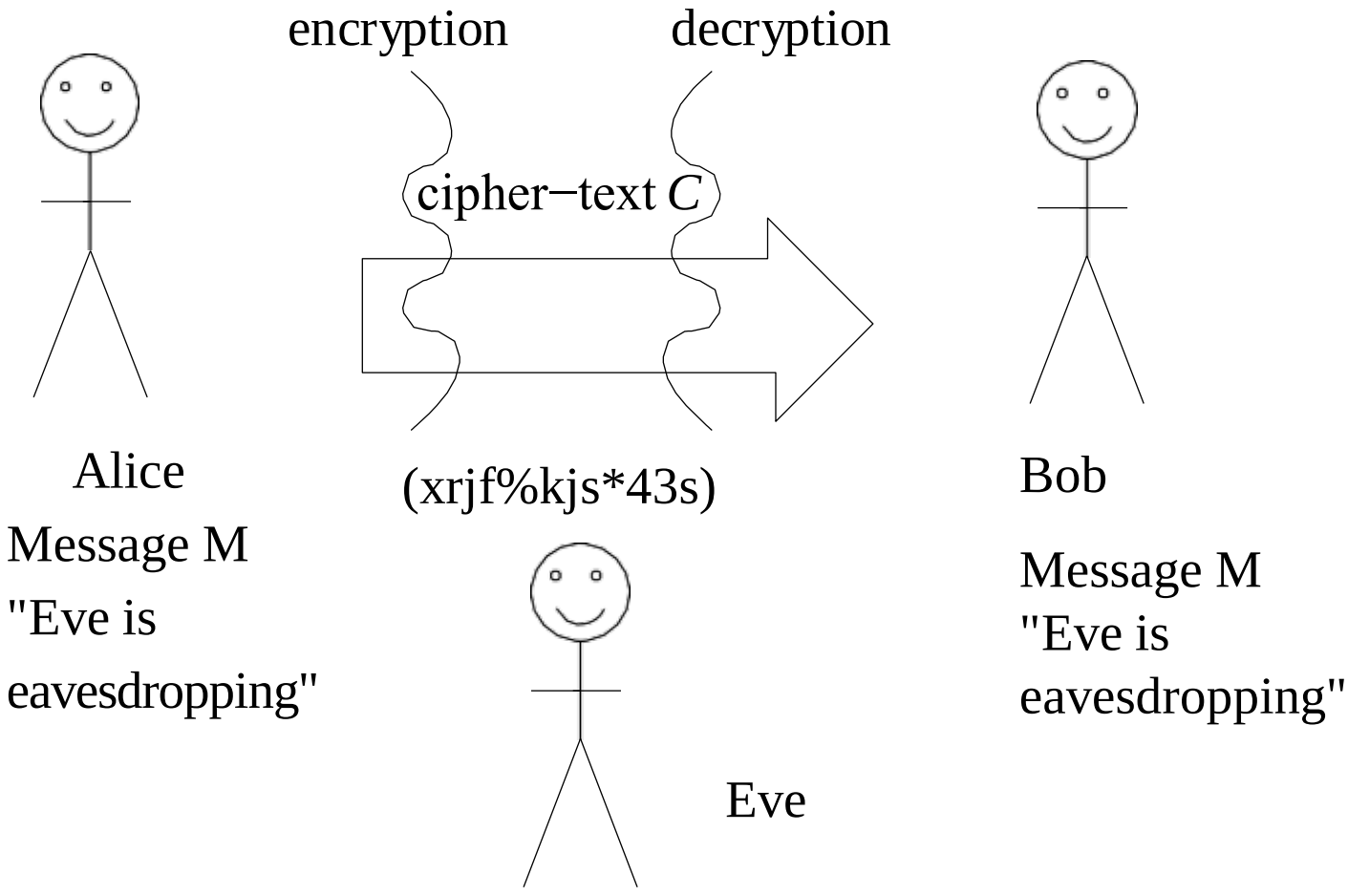
A variety of *cryptographic* methods have been developed to facilitate this type of communication. These include encryption/decryption transformations and digital signatures.

Encryption schemes

Confidentially in communication can be achieved by *encryption schemes*, or *ciphers*.

The general idea behind these schemes is that the message *M* to be sent, often referred to as the *plaintext*, is encrypted into an unrecognizable string of characters *C*, the *ciphertext*.

The ciphertext *C* is then transmitted to the recipient who decrypts *C* to recover the original message *M*.



- Scenario:
  - Alice wants to send a message (plaintext p) to Bob.
  - The communication channel is insecure and can be eavesdropped
    If Alice and Bob have previously agreed on an encryption scheme (cipher), the message can be sent encrypted (ciphertext c)

Symmetric encryption schemes and secret keys

In a traditional encryption scheme a common secret key, *K*, is shared by Alice and Bob.

This common key *K* is used for both encryption and decryption of messages.

Such an encryption scheme is called *symmetric* since the recipient and receiver both have access to the same secret key, and it is used for encryption and decryption.

Symmetric encryption schemes and secret keys

Substitution ciphers

A classic example of a symmetric cipher is a *substitution* cipher. In this case the secret key is a *permutation* π of the characters of the alphabet. (For example, each A gets replaced by the letter D, each *B* gets replaced by the letter *H*, etc.)

Encryption of the plaintext *M* is accomplished by mapping each character x with its corresponding character $y = \pi(x)$.

Decrypting the ciphertext *C* is easily accomplished with knowledge of the permutation π, i.e. each character y of *C* is replaced with $x = \pi^{-1}(y)$.

Symmetric encryption schemes and secret keys

The Caesar cipher

The *Caesar cipher* is an early example of a substitution cipher wherein each character x is replaced by the character $y = (x + k) \bmod n$, where n is the size of the alphabet and the integer k, $1 \leq k < n$, is the secret key.

This type of cipher is called the “Caesar cipher” because Julius Caesar is known to have used it with $k = 3$.

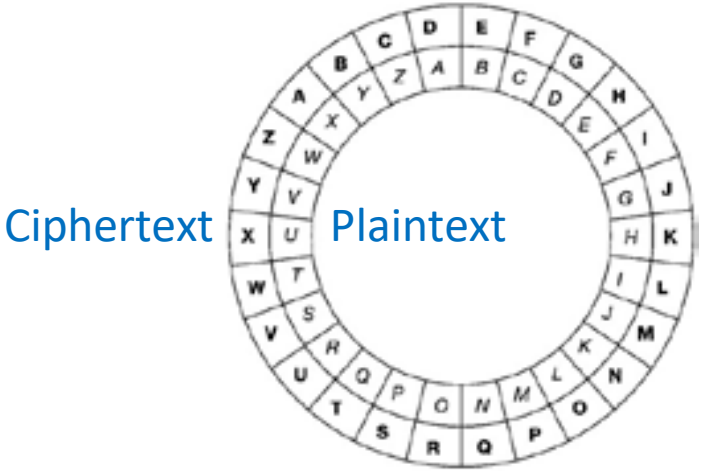
Symmetric encryption schemes and secret keys

Traditional Cryptography

- Ciphers were already studied in ancient times
- Caesar's cipher:
  - replace a with d
  - replace b with e
  - ...
  - replace z with c
- Caesar's cipher is an example of a monoalphabetic substitution cipher, which permutes the characters

N=3

Substitution cipher takes the plaintext alphabets and replace them by other alphabets to generate the ciphertext.



Symmetric encryption schemes and secret keys

Breaking substitution ciphers

While very easy to use, substitution ciphers are not secure.

The secret key of a substitution cipher is very easily broken by using frequency analysis, based on knowledge of the frequency of the various letters, or groups of letters, in the alphabet being used.

Statistical Attacks

- Armed with statistical knowledge about the plaintext language, one can easily break a monoalphabetic substitution cipher
  - Most frequent characters in English: e, t, o, a, n, i, ...
  - Most frequent digrams: th, in, er, re, an, ...
  - Most frequent trigrams: the, ing, and, ion, ...
- The first description of the frequency analysis attack appears in a book written in the 9th century by the Arab philosopher al-Kindi
- Example (S. Singh, The Code Book, 1999):

PCQ VMJYPD LBYK LYSO KBXBJXWXV BXV ZCJPO EYPD KBXBJYUXJ LBJOO KCPK. CP LBO LBCMKXPV XPV IYJKL PYDBL, QBOP KBO BXV OPVOV LBO LXRO CI SX'XJMI, KBO JCKO XPV EYKKOV LBO DJCMPV ZOICJO BYS, KXUYPD: "DJOXL EYPD, ICJ X LBCMKXPV XPV CPO PYDBLK Y BXNO ZOOP JOACMPLYPD LC UCM LBO IXZROK CI FXKL XDOK XPV LBO RODOPVK CI XPAYOPL EYPDK. SXU Y SXEO KC ZCRV XK LC AJXNO X IXNCMJ CI UCMJ SXGOKLU?"

OFYRCDMO, LXROK IJCS LBO LBCMKXPV XPV CPO PYDBLK

Frequency Analysis (1)

- We identify the most common characters, digrams and trigrams in the ciphertext
- Example

  PCQ VMJYPD LBYK LYSO KBXBJXWXV BXV ZCJPO EYPD KBXBJYUXJ LBJOO KCPK. CP LBO LBCMKXPV XPV IYJKL PYDBL, QBOP KBO BXV OPVOV LBO LXRO CI SX'XJMI, KBO JCKO XPV EYKKOV LBO DJCMPV ZOICJO BYS, KXUYPD: "DJOXL EYPD, ICJ X LBCMKXPV XPV CPO PYDBLK Y BXNO ZOOP JOACMPLYPD LC UCM LBO IXZROK CI FXKL XDOK XPV LBO RODOPVK CI XPAYOPL EYPDK. SXU Y SXEO KC ZCRV XK LC AJXNO X IXNCMJ CI UCMJ SXGOKLU?"

  OFYRCDMO, LXROK IJCS LBO LBCMKXPV XPV CPO PYDBLK

- First guess:
  - LBO is THE

Frequency Analysis (2)

- Assuming LBO represents THE, we replace L with T, B with H, and O with E and get

PCQ VMJYPD THYK TYSE KHXHJXWXV HXV ZCJPE EYPD KHXHJYUXJ THJEE KCPK. CP THE THCMKXPV XPV IYJKT PYDHT, QHEP KHO HXV EPVEV THE LXRE CI SX'XJMI, KHE JCKE XPV EYKKEV THE DJCMPV ZEICJE HYS, KXUYPD: "DJEXT EYPD, ICJ X THCMKXPV XPV CPE PYDHTK Y HXNE ZEEP JEACMPTYPD TC UCM THE IXZREK CI FXKT XDEK XPV THE REDEPVK CI XPAYEPT EYPDK. SXU Y SXEE KC ZCRV XK TC AJXNE X IXNCMJ CI UCMJ SXGEKTU?"

EFYRCDME, TXREK IJCS THE THCMKXPV XPV CPE PYDBTK

Symmetric encryption schemes and secret keys

Decryption

- Code:

X	Z	A	V	O	I	D	B	Y	G	E	R	S	P	C	F	H	J	K	L	M	N	Q	T	U	W
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

- Ciphertext:

PCQ VMJYPD LBYK LYSO KBXBJXWXV BXV ZCJPO EYPD KBXBJYUXJ LBJOO KCPK. CP LBO LBCMKXPV XPV IYJKL PYDBL, QBOP KBO BXV OPVOV LBO LXRO CI SX'XJMI, KBO JCKO XPV EYKKOV LBO DJCMPV ZOICJO BYS, KXUYPD: "DJOXL EYPD, ICJ X LBCMKXPV XPV CPO PYDBLK Y BXNO ZOOP JOACMPLYPD LC UCM LBO IXZROK CI FXKL XDOK XPV LBO RODOPVK CI XPAYOPL EYPDK. SXU Y SXEO KC ZCRV XK LC AJXNO X IXNCMJ CI UCMJ SXGOKLU?"

OFYRCDMO, LXROK IJCS LBO LBCMKXPV XPV CPO PYDBLK

- Plaintext:

Now during this time Shahrazad had borne King Shahriyar three sons. On the thousand and first night, when she had ended the tale of Ma'aruf, she rose and kissed the ground before him, saying: "Great King, for a thousand and one nights I have been recounting to you the fables of past ages and the legends of ancient kings. May I make so bold as to crave a favour of your majesty?"
Epilogue, Tales from the Thousand and One Nights

Brute force

Symmetric encryption schemes and secret keys

Secret-Key Encryption

- A secret-key cipher uses a unique key K to encrypt and decrypt
- Caesar's generalized cipher uses the modular addition of each character (viewed as an integer) with the key:

$$C[\mathrm{i}] = P[\mathrm{i}] + K \bmod m$$

$$P[\mathrm{i}] = C[\mathrm{i}] - K \bmod m$$

- More secure secret-key encryption schemes have been devised in this century
- Examples:
  - DES
  - 3DES
  - IDEA
  - BLOWFISH
- With private-key encryption, a distinct secret key must be established for every pair of parties

The One-time pad

Secure symmetric ciphers do exist!

In fact, the most secure cipher known is the symmetric cipher that's referred to as the *one-time pad*.

For this encryption scheme Alice and Bob share a *random* bit string *K* that is as long as any message that they are going to send. This key *K* is the symmetric key that is used for the encryption and decryption process.

The length of the key should be greater than or equal to the message key. The key must be one-time and generated randomly each time.

The One-time pad (encryption)

To encrypt the message M, Alice computes $C = M \oplus K$, where the $\oplus$ symbol denotes the bitwise "exclusive-or" operation.

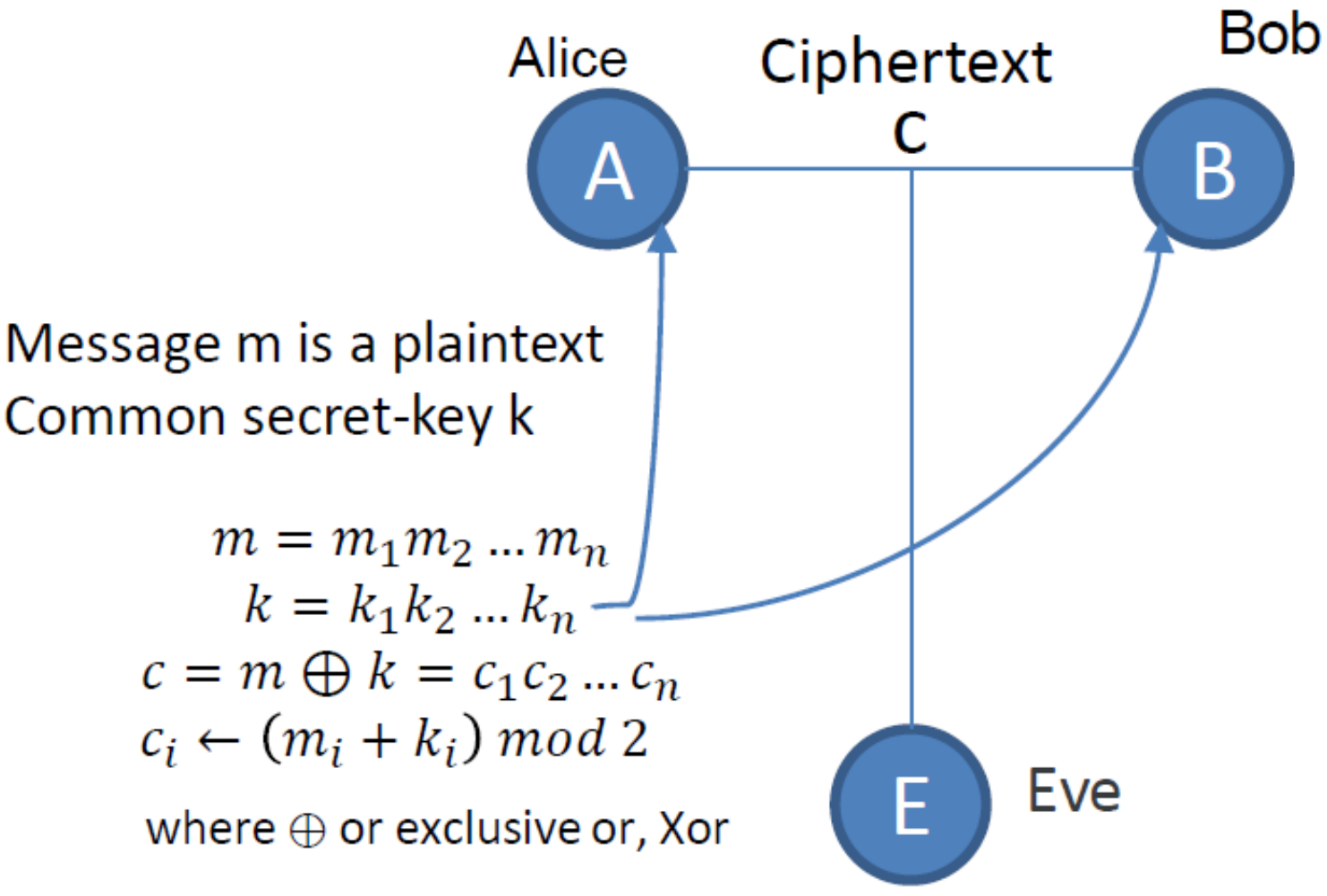
(Note: $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$.)

Encryption with the one-time pad is based on the exclusive-or (XOR) operation. The exclusive-or of two identical bits (two zeros or two ones) produces a zero and the exclusive-or of two different bits (a zero and a one) produces a one.

Alice then sends C to Bob on any reliable communications channel.

The communication is secure because C is computationally indistinguishable from a *random* bit string (This relies highly on the fact that K was selected randomly!).

The One-time pad (encryption)



Example

Encrypt the message ONETIME using a one-time pad.

```
Message: ONETIME
O: 01001111
N: 01001110
E: 01001010
T: 01010100
I: 01001001
M: 01001101
E: 01001010
Encoded: 01001111 01001110 01001010 01010100 01001001 01001101 01001010
```

One-time pad

```
11010111 11100101 10001111 00110000 10100010 00001010 01000000
```

Encryption

Letter	Binary	Letter	Binary
A	01000001	a	01100001
B	01000010	b	01100010
C	01000011	c	01100011
D	01000100	d	01100100
E	01000101	e	01100101
F	01000110	f	01100110
G	01000111	g	01100111
H	01001000	h	01101000
I	01001001	i	01101001
J	01001010	j	01101010
K	01001011	k	01101011
L	01001100	l	01101100
M	01001101	m	01101101
N	01001110	n	01101110
O	01001111	o	01101111
P	01010000	p	01110000

Table 1. Encoding letters in binary by ASCII.

```
Plaintext:    01001111 01001110 01000101 01010100 01001001 01001101 01000101
OneTimePad: ^ 11010111 11100101 10001111 00110000 10100010 00001010 01000000
Ciphertext: = 10011000 10101011 11001010 01100100 11101011 01000111 00000101
```

The One-time pad (decryption)

Bob can easily decrypt the ciphertext C to recover M in the following fashion: (Note: $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$.)

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus \mathbf{0} = M$$

where **0** represents the all-zero string with the same length as M.

This is clearly a symmetric scheme as Alice and Bob use the same key K for encryption and decryption.

The secret pad:
- Randomly generated
- Not-reused

Encrypt the message ONETIME using a one-time pad.

```
Message: ONETIME
O: 01001111
N: 01001110
E: 01001010
T: 01010100
I: 01001001
M: 01001101
E: 01001010
Encoded: 01001111 01001110 01001010 01010100 01001001 01001101 01001010
```

One-time pad

```
11010111 11100101 10001111 00110000 10100010 00001010 01000000
```

Encryption

Letter	Binary	Letter	Binary
A	01000001	a	01100001
B	01000010	b	01100010
C	01000011	c	01100011
D	01000100	d	01100100
E	01000101	e	01100101
F	01000110	f	01100110
G	01000111	g	01100111
H	01001000	h	01101000
I	01001001	i	01101001
J	01001010	j	01101010
K	01001011	k	01101011
L	01001100	l	01101100
M	01001101	m	01101101
N	01001110	n	01101110
O	01001111	o	01101111
P	01010000	p	01110000

Table 1. Encoding letters in binary by ASCII.

```
Plaintext:    01001111 01001110 01000101 01010100 01001001 01001101 01000101
OneTimePad: ^ 11010111 11100101 10001111 00110000 10100010 00001010 01000000
Ciphertext: = 10011000 10101011 11001010 01100100 11101011 01000111 00000101
```

Decryption

```
Ciphertext:   10011000 10101011 11001010 01100100 11101011 01000111 00000101
OneTimePad: ^ 11010111 11100101 10001111 00110000 10100010 00001010 01000000
Plaintext:  = 01001111 01001110 01000101 01010100 01001001 01001101 01000101
```

The One-time pad (analysis)

Advantages:

- Computationally efficient since the bitwise exclusive-or is easy to perform.
- Very secure (provided K is chosen randomly)

Disadvantages:

- Alice and Bob must share a very long key K.
- Security depends on the fact that the key is used *only once*

- In practice, we prefer secret keys that can be reused, and that the keys we use are much shorter than the messages that we must transmit.

How can we do this?

Public-key cryptography

A major problem with symmetric encryption schemes is *key distribution*, or how to *securely* distribute the secret keys.

Another idea is to dispense with using *symmetric* encryption schemes and seek another method for generating (and deciphering) the ciphertexts.

In 1976 Diffie and Hellman described an *abstract* system that overcomes the problem of key distribution – *Public-key cryptosystems*.

Public-key cryptosystems

A public-key cryptosystem consists of an encryption function E and a decryption function D. For any message M, the following properties must hold:

- $D(E(M)) = M$.
- Both E and D are easy to compute.
- It is *computationally infeasible* to derive D from E.
- $E(D(M)) = M$.

Public-key cryptosystems (cont.)

The third property is the particularly important one. It means that knowledge of the encryption method gives *no information* about the decryption scheme. Anybody can send a private message to the holder of the function *D*, but only that person knows how to decrypt it.

For this reason *E* is referred to as a *one-way* function.

In this kind of encryption method *E* is made *public* and *D* is kept *private*.

The RSA encryption scheme

Rivest, Shamir, and Adleman proposed a public-key encryption method that is probably the most well-known, and is still in use today for communications via web- browsers, etc.

Their method is tied to the difficulty of *factoring* large numbers. The first step is to select two large *prime* numbers p and q.

Let $n = p \cdot q$ and define $\varphi(n) = (p - 1)(q - 1)$.

We then choose two numbers e and d such that

1. e and $\varphi(n)$ are relatively prime, i.e. $\gcd(e, \varphi(n)) = 1$
2. $ed \equiv 1 \pmod{\varphi(n)}$ (by Extended Euclidean algorithm)

The pair of values e and n form the *public* key and d is the *private* key

Correctness of RSA

The plaintext M is encrypted using the public keys e and n by the following operation:

$$C \leftarrow M^e \bmod n.$$

Decryption is again handled by modular exponentiation:

$$M \leftarrow C^d \bmod n.$$

The correctness of the RSA method is guaranteed because it can be shown that with the choices of e, n, and d (with the properties listed earlier), then for every integer $0 < x < n$ we have

$$x^{ed} \equiv x \pmod{n}$$

RSA Cryptosystem

- **Setup:**
  - $n = pq$, with p and q primes
  - e relatively prime to $\phi(n) = (p-1)(q-1)$
  - d inverse of e in $Z_{\phi(n)}$
- **Keys:**
  - Public key: $K_E = (n, e)$
  - Private key: $K_D = d$
- **Encryption:**
  - Plaintext M in Z_n
  - $C = M^e \bmod n$
- **Decryption:**
  - $M = C^d \bmod n$

RSA Cryptosystem

- **Setup:**
  - $n = pq$, with p and q primes
  - e relatively prime to $\phi(n) = (p-1)(q-1)$
  - d inverse of e in $Z_{\phi(n)}$
- **Keys:**
  - Public key: $K_E = (n, e)$
  - Private key: $K_D = d$
- **Encryption:**
  - Plaintext M in Z_n
  - $C = M^e \bmod n$
- **Decryption:**
  - $M = C^d \bmod n$

- **Example**
  - Setup:
    - $p = 7,\ q = 17$
    - $n = 7 \cdot 17 = 119$
    - $\phi(n) = 6 \cdot 16 = 96$
    - $e = 5$
    - $d = 77$
  - Keys:
    - public key: $(119, 5)$
    - private key: 77
  - Encryption:
    - $M = 19$
    - $C = 19^5 \bmod 119 = 66$
  - Decryption:
    - $M = 66^{77} \bmod 119 = 19$

Complete RSA Example

- **Setup:**
  - $p = 5, q = 11$
  - $n = 5 \cdot 11 = 55$
  - $\phi(n) = 4 \cdot 10 = 40$
  - $e = 3$
  - $d = 27$ $(3 \cdot 27 = 81 = 2 \cdot 40 + 1)$
- **Encryption**
  - $C = M^3 \bmod 55$
- **Decryption**
  - $M = C^{27} \bmod 55$

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
C	1	8	27	9	15	51	13	17	14	10	11	23	52	49	20	26	18	2
M	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
C	39	25	21	33	12	19	5	31	48	7	24	50	36	43	22	34	30	16
M	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
C	53	37	29	35	6	3	32	44	45	41	38	42	4	40	46	28	47	54

Digital signatures

RSA cryptosystem supports *digital signatures*. Suppose that Bob sends a message M to Alice and that Alice wants to *verify* that it was Bob who sent it. Bob can create a *signature* using the decryption function applied to M:

$$S \leftarrow M^d \bmod n.$$

Alice verifies the digital signature using the encryption function, that is by checking that

$$M \equiv S^e \pmod{n}.$$

Since only Bob knows the decryption function, this will verify that it was indeed Bob who sent the message. (Of course, *any* person can use the encryption function as well to reconstruct the message M, so this is not a method to *secretly* pass information from Bob to Alice.)

The difficulty of breaking RSA

Note that even knowing e doesn't allow us to figure out d, unless we know $\varphi(n)$.

As the ability to factor larger numbers increases, we simply have to choose larger primes p and q so that $n = pq$ is outside of the current factoring capabilities.

In 1999, a 512-bit number was factored in 4 months using the following computers:

- 160 175-400 MHz SGI and Sun
- 8 250 MHz SGI Origin
- 120 300-450 MHz Pentium II
- 4 500 MHz Digital/Compaq

Estimated resources needed to factor a number within one year

Bits	PCs	Memory
430	1	128MB
760	215,000	4GB
1,020	342×10^6	170GB
1,620	1.6×10^{15}	120TB

Fast exponentiation

A possible bottleneck in the RSA algorithm is computing expressions of the form

$$x^k \mod n.$$

The "naive approach" is to calculate $x^2 \mod n$, then use that to get $x^3 \mod n$, then $x^4 \mod n$, etc. So the complexity should be in O(k).

Fast exponentiation

$3^{94} \pmod{17}$

Any number can be represented as the sum of distinct powers of two.

94=64+16+8+4+2

94 =	1	0	1	1	1	1	0
	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1*64	0*32	1*16	1*8	1*4	1*2	0*1

$$3^{94} \equiv 3^{64+16+8+4+2} \pmod{17}$$

$$\equiv 3^{64}3^{16}3^{8}3^{4}3^{2}$$

$$\equiv (1)(1)(-1)(-4)(9)$$

$$\equiv 36 \equiv 2$$

Fast exponentiation

A possible bottleneck in the RSA algorithm is computing expressions of the form

$$x^k \mod n.$$

We can do much better with an algorithm based on "repeated squaring." For example, if we wanted to compute x^{16}, we could first find x^2, then $(x^2)^2 = x^4$, then $(x^4)^2 = x^8$, and finally $(x^8)^2 = x^{16}$. This requires only four multiplications instead of fifteen with the "naive method." So the complexity should be in O(log k).