Chapter 4 Cryptography

4.0. Terminology.

The *plaintext* or original source message

encode encrypt encipher	$\downarrow$	$\uparrow$	decode decrypt decipher

The *ciphertext*, coded message or cryptogram

A *cipher system* consists of a general method of encipherment, which remains invariant but uses a variable *key*. It is wise to assume that the enemy knows the general method and is missing only the key.
[Known plaintext attack, chosen plaintext attack. Short term, long term (Venona 1942–45).]

4.1. Classical cryptography.

We shall assume that the source alphabet is a–z, with no punctuation. Spaces are either left unchanged or (better) are omitted.

A. Monoalphabetic or substitution ciphers. These use a simple permutation of the alphabet; the key is the permutation. Special cases are: [Stealable.]

A1. Caesar ciphers or additive ciphers. Identify the letters with the integers modulo 26 ($a = 0, \ldots, z = 25$). Encipher by adding κ (mod 26) to each letter, decipher by subtracting κ (mod 26). Julius Caesar used $\kappa = 3$, so the mapping is

plain	a	b	c	…	w	x	y	z
cipher	D	E	F	…	Z	A	B	C

The key is the value of κ, here 3 or D. There are only 26 possible keys, so one can easily try them all.

A2. Affine ciphers. Encipher by mapping letter λ to letter $\alpha\lambda + \beta$ (mod 26), decipher by mapping μ to $\alpha^{-1}(\mu - \beta)$.

$$[\ \mu = \alpha\lambda + \beta \iff \alpha\lambda = \mu - \beta \iff \lambda = \alpha^{-1}(\mu - \beta).\ ]$$

We need α to be coprime to 26, otherwise α^{-1} does not exist. The key is the pair (α, β). There are 12×26 keys. But it is easy to check guesses, because any two letters (not 13 letters apart) determine the cipher. For example, suppose we guess a $\to$ P, e $\to$ R, so that $\mathrm{P} = \alpha\mathrm{a} + \beta$, $\mathrm{R} = \alpha\mathrm{e} + \beta$, $\mathrm{R} - \mathrm{P} = \alpha(\mathrm{e} - \mathrm{a})$. Now, $\mathrm{R} - \mathrm{P} = 2$ and $\mathrm{e} - \mathrm{a} = 4$, so we want $4\alpha \equiv 2 \equiv 28 \equiv 54 \equiv \ldots \pmod{26}$. By trial and error we find $\alpha = 7$, and the mapping is

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

A3. Keyphrase ciphers. The key consists of a key phrase and a key letter — e.g., CAPTAIN COOK and F. Remove repeated letters from the key phrase — CAPTINOK — and form the code as follows:

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

So the message	w e	l e a v e	a t	d a w n
becomes	QZ	OZVMZ	VJ	YVQB

(Better if the key phrase has a large number of different letters.) There are now many more possible keys.

However, all monoalphabetic ciphers can be broken if one knows the frequencies of the different letters in the relevant language. In English the order is, roughly,

e | t a o i n s h r | d l | u c m f w y p b g | v k q j x z

The most common digrams (letter pairs) are

th he | in er an re ed | on es st en at to nt ha nd ou ea ti

The most common trigrams are

the | ing and | her ere ent tha nth was tio ion for nde has

With such information, one can often break a ciphertext 25–30 letters long if one knows it is a monoalphabetic cipher of English text.

[Yaqub Ibn Ishaq Al-Kindi, 9th c Baghdad; Italy 15–16th c; François Viète & Philip II of Spain, 2nd half 16th c. Beware misleading frequencies.]

ZAXIY OC IPC MAWVM ZA I DAA

QXQF VFLR TXLG VLWD PRUA

B. Randomization. All ciphers can be improved, and poor ciphers can be improved greatly, by making the ciphertext more random. Methods include:

(0) Use abbreviations, acronyms, codenames, unexpected languages, etc., wherever possible.

(1) Delete the word 'the' wherever it occurs.

(2) Replace each space by one of the letters v k q j x z chosen at random.

(3) Insert two random letters before every plaintext letter before encryption. Or let the number of random letters inserted be given by the decimal expansion of π.

(4) Do the same with the ciphertext after encryption.

(5) Encode the ciphertext using an e-error-correcting code, but make e random 'errors' in each codeword. This will make no difference to legitimate readers but it will confuse the enemy.

[Homophonic ciphers (must be written down).]

C. Polyalphabetic ciphers. (L. B. Alberti, c. 1460.) A polyalphabetic cipher of period p consists of p monoalphabetic ciphers used in rotation, usually by means of a keyword that is repeated indefinitely. In the *Vigenère* cipher (B. de Vigenère, 1586) the key letter is added (mod 26) to the plaintext in encryption and subtracted in decryption, as in the Caesar cipher. In the *Beaufort* cipher, both encryption and decryption involve subtraction (mod 26) from the key letter — but $a = 1, \ldots, y = 25, z = 0$!

$$\left[\begin{array}{c|cccccccccccccccccccccccccc}
 & a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\
V & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\
B & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26
\end{array}\right]$$

Example plaintext	t h e r e i s a t a v e r n i n t h e t o w n
($p = 4$) keyword	w i n e w i n e w i n e w i n e w i n e w i n
Vigenère ciphertext	$\underline{\text{P P R}}$ V A Q F E P I I I N V V R $\underline{\text{P P R}}$ X K E A
Beaufort ciphertext	$\underline{\text{C A I}}$ M R Z U D C H R Z E U E Q $\underline{\text{C A I}}$ K H L Z

Since a letter in the ciphertext can now correspond to any one of p plaintext letters, its frequency of occurrence is the average of the frequencies of these p letters, and so we cannot apply letter counts directly. However, every pth letter uses the same monoalphabetic cipher, and letter counts can help to break the code if we can find the period p. Here is one way of doing this.

Suppose we compare two passages of English text, or a passage against a shift of itself, character by character:

$$\begin{array}{r|l|l}
\text{t h e} & \text{r e i s a t a v e r n i n t h e } \boxed{\text{t}} \text{ o w n} & \\
 & \text{t h e r e i s a t a v e r n i n } \boxed{\text{t}} \text{ h e t} & \text{o w n}
\end{array} \qquad 1/20 = 5\%$$

The proportion of places where they have the same letter should be about $\sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)^2$, where $f(\lambda)$ is the frequency with which the letter λ occurs in English. Now,

$$f(\mathrm{a}) \approx 0.082, \ \ f(\mathrm{b}) \approx 0.015, \ \ \ldots, \ \ f(\mathrm{z}) \approx 0.0074, \ \text{ and}$$

$$\textstyle\sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)^2 \ \approx \ 0.082^2 + 0.015^2 + \ldots + 0.00074^2 \ \approx \ 0.0655.$$

So we expect agreement in about $6\frac{1}{2}\%$ of places. However, suppose we apply a permutation π of the alphabet to all letters in one of the two passages:

$$\begin{array}{r|l|l}
\text{t h e} & \text{r e i } \boxed{\text{s}} \text{ a t a v e r n i n t h e t o w n} & \\
 & \text{u i f } \boxed{\text{s}} \text{ f j t b u b w f s o j o u i f u} & \text{p x o}
\end{array} \qquad 1/20 = 5\%$$

Then we expect agreement in a proportion of places equal to

$\sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)f(\pi(\lambda))$. But

$$\begin{aligned}
&\sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)^2 - \sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)f(\pi(\lambda)) \\
&= \tfrac{1}{2}\left[\sum_{\lambda} f(\lambda)^2 + \sum_{\lambda} f(\pi(\lambda))^2\right] - \sum_{\lambda} f(\lambda)f(\pi(\lambda)) \\
&= \tfrac{1}{2}\sum_{\lambda} [f(\lambda) - f(\pi(\lambda))]^2 > 0.
\end{aligned}$$

So $\sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)f(\pi(\lambda)) \;<\; \sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)^2 \;\approx\; 6.55\%$.

Now let C_0 be a period-p polyalphabetic ciphertext obtained by using permutations $\pi_1, \ldots, \pi_p$ of the alphabet in rotation, and let C_k be C_0 shifted by k symbols. Suppose we write C_0 above C_k, and let $a(k)$ be the proportion of positions where they agree. If k is a multiple of p, then C_0 and C_k agree precisely where the corresponding plaintexts P_0 and P_k agree, so we expect $a(k) \approx 6\frac{1}{2}\%$. But if k is *not* a multiple of p then C_0 and C_k agree in position i iff the letters λ and μ occurring in P_0 and P_k in that position satisfy $\pi_{i+k}(\lambda) = \pi_i(\mu)$, i.e., $\mu = \pi_i^{-1}(\pi_{i+k}(\lambda))$ (subscripts taken modulo p). So we expect $a(k) \approx \frac{1}{p}\sum_{i=1}^{p}\sum_{\lambda=\mathrm{a}}^{\mathrm{z}} f(\lambda)f(\pi_i^{-1}\pi_{i+k}(\lambda)) < 6\frac{1}{2}\%$. [perhaps $\approx \frac{1}{26} \approx 3.8\%$?]

Example from Beker and Piper, for a ciphertext 223 letters long:

k	1	2	3	4	5	6	7	8	9	10	11
$a(k)$ (%)	4.05	1.36	<u>6.36</u>	4.57	5.96	<u>8.76</u>	2.31	4.19	5.14	3.76	4.24

k	12	13	14	15	16	17	18	19	20	...
$a(k)$ (%)	4.74	3.33	4.78	<u>6.25</u>	3.86	3.88	4.88	1.96	4.43	...

We deduce that probably $p = 3$. A similar technique can be used to decide whether or not two ciphertexts with the same period actually used the same cipher. [Alt. method Charles Babbage 1854, Friedrich Kasiski 1863.]

D. Nonperiodic substitution ciphers.

D1. The autokey cipher. Here the plaintext is used to continue the keyword in an additive cipher. E.g.,

keyword	p	u	n	t															
plaintext	q	u	e	e	n	t	o	a	r	r	i	v	e	a	t	n	o	o	n
autokey	p	u	n	t	q	u	e	e	n	t	o	a	r	r	i	v	e	a	t
ciphertext	F	O	R	X	D	N	S	E	E	K	W	V	V	R	B	I	S	O	G

If we can guess that the keyword has 4 letters, we can break this by turning it into a Vigenère ciphertext with period 8 and keyword 'puntquee':

plaintext	q u e e n t o a	r r i v e a t n	o o n
keyword	p u n t q u e e	p u n t q u e e	p u n
Vigenère ciphertext	F O R X D N S E	G U	D

To see how, note that (looking at every fourth position)

plain	+	key	=	cipher	
q	+	p	=	F	
n	+	q	=	D	
r	+	n	=	E	so $\mathrm{r} + \mathrm{p} = \mathrm{E} - \mathrm{D} + \mathrm{F} = \mathrm{G}$
e	+	r	=	V	so $\mathrm{e} + \mathrm{q} = \mathrm{V} - \mathrm{E} + \mathrm{D} = \mathrm{U}$
o	+	e	=	S	so $\mathrm{o} + \mathrm{p} = \mathrm{S} - \mathrm{V} + \mathrm{E} - \mathrm{D} + \mathrm{F} = \mathrm{S} - \mathrm{V} + \mathrm{G} = \mathrm{D}$

Of course, we will not know that the keyword has 4 letters, so we must try possible keyword lengths $l = 1, 2, 3, \ldots$ in turn, each time testing the modified ciphertext by shifting it against itself by multiples of $2l$, until we find an l where the average agreement is around $6\frac{1}{2}\%$, at which point it is very likely that we have a Vigenère ciphertext with period $2l$ which we can now try to break using letter frequencies.

D2. Pseudorandom sequences. (Not for examination.) The sequence of substitutions is generated by a pseudorandom process that forms the key and is known to both sender and receiver. This could be by addition, Vigenère-style, using a 'keyword' as long as the plaintext (e.g. a page of a book). Some military codes (such as the German Enigma), used up to about 1950, were nonadditive examples of this type, generated by multi-rotor machines.

D3. The one-time pad. (Not for examination.) This is an additive cipher in which the key sequence is genuinely random (so both sender and receiver must have copies of it in advance). It is genuinely unbreakable.

E. Block substitution ciphers. (Not for examination.) The plaintext is divided into blocks of length d, each of which is encrypted separately.

E1. Permutation ciphers. E.g., if $d = 5$ and the permutation is $\left(\begin{smallmatrix}1&2&3&4&5\\4&1&3&5&2\end{smallmatrix}\right)$:

plaintext	s e n d m	e r e i n	f o r c e	m e n t s
ciphertext	E M N S D	R N E E I	O E R F C	E S N M T

E2. Linear transformations. Here the number of symbols in the code alphabet must be a prime number p. The key is a nonsingular $d \times d$ matrix A with entries mod p. Each block of d plaintext letters is multiplied by A (mod p) to encrypt it.

E3. The Playfair cipher. (L. Playfair and C. Wheatstone, 1854.) Here $d = 2$. Spaces are omitted and J is represented by I. The remaining 25 letters are written in a 5×5 grid formed by using a key phrase as in a keyphrase cipher:

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

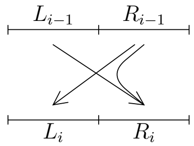
The pairs of letters are encrypted as follows:

plaintext	s e	n d	m e	r e	i n	f o	r c	e m	e n	t s
ciphertext	F B	M E	N D	G U	F H	S G	C O	D N	N Z	S A

E4. Feistel ciphers. (H. Feistel, c. 1970) These encipher *binary* plaintext by a form of *cipher block chaining*, where a relatively simple algorithm is repeated for several *rounds*, with the output from each round forming the input for the next round. In Feistel ciphers, a $2t$-bit block of plaintext is split into two t-bit blocks L_0, R_0. In the ith round, define

$$L_i := R_{i-1}, \quad R_i := L_{i-1} + f(R_{i-1}, K_i) \quad \text{(bitwise + mod 2)}$$

where K_i is a *subkey* and f is some function. This is reversed in decryption by



$$R_{i-1} := L_i, \quad L_{i-1} := R_i + f(L_i, K_i).$$

The left and right blocks are interchanged after the final round, so that decryption uses the same algorithm as encryption but with the subkeys in the reverse order. The best-known example is the *Data Encryption Standard* (*DES*), proposed in 1975 and adopted in 1976. This has 16 rounds, a block length of 64 bits and a key length of 56 bits. The subkeys have 48 bits, extracted from permutations of the key. The function f involves

(1) expansion of R_{i-1} to 48 bits: 32 1 2 3 4 5 4 5 6 7 8 9 8 9 … 30 31 32 1;
(2) bitwise addition of K_i;
(3) reduction to 32 bits by eight 'S-boxes' (lookup tables), each reducing 6 bits to 4;
(4) a fixed permutation of the bits.

Increased computer power has now made the 56-bit key length insecure. In July 1998, a purpose-built computer broke a DES-encrypted message in 56 hours.

[DES used by banks for electronic transfer of funds, and to protect civilian satellite communications. 'DES Cracker' built by Electronic Frontier Foundation for $<$ \$250 000 in $<$ 1 year. DES's successor AES (Advanced Encryption Standard) announced November 2000: Rijndael, based on properties of Galois field GF(8). Competes with triple DES.]

4.2. Interlude: some number theory.

Theorem 4.1. Let $p > 2$ be a prime number and let s be a number not divisible by p.

(a) There is a number x such that $xs \equiv 1 \pmod p$.

(b) (Fermat's theorem.) $s^{p-1} \equiv 1 \pmod p$.

(c) (Euler's criterion, easy part.) If there is a number m such that $s \equiv m^2 \pmod p$, then $s^{(p-1)/2} \equiv 1 \pmod p$.

Proof. (Not for examination.) If $is \equiv js \pmod p$ then $(i-j)s \equiv 0 \pmod p$ and so $i-j$ is divisible by p (since s isn't). Therefore the numbers

$$s \quad 2s \quad 3s \quad \ldots \quad (p-1)s$$

are distinct and nonzero (mod p), and hence are congruent (mod p) to the numbers

$$1 \quad 2 \quad 3 \quad \ldots \quad (p-1)$$

in some order. Thus one of them, say xs, $\equiv 1 \pmod p$. This proves (a). To prove (b), note that

$$(s)(2s)(3s)\ldots((p-1)s) \equiv (1)(2)(3)\ldots(p-1) \pmod p,$$

i.e., if $t := (p-1)!$ then

$$ts^{p-1} \equiv t \pmod p.$$

Since t is not divisible by p, $\exists\, y$ s.t. $yt \equiv 1 \pmod p$ by (a), and then

$$s^{p-1} \equiv yts^{p-1} \equiv yt \equiv 1 \pmod p.$$

This proves (b). Now (c) follows, because m is clearly not divisible by p and so $s^{(p-1)/2} \equiv (m^2)^{(p-1)/2} \equiv m^{p-1} \equiv 1 \pmod p$ by (b). //

The **Euclidean Algorithm** is a procedure whereby, given numbers r_0 and r_1 ($r_0 > r_1$), one can find numbers a and b such that $ar_0 + br_1 = \operatorname{hcf}(r_0, r_1)$ ($= 1$, in our case). One can use it to find the x in Theorem 4.1(a). It involves constructing a sequence $r_0 > r_1 > r_2 > \ldots = 1$ by using quotient and remainder:

$$\begin{aligned} r_0 &= q_1 r_1 + r_2, \\ r_1 &= q_2 r_2 + r_3, \end{aligned}$$

etc.

Example 1. To find x s.t. $17x \equiv 1 \pmod{47}$.
Take $r_0 = 47$, $r_1 = 17$, $r_2 = 13$, $r_3 = 4$, $r_4 = 1$.

$$\begin{aligned} 47 &= 2 \times 17 + 13, \\ 17 &= 1 \times 13 + \ \ 4, \\ 13 &= 3 \times \ \ 4 + \ \ 1. \end{aligned} \qquad \begin{aligned} 1 &= 13 - 3 \times 4 \\ &= 13 - 3(17-13) = 4 \times 13 - 3 \times 17 \\ &= 4(47 - 2 \times 17) - 3 \times 17 \\ &= 4 \times 47 - 11 \times 17. \end{aligned}$$

So $-11 \times 17 \equiv 1 \pmod{47}$,
and $x \equiv -11 \equiv 36 \pmod{47}$—i.e., $17 \times 36 \equiv 1 \pmod{47}$.

Example 2. To find a and b such that

$$a \equiv \begin{cases} 1 \pmod{17} \\ 0 \pmod{19} \end{cases} \quad \text{and} \quad b \equiv \begin{cases} 0 \pmod{17} \\ 1 \pmod{19}. \end{cases}$$

[If $1 = 17x + 19y$, we can take $a = 19y$ and $b = 17x$.]

Take $r_0 = 19$, $r_1 = 17$, $r_2 = 2$, $r_3 = 1$.

$$\begin{aligned} 19 &= 1 \times 17 + 2, \\ 17 &= 8 \times \ \ 2 + 1, \end{aligned} \qquad \begin{aligned} 1 &= 17 - 8 \times 2 \\ &= 17 - 8(19 - 17) \\ &= 9 \times 17 - 8 \times 19. \end{aligned}$$

So we could choose $a = -8 \times 19$ and $b = 9 \times 17$, giving $a + b = 1$. If we want $a, b > 0$, choose $a = (17-8) \times 19 = 9 \times 19 = 171$ and $b = 9 \times 17 = 153$, so that $a + b = 1 + 17 \times 19$. Then

$$\begin{aligned} a &= 1 + 17 \times 19 - 9 \times 17 \equiv 1 \pmod{17}, \\ b &= 1 + 17 \times 19 - 9 \times 19 \equiv 1 \pmod{19}. \end{aligned}$$

4.3. Three public-key cryptosystems. A *one-way function* is a function that is feasible to compute but 'impossible' to invert.

Example 1. Let Π_k be the set of prime numbers with at least k decimal digits. Define $f : \Pi_k \times \Pi_k \to \mathbb{Z}$ by $f(p,q) := pq$. Then f

is a one-way function provided that (around 2000) $k \gtrsim 150$ (and p and q are not too close together).

Example 2. Let $p \in \Pi_k$ and let a be a *primitive root* mod p: that is, $\{a, a^2, a^3, \ldots, a^{p-1}\} = \{1, 2, 3, \ldots, p-1\} \pmod p$. Define $f : \mathbb{Z}_p \to \mathbb{Z}_p$ by $f(x) := a^x$. This is a one-way function provided that (around 2000) $k \gtrsim 150$ (and x is not too small).

To invert these two functions involves *factoring* and *taking discrete logarithms*, respectively, both of which are computationally hard.

A *trapdoor function* is a one-way function that becomes easy to invert if one knows an extra piece of information, the *trapdoor*.

A. The RSA system. (R. L. Rivest, A. Shamir and L. Adleman, 1978.) [MIT; Clifford Cocks at GCHQ; RSA Data Securities Inc.; internet security.]

Preparation. Each participant

(a) chooses two large ($\gtrsim$ 150-digit) primes p and q,
(b) computes $n = pq$,
(c) chooses d large and relatively prime to $(p-1)(q-1)$ (e.g., a prime $> \max(p, q)$),
(d) computes (by the Euclidean algorithm) the unique $e \in \{1, 2, \ldots, (p-1)(q-1)\}$ s.t. $de \equiv 1 \pmod{(p-1)(q-1)}$,
(e) publishes n and e, which form the *public key*, while keeping secret p, q and d, the *trapdoor* or *private key*.

Encryption. Alice, to send a secret message to Bob, uses Bob's public key $(n_{\mathrm{B}}, e_{\mathrm{B}})$; breaks the message into blocks, each of which can be represented as an integer $M \in \{0, 1, \ldots, n_{\mathrm{B}} - 1\}$; and encrypts M into $C \equiv M^{e_{\mathrm{B}}} \pmod{n_{\mathrm{B}}}$.

Decryption. Bob, receiving C, computes $D \in \{0, 1, \ldots, n_{\mathrm{B}} - 1\}$ by $D \equiv C^{d_{\mathrm{B}}} \pmod{n_{\mathrm{B}}}$, using his private key d_{B}.

Theorem 4.2. $D \equiv M \pmod{n_{\mathrm{B}}}$; hence $D = M$.

Proof. Write $n_{\mathrm{B}} = n = pq$, etc. Then $D \equiv C^d \equiv M^{de} \pmod n$. But $de = t(p-1)(q-1) + 1$ for some positive t, so $D \equiv M^{t(p-1)(q-1)+1}$. Since p is prime, by Fermat's theorem

$$M^{p-1} \equiv \begin{cases} 1 & \text{if } M \not\equiv 0 \pmod p, \\ 0 & \text{if } M \equiv 0 \pmod p, \end{cases}$$

so

$$D \equiv M^{t(p-1)(q-1)+1} \equiv \left\{ \begin{matrix} 1^{t(q-1)}M \\ 0 \end{matrix} \right\} \equiv M \pmod p.$$

Similarly $D \equiv M \pmod q$. Thus $D - M$ is divisible both by p and by q, hence by $pq = n$, whence $D \equiv M \pmod n$. //

At present, nobody knows of any way of breaking this code without factoring n, which is computationally intractable.

B. Rabin's public-key system. (M. O. Rabin, 1979.)

Preparation. Each participant

(a) chooses two large ($\gtrsim$ 150-digit) primes $p = 4k - 1$, $q = 4l - 1$,
(b) computes $n = pq$,
(c) computes (by the Euclidean algorithm) the unique integers a and b in $\{1, 2, \ldots, n\}$ such that

$$a \equiv \begin{cases} 1 \pmod p \\ 0 \pmod q \end{cases} \quad \text{and} \quad b \equiv \begin{cases} 0 \pmod p \\ 1 \pmod q, \end{cases}$$

(d) chooses an integer $r < n$,
(e) publishes n and r, which form the *public key*; p, q, k, l, a and b form the *trapdoor* or *private key*.

Encryption. Alice, to send a secret message to Bob, uses Bob's public key (n, r); represents the message as integers $M \in \{0, 1, \ldots, n-1\}$; and encrypts M into $C \equiv M(M + r) \pmod n$.

Decryption. Bob, receiving C, uses his private key to compute $y \in \{0,\ldots,p-1\}$, $z \in \{0,\ldots,q-1\}$ and $D \in \{0,\ldots,n-1\}$ such that $y \equiv \pm(C+\frac{1}{4}r^2)^k - \frac{1}{2}r \pmod p$, $z \equiv \pm(C+\frac{1}{4}r^2)^l - \frac{1}{2}r \pmod q$ and $D \equiv ay+bz$ (mod $n = pq$). There are four possible values for D; presumably only one of them will make sense.

Theorem 4.3. One of the four possible values of D is equal to M.

Proof. Define $y' \in \{0,\ldots,p-1\}$ and $z' \in \{0,\ldots,q-1\}$ s.t. $y' \equiv M \pmod p$ and $z' \equiv M \pmod q$. Since $a \equiv 1$ and $b \equiv 0 \pmod p$, and *vice versa* (mod q), $M \equiv ay'+bz'$ (mod p and mod q, hence mod $n = pq$). Let $s = C + \frac{1}{4}r^2$. Then, working mod p,

$$(y'+\tfrac{1}{2}r)^2 \equiv (M+\tfrac{1}{2}r)^2 = M(M+r)+\tfrac{1}{4}r^2 \equiv C+\tfrac{1}{4}r^2 = s \pmod p.$$

Let $m = M + \frac{1}{2}r$ and recall $k = \frac{1}{4}(p+1)$. Since $s \equiv m^2 \pmod p$, Theorem 4.1(c) implies $s^{(p-1)/2} \equiv 1 \pmod p$ if $p \nmid s$, and so, by the definition of y,

$$\begin{aligned}(y+\tfrac{1}{2}r)^2 \equiv (C+\tfrac{1}{4}r^2)^{2k} = s^{2k} = s^{(p+1)/2} = s^{(p-1)/2}s \\ \equiv s \equiv (y'+\tfrac{1}{2}r)^2 \pmod p.\end{aligned}$$

(If $p \mid s$ then clearly $y' + \frac{1}{2}r \equiv y + \frac{1}{2}r \equiv 0 \pmod p$.) But, mod p, if $x_1^2 \equiv x_2^2$ then $x_1^2 - x_2^2 = (x_1+x_2)(x_1-x_2) \equiv 0$ and so $x_1 \equiv \pm x_2 \pmod p$. [This is not true mod pq: e.g., $2^2 \equiv 9^2 \equiv 4 \pmod{77}$, but $2 \not\equiv \pm 9$.] Hence $y + \frac{1}{2}r \equiv \pm(y' + \frac{1}{2}r) \pmod p$; that is, y' is one of the two possible values for y. Likewise, z' is one of the two possible values for z. Since $D \equiv ay+bz$ and $M \equiv ay'+bz' \pmod n$, it follows that M is one of the four possible values for D. //

Rabin proved that breaking this code is as hard as factoring.

[Quantum computing.]

C. ElGamal's public-key system. (T. ElGamal, 1985.)

Preparation. All participants agree on a large ($\gtrsim$ 150-digit) prime p and a primitive root $a \pmod p$. Each participant chooses a *private key* $d \in \{1,2,\ldots,p-1\}$ and publishes the *public key* $e \in \{1,2,\ldots,p-1\}$ s.t. $e \equiv a^d \pmod p$.

Encryption. Alice, to send to Bob, uses his public key e; represents the message as integers $M \in \{0,1,\ldots,p-1\}$; chooses at random a k in $\{1,2,\ldots,p-1\}$; and encrypts M into the pair $C_1 \equiv a^k$, $C_2 \equiv Me^k \pmod p$.

Decryption. Note that $M \equiv C_2/e^k \equiv C_2/a^{dk} \equiv C_2/C_1^d \pmod p$. So Bob, receiving (C_1, C_2), uses his private key d to compute $M \equiv C_2/C_1^d \pmod p$.

At present, nobody knows how to break this system without computing discrete logarithms.

4.4. Five ingenious ideas.

A. The Diffie–Hellman key-distribution system. (W. Diffie and M. E. Hellman [and R. Merkle?], 1976.) This is a method whereby two people can agree on a common key without an interceptor being able to deduce it (assuming that an arbitrary number is acceptable as the key). As in ElGamal's system (1985), all participants agree on a large prime p and a primitive root $a \pmod p$, and each chooses a *private key* $d \in \{1,2,\ldots,p-1\}$ and publishes the *public key* $e \equiv a^d \pmod p$. Alice and Bob then communicate using the key $K \equiv a^{d_\mathrm{A} d_\mathrm{B}} \pmod p$. Alice calculates $K \equiv e_\mathrm{B}^{d_\mathrm{A}}$ and Bob calculates $K \equiv e_\mathrm{A}^{d_\mathrm{B}} \pmod p$. No other user can compute K without finding a discrete logarithm. [Implemented by Hewlett Packard and the Mitre Corporation.]

B. A cryptosystem with no shared keys. (A. Shamir, c. 1980.) Alice and Bob agree on a large prime p. Alice chooses integers $a, a' \in \{1,2,\ldots,p-1\}$ s.t. $aa' \equiv 1 \pmod{p-1}$. Bob chooses b, b' similarly. Alice, to send a message $M \in \{1,2,\ldots,p-1\}$ to Bob, first sends an integer $C \equiv M^a \pmod p$. Bob returns to Alice $D \equiv C^b \pmod p$. Alice responds with $E \equiv D^{a'} \pmod p$. Finally, Bob deciphers the message as $F \equiv E^{b'} \pmod p$, $F \in \{1, 2, \ldots, p-1\}$.

Theorem 4.4. $F = M$.

Proof. $F \equiv E^{b'} \equiv D^{a'b'} \equiv C^{ba'b'} \equiv M^{aba'b'} \pmod p$. But $aba'b' \equiv 1 \pmod{p-1}$, and $M^{p-1} \equiv 1 \pmod p$ by Fermat's theorem, so $F \equiv M^{aba'b'} = M^{t(p-1)+1} \equiv M \pmod p$, and so $F = M$. //

C. The password problem. Access to a computer is by means of a password, but the password file cannot be kept secure. The solution is to store *encrypted* passwords. Any one-way function will do here—no trapdoor is needed. [No longer secure.]

D. Authentication. Alice receives a message, ostensibly from Bob. How can she confirm that he really sent it? Using any public-key cryptosystem, she can encrypt some arbitrary item using Bob's public key and ask him to send back the decrypted version. Only Bob can do this, using his private key.

E. 'Signed messages'. To create these, we can use any public-key cryptosystem in which every message is the encryption of some other (nonsense) message. Then Bob, to send a signed message to Alice, treats it as ciphertext and '*decrypts*' it using his private key, to obtain the *signature* S. Alice, receiving S, treats it as plaintext and '*encrypts*' it using Bob's public key. Now neither Alice nor Bob can claim that a different message was sent, since Alice can produce the signature S for the real message M but not for any other message.

E.g., with RSA, suppose Alice and Bob have public keys $(n_{\mathrm{A}}, e_{\mathrm{A}})$ and $(n_{\mathrm{B}}, e_{\mathrm{B}})$ and private keys d_{A} and d_{B}. Bob, to send a message $M \in \{1, 2, \ldots, n_{\mathrm{B}}\}$ to Alice, sends $S \equiv M^{d_{\mathrm{B}}} \pmod{n_{\mathrm{B}}}$. Alice recovers M by $M \equiv S^{e_{\mathrm{B}}} \pmod{n_{\mathrm{B}}}$. If secrecy is required

as well as signature, then Bob, having calculated S, sends Alice $C \equiv S^{e_{\mathrm{A}}} \pmod{n_{\mathrm{A}}}$ and Alice recovers S by $S \equiv C^{d_{\mathrm{A}}} \pmod{n_{\mathrm{A}}}$ before calculating M. (If $S > n_{\mathrm{A}}$, split S into blocks $< n_{\mathrm{A}}$. Alternatively, everybody has *two* public-key pairs, the smaller modulus ($< 10^{300}$, say) for signature-verification and the larger ($> 10^{300}$) for encryption.)