MATHEMATICAL PROBLEMS IN CRYPTOLOGY

N. P. Varnovsky, A. I. Verchenko, and E. A. Primenko UDC 971.03.35

Introduction

By private (secret) information we mean information which, by mutual agreement between its sender and receiver, is not intended for a third party. Apparently there is no need to argue the need to protect private information and to explain the motivation of the third party to make considerable efforts to obtain it.

In referring to the above-mentioned "third party" we use the term "adversary," borrowed from military applications.

Cryptology is concerned with studying and developing mathematical methods of protecting private information transmitted by the sender to the receiver over public communication channels.

Cryptology consists of two parts: *cryptosynthesis* (or theoretical cryptography) and *cryptoanalysis.*

Cryptography develops information-protecting techniques. Private information is contained in the sender-composed *plaintext*, which is a sequence of letters of a finite alphabet. An intended transformation of plaintext, complicating the extraction of private information contained in it, is called *encryption.*

The procedure used for this purpose is called the *cipher* or *cryptographic system.* The latter includes a certain element, known only to the sender and (or) receiver, which is called the *private* (secret) *key.* The result of encryption is a *ciphertext* or *cryptogram.* It is evident that encryption should be reversible, i.e., allow one to uniquely recover plaintext from the cryptogram (decryption).

Cryptoanalysis, in contrast to cryptography, studies the possibilities of cracking the cryptosystem, i.e., extracting plaintext from a cryptogram by an adversary who does not know the private key.

Cryptography has been known to exist since ancient times. As Suetonius noted, Julius Caesar used letter substitution in his correspondence: he substituted D for A, E for B, F for C, and so on. This procedure is a particular case of a *simple substitution*, which consists in substituting each plaintext letter by a certain equivalent—a letter of the same or another alphabet; moreover, different letters are substituted by different equivalents. Since the frequencies of letters and of their combinations in plaintext coincide with the frequencies of the corresponding equivalents in a cryptogram, cracking such a cryptosystem is an easy task, provided the cryptogram is of sufficient length.

In terms of computer science a simple substitution is an alphabetic coding [163].

Principally the other type of cryptosystem is the *block* one, in which plaintext is divided into sequential fixed-length blocks and then the same encrypting algorithm is applied to each block. An example of a block cryptosystem is a *permutation cipher* in which the same r-permutation is applied to each block of length r. A method of extracting the plaintext with an unknown key (permutation) is described in detail in [6].

Polygram substitution is also a block cryptosystem. It is analogous to a simple substitution, but differs in that the substitution unit is not a separate letter, but a bigram, trigram, and so on. A witty version of a bigram substitution cipher is the Playfair cipher [131]. The secret key is a square composed of 25 letters of the Latin alphabet: the cipher was developed in an English-speaking country, in whose cryptographic practice the letter J was omitted due to its nonfrequent occurrence. Plaintext is divided into bigrams. In order to suppress bigrams, consisting of two occurrences of the same letter, some nonfrequent letter is put between them, if necessary. If both letters of a bigram are located in the same row (in the same column) then, when encrypting, each of them is substituted by the next letter to the right (next below), and, moreover, each row and each column are treated in a cyclic manner. If the bigram letters are in different rows and columns, then they are located in opposite corners of a certain rectangle and are substituted by the letters located in two other opposite corners, and, moreover, each of them is substituted by the

Translated from Itogi Nauki i Tekhniki, Seriya Sovremennaya Matematika i ee Prilozheniya. Tematicheskiye Obzory. Vol. 2, Teoriya Veroyatnostey i Matematicheskaya Statistika – 1, 1993.

letter located in the same row. Recovering the plaintext is not a difficult task, as in the case of a simple substitution. Cracking this cryptosystem is described in detail in [131].

We may consider the letters of an alphabet as distinct elements of the ring Z_n, where n is the cardinality of the alphabet. The encryption procedure may be defined as an elementwise sum of plaintext and a certain sequence of Z_n elements—a secret key. If this sequence is periodic, then this cipher is called a *Vigenere cipher*. A Vigenere cipher with period equal to 1 is a Caesar cipher (credited to August Caesar, not to be confused with the above-mentioned Julius Caesar). If a private key is a nonperiodic sequence, then the Vigenere cipher is a *Vernam cipher* [159]. An example of cracking a Vigenere cipher is considered in [96]. The Vernam cipher is recalled in §1.

Hill [76, 77] proposed a cryptosystem in which the letters are also considered to be the elements of the ring Z_n and the plaintext is divided into fixed-length blocks. Encryption consists in multiplying each plaintext block by the $r \times r$ nonsingular matrix, which is a private key.

Though the cited examples seem to be rather simple, their underlying ideas are so fruitful that even now they are still used to design cryptosystems.

The present paper is aimed at attracting the attention of mathematicians to the problems of cryptology.

The authors have to note that since a number of references are unavailable to them, some results are cited based on abstracts and references from other papers.

§1. Background Remarks

Two periods can be conventionally outlined in the development of cryptology. The rare publications of the first period allow one to conclude that the cryptology of this period was an art rather than a science [7, 52, 56, 94, 95, 153].

The end of this period fell approximately in the early 1940s when the problems of cryptology attracted the attention of mathematicians and cyberneticians. Efforts were made to develop a formal theory of cryptographic systems. One of the first works stating the foundations of this theory was Shannon's paper [145].

A cryptosystem is defined abstractly as a certain set of mappings from the set of plaintexts to the set of possible cryptograms. Each particular mapping in this set corresponds to encryption with a particular key.

These mappings should be one-to-one; thus, if the key is known, the result of decryption is unique. Each key (and, hence, each mapping) has a certain corresponding a priori probability, with which this key is chosen. Analogously, to each possible plaintext there corresponds an a priori probability defined by the distribution of plaintexts. The probabilities of different keys and plaintexts are a priori probabilities for the adversary and characterize his a priori knowledge of the cryptosystem.

If x is a plaintext, k is the key, and y is a cryptogram, then

$$y = f(x, k).$$

It is more suitable to consider y as a (one-parameter) family of mappings

$$y = T_i x,$$

rather than a two-variable function.

The mapping T_i, applied to plaintext x, produces cryptogram y. The index i corresponds to a particular key.

In what follows it is assumed that the adversary knows the family T_i and the probabilities with which the keys are chosen.

This assumption is common in cryptography, since one can suppose that the adversary will succeed, sooner or later, in learning any cryptosystem, at least up to the unknown key.

Shannon defines a *perfect secrecy* as follows: for all y, a posteriori probabilities $P(x|y)$ are equal to a priori probabilities $P(x)$, independently of their values.

Theorem [145]. *For a perfect secrecy it is necessary and sufficient that*

$$P(y|x) = P(y)$$

for all x and y.

In other words, the total probability of all keys mapping plaintext x to a certain cryptogram y is equal to the total probability of all keys mapping any other plaintext to the same cryptogram.

In [145] it is also proved that in a perfectly secure cryptosystem a key must be at least as long as the plaintext. An example of such a system is the above-mentioned Vernam cipher.

One can distinguish two main types of cryptosystems, which are conventionally called ***private-key cryptosystems*** and ***public-key cryptosystems.*** All of the above examples of cryptosystems are of the first type. Their main feature is that knowlege of the ***encryption key*** allows the adversary to compute the ***decryption key*** in a rather simple way. Therefore, both encryption and decryption keys should be kept private.

In the case of public-key cryptosystems, the computation of the decryption key, when the encryption key is known, is a hard problem from the complexity-theoretic point of view. The encryption key in these systems is published in a public directory, while the decryption key is known only to the receiver of information.

It is necessary to note that cryptology studies not only cryptosystems, but also ***cryptographic protocols***, including ***authentication schemes***, ***electronic signatures***, etc. (see §4).

The authentication of information consists in establishing by the receiver of the fact that information was sent by a certain sender and that it was not substituted or corrupted by the adversary [148].

The information that can be presented by the receiver to prove this to the sender or to the third party (judge) is called an electronic signature.

The cryptological dependability measure is called ***security.*** Cryptosystem security depends on the assumptions concerning information and resources that are at the adversary's disposal. As was pointed out above, the assumption that the adversary knows not only the cryptogram but also a cryptosystem up to the key is common in cryptology. Moreover, in public-key cryptosystems the encryption key is also publicly available.

Cracking the cryptosystem is usually considered to be a procedure of recovering the plaintext or part of it from the cryptogram (not necessarily by determining the private key). It is also assumed that plaintext is obtained solely by analyzing the cryptosystem weaknesses.

An attempt by the adversary to crack the cryptosystem is called an ***attack*** on the cryptosystem (another term borrowed from military applications). We distinguish the following types of attack.

Chosen-plaintext attack. The adversary is assumed to have the opportunity to choose the necessary number of plaintexts and obtain their cryptograms. The choice of the next plaintext is based on all previously seen cryptograms.

Chosen-ciphertext attack. The adversary has the opportunity to choose the necessary number of cryptograms and get the corresponding plaintexts. The choice of the next cryptogram is based on the results of the analysis of all previously obtained plaintexts.

Chosen-text attack. The adversary has the opportunity to choose both cryptograms (and to decrypt them) and plaintexts (and to encrypt them) based on the results of the analysis of all previously obtained plaintexts and cryptograms.

The cryptosystem security with respect to a particular type of attack is proportional to the time required for the adversary to crack the cryptosystem with an attack of this type.

Usually a cryptosystem is considered to be secure if there is no polynomial-time cracking algorithm.

In addition to the attacks on cryptosystem an attack on the cryptographic protocol is also possible. If the number of possible cryptograms is relatively small, then the adversary, having guessed the plaintext and verified his guess, can obtain the plaintext with a relatively high probability. This type of attack is called a ***verifiable-text attack.*** The adversary can succeed in this type of attack without cracking the underlying cryptosystem.

§2. The Data Encryption Standard (DES)

The DES cryptosystem was designed by IBM and issued in 1977 [34] as the USA federal standard.

DES is a block private-key cryptosystem [37, 114]. The length of a plaintext block is equal to 64 bits; the keylength is 56 bits. The key K is padded with 8 check bits to get a tuple $\tilde{K} = (k_1, k_2, \ldots, k_{64})$, where the $8j$th bit is a check bit and satisfies the following condition (parity check):

$$k_{8j} = \oplus \sum_{i=1}^{7} k_{8(j-1)+i}, \; j = 1, 2, \ldots, 8.$$

The total number of different keys in a DES cryptosystem is 2^{56}. This makes the guessing of a key at random practically infeasible.

Let us describe the encryption algorithm of the DES cryptosystem.

Each plaintext block x is subjected to some permutation $\mathrm{IP} \in \boldsymbol{S}_{64}$, i.e.,

$$\mathrm{IP}(x) = (x_{\mathrm{IP}(1)}, x_{\mathrm{IP}(2)}, \ldots, x_{\mathrm{IP}(64)}) = x_0.$$

Block x_0 is divided into two blocks of length $n = 32$, i.e., $x_0 = L_0 \cdot R_0$, where L_0 contains the first n bits of the block x_0 and R_0 contains the last n bits. Then the following recurrent computational procedure is applied:

$$L_i = R_{i-1}; \; R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \; i = 1, 2, \ldots, 16, \tag{1}$$

where for each K_i, $f_{K_i} = f(\cdot, K_i)$ is a specific one-to-one mapping from $\{0,1\}^{32}$ to $\{0,1\}^{32}$ and K_i is a key block of length 48, generated in a definite way from the initial key K and satisfying the condition

$$K_i = K_{16-i+1}, \; i = 1, 2, \ldots, 8. \tag{2}$$

The permutation IP^{-1} is applied to the block $y_0 = R_{16} \cdot L_{16}$ to get a cryptogram $y = \mathrm{IP}^{-1}(y_0)$.

It follows from (1) and (2) that for decrypting the above encryption algorithm should be applied to the cryptogram y with the same key K.

Let us describe the main cryptosystem parameters: IP, K_i, f. IP is chosen to be a permutation having the following product-of-cycles decomposition:

$$\begin{gathered} \mathrm{IP} = (1,40,28,13,55,58)(2,8,32,29,53,50)(3,48,27,45,51,42)(4,16,31,61,49,34)(5,56,26) \\ (6,24,30,21,54,18)(7,64,25,37,52,10)(9,39,60)(11,47,59,41,35,44)(12,15,63,57,33,36) \\ (14,23,62,17,38,20)(19,46)(22)(43). \end{gathered}$$

A key block K_i of length $l = 48$ is generated from the initial key K as follows:

If $K = (k_1, \ldots, k_7, k_9, \ldots, k_{15}, \ldots, k_{49}, \ldots, k_{63}) = (\gamma_1, \gamma_2, \ldots, \gamma_{56})$, then $K_i = (\gamma_{c_1(i)}, \gamma_{c_2(i)}, \ldots, \gamma_{c_{48}(i)})$, $i = 1, 2, \ldots, 16$, where $c_1(i), c_2(i), \ldots, c_{48}(i)$ is an ordered combination of $\{1, 2, \ldots, 56\}$ taken l at a time without replacement.

These combinations are produced as compositions of permutation mappings and cyclic shifts in such a way that condition (2) is satisfied.

The values of the mapping f, playing a pivotal role in supporting the DES cryptosystem security, are calculated as follows:

$$f(R, Z) = P(S(E(R) \oplus Z)),$$

where P is a bit permutation of a 32-bit vector, $E : \{0,1\}^n \to \{0,1\}^l$, $S : \{0,1\}^l \to \{0,1\}^n$, $n = 32$, $l = 48$.

The permutation P has the following cycle structure:

$$\begin{gathered} (1,9,24,19,25,32,21,4,31,15,10,16) \\ (2,17,8,18,14,20,3,23,11,30,27,22,29,5,13,26,12,6,28,7). \end{gathered}$$

If $R = (r_1, r_2, \ldots, r_{32})$, $E(R) = (e_1, e_2, \ldots, e_{48})$, then

$$\begin{aligned} e_1 &= r_{32}, \\ e_{6k+i} &= r_{4k+i-1}, \ i = 1,2,\ldots,6; \ k = 0,1,\ldots,7; (i,k) \neq (1,0); \ (i,k) \neq (6,7), \\ e_{48} &= r_1. \end{aligned}$$

The mapping S is defined as a direct product of mappings S_j: $\{0,1\}^6 \to \{0,1\}^4$, $j = 1,2,\ldots,8$, i.e., if $U = (u_1, u_2, \ldots, u_{48}) \in \{0,1\}^{48}$, then

$$S(U) = (S_1(u_1, \ldots, u_6), S_2(u_7, \ldots, u_{12}), \ldots, S_8(u_{43}, \ldots, u_{48})).$$

Each mapping S_j is defined by a set of four Boolean functions S_{j1}, S_{j2}, S_{j3}, S_{j4}, whose values are given by a 4×16 matrix A_j. An element of this matrix is the set of values of functions S_{j1}, S_{j2}, S_{j3}, S_{j4}, represented by the corresponding decimal number.

If $\tilde{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_6) \in \{0,1\}^6$; $2\sigma_1 + \sigma_6 = r$; $2^3\sigma_2 + 2^2\sigma_3 + 2\sigma_4 + \sigma_5 = t$, and element a^j_{rt} of the matrix A_j is equal to $2^3\varepsilon_1 + 2^2\varepsilon_2 + 2\varepsilon_3 + \varepsilon_4$, $\varepsilon_q = 0,1$; $q = 1,2,3,4$, then

$$S_{jq}(\tilde{\sigma}) = \varepsilon_q, \ q = 1,2,3,4.$$

Matrices A_j were chosen in such a way that functions S_{jq} are nonlinear and all sixteen tuples of values S_j in $\{0,1\}^4$ have the same frequency.

Note that in spite of a large number of attempts to crack the cryptosystem with attacks of different types, the issue of DES security is still open [35, 42, 43].

In this connection we should mention Shamir's communication at "Securicom 89" [39] that he had found in DES cryptograms certain dependencies, which can help in cryptoanalysis. This possibility was demonstrated by an example of cracking a simplified DES cryptosystem (20-bit keys and 8 cycles of encryption). For evident reasons Shamir did not publish his cracking method, and its applicability to the standard DES cryptosystem is still unclear.

§3 Public-Key Cryptosystems

Recall that this term is used to refer to cryptosystems in which the encryption key is made public, while the decryption key is kept private.

The cryptosystem includes the following publicly known efficient algorithms: the encryption algorithm E, the decryption algorithm D, and the *key generator* G. The potential receiver (A) generates, using the algorithm G and random coin tosses, two keys: K_1 is a private one and K_2 is that published in a public directory. An arbitrary sender (B) encrypts plaintext x, using the algorithm E with key K_2 and sends the cryptogram $y = E(K_2, x)$ to A. The receiver A decrypts cryptogram y, using the algorithm D with key K_1, and recovers plaintext $x = D(K_1, y)$.

The problem of computing the private key, given a public one, is assumed to be computationally intractable. Let us emphasize that this assumption, even if it is true, does not imply that cryptosystem cracking is also a computationally intractable problem. The existence of a public-key cryptosystem with provably high cracking complexity is currently an open problem [25, 40, 135].

The cryptosystems surveyed below are based on well-known mathematical problems that are commonly believed to be intractable.

1. Knapsack system. This cryptosystem was named after the well-known combinatorial knapsack problem [121].

The public-key of the cryptosystem is an integer n-dimensional vector a, the plaintext x is a binary n-dimensional vector, the cryptogram y is a scalar product of a and x.

In general the problem of solving the equation

$$y = ax \tag{3}$$

in x is NP-complete [121].

If vector a is superincreasing, i.e., if each of its coordinates exceeds the sum of all preceding coordinates, then solving Eq. (3) is an easy task. Merkle and Hellman [40, 111] proposed to "mask" easy instances of the knapsack problem in such a way that they look like NP-complete ones: let vector b be superincreasing, n, e, d be integers such that $ed = 1(\mathrm{mod} n)$, n be larger than the sum of all coordinates of vector b. Then vector

$$a = eb(\mathrm{mod} n)$$

is, in general, not superincreasing and does not have the structure of vector b. Solving Eq. (3) with unknown n, e, d seems to be a more complex problem. The vector b and numbers n, e, d constitute a private key of the cryptosystem.

The decryption consists in solving the equation

$$dy = bx(\mathrm{mod} n)$$

in x.

An obvious improvement of the cryptosystem consists in multiplying vector b sequentially by $e_1(\mathrm{mod}\ n_1)$, $e_2(\mathrm{mod} n_2), \dots, e_k(\mathrm{mod} n_k)$ and permuting the coordinates of vector a. Other modifications were also proposed.

Neither for the Merkle–Hellman cryptosystem, nor for its numerous modifications [30, 69, 102, 143] did anyone succeed in proving that its cracking problem is polynomially equivalent to the knapsack problem. Moreover, Shamir [139, 141] proposed an algorithm for transforming vector a into superincreasing vector b, which resulted in cracking the cryptosystem.

2. Cryptosystems based on linear codes. Let the plaintext x and the cryptogram y be vectors of dimension n and m, respectively, over field $\mathrm{GF}(q^r)$, and G be an $m \times n$ matrix over this field, and, in addition,

$$y = xG. \tag{4}$$

In linear coding theory [16] y is called a codeword and the matrix G is called a generator matrix for the code. In the case of an arbitrary linear code, the problem of decoding, i.e., solving Eq. (4) in x, is NP-complete.

If the matrix G is a generator matrix for the error-correcting code, then fast decoding algorithms are known [17].

McEliece proposed a cryptosystem [107], in which the generator matrix for the t-error correcting code is "masked" in such a way that it looks like a matrix K for some linear code:

$$K = SGP, \tag{5}$$

where G is an $m \times n$ matrix, S is a nonsingular $m \times m$ matrix, P is a permutation $n \times n$ matrix, $m = n + t$.

The matrix K is the public key; the matrices S, G, P constitute the private key.

The encrypting procedure is given by the equation

$$y = xK + z, \tag{6}$$

where z is a vector introducing at most t errors. This vector is chosen by each sender at random and it does not affect the result of decryption since all the errors are corrected when decoding. To decrypt, the receiver right-multiplies the cryptogram y by the private matrix P^{-1}. Since P^{-1} is a permutation matrix, the vector zP^{-1} also introduces at most t errors. Then the receiver applies the well-known decoding algorithm to the vector

$$yP^{-1} = xSG + zP^{-1}$$

and obtains the vector xS and, finally, right-multiplying it by the private matrix S^{-1}, he obtains the initial plaintext x.

If the adversary tries to decipher the cryptogram by solving Eq. (6) in x, then he can do this, in general, in $O\left(m^3\binom{n}{m}/\binom{n-t}{m}\right)$ operations [107].

Gabidulin, Paramonov, and Tretjakov [54] proposed a modification of the McEliece scheme, based on the maximum distance codes [16].

If $x \in F^N$, where $F = \mathrm{GF}(q^s)$, then its coordinates $x_0, x_1, \ldots, x_{N-1}$ may be considered as the elements of an s-dimensional space over the field $\mathrm{GF}(q)$ [155].

Let $r(x)$ be the rank of the vector system $\{x_0, x_1, \ldots, x_{N-1}\}$; S be a nonsingular $m \times m$ matrix; G be a random generator $m \times n$ matrix for the maximum distance t-error correcting code, α be a nonzero column vector in F^m; e_g be a nonzero vector in F^n such that

$$r(e_g) \leq t_g < t,$$

where t_g is a parameter to be evaluated.

S, G, α, and e_g constitute the private key of the cryptosystem. The matrix $K = SG + \alpha e_g$ is the public key.

From the choice of α it follows that if the word z introduces at most $t_e = t - t_g$ errors, then the vector $x\alpha e_g + z$ may be considered as a vector introducing into codeword xSG at most t errors. This implies that to decrypt the cryptogram

$$y = xK + z,$$

it is necessary first to apply the well-known decoding algorithm [16, 109], and then to apply the private matrix S^{-1} to the obtained word. If the adversary tries to decipher the cryptogram by solving the system (6) with the algorithm given in [107], then it would take $O(m^3 L(m, t_e))$ [54] operations, where

$$L(m,i) = \begin{bmatrix} m \\ i \end{bmatrix} (q^n - 1)(q^n - q)\ldots(q^n - q^{i-1}),$$
$$\begin{bmatrix} m \\ i \end{bmatrix} = \frac{(q^m - 1)(q^m - q)\ldots(q^m - q^{i-1})}{(q^i - 1)(q^i - q)\ldots(q^i - q^{i-1})}.$$

A cryptosystem, slightly different from the McEliece system, is considered by Niederreiter in [117]. Let $F_q = \mathrm{GF}(q)$, $F = F_q \cup \{\infty\}$ be a field with adjuncted element ∞. Let us consider the matrix A

$$A = \begin{pmatrix} z_1\alpha_1^0 & z_2\alpha_2^0 & \ldots & z_N\alpha_N^0 \\ z_1\alpha_1^1 & z_2\alpha_2^1 & \ldots & z_N\alpha_N^1 \\ \ldots & \ldots & \ldots & \ldots \\ z_1\alpha_1^s & z_2\alpha_2^s & \ldots & z_N\alpha_N^s \end{pmatrix},$$

where $\alpha_i \in F$, $z_i \in F_q \backslash \{0\}$, $\alpha_i \neq \alpha_j$ if $i \neq j$ and the jth column of the matrix A has the form $z_j(0, 0, \ldots, 0, 1)^T$ if $\alpha_j = \infty$.

The matrix A is a check matrix of q-ary generalized Reed–Solomon code [109]. Let us consider the collection

$$\mathcal{E} = \{HA : H \in \mathrm{Gl}(n, F_q)\}.$$

An arbitrary matrix $K \in \mathcal{E}$ has the form

$$K = HA = \begin{pmatrix} z_1 f_1(\alpha_1) & z_2 f_1(\alpha_2) & \ldots & z_N f_1(\alpha_N) \\ z_1 f_2(\alpha_1) & z_2 f_2(\alpha_2) & \ldots & z_N f_2(\alpha_N) \\ \ldots & \ldots & \ldots & \ldots \\ z_1 f_{s+1}(\alpha_1) & z_2 f_{s+1}(\alpha_2) & \ldots & z_N f_{s+1}(\alpha_N) \end{pmatrix}, \tag{7}$$

where the polynomials $f_j(x)$ are linearly independent over F_q and their degrees are at most s.

The cracking of the knapsack cryptosystem encouraged researchers to develop algorithms for finding the representation of the matrix K (public key) in form (5) (for the McEliece scheme) or in form (7) (for the Niederreiter scheme).

Sidelnikov and Shestakov [146] developed an algorithm of complexity $O(s^4 + sN)$, allowing one to find matrices H and A, given the matrix K, in the Niederreiter cryptosystem.

3. Cryptosystems based on the theory of L-systems. Salomaa [132] proposed to use the Lindenmayer theory of formal languages (the theory of L-systems) in designing public-key cryptosystems [86, 87, 130, 151].

Let Σ^* and Δ^* be the sets of words over finite alphabets Σ and Δ, respectively. The mapping $\Sigma^* \to \Delta^*$ is called a *morphism* if the image in Δ^* of any word v from Σ^* coincides with the sequence of images of its letters. A letter a in the alphabet Σ is *dummy* if its image in Δ^* under the given morphism is an empty word.

Now let h_0 and $h_1 : \Sigma^* \to \Sigma^*$ be two morphisms such that for any v in Σ^* the equality

$$h_{i_1}(h_{i_2}(\dots h_{i_n}(v))) = h_{j_1}(h_{j_2}(\dots h_{j_m}(v))),$$

where $i_k \in \{0,1\}$, $j_k \in \{0,1\}$, holds iff

$$m = n,\ i_1 = j_1,\ i_2 = j_2, \dots,\ i_n = j_n.$$

Further, let $g : \Delta^* \to \Sigma^*$ be a morphism such that the images in Σ^* of some letters from Δ are empty words, and let w and u be nonempty words in Σ^* and Δ^* respectively such that

$$w = g(u). \tag{8}$$

Let, finally, σ_0 and $\sigma_1 : \Delta^* \in \Delta^*$ be two multivalued morphisms such that for any word v in δ^* and each of its images in $\sigma_k(v)$

$$g(\sigma_k(v)) = h_k(g(v)),\ k = 0, 1. \tag{9}$$

Kari [86, 87] proposed a cryptosystem in which the public key is a T0L-system $(\Delta, \sigma_0, \sigma_1, u)$ and the plaintext x is a binary sequence. The cryptogram y is defined as the image of the word u in Δ^* under a sequence of morphisms, composed of σ_0 and σ_1 in the order determined by the sequence of plaintext letters, i.e., if

$$x = x_1, x_2, \dots, x_n;\ x_k \in \{0,1\},\ k = 1, 2, \dots, n,$$

then

$$y = \sigma_{x_n}(\dots \sigma_{x_2}(\sigma_{x_1}(u))).$$

The cryptosystem private key consists of the DT0L-system (Σ, h_0, h_1, w) and the morphism g. Expressions (8) and (9) imply that

$$g(y) = h_{x_n}(\dots h_{x_2}(h_{x_1}(w)))$$

and the above-mentioned properties of morphisms h_0 and h_1 allow one to recover the sequence $x_n, \dots, x_1$ uniquely. It is obvious that the cryptogram considerably exceeds the plaintext in length. In [87] a condition on the public key is given, sufficient for this difference in length to be at most polynomial.

The T0L-system H is *polynomially growing* if there exists a polynomial p such that any plaintext of length n can be encrypted only to cryptograms of length at most $p(n)$.

Theorem [87]. *Let $H = (\Delta, \sigma_0, \sigma_1, u)$ be a* T0L-*system. Suppose there exists a function $f : \Delta \to N$ satisfying the following conditions for any letter $d \in \Delta$ and any word $d_1 d_2 \dots d_n \in \sigma_i(d)$:*

(a) $f(d) \le f(d_j)$, $j = 1, 2, \dots, n$;

(b) $f(d) = f(d_j)$ *for at most one* $j \in \{1, 2, \dots, n\}$.

Then the T0L-*system H is polynomially growing.*

In addition, Kari showed that one of the approaches to the cracking problem for this cryptosystem leads to an NP-hard problem.

Theorem [87]. *Given the* T0L-*system* $H = (\Delta, \sigma_0, \sigma_1, u)$, *the problem of constructing the interpretation morphism* $g : \Delta^* \to \Sigma^*$, *mapping* H *into the* DT0L-*system* $G = (\Sigma, h_0, h_1, w)$ *and satisfying the following conditions:*

(a) *for each letter* $d \in \Delta$, $g(d) \in \Sigma \cup \{\lambda\}$;

(b) *there exists a letter* $d \in \Delta$ *such that* $g(d) \in \Sigma$;

(c) *for any letter* $d \in \Delta$, *index* $i \in \{0, 1\}$, *and word* $x \in \sigma_i(d)$, $g(x) = h_i(g(d))$;

(d) $g(u) = w$,

is an NP-hard problem.

Note that this theorem proves the NP-hardness of the following problem: given the public key H, find the private key (G, g). The cryptosystem cracking consists in computing the plaintext from the cryptogram and the public key. Therefore, this result does not preclude the possibility that there exist polynomial algorithms that crack this cryptosystem.

4. Cryptosystems based on partially linear transformations. In [70, 71] Guan studied partially linear transformations from the standpoint of their applicability in designing public-key cryptosystems.

The mapping $F : S^m \to S^m$, where the ring S is an initial alphabet and m is the plaintext block length, is given by the set of functions $\{f_1, f_2, \dots, f_m\}$:

$$\begin{cases} y_1 = f_1(x_1, x_2, \dots, x_m) \\ y_2 = f_2(x_1, x_2, \dots, x_m) \\ \dots\dots\dots\dots \\ y_m = f_m(x_1, x_2, \dots, x_m) \end{cases} \tag{10}$$

In general the problem of inverting such mappings is reduced analytically to solving systems of nonlinear equations over a finite ring. In particular, the problem of solving system (10) in $x_1, x_2, \dots, x_m$ over the field GF(2) is NP-complete [72].

For the sake of simplicity we consider below the case $S = \mathrm{GF}(2)$.

A class of nonlinear transformations $S^m \to S^m$, defined by the composition of "simple" nonlinear transformations and called partially linear, is proposed for designing cryptosystems. The set of variables $X = \{x_1, x_2, \dots, x_m\}$ is divided into s disjoint classes:

$$X = X_1 \cup X_2 \cup \ldots \cup X_s, \ |X_j| = k_j, \ X_i \cap X_j = \varnothing, \ i \neq j.$$

The set of functions $\{f_i\}$ is also divided into s classes in such a way that the functions in the class corresponding to the set X_j are linear in variables of the set X_j and the matrix of coefficients of these variables is nonsingular. These functions can depend also on the variables in the preceding classes X_i, $i = 1, 2, \dots, j-1$. The following transformation is an example of a partially linear transformation:

$$\begin{cases} y_1 = a_{11}x_{i1} \oplus a_{12}x_{i2} \oplus \ldots \oplus a_{1k}x_{ik} \oplus a_1 \\ y_2 = a_{21}x_{i1} \oplus a_{22}x_{i2} \oplus \ldots \oplus a_{2k}x_{ik} \oplus a_2 \\ \dots\dots\dots\dots\dots\dots \\ y_k = a_{k1}x_{i1} \oplus a_{k2}x_{i2} \oplus \ldots \oplus a_{kk}x_{ik} \oplus a_k \\ y_{k+1} = x_{ik+1} \oplus g_1(x_{i1}, x_{i2}, \dots, x_{ik}) \\ y_{k+2} = x_{ik+2} \oplus g_2(x_{i1}, x_{i2}, \dots, x_{ik}) \\ \dots\dots\dots\dots\dots\dots \\ y_m = x_{im} \oplus g_{m-k}(x_{i1}, x_{i2}, \dots, x_{ik}) \end{cases}$$

where $g_1, g_2, \dots, g_{m-k}$ are arbitrary Boolean functions and the matrix $\|a_{ij}\|$ is nonsingular.

A public-key cryptosystem can be based on the choice of some sequence of partially linear transformations $F_1, F_2, \dots, F_n$. The public key is the composition of these transformations

$$F = F_n \circ F_{n-1} \circ \ldots \circ F_1,$$

and the private key is the sequence

$$F_1^{-1}, F_2^{-1}, \dots F_n^{-1}.$$

Note that the issue of practical ways of implementing these cryptosystems remains open.

5. The Rivest–Shamir–Adleman cryptosystem (RSA). In [128] Rivest, Shamir, and Adleman proposed a public-key cryptosystem known as RSA. The public key of the RSA consists of an integer n, which is equal to the product of two large primes p and q, and a number e, relatively prime to $\varphi(n) = (p-1)(q-1)$.

A private key is a triple (p, q, d), where d is such that

$$e * d = 1(\mathrm{mod}\varphi(n)). \tag{11}$$

From the Euclidean algorithm of finding the greatest common divisor of two numbers it follows that such a d is unique.

For encrypting the plaintext is divided into blocks x; each of them may be considered as an integer not exceeding $n - 1$. The corresponding cryptogram y is computed by the formula

$$y = x^e(\mathrm{mod}n).$$

The decryption is executed by raising the block y into power d.

The assumption on the security of the RSA cryptosystem is based on the hypothesis that integer factorization is a computationally intractable problem.

If we put $e = 2$ for the RSA cryptosystem and take a pair p, q as a private key, then this modification is called the *Rabin cryptosystem*. In this cryptosystem decryption consists in solving the equation

$$y = x^2(\mathrm{mod}pq), \tag{12}$$

which is solvable for a quarter of the possible values of y and for all these y it has, as a rule, four different solutions.

Since the RSA cryptosystem was first published, considerable efforts have been made to study its security and that of its modifications. Rabin [124] proved that the cracking problem of his cryptosystem is polynomially equivalent to the integer factorization problem. Williams [162] proved a similar result for another modification of the RSA cryptosystem. For the RSA cryptosystem itself no such result is known. However, Kurosawa and Matsu [91] showed that the RSA cracking problem is equivalent to the problem of computing $x(\mathrm{mod}3)$ with probability $1/3 + 1/p$, where p is some polynomial.

Håstad [73] and Wiener [161] showed the insecurity of the RSA cryptosystem with small e and d. For instance, the cracking method, proposed by Wiener, is applicable if the length of d is at most a quarter of the module length.

There are also some results concerning the bitwise hardness of the RSA cryptosystem. The bitwise hardness of one-way functions is considered in §6. Here we give the statements of some results.

Let $E(x) = x^e(\mathrm{mod}n)$ for the RSA cryptosystem and $E(x) = x^2(\mathrm{mod}n)$ for the Rabin cryptosystem. The symbol x_i of the plaintext x is λ-secure, $0 < \lambda < 1$, if any algorithm computing x_i, given $E(x)$, with error probability less than $1 - \lambda$ may be transformed into the algorithm of the same complexity computing x, given the $E(x)$. U. Vazirani and V. Vazirani [158] proved that for the RSA cryptosystem the least significant bit is $(0,732 + \varepsilon)$-secure and all the other bits are at least $(15/16 + \varepsilon)$-secure. Alexi et al. [1, 2] showed that for the RSA and Rabin cryptosystems the least significant bit is $(1/2 + 1/(\log n)^c$-secure for any $c > 0$ and all sufficiently large n. $\log\log n$ lower order bits have the same security.

Cryptographical requirements and, in particular, investigation of the RSA cryptosystem, renewed interest in the development of integer factorization techniques.

There are several surveys on the *factorization*; see, for instance, [127, 150, 160]. Here we only note the main achievements. The existing factorization algorithms have the complexity $\exp\{(c+o(1))(\ln n \ln\ln n)^{1/2}\}$ with various constants c. Such estimates are frequently based on some unproved assumptions. One of the exceptions is the Vallee algorithm [154] of complexity $\exp\{((4/3)^{1/2} + o(1))(\ln n \ \ln\ln n)^{1/2}\}$, having a sufficiently rigorous proof of the estimate.

Pomerance proposed the quadratic sieve method. In [123] he described various modifications of this method and gave the following complexity estimate:

$$\exp\{((9/8)^{1/2} + o(1))(\ln n \ \ln\ln n)^{1/2}\}.$$

Bach and Shallit [5] developed a probabilistic factorization algorithm, using cyclotomic polynomials.

Seysen [136] described a probabilistic factorization algorithm with quadratic forms, having the complexity

$$\exp\{((5/4)^{1/2} + o(1))(\ln n \ \ln \ln n)^{1/2}\}.$$

Lenstra [97] gives an algorithm for finding a nontrivial factor of an integer n with the probability at least $1/2 - o(1)$ in $\exp\{(1+o(1))(\ln n \ \ln \ln n)^{1/2}\}$ steps, but this estimate is valid only under the assumption of the generalized Riemann hypothesis.

There are several factorization algorithms for integers of special form. For integers of the form $n = r^l + s$, where r and s are not large, A. Lenstra et al. [98] proposed the number field sieve method with complexity estimate, under some heuristic assumptions, of the form

$$\exp\{(1 + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}\}.$$

To implement the RSA cryptosystem one must possess algorithms for generating sufficiently large primes and algorithms for primality testing. See Vasilenko's survey [157] for primality testing techniques.

§4. Cryptographic Protocols

Cryptology, founded initially as a science of encrypting, in the last decade has been significantly extended, owing to the emergence of a new research field, namely that of cryptographic protocols.

In the field of programming the protocol is usually considered to be a distributed algorithm for some problem, together with assumptions on the format of transmitted messages, on the responses to failures, synchronization of computational processes, etc. As a rule, a cryptographic protocol fits this scheme, but there are no common terms. The main difficulty is that the class of problems solved by cryptographic protocols is rather large: on the one hand, it merges with encryption problems, on the other—with problems of distributed computations and teleprocessing. In both cases, it seems to be impossible to draw the precise lines.

In contrast to cryptosystems, cryptographic protocols are intended not only for data encryption, and even mainly not for encryption, but for ensuring *information integrity*, which means ensuring that information comes unaltered and from the right source. In some cases data is not required to be encrypted at all—messages are transmitted in plaintext form. The role of the adversary also changes. First, the adversary may become active, i.e., not limit his activities to ciphertext tapping and deciphering, but also try to alter information and to deceive its receiver. Second, the adversary now is not necessarily a "third party." He may be one or several protocol participants. Thus, one of the main tasks of cryptographic protocols is to protect the usage of public communication networks from the malicious and involuntary faults of partners.

We may conventionally divide cryptographic protocols into two classes: application and primitive ones. Application protocols are intended for solving particular cryptographical problems that occur in practice. These include the above-mentioned authentication and digital signature schemes. Authentication schemes, because of their particular importance for cryptography, are considered below in more detail. For the theoretical background for designing secure digital signature schemes, see §6.

Key distribution protocols allow participants to produce the common private key as a result of message exchange over public communication lines. This is also one of the most important protocol types that is considered below in more detail.

A *t-private protocol* for computing a function f over an integer domain is defined as follows. Let initially an ith participant of the protocol hold a value of x_i and a random string r_i, $1 \leq i \leq n, n \geq 3$. The protocol computes the function f with ε-advantage, $0 < \varepsilon \leq 1/2$, if all the parties eventually receive an identical result, equal to $f(x_1, x_2, \ldots, x_n)$ with probability at least $1/2 + \varepsilon$. A protocol is t-private if any coalition of no more than t parties does not learn any additional information from the protocol execution (for instance, no information about the values of other arguments of the function).

A function f is *t-private* if there exists a t-private protocol that computes f with ε-advantage for some ε, $0 < \varepsilon \leq 1/2$.

Ben-Or et al. [12] and Chaum et al. [27] have shown that if the domain of f is finite, then f is $\lfloor\frac{n-1}{2}\rfloor$-private. In addition, some functions, like $\sum x_i$, are n-private, while others, including $\bigvee x_i$, are not $\lceil\frac{n}{2}\rceil$-private [8, 12, 29, 92].

Chor et al. [28] considered functions over countable domains. The possibility of computing the function f privately is shown to be closely related to its *communication complexity* (the notion of communication complexity for distributed computations was defined by Yao [164]). As a result, it is demonstrated that the function $\sum x_i$, defined over Z_+, is $\lfloor\frac{n-1}{2}\rfloor$-private, but the same function, defined over Z, is not even 1-private. t-Private Boolean-valued functions over countable domains are completely characterized in [28]. It is shown that such a function is 1-private iff its communication complexity is bounded. If the function f is 1-private, then it is also $\lfloor\frac{n-1}{2}\rfloor$-private.

Chor and Kushilevitz [29] proved that an $\lceil\frac{n}{2}\rceil$-private Boolean-valued function is also n-private. Hence, it is shown that every Boolean-valued function over a countable domain is either n-private, $\lfloor\frac{n-1}{2}\rfloor$-private, but not $\lceil\frac{n}{2}\rceil$-private, or not even 1-private [28].

The *secret-sharing problem* is closely related to that of computing functions privately. In this problem some private information m is divided into n shares distributed among n parties in a one-to-one fashion. What is required is a (t, n)-*threshold scheme*: the information must be uniquely recoverable from any t shares, while any $t - 1$ shares are insufficient for this purpose.

(t, n)-Threshold schemes were first described by Shamir in 1979 [137]. His idea was based on the following fact: any polynomial of degree $t - 1$ is recoverable in a unique way from its values in any t points with the help of the Lagrangian interpolation formula. In 1979 Blakley independently developed another implementation of the (t, n)-threshold scheme [18]. The title of Blakley's paper, "Safeguarding cryptographic keys," reflects one of the main applications of (t, n)-threshold schemes.

The secret-sharing problem may be stated in terms of protocols.

A protocol for a (σ, ρ)-secret-sharing problem [45] is a pair of n-processor protocols (P_1, P_2). A distinguished processor p_0 initially has some private information m. During P_1 processor p_0 distributes shares of m in such a way that no coalition of σ processors, not including p_0, can get any additional information about the secret m. P_2 is a protocol for recovering m. If p_0 remains nonfaulty during P_1 and the total number of failed processors is at most ρ, then the information m will be recovered.

Two adversaries are considered.

Passive A_L, listening to at most σ processors, i.e., any information sent or received by these processors is available to him.

Active A_D, who enjoys total control over at most ρ processors.

Let Q be a finite set from which the private information m is chosen.

The following two requirements to be satisfied by the protocol are stated:

Secrecy: $\forall m, m' \in Q$, no passive adversary can get any information on what secret the protocol is working with.

Resiliency: no active adversary A_D can prevent the correct recovering of m with protocol P_2.

A protocol satisfying secrecy and resiliency requirements is called (σ, ρ)-*unverified secret sharing*. In the case where $\sigma = \rho = t$, such a protocol is called t-unverified secret sharing.

A *t-verifiable secret-sharing protocol* is a t-unverified secret-sharing protocol with the following additional condition: if, during P_1, processor p_0 becomes faulty, but some subset of at least $n - t$ processors is correct at the end of P_1 (i.e., these processors received their shares of m) and the total number of faulty processors is at most t, then the secret m will be recovered correctly.

Dolev et al. [45] showed that $\sigma + 2\rho + 1$ processors are necessary for (σ, ρ)-unverified secret sharing. This bound can be achieved [12, 108]. Rabin and Ben-Or [125] proved that for any k, t-unverified secret sharing can be achieved with $2t + 1$ processors and with error probability at most 2^{-k}. This implies the possibility of (σ, ρ)-unverified secret sharing with $\sigma + \rho + 1$ processors and with small error probability. Dolev et al. [45] proved that $3t + 1$ processors are necessary for t-verifiable secret sharing.

Secret-sharing techniques play a fundamental role in the solution of the *secret message transmission problem*. The sender, connected with the receiver by n communication lines, wishes to send him a secret message. As above, two adversaries are considered: passive A_L, listening to at most σ lines, and active A_D, controlling at most ρ lines. The problem is to design a message transmission protocol that satisfies secrecy and resiliency requirements. Resiliency, in this case, means guaranteed delivery of message to receiver, independently of active adversary behavior.

Dolev et al. [45] proved that $n \geq \max\{\sigma + \rho + 1, 2\rho + 1\}$ lines are necessary and sufficient to solve this problem. The complexity of the proposed 3-round protocol is bounded by a polynomial in n.

Note that if $\rho = 0$ (there is no active adversary), the problem considered is exactly the problem of secret data transmission, which is traditionally solved with cryptosystems. Moreover, the solution obtained in [45] is absolute, i.e., it depends on no unproved assumption and is applicable to any n (is not asymptotic).

Thus, if future technological progress would permit implementing communication networks in which any pair of users is connected by a large number of lines, then the problem of secret communication could be solved with such protocols.

Certified mail ensures a correspondence exchange among mutually suspicious partners. One of the protocol participants wishes to send a message to another participant, but in such a way that the receiver could learn nothing about its contents till the sender receives a certificate indicating that the message was delivered [19, 21].

Contract-signing protocol allows two partners to do so, exchanging messages through the communication network in such a way that any participant can interrupt the protocol with the commitment of another, and not having committed himself, only with negligibly small probability [48, 57].

Voting protocol allows to vote using the communication network. The voting results become public, while any information on participant individual votes remains unavailable [10, 13, 31].

Primitive cryptographic protocols are used in constructing application protocols. It is clear that the partition of protocols into primitive and application is rather conventional. For instance, certified mail can be used as a primitive in designing contract-signing protocols.

Note that the problem of dependable protection of private information transmitted over the public communication network may be considered as a problem of constructing the related cryptographic protocol. In this case the cryptosystem and key distribution protocol would be primitives. In these terms, the security of the cryptosystem itself is not sufficient for such a protocol to be secure. Moore [113] considered related problems. Li Gong [100] proposed a cryptographic protocol capable of withstanding a verifiable-text attack, mentioned in §1.

The main primitives used in cryptographic protocol design are the so-called *zero-knowledge proof protocols*. Section 10 is devoted to them.

A *distributed coin-flipping protocol* allows two or more mutually suspicious parties to flip a coin, exchanging information through communication lines [20]. Dwork et al. proposed an n-processor coin-flipping protocol [46]. The protocol tolerates up to $cn/\log n$ malicious processor faults, and the number of rounds of interprocessor message exchange is bounded by a constant independent of n.

Yao [167] showed how two participants A and B can interactively produce a random integer $n = pq$ in such a way that its secret (i.e., prime factors p and q) is unavailable to each participant, but is recoverable, if desired, in cooperation. This result can be used, for instance, to design a protocol that permits A and B to exchange a pair of secrets S_A and S_B in such a way that S_A becomes computable by B when and only when S_B becomes computable by A.

Brassard et al. [24] studied protocols that allow their participant A, having a certain number of secrets, to disclose one of them to participant B. B may choose any secret, but he does not wish A to know which one. On the other hand, A is not willing to give B a chance to know any information about more than one secret. The *"all-or-nothing" disclosure protocol* [24] is such a protocol, in which during its execution, as soon as B obtains some information about one of the secrets, he has lost any chance to learn anything about the other secrets.

The theoretical background of authentication schemes was developed by Simmons [147]. According to Simmons, in an *authentication system* the sender and receiver choose some encoding rule $e \in E$ and keep it private. The sender observes some state of the source $s \in S$ and, using an encoding rule e, determines a message $m \in M$ to be sent to the receiver.

The adversary can use two different attacks on the authentication system: *impersonation*—a valid message is formed when in fact nothing has been sent; *substitution*—an adversary waits for a message to be sent by the sender and replaces it with another valid message. P_I and P_S denote the adversary's best-possible probability of success in impersonation and substitution attack, respectively.

Let P_d be the probability that the adversary succeeds by choosing optimally between impersonation and substitution.

Under the assumption that the adversary knows all the authentication system's statistics, excluding the probability distribution $P(e)$ for the encoding rule, which is assumed to be independent of the source state, Simmons [147] showed that $P_d \geq \max\{P_I, P_S\}$ and that for constructing a P_d-secure authentication scheme the cardinality of the set of encoding rules is required to be at least $1/P_d^2$. If the adversary also knows $P(e)$, then, as Massey proved [104], $P_d = \max\{P_I, P_S\}$.

Simmons [147] also proved that $P_I \geq 2^{-I(M,E)}$, where $I(M,E)$ is the mutual information, and gave the necessary and sufficient condition for equality in this relation: first, for each message m, probabilities $P(m|e)$ are the same for all encoding rules for which m is a valid message; second, another condition is satisfied, which implies that choosing a valid message completely at random is an optimum impersonation attack.

A number of other results of practical significance was obtained in [147]. The reader may become familiar with these results in Simmons's survey [148].

Fiat and Shamir proposed [51, 140] an authentication protocol with the same underlying ideas as the RSA cryptosystem. In contrast to the Simmons theoretical scheme, in this case we consider a scheme that permits one only to identify communication network users. Such a scheme is sometimes called a *simple authentication.*

Let A and B be communication network users and let B ask A to prove his identity. A is supposed to have some secret information S and must prove this to B, not disclosing S.

In the proposed authentication scheme S consists of k integers $s_1, s_2, \ldots, s_k$ and primes p, q. A computes and publicizes numbers v_j such that $v_j s_j^2 = 1 (\text{mod } n)$, $j = 1, 2, \ldots, k$, where the public module n is the product of p and q.

For the authentication A and B execute the following protocol:

(1) A chooses t random numbers $r_i \in [0, n)$ and sends to B t numbers $x_i = r_i^2 (\text{mod } n)$, $i = 1, 2, \ldots, t$.

(2) B sends to A a random $t \times k$ Boolean matrix $\|e_{ij}\|$.

(3) A sends to B t numbers $y_i = r_i \prod s_j (\text{mod } n)$, where the product is taken over all j such that $e_{ij} = 1$.

(4) B accepts if for all $i, 1 \leq i \leq t, x_i = y_i^2 \prod v_j (\text{mod } n)$, where the product is taken over the values of j given in (3).

Let $\bar{A}$ and $\bar{B}$ be the parts of a protocol executed by A and B respectively. An adversary C, impersonating B, may ask A to execute an authentication protocol a number of times. After an analysis of the information he has received, C might try to simulate the behavior of A during the execution of an authentication protocol and thereby deceive the user B.

Let $\tilde{A}$ and $\tilde{B}$ be the versions of the authentication protocol parts executed by the adversary C.

Theorem [140]. *Suppose that after g-times execution of protocol $(\bar{A}, \tilde{B})$ the adversary C can execute the protocol $(\tilde{A}, \bar{B})$ with probability of success higher than $(1 + \varepsilon)2^{-kt}$, where $\varepsilon > 0$. Then the number n may be factorized in time of order*

$$g|(\bar{A}, \tilde{B})| + ((2 + \varepsilon)/(1 + \varepsilon))2^{kt}|(\tilde{A}, \bar{B})|$$

with probability higher than $\varepsilon(1 - e^{-1})/2(1 - \varepsilon)$, where $|Z|$ is the complexity of protocol Z.

Note that under some assumptions the security of the Fiat–Shamir protocol was considered to be proved, i.e., this scheme was considered to be provably a zero-knowledge proof protocol (see §10). Nevertheless, in the thesis of the First International Workshop on the Mathematical Concepts of Dependable Systems, held in 1990 in Oberwolfach, it is noted [105] that Desmedt and Burmester have found a flaw in the Fiat–Shamir scheme. As can be concluded from this short thesis, this scheme is not sound. Many aspects of cryptographic protocol security were also critically reconsidered in [105].

Okamoto and Tanaka [119] proposed a message authentication (digital signature) and key distribution scheme for computer networks, also based on the ideas underlying the RSA cryptosystem.

A network has a *key distribution center* that holds a private key: factors p and q of the module n and degree d for decryption. Public information consists of identifiers $I_1, I_2, \dots$ for all network users, module n, degree e for encryption, integer α—a primitive element of fields GF(p) and GF(q)—and some hard-to-invert hash function f, with values less than e (for more information on the hash functions, see §6).

At the request of the ith user the center computes $S_i = I_i^d$ and sends him over the private communication line. When the ith user wishes to send an authenticated message to the jth user, he stamps time T, randomly chooses a number r_i, computes

$$X_i = \alpha^{er_i} (\text{mod } n),$$
$$C = f(X_i, T, M),$$
$$Y_i = S_i \alpha^{Cr_i} (\text{mod } n),$$

and sends X_i, Y_i, T, M over the network to the jth user.

On receiving these data, the jth user computes C, and then checking the equality

$$Y_i^e = I_i X_i^C (\text{mod } n) \tag{13}$$

becomes sure that the message was sent just by the ith user.

It is clear that if M changes, C changes also, and equality (13) no longer holds. Hence, message substitution by the adversary, not knowing S_i, seems unlikely.

The dependence of the hash function f on time T prevents the possibility of the following attack on protocol: the adversary, having once tapped a message, further sends it to the receiver on behalf of the sender a number of times.

The same principal idea underlies the key distribution scheme proposed by the same authors [119].

In 1976 Diffie and Hellman [41] suggested a key distribution scheme based on discrete exponentiation. The problem opposite to discrete exponentiation is known as the *problem of computing discrete logarithms*. The *discrete logarithm* is considered to be a hard-to-compute function [118], which makes it rather attractive for cryptographic applications.

In the *Diffie and Hellman scheme* [41] two users A and B, having no secret information in common, produce a common private key. The public information consists of a prime number p and an α, which is a primitive element of GF(p). Users A and B, wishing to exchange some private information, each choose a random number in GF(p), say X_A and X_B, respectively; then each computes $Y_A = \alpha^{X_A} (\text{mod } p)$ and $Y_B = \alpha^{X_B} (\text{mod } p)$ and puts this into the public file, keeping X_A and X_B private. Since $Y_A^{X_B} = Y_B^{X_A} (\text{mod } p)$, A and B can compute the value $K_{AB} = \alpha^{X_A X_B} (\text{mod } p)$, which is used as a private key.

El Gamal [47] proposed the following modification of the Diffie and Hellman scheme.

Sender A chooses a random number k in GF(p) and computes $K = Y_B^k (\text{mod } p)$ from the value Y_B. Then he computes

$$Y_1 = \alpha^k (\text{mod } p),$$
$$Y_2 = Kx (\text{mod } p),$$

where x is a plaintext, and sends to B a cryptogram $y = Y_1 \cdot Y_2$, where the dot denotes concatenation.

B decrypts in two stages. First he determines K, raising Y_1 to the power X_B, known only to him, and then computes x.

An important unsolved issue is the polynomial equivalence of the problem of computing key K_{AB} in the Diffie–Hellman scheme and the discrete logarithm problem.

McCurley [106] obtained a partial result on the security of one modification of the Diffie–Hellman scheme.

Let $n = pq$, where p and q are primes such that $p = 8r + 3$, $q = 8s - 1$, integers $2r + 1$ and s have a large prime factor, numbers $4r + 1$ and $4s - 1$ are primes. Let $G = (Z/nZ)^*$.

Theorem [106]. *Let $0 < \sigma < 1$. Let A_σ be an algorithm, computing the values of a function*

$$\text{DH}(g, n, a, b) = g^{xy} (\text{mod } n),$$

where $a = g^x$, $b = g^y$, for at least $\sigma(\mathrm{ord}_n 16)^2$ input pairs $(a,b) \in \langle 16\rangle_n \times \langle 16\rangle_n$ (where $\langle 16\rangle_n$ is a G subgroup, generated by element 16; $\mathrm{ord}_n 16$ is its order). If A_σ executes no more than $R(n)$ bit operations, then an algorithm B_σ exists that outputs p and q with probability at least 0.5 in at most $O(\sigma^{-1}(R(n) + \log^2 n))$ bit operations.

The bitwise complexity of the discrete logarithm was also studied. For instance, Long and Wigderson [101] showed that extracting any partial information about the $O(\log|p|)$ most significant bits of x, given $g^x(\bmod\ p)$, even with a tiny advantage over simple guessing, is equivalent to computing logarithms modulo p. The bitwise complexity of the discrete logarithm modulo n, where n is a Blum integer, is discussed in more detail in §6.

The best known algorithms for computing discrete logarithms in a finite field $\mathrm{GF}(q)$ have complexity

$$\exp\{(c + o(1))(\ln q\ \ln\ln q)^{1/2}\}$$

with various constants c. A survey of such algorithms can be found in [118].

For a field $\mathrm{GF}(p)$, where p is prime, Coppersmith et al. [33] suggested an algorithm for computing discrete logarithms, with complexity

$$\exp\{(1 + o(1))(\ln p\ \ln\ln p)^{1/2}\}.$$

The record is held by the Coppersmith algorithm [32] for a field $\mathrm{GF}(2^n)$ that has complexity

$$\exp\{cn^{1/3}(\ln n)^{2/3}\},$$

where c is a certain constant. However, it should be noted that this estimate was derived using some unproved assumptions.

§5. Tasks of Complexity Theory in Cryptography

Cryptographers are interested in complexity theory because of the wide adoption of the following thesis: a cryptosystem security measure is the time (measured in steps executed by a Turing machine), required for cracking this cryptosystem by a cryptoanalyst. This thesis appears to be quite natural and allows one to state cryptographical problems in terms of the most common and thoroughly studied complexity measure. Cryptography immediately proposes one such problem "in a ready-made form": determine the complexity of the cracking problem for a given cryptosystem. However, the current state of complexity theory does not allow one to obtain nontrivial lower bounds for the complexity of particular problems. Thus, we may state that the main task of complexity theory in cryptography is to answer the following question: do there exist cryptosystems with provably high cracking complexity?

We may outline the following major approaches:

—seeking necessary and sufficient conditions for the existence of provably secure cryptosystems; studying the properties of related objects;

—determining reducibilities among cryptosystems. This approach is borrowed from complexity theory itself: we cannot prove a high lower bound on the complexity of a particular problem, but we may prove, for instance, its NP-completeness, which means that this problem is in some, rigorously formalizable, sense not less complex than any other problem in NP;

—studying the methods of designing provably secure cryptosystems under the assumption that objects with the required properties do exist (and are "available").

Below we consider these approaches in more detail.

§6. One-Way Functions

A central concept in the field of cryptographic applications of complexity theory is that of a *one-way function*, which is usually considered to be a poly-time computable function f such that its inverting problem, that is, solving equation $f(x) = y$ in x, is undecidable in polynomial time.

Let $\Sigma = \{0,1\}$. PSV denotes the class of all single-valued functions $f : \Sigma^* \to \Sigma^*$, each having a poly-time algorithm that computes f on all inputs $x \in \mathrm{dom}(f)$.

Definition. *A function f is honest (or polynomially honest) if there is a polynomial P such that $P(|f(x)|) \geq |x|$ for all $x \in \mathrm{dom}(f)$, where $|y|$ denotes the length of a string y.*

Definition. *An honest function $f \in$ PSV is one-way if there exists no poly-time algorithm that, for each $y \in \mathrm{range}(f)$, computes a value $x \in \mathrm{dom}(f)$ such that $f(x) = y$.*

Note that in the definitions of the one-way function encountered in the literature, the honesty requirement is often omitted, but assumed implicitly since otherwise any function $f(x)$ such that $|x| >> |f(x)|$ would be one-way.

Let UP be the class of all languages accepted in polynomial time by nondeterministic Turing machines which on each input have at most one accepting computation. It is evident that UP $\subseteq$ NP, thus P $\neq$ UP trivially implies P $\neq$ NP. For any complexity class C, coC $= \{L : \bar{L} \in \mathrm{C}\}$, where $\bar{L}$ denotes the complement of a language L (considered as a set of strings).

Assertion. 1. *A one-way function exists iff* P $\neq$ NP.

2. *A one-way one-to-one function exists iff* P $\neq$ UP *[135].*

3. *A one-way one-to-one surjective function exists iff* P $\neq$ UP $\cap$ coUP.

It should be emphasized that from the complexity-theoretic standpoint the one-way function is a hypothetical object, while in cryptographic literature the term "one-way function" is used sometimes to denote real number-theoretic functions with hard-to-invert hypotheses based on assumptions that appear to be stronger than P $\neq$ NP (e.g., on the assumption that there is no polynomial-time algorithm for integer factorization).

In the following discussion, unless otherwise noted, one-way functions are assumed to be one-to-one.

In connection with the problem of the existence of one-way functions, one should note the Joseph–Young conjecture [84, 168]: if one-way functions do exist, then there are NP-complete sets, which are not polynomially isomorphic. The following result of Rackoff is also of interest [126]; he constructed an oracle A such that $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$, that is, in his relativized world it is true that P $\neq$ NP, but there are no one-way functions.

The above definition can be satisfied by any function that is hard to invert only on some infinite sequence of values. It is clear that such functions are of no practical value for cryptography. In addition, one may attack the problem of inverting functions not only with deterministic, but also with probabilistic algorithms.

Definition [74]. *An honest function $f \in$ PSV is strongly one-way if for any algorithm A which runs in random polynomial time and any polynomial Q the following is true. Given input $f(x)$, where x is chosen uniformly among strings of length n, the probability that A outputs a y such that $f(y) = f(x)$ is bounded by $1/Q(n)$ for sufficiently large n.*

In this definition the function f is not assumed to be necessarily one-to-one.

Thus, a strongly one-way function is not invertible in polynomial time on all but less than polynomial fraction of argument values.

Goldreich et al. [59] demonstrated that, with some restrictions, functions that are hard to invert almost everywhere, can be constructed from functions that are hard to invert only on some, relatively small, fraction of argument values.

Definition [59]. *A polynomial-time computable function f is $\alpha(n)$-one-way with security $s(n)$ if every randomized algorithm inverts f in time $t(|x|)|x|^{O(1)}$ on fraction $< 1 - \alpha + t/s$ of inputs $f(x)$ and internal coin flips.*

Functions $s(n)$ and $s(n)^{\theta(1)}$ are called comparable.

Theorem [59]. *For any $n^{-O(1)}$-one-way permutation f there exists a 1/2-one-way permutation F with the same security.*

Theorem [59]. *For any 1/2-one-way permutation f with security $s(n) = s'(n + 10 \log s(n))$ there exists a 1-one-way permutation with security $s'(n)$.*

In the following definition and theorem a.e. means for all except $< 1/s$ fraction of instances.

Definition [59]. *A one-way function f is regular if there is a polynomial-time computable function $m(x)$ such that $|f^{-1}(f(x))| \leq 2^{m(x)}$ a.e. and $|f^{-1}(f(x))| \geq 2^{m(x)}/s(|x|)^{o(1)}$ on a polynomial fraction of hard-instances.*

It should be noted that this definition of regularity is rather weak: usually a function is called regular if all its values have the same number of pre-images.

Theorem [59]. *If a regular $n^{-O(1)}$-one-way function of security $s(n) = s'(n + \theta(\log(s(n)))^2)$ exists, then there is a 1-one-way a.e. one-to-one function of security comparable to s'.*

In the nonuniform model of computation a one-way function is defined as follows [83, 134]: f is one-way if it is computable by a family of polynomial-size Boolean circuits but is invertible by no family of polynomial-size Boolean circuits.

Goldreich and Levin [63] proved that for any strongly one-way function $f(x)$ there exists a *hard-core predicate*. For each value $f(x)$ and each Boolean vector p such that $|p| = |x|$ this predicate is defined as a Boolean scalar product $B(x, p)$. For any given function s the following conditions are proved to be equivalent:

(1) There exists a probabilistic algorithm, which in time $s(|x|)^{O(1)}$ inverts function f on at least a fraction $s(|x|)^{-O(1)}$ of values.

(2) There exists a probabilistic algorithm which, given $f(x)$ and p, computes $B(x,p)$ in time $s(|x|)^{O(1)}$ with probability at least $1/2 + s(|x|)^{-O(1)}$.

Among the properties of one-way functions, the *bitwise hardness* has been studied rather thoroughly.

Definition [134]. *A bit of an argument x of the function f is said to be hard if no family of polynomial-size Boolean circuits can, given the value $f(x)$, compute this bit with probability of success at least $1/2 + 1/p(n)$ for any polynomial p.*

In this definition, as in [22], the bit hardness is defined directly.

In some other papers (e.g., in [101]) the bit is defined to be hard if its computation is as hard as inverting f. Goldreich and Levin [63] proved that any one-way function has at least a logarithmic number of hard (under the above definition) bits. It is unlikely that this result could be extended since a function may depend essentially on only a small fraction of bits and still be one-way. There is a number of results on the bitwise hardness of certain number-theoretic functions. Schrift and Shamir [134] have studied the bitwise hardness of discrete logarithm modulo composite in the nonuniform model of computation.

Definition. *A number $n = pq$, where p and q are primes of the same length and $p = q = 3(\text{mod } 4)$, is called a Blum integer.*

Blum integers find various applications in cryptography (see, e.g., [15, 23, 116]).

Consider a function $f_{g,N}(x) = g^x(\text{mod} N)$, where N is a Blum integer. A triplet (g, N, y) is said to be admissible if g is a quadratic residue $(\text{mod} N)$ and there exists an x such that $f_{g,N}(x) = y$. If the problem of inverting $f_{g,N}$ is considered as the problem of finding the argument x given the admissible triplet, then this function is one-way under the assumption that no family of polynomial-size Boolean circuits can factor a polynomial fraction of Blum integers. Note that this assumption appears to be stronger than P $\neq$ NP: its validity is assumed in the statements of the two following theorems.

Theorem [134]. *For every $i, 1 \leq i \leq (1-\varepsilon)n$, with ε an arbitrarily small constant, the ith bit of $f_{g,N}$ is hard.*

Thus, almost all bits of the discrete logarithm are hard. It was also noted in [134] that Håstad proved in an unpublished paper the hardness of the ith bit of the discrete logarithm for every

$$i: \; n/2 \leq i \leq n - O(\log n).$$

The $O(\log n)$ most significant bits of the discrete logarithm are known to be *biased* and therefore they can be predicted with probability of success significantly greater than $1/2$. Schrift and Shamir [134] introduced a special definition of hardness for biased bits and proved that under this definition $O(\log n)$

most significant bits of the discrete logarithm are hard. Thus, we can state that all n bits of the discrete logarithm are hard.

Let P_j^k be a function that, given a bit string, picks out the bits from jth to kth.

Definition [134]. *The bits of an argument x from jth to kth are simultaneously hard if no family of polynomial-size Boolean circuits can distinguish $(P_j^k(f_{g,N}^{-1}(y)), y)$ from (x_j^k, y) with probability at least $1/p(n)$ for any polynomial p, where (g, N, y) is a randomly chosen admissible triplet and x_j^k is a $(k-j+1)$-bit random string.*

Theorem [134]. *The $n/2$ right-hand bits of the discrete logarithm are simultaneously hard.*

In the above-mentioned paper Håstad proved that $n/2$ left-hand bits of the discrete logarithm are also simultaneously hard.

One can estimate these results from different points of view. On the one hand, we may take that of Schrift and Shamir, stated in the title of their paper [134]. But, on the other hand, we may treat such strong results as evidence against the truth of the assumption from which they were inferred.

Among all the papers on one-way functions, Homer and Wang [79] stand apart. In this paper the notion of the one-way function was somewhat weakened, but, on the positive side, absolute results (which depend on no unproved assumptions) were proved.

Let $\mathrm{P}^k(\mathrm{NP}^k)$ be the class of languages, accepted by deterministic (nondeterministic) Turing machines, that run in time $O(n^k)$; UP^k is defined similarly. It is evident that $\mathrm{P} = \bigcup_k \mathrm{P}^k$, $\mathrm{NP} = \bigcup_k \mathrm{NP}^k$, $\mathrm{UP} = \bigcup_k \mathrm{UP}^k$. Further, let PSV^k denote the set of all partial single-valued functions computed by deterministic Turing machines which run in time $O(n^k)$.

Definition [79]. *A polynomially honest function f is k-honest if there is a polynomial p of degree k such that for all $x \in \mathrm{dom}(f)$, $p(|f(x)|) \geq |x|$.*

A set S is k-sparse if the number of words of length n in it is bounded by a polynomial of degree k. S is polynomially sparse if it is k-sparse for some k.

Definition [79]. *Function f is k-one-way if f is an element of PSV^2, f is one-to-one and polynomially honest, and $f^{-1} \notin \mathrm{PSV}^k$.*

Thus, the notion of the k-one-way function is a natural "weakening" of the notion of the one-way function: the requirement that f be poly-time computable is replaced by the requirement of its computability in quadratic time, and the requirement that f^{-1} be noncomputable in polynomial time is replaced by that of noncomputability in time $O(n^k)$.

Theorem [79]. *For any $k > 0$ there is a function f such that the following conditions hold simultaneously: f is a $(2k-1)$-one-way, k-honest, surjective function and $\mathrm{dom}(f)$ is not polynomially sparse.*

This function can be made total if the surjectivity requirement is dropped.

The function constructed in this theorem may turn out to be hard to invert only on some infinite sequence of argument values.

Definition [79]. *A one-to-one function f is strongly k-hard to invert if for every function $g \in \mathrm{PSV}^k$ the set $\{x : f^{-1}(x) = g(x)\}$ is finite.*

Theorem [79]. *For any $k > 0$ there is a function f such that the following conditions hold simultaneously: f is a $(2k-1)$-one-way, k-honest, strongly $(2k-1)$-hard to invert, surjective function and $\mathrm{dom}(f)$ is not polynomially sparse.*

These results are near optimal as follows.

Theorem [79]. *Given $k \geq j > 0$. If there is a k-one-way function with j-honesty, then* $\mathrm{UP}^{2j} - \mathrm{P}^{k-j} \neq \varnothing$.

Corollary [79]. *If for some k there is a $3k$-one-way k-honest function, then* $\mathrm{UP}^{2k} \neq \mathrm{P}^{2k}$.

Thus, the constructed k-honest $(2k-1)$-one-way function can be "improved" by similar techniques at most to $(3k-1)$-one-way. A further advantage in this direction would require the development of essentially more powerful techniques that allow separation of certain subclasses of the class UP from certain subclasses of the class P. Note that the very possibility of such a separation is currently an open problem. (It is clear that if $\mathrm{P} \neq \mathrm{UP}$, then for some j, $\mathrm{UP}^j - \mathrm{P} \neq \varnothing$ and, hence, $\mathrm{UP}^j - \mathrm{P}^k \neq \varnothing$ for all k.)

So far the time required to compute the function f^{-1} has been evaluated as a function of the length of $f(x)$. If this time is evaluated as a function of the length of x, then for the constructed functions it is quadratic. The following question comes up naturally: could this second-degree polynomial be replaced by a polynomial of degree k?

Definition [79]. *A function f is strongly k-one-way if f is k-one-way and f^{-1} cannot be computed in time $O(|x|^k)$.*

Definition [79]. *A function f is exactly k-honest if there exist constants $C_0 > 0$ and $C_1 > 0$ such that for all $x \in \mathrm{dom}(f)$, $C_0|x| \geq |f(x)|^k \geq C_1|x|$.*

Theorem [79]. *Given $k \geq j > 0$. If there is a function $f \in \mathrm{PSV}^1$ such that f is strongly k-one-way with exact j-honesty, then* $\mathrm{UP}^{2j} - \mathrm{P}^{j(k-1)} \neq \varnothing$.

Corollary [79]. *If there is a strongly 2-one-way function $f \in \mathrm{PSV}^1$ with exact 1-honesty, then* $\mathrm{UP}^1 \neq \mathrm{P}^1$.

Thus, only the following question remains open: is it possible, without separation of subclasses UP^1 and P^1, to construct a one-to-one function f computable in linear time such that the computation of $f^{-1}(y) = x$ requires time $O(|x|^2)$.

These results were used in constructing a public-key cryptosystem. In the notation of §3, the cracking problem for such a cryptosystem is the problem of finding an efficient algorithm which computes a (partial) extension of the partial function $Crack$, where $Crack(n, K_2, y) = x \iff |x| = n$, and $E(K_2, x) = y$.

Definition [79]. *A public-key cryptosystem is k-secure if G, E, and D are in PSV^2 and there is a $j \leq k$ such that E is j-honest, but no extension of $Crack$ is in PSV^k.*

For any $k > 0$ the existence of a k-secure cryptosystem is proved. Though this result uses the traditional complexity measure—worst-case complexity—and the whole construction is based on the set built by the diagonal technique, it should be emphasized that this result is absolute, that is, it depends on no unproved assumption.

It is pointed out that in a similar way a cryptosystem can be constructed which has an encoding function with range(f) not polynomially sparse, and for any function $g \in \mathrm{PSV}^k$ the set $\{x : g(x) = f^{-1}(x)\}$ is finite.

As an example of the application of one-way functions in cryptography, let us consider a digital signature scheme. Such a scheme is defined [129] by the following components:

a message space Σ^k, where $\Sigma = \{0, 1\}$;

a probabilistic poly-time key generation algorithm which on input 1^k computes public key P_k and a matching secret key S_k;

a probabilistic poly-time algorithm, which, given a message $M \in \Sigma^k$ and a pair of keys P_k and S_k, computes a signature s;

a poly-time algorithm, which, given s, M, and P_k, verifies whether s is a valid signature of M with respect to P_k.

The security of a digital signature scheme against the chosen message attack is defined—it is assumed that the adversary can choose adaptively a polynomial number of messages and get their signatures. The scheme is secure against the chosen message attack if, for any polynomial p, the adversary, after this attack, can produce a forged message in polynomial time with success probability less than $1/p(k)$.

Lamport [93] proposed a digital signature scheme, secure against the chosen message attack, in which the public file contains a one-way function f and some values

$$\alpha_1^0, \alpha_1^1, \alpha_2^0, \alpha_2^1, \ldots, \alpha_m^0, \alpha_m^1.$$

To sign a bit b_i the sender transmits a value $f^{-1}(\alpha_i^{b_i})$. However, this scheme allows only m bits to be transmitted. Naor and Yung [115] overcame this disadvantage with the aid of hash-functions.

A *k-universal family of hash-functions* [26] is a family $\{h_i : \Sigma^m \to \Sigma^l\}$ of poly-time computable functions such that

(1) $P\{h_i(x_1) = y_1, \ldots, h_i(x_k) = y_k\} = 2^{-kl}$ for any $x_1, \ldots, x_k \in \Sigma^m$ and $y_1, \ldots, y_k \in \Sigma^l$;

(2) there exists a probabilistic poly-time algorithm, which for given $j \leq k$ and any $x_1, \ldots, x_j \in \Sigma^m$ and $y_1, \ldots, y_j \in \Sigma^l$ uniformly samples h_i such that $h_i(x_1) = y_1, \ldots, h_i(x_j) = y_j$.

A *family of one-way hash functions* [115] is such that no collection of polynomial-size Boolean circuits exists, each of which first outputs some value x and then, for a function f randomly selected from this family, computes $x' \neq x$ such that $f(x') = f(x)$.

A modification of Lamport's scheme, introduced in [115], consists in transmitting at each stage a new set of values

$$\alpha_1^0, \alpha_1^1, \alpha_2^0, \alpha_2^1, \ldots, \alpha_m^0, \alpha_m^1,$$

and the signature for the value of the hash function on this set. Provided the family of one-way hash functions exists, this scheme is secure. In [115] a family of one-way hash functions was constructed from a one-way permutation.

A decisive solution of this problem was obtained by Rompel [129]. He constructed a family of one-way hash functions based on the assumption that a (strongly) one-way function exists. As a corollary, it was shown that provably secure digital signature schemes exist iff one-way functions exist.

One special class of one-way functions should be mentioned, which is studied mainly in theoretical cryptography. These are the so-called *trapdoor functions* [166]. A trapdoor function is a one-way function which can, nevertheless, be easily inverted if some additional information, called a *trapdoor*, is known.

§7. Pseudorandom Generators

The emergence in complexity theory of such a concept as a *pseudorandom generator*, or, more completely and precisely, *pseudorandom bit generator*, has several motivations. One of them is due to the Shannon theorem on perfectly secure encryption with random one-time pad (see §1). Since it is impossible to produce random strings in practice, algorithms are considered, which from a given fixed-length seed (bit string) generate longer sequences having some properties similar to the properties of random strings. The sequence is pseudorandom if no probabilistic poly-time algorithm can distinguish it from a random one. More formally:

Definition. *Let A be a probabilistic poly-time algorithm. Let g be a poly-time algorithm that takes a seed of length n as input and produces a sequence $g(s)$ of length n^k. Further, let P be the probability that A accepts x, where x is a random string of length n^k, and Q be the probability that A accepts $g(s)$, where s is a seed, chosen randomly from the set of all strings of length n. We say that g is a (secure) pseudorandom bit generator if for any polynomial p, for any algorithm A, and for all sufficiently large n we have $|P - Q| < 1/p(n)$.*

It seems that for specialists in cryptography the following definition of pseudorandom generator would be more natural and clear: g is a pseudorandom bit generator if for any polynomial $p(n)$ no probabilistic poly-time algorithm, given the initial segment of the sequence generated by g, can predict the next bit with probability at least $1/2 + 1/p(n)$. Yao [166] proved the equivalence of the above two definitions of pseudorandom bit generators.

The existence of pseudorandom generators was first discovered by Blum and Micali in 1982 [22]. They proved the existence of such generators under the assumption of infeasibility of the discrete logarithm problem. Further investigations were aimed at seeking more and more weak assumptions, sufficient for the existence of pseudorandom generators.

Yao [166] constructed a pseudorandom generator under the assumption that one-way permutations exist. Levin [99] proved that the existence of a function which is one-way on its iterates is necessary and sufficient for the existence of pseudorandom generators. Goldreich et al. [62] constructed such a generator under the assumption that regular one-way functions exist.

Impagliazzo et al. [83] considered the problem in the nonuniform model of computation. In this model a generator is secure if it produces a pseudorandom sequence which is indistinguishable from the random sequence by any family of polynomial-size Boolean circuits. They proved that the existence of a one-way function in the nonuniform model implies the existence of a pseudorandom generator.

Finally, Håstad [74] proved an analogous result in the uniform model of computation.

Theorem [74]. *A secure pseudorandom generator exists iff a strongly one-way function exists.*

The construction proposed in the proof of this theorem is inefficient in the following sense: n-bit input values of a one-way function have the corresponding seeds of a pseudorandom generator, which are of length n^8. A paper of four authors (Impagliazzo, Levin, Luby, and Håstad) has been announced, in which this inefficiency will be removed.

Allender [3], citing an unpublished paper of Boppana and Hirschfield, gives the definition of a ***pseudo-random extender***, which is a pseudorandom generator that, given the seed of length n, produces a sequence of length $n + 1$. The following result, also due to Boppana and Hirschfield, is formulated: pseudorandom generators exist iff pseudorandom extenders exist.

Allender [3] considered stronger requirements, imposed on the pseudorandom sequences. Generator g passes a $(T(n), e(n))$-test if no probabilistic algorithm that runs in time $T(n)$ can distinguish the sequence generated by g from a random sequence with probability at least $e(n)$. From the assumption that very secure generators exist, a series of results is inferred, concerning the relationships among complexity classes. For instance, subexponential functions were considered as $T(n)$ both in uniform and nonuniform models of computation.

The concept of a pseudorandom bit generator is closely related with that of a ***pseudorandom function generator***. Let $F^n = \{f : \{0,1\}^n \to \{0,1\}^n\}$ and $l(n)$ be a certain polynomial. A ***function generator*** is a collection $\{f^n\}$, where f^n specifies for each $k \in \{0,1\}^{l(n)}$ a function $f_k^n \in F^n$. In addition, it is required that given k and $\alpha \in \{0,1\}^n$, $f_k^n(\alpha)$ can be computed in time polynomial in n. The string $k \in \{0,1\}^{l(n)}$ is called a key.

Intuitively, a generator is pseudorandom if it passes a "black-box" test, suggested essentially by Turing [78].

Definition [103]. *A function generator is pseudorandom if for any polynomial $p(n)$ there is no poly-time algorithm which, given a polynomial number of pairs of input and output values of the function, can distinguish a function randomly selected from F^n from the function f_k^n for a randomly selected key k, with probability at least $1/p(n)$. The algorithm is allowed to choose the input values adaptively, based on the analysis of previously seen input and output values.*

Goldreich et al. [60] proved that pseudorandom function generators exist iff pseudorandom bit generators exist.

If in the above definition the set of all functions f is replaced by the set of all permutations, we get the definition of a ***pseudorandom permutation generator***. Luby and Rackoff [103] proved that pseudorandom permutation generators exist iff pseudorandom function generators exist.

Consequently, a chain of equivalences was deduced for the assumptions that the following objects exist: pseudorandom permutation generators, pseudorandom function generators, pseudorandom bit generators, (strongly) one-way functions.

The paper of Luby and Rackoff [103] is of particular interest because for the construction of a pseudorandom permutation generator from a pseudorandom function generator they used a modified version of the well-known private-key block cipher—DES (Data Encryption Standard). Since DES was described in §2, here we only briefly outline the main differences of the modified version. First, a plaintext is divided into blocks of length $2n$. Given the key K of length $l(n)$, let f_K^n be a function produced by a pseudorandom function generator. Let $g(L \cdot R) = R \cdot [L \oplus f_K^n(R)]$, where the dot denotes concatenation, and L and R are the left and right halves of the plaintext block. Second, in the modified version it is assumed that K is a part of a full key of sufficient length and this key was selected randomly from the set of all strings of this length, not generated as in DES.

It is proved that $h = g \circ g \circ g$ is a pseudorandom permutation generator, while $g \circ g$ is not such a generator. As a corollary, the above cryptosystem with encryption function h is shown to be secure against a chosen-plaintext attack. Moreover, the same cryptosystem with encryption function $g \circ g \circ g \circ g$ is secure against a chosen-text attack. It is clear that this result says nothing about the security of DES itself, but serves, to a certain degree, as a theoretical justification of the ideas underlying its construction.

§8. Probabilistic Encryption

So far we have considered cryptosystems with a deterministic encryption algorithm, that is, given a fixed key, the ciphertext was always defined uniquely for any plaintext. Goldwasser and Micali [66] proposed an essentially different approach. They designed a public-key cryptosystem, which they called a *probabilistic encryption system.*

Probabilistic encryption systems may be considered as a marginal case of block encryption—they apply an encryption algorithm either to each plaintext bit separately, or to blocks consisting of a few bits. In the next discussion, following Goldwasser and Micali, we suppose that an encryption algorithm is applied to each bit separately.

A plaintext bit b is represented in ciphertext by a random string y such that $B(y) = b$, where B is a specific predicate. Thus, each plaintext has a set (generally, of exponential cardinality) of corresponding ciphertexts. The predicate B, however, is chosen in such a way that always permits a unique decryption.

The construction proposed in [66] is based on the notion of an *unapproximable trapdoor predicate.*

Definition [66]. *A Boolean circuit C ε-approximates the predicate $B : \Omega \to \{0,1\}$ if $C[y] = B[y]$ for at least a fraction $1/2 + \varepsilon$ of the values $y \in \Omega$.*

Let N' be an infinite set of integers. For any $k \in N'$, S_k denotes a subset of k-bit integers and for any $i \in S_k$ let Ω_i be a subset of integers with length at most k. Let $B_k = \{b_i : \Omega_i \to \{0,1\}, i \in S_k\}$ and $B = \bigcup_{k \in N'} B_k$.

Definition [66]. *B is an unapproximable trapdoor predicate if:*

(1) *Let p_1 and p_2 be any polynomials and $k \in N'$. Let c_k be the size of the minimal Boolean circuit $C[\cdot,\cdot]$ such that $C[\cdot, i]$ $(1/p_1(k))$-approximates b_i for at least $(1/p_2(k))$-fraction of the values $i \in S_k$. Then c_k grows faster than any polynomial in k.*

(2) *Let $\Omega_i^v = \{y \in \Omega_i : b_i(y) = v\}$ with $v \in \{0,1\}$.*

(2.1) *There exists a probabilistic Turing machine with running time bounded by a polynomial in k, which on input (i, v), where $i \in S_k$ and $v \in \{0,1\}$, selects $y \in \Omega_i^v$ with uniform probability.*

(2.2) *There exists a function $\sigma : \bigcup_{k \in N'} S_k \to N$ such that for a certain polynomial Q and any y $|\sigma(y)| < Q(|y|)$ and there exists a polynomial-time Turing machine which on input $(i, \sigma(i), y)$ computes $b_i(y)$ for any $i \in S_k$ and any $y \in \Omega_i$.*

(2.3) *For any $k \in N'$ it is possible in probabilistic polynomial (in k) time to select any pair $(i \in S_k, \sigma(i))$ with probability $1/|S_k|$.*

A message generator is a probabilistic polynomial-time Turing machine MG that takes a number k in unary as input and outputs a string referred to as a message.

Definition [66]. *A public-key cryptosystem is a probabilistic polynomial time Turing machine Π which on input (k, MG) outputs the description of two algorithms E and D such that:*

(1) *for some constants c, on inputs of length n the algorithms E and D halt within n^c steps;*

(2) *for all* $x \in \mathrm{MG}[k]$, $D(E(x)) = x$.

The encryption algorithm E may be probabilistic. A probabilistic public-key cryptosystem is a public-key cryptosystem Π which on input (k, MG) produces a pair $(i, \sigma(i))$, where $i \in S_k$. The number $i \in S_k$ specifies an encryption algorithm as follows: any given l-bit message $x = x_1, \ldots, x_l$ is represented by ciphertext $y_1, \ldots, y_n$, where for each j, y_j is a randomly selected element of $\Omega_i^{x_j}$. The value $\sigma(i)$ specifies a decryption algorithm: item 2.2 of the unapproximable trapdoor predicate definition ensures that for each j the value $b_i(y_j) = x_j$ can be computed efficiently, given the triplet $(i, \sigma(i), y_j)$.

For probabilistic cryptosystems a new stronger notion of security is defined. Informally, a public-key cryptosystem is *polynomially secure* if for any message space M with any probability distribution no polynomially bounded probabilistic algorithm F can find two messages whose encryptions are distinguishable by the polynomially bounded algorithm T.

More formally, let T_k be a polynomial-size Boolean circuit that takes (E, y) as input, where $y \in E(x)$, $E \in \Pi(k, \mathrm{MG})$, $x \in \mathrm{MG}[k]$. Let P_r^E be the probability with which T_k outputs 1 on inputs $E \in \Pi(k, \mathrm{MG})$, $y \in E(x_r)$, where $r = 0, 1$. The circuit T_k p-distinguishes x_0 from x_1 with respect to E if $|P_1^E - P_0^E| > 1/p(k)$. Let F_k be a Boolean circuit which on input E outputs two messages $x_1, x_2 \in \mathrm{MG}[k]$.

Definition [66]. *Let Q, p_1, p_2 be polynomials. Let s_k^T be the size of a minimal Boolean circuit F_k which with probability at least $1/p_1(k)$ on input E produces two messages x_0, x_1 that are p_2-distinguishable by the circuit T_k. Cryptosystem Π is polynomially secure with respect to* MG *if for any sequence $\{T_k\}$, s_k^T grows faster than any polynomial in k. Π is polynomially secure if it is polynomially secure with respect to any message generator.*

Theorem [66]. *Each probabilistic public-key cryptosystem is polynomially secure.*

This is true for cryptosystems constructed as above with unapproximable trapdoor predicates. Yao proved that unapproximable trapdoor predicates exist if one-to-one trapdoor functions exist.

Definition [66]. *Let, for any $x \in \mathrm{MG}[k]$, P_x denote the probability that* MG *will output x on input 1^k. Let $f_{\mathrm{MG}} = \{f_E : \mathrm{MG}[k] \to V;\ E \in \Pi(k, \mathrm{MG})\}$. For each $E \in \Pi(k, \mathrm{MG})$ let $P_E = \max\limits_{v \in V} \left\{ \sum\limits_{x \in f_E^{-1}(v)} P_x \right\}$.*

Let C be a Boolean circuit which on input $E \in \Pi(k, \mathrm{MG})$ and $y \in E(x)$, where $x \in \mathrm{MG}[k]$, outputs a string α. Let p, Q be polynomials. We say that the circuit C (p, Q, k)-computes f_{MG} if $P\{\alpha = f_E(x) : x \in \mathrm{MG}[k],\ y \in E(x)\} > P_E + 1/Q(k)$ for all E belonging to a subset $S \subseteq \Pi(k, \mathrm{MG})$ having probability at least $1/P(k)$. Let $c_k^{Q,p}$ denote the size of a minimal circuit C that (p, Q, k)-computes f_{MG}. A cryptosystem Π is *semantically secure* if for any message generator MG, for all f_{MG}, for all p and Q, $c_k^{Q,p}$ grows faster than any polynomial in k.

Theorem [66]. *Each polynomially secure public-key cryptosystem is semantically secure.*

Micali et. al. [112] proved the inverse implication: if a cryptosystem is semantically secure, then it is also polynomially secure. Goldreich [58] showed the validity of an analogous result for the uniform model of computation. Thus, the notions of polynomial security and semantic security are equivalent.

The probabilistic encryption system essentially differs from a common cryptosystem in that the former is secure for any message space with any probability distribution. In the traditional approach, the proofs of cryptosystem security essentially depend on the assumption that the message space is "dense" in the set of all strings and that the probability distribution is uniform.

Semantic security is essentially a polynomially bounded analog of Shannon's perfect security: from the point of view of the polynomially bounded adversary, the a posteriori probabilities of various plaintexts, given the intercepted cryptogram, are the same as the a priori probabilities of the same plaintexts.

Goldwasser and Micali [66] suggested a particular implementation of the probabilistic encryption cryptosystem. Let $k \in N$, p_1 and p_2 be primes. Let

$$H_k = \{n : n = p_1 p_2, \text{ where } |p_1| = |p_2| = k\},$$
$$Z_n^* = \{x \le n : (x, n) = 1\}.$$

Let Z_n^1 denote the subset of Z_n^* consisting of the elements with Jacobi symbol +1. For all $x \in Z_n^1$

$$Q_n(x) = 0 \text{ if } x \text{ is a quadratic residue mod } n,$$
$$Q_n(x) = 1 \text{ if } x \text{ is a quadratic nonresidue mod } n.$$

Let $S_{4k} = \{n\#y : n \in H_k,\ y \in Z_n^1$ be a quadratic nonresidue mod $n\}$. The symbol # denotes here the concatenation of strings. Let $\Omega_{n\#y} = Z_n^1$ and $Q_{n\#y}(x) = Q_n(x)$ for all $x \in Z_n^1$.

Under the assumption that no family of polynomial-size Boolean circuits can compute $Q_n(x)$ for all $x \in Z_n^1$ on at least a polynomial fraction of $n \in H_k$, it is proved that $\{Q_{n\#y} : n\#y \in S_{4k}\}$ is an unapproximable trapdoor predicate.

In the cryptosystem being considered, the pair (n, y) is a public key and the pair (p_1, p_2) is a private key. Bit 1 is represented in ciphertext by the value $yx^2(\text{mod} n)$, and bit 0 by the value $x^2(\text{mod} n)$, where x is a randomly selected element of Z_n^1.

Since the message receiver knows p_1 and p_2, he can decrypt the message by computing Q_n.

For another example of a particular probabilistic cryptosystem, suggested by Kari, see §9.

Naor and Yung [116] used probabilistic encryption to construct a public-key cryptosystem, provably secure against a chosen-ciphertext attack. They also note that this result can be strengthened by constructing a public-key cryptosystem secure against the following very strong attack, suggested in an unpublished paper of Rackoff and Simon. The adversary first gets the ciphertext he is interested in cracking. Then he can adaptively choose a polynomial number of ciphertexts and get the corresponding plaintexts. The only restriction is that he is not allowed to include in this number exactly the same ciphertext he wants to crack.

§9. Cryptographic Security and NP-Completeness

The emergence of NP-completeness theory raised among cryptographers the hopes of designing cryptosystems with NP-complete cracking problems. However, further research showed that these hopes were unfounded. First, no cryptosystem was found with the NP-hard cracking problem. Second, as Selman notes, the requirement of NP-hardness is, from the cryptographic point of view, simultaneously too strong and too weak. Too strong because the cryptosystem cracking problem must be hard, but this does not mean that it must be NP-hard. Too weak, since the cryptosystem cracking problem must be hard "almost everywhere," while NP-hardness guarantees (hypothetically!) the polynomial-time infeasibility of the problem only "infinitely often."

Anyhow, let us note that, for any sound definition of security, secure cryptosystems exist only if cryptosystems with an "infinitely often" hard cracking problem exist. Thus, the investigation of NP-completeness issues in cryptography is of undoubted theoretical importance.

Even et al. [49] studied these issues for public-key cryptosystems. For a further discussion we need some complexity-theoretic concepts defined by Even and Yacobi in [50].

Definition [50]. *A promise problem is the following structure:*

$$\begin{array}{ll} \textit{input} & x, \\ \textit{promise} & Q(x), \\ \textit{property} & R(x), \end{array}$$

where Q and R are predicates.

A promise problem is denoted by (Q, R). A deterministic Turing machine M solves the promise problem (Q, R) if $\forall x[Q(x) \Rightarrow [M(x)\downarrow \&[M(x) = \text{"yes"} \Leftrightarrow R(x)]]]$, where the notation $M(x)\downarrow$ means that M halts on input x.

If such a machine M exists, then the problem (Q, R) is called solvable and the language $L(M)$ is its solution.

Each promise problem (Q, R) with recursive predicates Q and R is solvable: $Q \cap R$ is the minimal solution, R is a solution, and $(Q \cap R) \cup \bar{Q}$ is the maximal solution (here we consider sets denoted by the same letters as their corresponding predicates).

NPP is the class of all promise problems that have a solution in NP. coNPP is the class of all promise problems (Q, R) such that $(Q, \bar{R}) \in$ NPP.

Definition [50]. *A promise problem (Q, R) is* NP*-hard if each of its solutions is* NP*-hard.*

The problem of cracking a public-key cryptosystem is considered. A triplet (n, K_2, y) is *legal* if K_2 is a legal encryption key and there is a message x of length n such that $E(K_2, x) = y$.

Then the cracking problem can be stated as the following promise problem (CP):

input	n, K_2, y, x',
promise	(n, K_2, y), a legal triplet,
property	$x' \leq x$ such that $E(K_2, x) = y$.

This problem is proved to be polynomially equivalent to the computational form of the cracking problem in which the recognition of the above property is replaced by the requirement of finding a message x such that $E(K_2, x) = y$.

From the statement of the problem CP it is clear that its promise is in NP, and the problem itself is in NPP$\cap$coNPP.

The following conjecture was stated in [135].

There exists no promise problem (Q, R) such that

(1) Q $\in$NP,

(2) (Q, R) $\in$NPP$\cap$coNPP,

(3) (Q, R) is NP-hard.

If this conjecture is true, then there is no public-key cryptosystem with the NP-hard cracking problem. The supposed proof of this conjecture must be a rather nontrivial one since it is known to imply NP $\neq$ coNP and also NP $\neq$ UP.

A similar direction was studied also by Kari [88]. He considered a public-key probabilistic cryptosystem.

The public key of this cryptosystem consists of two statements p_0 and p_1 of propositional calculus. The secret decryption key is some truth assignment for the variables such that p_1 is true and p_0 is false on this assignment. The encryption of bit i is done by transformation of the statement p_i into an equivalent but differently looking statement p'_i. Decryption consists in computing the value of p_i on the secret truth assignment.

The cracking problem for this cryptosystem is proved to be NP$\cap$coNP-hard. Moreover, this cryptosystem is maximal in the sense that if there is a cracking oracle for this system, then any other cryptosystem can be cracked in polynomial time.

On the other hand, the real security of the proposed cryptosystem is not clear due to inadequacy of the "worst case" complexity measure for the needs of cryptography. For instance, the same result can be shown for the following degenerate cryptosystem: the public key consists of two statements p_k, $k = 0, 1$, of propositional calculus, one of which is everywhere false, while the other is satisfiable. The private key is equal to the index k of a satisfiable statement. The encryption of bit i, $i = 0, 1$, consists in choosing a random truth assignment for variables and computing p_i under this assignment. If the result is 1, then a special symbol $*$ is transmitted; otherwise the bit i is transmitted itself. The decryption algorithm simply replaces $*$ by k and leaves 0 and 1 unaltered. If p_k has many satisfying truth assignments, the adversary can determine, with sufficiently high probability, which of the statements p_0 or p_1 is satisfiable, and decide the meaning of $*$. If the number of satisfying truth assignments is low, then the symbol $*$ rarely occurs in ciphertext. In any case, it is evident that the cryptosystem is insecure.

§10. Zero-Knowledge Proofs

Interactive zero-knowledge proofs is a new and rapidly developing field of research, whose results are important for cryptography as well as for complexity theory. The coverage of this section is rather short and by no means complete. Unfortunately, we cannot point out any survey deserving attention in this field.

Informally, an *interactive proof system* for a language L is a protocol with two participating sides—P and V by which the prover (P) can "convince" a verifier (V) that an input string x, known to both participants, belongs to language L, if it indeed belongs, but cannot do so otherwise. As a rule, P is supposed to have unbounded computational power, while V has a probabilistic polynomial-time machine.

Alternatively, we may consider the input x as some statement whose truth the prover tries to demonstrate to the verifier.

An interactive proof is said to be zero-knowledge if the verifier can get from this proof no additional information about why x belongs to L (or why the statement is true). In other words, everything V can get from this protocol, he can compute himself in probabilistic polynomial time, provided he knows the input x and is sure that x belongs to L.

Let us give more precise definitions. Let P be a probabilistic Turing machine with unbounded resources (time and space), V be a probabilistic polynomial-time Turing machine, x be the common input. P and V interact, exchanging messages. Each of the machines, after the current message is sent, takes a transition to the wait-state, and leaves this when a new message is received. Such interaction is denoted by $P \longleftrightarrow V$. $P \longleftrightarrow V(x)$ accepts (input x) if V eventually stops in an accepting state.

Definition [53]. *P and V form an interactive proof protocol for a language L if:*
(1) *if $x \in L$, then $Pr\{P \longleftrightarrow V(x)$ accepts$\} > 1 - 1/p(|x|)$ for all polynomials p (completeness),*
(2) *if $x \notin L$, then $\forall P^* Pr\{P^* \longleftrightarrow V(x)$ accepts$\} < 1/p(|x|)$ for all polynomials p (soundness).*

Completeness means that if both P and V follow the protocol and $x \in L$, then the verifier will accept the proof with probability close to 1. Soundness means that if $x \notin L$, then the prover, using any algorithm P^*, cannot convince the verifier with nonnegligible probability that $x \in L$.

Let IP be the class of all languages that have interactive proofs. A *round of an interactive protocol* is a step consisting of a message from V to P, followed by a message from P to V.

Babai [4] proposed a somewhat different definition of an interactive protocol. He defined the class AM as the class of all languages that have one-round interactive proof protocols in which all of the verifier's messages consist exactly of his coin tosses. The corresponding protocol is called an AM-*protocol* (AM is an abbreviation for the "Arthur–Merlin" game). Babai proved [4] that any bounded round AM-protocol is equivalent to a one-round AM-protocol. He also showed that $\mathrm{AM} \subseteq \mathrm{NP}^R$ with probability 1 for a random oracle R and that $\mathrm{AM} \subseteq \Pi_2^P$, where Π_2^P is a second level class of polynomial-time hierarchy.

At first glance Babai's model may seem weaker than an interactive protocol, since the behavior of the verifier is severely restricted and he cannot keep his coin tosses private. However, Goldwasser and Sipser [68] have shown that for any language that has a Q-round interactive proof protocol, there is a $Q+2$-round AM-protocol.

Shamir [142] proved that IP=PSPACE (PSPACE is the class of all languages accepted by deterministic, polynomial-space-bounded Turing machines).

The precise formal definition of an interactive zero-knowledge proof is rather cumbersome (see, e.g., [11]). In addition, there are three different notions of zero-knowledge. We give simplified definitions.

$P \underset{\text{view}}{\longleftrightarrow} V(x)$ denotes the set of all messages from P to V and all coin tosses of V during the proof. Let $P \longleftrightarrow V[x]$ denote the probability distribution of $P \underset{\text{view}}{\longleftrightarrow} V(x)$ over the coin tosses of P.

An interactive proof $P \longleftrightarrow V$ is zero-knowledge if for any probabilistic polynomial-time Turing machine V^* there exists a probabilistic polynomial-time *simulating Turing machine* M_{V^*} which on input x produces a probability distribution on the set of messages and coin tosses of V^* (denoted $M_{V^*}[x]$), which is indistinguishable from $P \longleftrightarrow V^*[x]$.

The proof is called:

computational zero-knowledge if for any polynomial r no probabilistic polynomial-time Turing machine can distinguish a family of distributions $\{M_{V^*}[x], x \in L\}$ from the family $\{P \longleftrightarrow V^*[x], x \in L\}$ with probability at least $1/r(|x|)$;

statistical zero-knowledge if the families $\{M_{V^*}[x], x \in L\}$ and $\{P \longleftrightarrow V^*[x], x \in L\}$ are statistically indistinguishable, i.e., for any polynomial r, for all x of sufficient length, the difference of probabilities of any subset with respect to these distributions is less than $1/r(|x|)$;

perfect zero-knowledge if the families of distributions $\{P \longleftrightarrow V^*[x], x \in L\}$ and $\{M_{V^*}[x],\ x \in L\}$ are equal.

The possibility of tossing a coin is essential: Oren [120] proved that statistical zero-knowledge proofs with deterministic P exist only for languages in BPP.

The applications of zero-knowledge proofs in cryptography are evident. For instance, an authentication protocol must provide:

close to 1 probability that correct authenticity proof is accepted (completeness);

close to 0 probability that incorrect (i.e., produced by someone who does not know the secret identification information) authenticity proof is accepted (soundness);

the possibility of replicating an authenticity proof at least a polynomial number of times. This requires that the secret identification information be noncomputable from data gathered during protocol execution (zero-knowledge).

To prove the security of a protocol its designer has to demonstrate its completeness, soundness, and construct a simulating machine. It should be noted, however, that this task, especially its last part, is not a trivial one. In addition, we should emphasize that such proofs are often based on some unproved assumptions (e.g., on the assumption that one-way functions of some kind exist).

Zero-knowledge proofs were defined by Goldwasser et al. [67] in 1985. Goldreich et al. [64] proposed a computational zero-knowledge interactive proof for the language of 3-colorable graphs. Their construction is based on the assumption that a secure probabilistic cryptosystem exists (see §8). Since the language of 3-colorable graphs is NP-complete, it is proved as a corollary that computational zero-knowledge proofs exist for all languages in NP.

Ben-Or et al. [11] extended this result and proved that if one-way functions exist, then any language having interactive proof also has a computational zero-knowledge proof.

On the other hand, for NP-complete languages no perfect zero-knowledge proofs are known. Moreover, Fortnow [53] showed that if such proofs do exist, then the polynomial-time hierarchy collapses to the second level.

Nevertheless there are languages which are known to have perfect zero-knowledge interactive proofs. These are, e.g., graph isomorphism and quadratic residuosity [9, 64]. It should be noted that in these cases the proofs of zero-knowledgeness depend on no unproved assumptions. The first published perfect zero-knowledge proof protocols for these languages have unbounded number of rounds. Bellare et al. [9] proposed for them 5-round protocols. If graph isomorphism and quadratic residuosity are unrecognizable in polynomial (may be, probabilistic) time, then these protocols are near optimal since Goldreich and Krawczyk [61] proved that for any language outside BPP each perfect zero-knowledge protocol requires at least 4 rounds. Note that in [9] and [61] a round is considered to be a message transmission in one direction (from P to V or vice versa).

Literature Cited

1. W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, "RSA/Rabin bits are $1/2 + 1/\text{poly}(\log n)$ secure," In: *Proc. 25th Annu. IEEE Symp. on Found. Comput. Sci.*, 1984, pp. 449–457.
2. W. Alexi, B. Chor, O. Goldreich, and C. Schnorr, "RSA/Rabin bits are $1/2 + 1/\text{poly}$ secure," *SIAM J. Comput.*, **17**, No. 2, 194–209 (1988).
3. E. W. Allender, "Some consequences of the existence of pseudorandom generators," In: *Proc. 19th Annu. ACM Symp. on Theory of Comput.*, 1987, pp. 151–159.
4. L. Babai, "Trading group theory for randomness," In: *Proc. 17th Annu. ACM Symp. on Theory of Comput.*, 1985, pp. 421–429.
5. E. Bach and J. Shallit, "Factoring with cyclothomic polynomials," *Math. Comput.*, **52**, No. 185, 201–219 (1989).
6. W. W. R. Ball and H. S. M. Coxeter, *Mathematical Recreation and Essays,* Toronto Univ. Press, (1974).
7. E. Bazeries, *Les Chiffres Secrets Devoiles*, Paris (1901).
8. D. Beaver, "Perfect privacy for two party protocols," Technical Report TR-11-89, Harvard University (1989).
9. M. Bellare, S. Micali, and R. Ostrovsky, "Perfect zero-knowlege in constant rounds," In: *Proc. 22nd Annu. ACM Symp. on Theory of Comput.*, 1990, pp. 482–493.
10. J. D. Benaloh (Cohen) and M. Yung, "Distributing the power of a government to enhance the privacy of voters," In: *Proc. of the 5th ACM Symp. on Principles of Distributed Comput.*, 1986.

11. M. Ben-Or et al., "Everything provable is provable in zero-knowlege," *CRYPTO-88 (proceedings). Goldwasser, S. (ed.), Springer-Verlag, Lect. Notes Computer Sci.*, **403**, 37–56 (1990).
12. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for noncryptographic fault-tolerant distributed computation," In: *Proc. 20th Annu. ACM Symp. on Theory of Comput.*, 1988, pp. 1–10.
13. M. Ben-Or and N. Linial, "Collective coin-flipping, robust voting schemes and minima of Banzhaf values," In: *Proc. 26th Annu. IEEE Symp. on Found. of Computer Sci.*, 1985, pp. 408–416.
14. B. V. Berezin and P. V. Doroshkevich, "Digital signature scheme based on traditional cryptography," *Zashchita Informatsii*, **2**, 148-167 (1992).
15. R. Berger, S. Kannan, and R. Peralta, "A framework for the study of cryptographic protocols," *Proc. CRYPTO-85, Springer. Lect. Notes Comput. Sci.*, No. 215, 87–103 (1985).
16. E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York (1968).
17. E. R. Berlekamp, "Factoring polynomials over large finite fields," *Math. Comput.*, **24**, 713–735 (1978).
18. G. R. Blakley, "Safeguarding cryptographic keys," In: *Proc. of AFIPS National Computer Conference*, Vol. 48, 1979, pp. 313–317.
19. M. Blum, "Three applications of the oblivious transfer. Part I: Coin flipping by telephone. Part II: How to exchange secrets. Part III: How to send certified electronic mail," Dept. EECS, University of California, Berkeley, California (1981).
20. M. Blum, "Coin flipping by telephone: A protocol for solving impossible problems," In: *Proc. 24th IEEE Compcon* 133–137 (1982), *SIGACT News*, Vol. 15, 1983, pp. 23–27.
21. M. Blum, "All-or-nothing certified mail," *Workshop on Mathematical Aspects of Cryptography*, Endicott House, MIT (1985).
22. M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM J. Comput.*. **13**, No. 4, 850–864 (1984).
23. J. Boyar, G. Brassard, and R. Peralta, "Subquadratic zero-knowledge," In: *Proc. 32nd Annu. IEEE Symp. on Found. of Comput. Sci.*, 1991, pp. 69–78.
24. G. Brassard, C. Crépeau, and J.-M. Robert, "Information theoretic reductions among disclosure problems," In: *Proc. 27th Annu. IEEE Symp. on Found. of Computer Sci.*, 1986, pp. 168–173.
25. E. F. Brickell and A. M. Odlyzko, "Cryptoanalysis: A survey of recent results," *Proc. IEEE*, **76**, No. 5, 578–593 (1988).
26. J. L. Carter and M. N. Wegman, "Universal classes of hash functions," *J. Comput. Syst. Sci.*, **18**, No. 2, 143–154 (1979).
27. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," In: *Proc. 20th Annu. ACM Symp. on Theory of Comput.*, 1988, pp. 11–19.
28. B. Chor, M. Geréb-Graus, and E. Kushilevitz, "Private computations over the integers," In: *Proc. 31st Annu. IEEE Symp. on Found. of Computer Sci.*, 1990, pp. 335–344.
29. B. Chor and E. Kushilevitz, "A zero-one law for Boolean privacy," In: *Proc. 21th Annu. ACM Symp. on Theory of Comput.*, 1989, pp. 62–72.
30. B. Cleor and R. Rivest, "A knapsack type public key cryptosystem based on arithmetic in finite fields," In: *Proc. CRYPTO-84, New York, NY: Springer-Verlag*, 1985, pp. 54–65.
31. J. Cohen and M. Fisher, "A robuts and verifiable cryptographically secure election scheme," In: *Proc. 26th Annu. IEEE Symp. on Found. of Computer Sci.*, 1985, pp. 372–382.
32. D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Trans. Inf. Theory*, **30**, No. 4, 587–594 (1984).
33. D. Coppersmith, A. M. Odlyzko, and R. Schroeppel, "Discrete logarithms in GF(p)," *Algorithmica*, **1**, 1–15 (1986).
34. Data Encryption Standard 1977, *Federal Information Processing Standard (FIPS)*, Publication 46, National Bureau of Standards. U. S. Department of Commerce: Washington, DC. (January, 1977).
35. R. M. Davis, "The Data Encryption Standard in perspective," *IEEE Communications Society Magazine*, **16**, 5–9 (1978).
36. A. de Grandpre, *La Cryptographie Pratique*, Paris (1905).
37. H. M. Deitel, *An Introduction to Operating Systems*, Addison-Wesley, Reading, Massachusetts (1984).

38. R. A. Demillo, N. A. Lynch, and M. J. Merritt, "Cryptographic protocols," In: *Proc. 14th Annu. ACM Symp. on Theory of Comput.*, 1982, pp. 383–400.

39. "DES-Algorithmus entschlusselt?" *Datenschutz-Berater*, **12**, No. 5, 3–5 (1989).

40. W. Diffie, "The first ten years of public-key cryptography," *Proc. IEEE*, **76**, No. 5, 560-577 (1988).

41. W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, **IT-22**, 644–654 (1976).

42. W. Diffie and M. E. Hellman, "A critique of the proposed Data Encryption Standard," *Comm. ACM*, **19**, 164-165 (1976).

43. W. Diffie and M. E. Hellman, "Exhaustive cryptoanalysis of the NBS data encryption standard," *Computer*, **10**, 74–84 (June 1977).

44. W. Diffie and M. E. Hellman, "Privacy and authentication: an introduction to cryptography," *Proc. IEEE*, **67**, No. 3, 397–427 (1979).

45. D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," In: *Proc. 31st Annu. IEEE Symp. on Found. of Computer Sci.*, 1990, pp. 36–45.

46. C. Dwork, D. Shmoys, and L. Stockmeyer, "Flipping persuasively in constant time," *SIAM J. Computing*, **19**, No. 3, 472–499 (1990).

47. T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, **IT-31**, 469–472 (1985).

48. S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, **28**, 637-647 (1985).

49. S. Even, A. Selman, and Y. Yacobi, "The complexity of promise problems with applications to public-key cryptography," *Inf. Control*, **61**, No. 2, 159–173 (1984).

50. S. Even and Y. Yacobi, "Cryptocomplexity and NP-completeness," In: *Proc. 8th Colloq. on Automata, Languages and Programming, Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1980, pp. 195–207.

51. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," (Technical Report), The Weizmann Institute of Science, Rehevot, Israel (1986).

52. A. Figl, *Sisteme des Chiffrierens*, Graz (1923).

53. L. Fortnow, "The complexity of perfect zero-knowledge," In: *Proc. 19th Annu. ACM Symp. on Theory of Comput.*, 1987, pp. 204–209.

54. E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, "Ideals over a non-commutative ring and their applications in cryptology," In: *Proc. EUROCRYPT'91, Lecture Notes in Computer Science*, No. 547, Springer-Verlag, New York, 1991.

55. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, New York (1979).

56. C. M. Givierge, *Course de Cryptographie*, Paris (1925).

57. O. Goldreich, "A simple protocol for signing contracts," In: *Proc. CRYPTO 83*, Plenum Press, 1984, pp. 133–136.

58. O. Goldreich, "A uniform complexity treatment of encryption and zero-knowledge," *Technion CS-TR 570* (June 1989).

59. O. Goldreich et al., "Security preserving amplification of hardness," In: *31st Annu. IEEE Symp. on Found. of Comput. Sci.*, 1990, pp. 318-326.

60. O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," In: *Proc. 25th Annu. IEEE Symp. on Found. of Computer Sci.*, 1984, pp. 464 -479.

61. O. Goldreich and H. Krawczyk, "On the composition of zero-knowledge proof systems," In: *Proc. 17th Internat. Coll. on Automata, Languages and Programming*, Springer, Berlin (1990).

62. O. Goldreich, H. Krawczyk, and M. Luby, "On the existence of pseudo-random generators," In: *Proc. 29th Annu. IEEE Symp. on Found. of Comput. Sci.*, 1988, pp. 12–24.

63. O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," In: *Proc. 21st Annu. ACM Symp. on Theory of Comput.*, 1989, pp. 25-32.

64. O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity and a methodology of cryptographic protocol design," In: *Proc. 27th IEEE Symp. on Found. of Computer Sci.*, 1986, pp. 174-187.

65. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," In: *Proc. 19th Annu. ACM Symp. on Theory of Comput.*, 1987, pp. 218-229.

66. S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, **28**, 270–299 (1984).

67. S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," In: *Proc. 17th Annu. ACM Symp. on Theory of Comput.*, 1985, pp. 291–304.

68. S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," In: *Proc. 18th Annu. ACM Symp. on Theory of Comput.*, 1986, pp. 59–68.

69. R. M. Goodman and A. J. McAuley, "A new trapdoor knapsack public key cryptosystem," In: *Advances in Cryptology, EUROCRYPT'84, New York, NY, Springer-Verlag*, 1985, pp. 150-158.

70. Guan Puhua, "Cellular automaton public-key cryptosystem," *Complex Systems*, **1**, 51–57 (1987).

71. Guan Puhua, "Public-key cryptosystem based on higher order cellular automata," *IEEE Trans. Inf. Theory* (1987).

72. Guan Puhua and H. Zassenhaus, "Solving systems of equations over finite fields," *J. Number Theory* (1987).

73. J. Håstad, "Solving simultaneous modular equations of low degree," *SIAM J. Comput.*, **17**, No. 2, 336 341 (1988).

74. J. Hastad, "Pseudo-random generators under uniform assumptions," In: *Proc. 22nd Annu. ACM Symp. on Theory of Comput.*, 1990, pp. 395–404.

75. M. E. Hellman, "A cryptoanalytic time-memory tradeoff," *IEEE Trans. Inf. Theory*, **IT 26**, 401-406 (1980).

76. L. S. Hill, "Cryptography in an algebraic alphabet," *Amer. Math. Monthly*, **36**, No. 6, 306–312 (1929).

77. L. S. Hill, "Concerning certain linear transformation apparatus of cryptography," *Amer. Math. Monthly*, **38**, No. 3, 135–154(1931).

78. A. Hodges, *Alan Turing; The Enigma of Intelligence*, Unwin Paperbacks (1985).

79. S. Homer and J. Wang, "Absolute results concerning one-way functions and their applications," *Math. Syst. Theory*, **22**, No. 1, 21–35 (1989).

80. M.-D. A. Huang and S.-H. Teng, "Secure and verifiable schemes on election and general distributed computing problems," In: *7th Annual ACM Symposium on Principles of Distributed Computing*, 1988, pp. 182-196.

81. M.-D. Huang and S.-H. Teng, "A universal problem in secure and verifiable distributed computation," In: *Advances in Cryptology CRYPTO'88, Lecture Notes in Comput. Sci.*, Springer-Verlag, New York, Berlin, 1988, pp. 336-351.

82. M.-D. Huang and S.-H Teng, "Security, verifiability and universality in distributed computing," *J. Algorithms*, **11**, 492–521 (1990).

83. R. Impagliazzo, L. Levin, and M. Luby, "Pseudo-random generation from one-way functions," In: *Proc. 21st Symp. on Theory of Comput.*, 1989, pp. 12-24.

84. D. Joseph and P. Young, "Some remarks on witness functions for nonpolynomial and noncomplete sets in NP," *Theor. Comput. Sci.*, **39**, 225–237 (1985).

85. D. Kahn, *The Codebreakers: The Story of Secret Writing*, Mac Millan, New York (1967).

86. J. Kari, "A cryptoanalytic observation concerning systems based on language theory," *Discr. Appl. Math.*, **21**, No. 23, 265-268 (1988).

87. J. Kari, "Observations concerning a public-key cryptosystem based on iterated morphisms," *Theoretical Computer Sci.*, **66**, No. 1, 45–53 (1989).

88. J. Kari, "Security of ciphering in view of complexity theory," *MTA Szamitas techn., es autom. kut. intez. tanulm.*, No. 208, 163–169 (1988).

89. Ko Ker-I, T. J. Long, and Du Ding-Zhu, "On one-way functions and polynomial-time isomorphisms," *Theor. Comput. Sci.*, **47**, No. 3, 263–276 (1986).

90. N. Koblitz, "Use of algebraic curves in public-key cryptography," *Cryptography Tagungsber. Math. Forschungsinst., Oberwolfach*, No. 41, 18 (1989).

91. K. Kurosawa and K. Matsu, "*M* mod 3 security of RSA," *Electronics Letters*, **25**, No. 7, 445–446 (1989).
92. E. Kushilevitz, "Privacy and communication complexity," In: *Proc. 30th IEEE Symp. on Found. of Computer Sci.*, 1989, pp. 416-421.
93. L. Lamport, "Constructing digital signatures from one-way functions," *SRI Intl.* **CSL-98**, (1979).
94. A. Lange, *De la Cryptographie*, Paris (1918).
95. A. Lange and E. A. Soudart, *Traité de Cryptographie*, Paris (1925).
96. M. Leclerc, "A linear algorithm for breaking periodic Vernam ciphers," *Ars Combinatoria*, **27**, 177–179 (1989).
97. A. K. Lenstra, "Fast and rigorous factorization under the generalized Riemann hypothesis," *Proc. A. Kon. Ned. Akad. Wetensch.*, **91**, No. 4, 443–454 (1983).
98. A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The number field sieve" (Preprint).
99. L. A. Levin, "One-way functions and pseudorandom generators," In: *Proc. 17th Annu. ACM Symp. on Theory of Comput.*, 1985, pp. 363–365.
100. Li Gong, "Verifiable-text attacks in cryptographic protocols," In: *IEEE INFOCOM'90: Conf. Comput. Commun.: 9th Annu. Jt. Conf. IEEE Comput. and Commun. Soc. "Multiple Facets, Integer," San-Francisco, Calif., June 3-7, 1990*, Vol. 2, Los Alamitos, California, 1990, pp. 686-693.
101. D. L. Long and A. Wigderson, "The discrete logarithm hides $O(\log n)$ bits," *SIAM J. Comput.*, **17**, No. 2, 363–372 (1988).
102. S. C. Lu and L. N. Lee, "A simple and effective public-key cryptosystem," *COMSAT Tech. Rev.*, 15-24, (1979).
103. M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions,"*SIAM J. Comput.*, **17**, No. 2, 373–386 (1988).
104. J. L. Massey, "An introduction to contemporary cryptology," *Proc. IEEE*, **76**, No. 5, 533–549 (1988).
105. "Mathematical concepts of dependable systems," *Tagungsber., Math. Forschungsinst., Oberwolfach.*, **17**, 1-20 (1990).
106. K. S. McCurley, "A key distribution system equivalent to factoring," *J. Cryptol.*, **1**, No. 2, 95–105 (1988).
107. R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," In: *Deep Space Network Progress Report 42-44*, Jet Propulsion Laboratory, Pasadena, 1978, pp. 114–116.
108. R. McEliece and D. Sarwate, "On sharing secrets and Reed-Solomon codes," *Commun. ACM*, **24**, No. 9. 583–584 (1981).
109. F. J. McWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam (1977).
110. R. Merkle, *Letters to the Editor, Time Magazine*, **120**, No. 20, 8 (Nov. 1982).
111. R. C. Merkle and M. E. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Trans. Inform. Theory*, **IT-24**, No. 5, 525–530 (Sept. 1978).
112. S. Micali, C. Rackoff, and R. Sloan, "Notions of security of public-key cryptosystems," *SIAM J. Comput.*, **17**, No. 2, 412-426 (1988).
113. J. H. Moore, "Protocol failures in cryptosystems," *Proc. IEEE*, **76**, No. 5, 594–602 (1988).
114. R. Morris, "The Data Encryption Standard—retrospective and prospects," *IEEE Comm. Soc. Mag.*, **16**, 11-14 (1978).
115. M. Naor and M. Yung, "Universal one-way hash functions and their crytographic application," In: *Proc. 21st Annu. ACM Symp. on Theory of Comput.*, 1989, pp. 33-43.
116. M. Naor and M. Yung, "Public-key crytosystems provably secure against chosen ciphertext attacks," In: *Proc. 22nd Annu. ACM Symp. on Theory of Comput.*, 1990, pp. 427-437.
117. H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems Control Inform. Theory*, **15**, 159–166 (1986).
118. A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," In: *Lect. Notes Comput. Sci.*, 1985, pp. 224–314.
119. E. Okamoto and K. Tanaka, "Identity-based information security management system for personal computer networks," *IEEE J. Selec. Areas Commun.*, **7**, No. 2, 290–294 (1989).

120. Y. Oren, "On the cunning power of cheating verifiers: some observations about zero-knowledge proofs," *Proc. 28th Annu. Symp. on Found. of Comput. Sci.*, 462-471 (1987).
121. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey (1982).
122. J. P. Pieprzyk, "On public-key cryptosystems built using polynomial rings," In: *Advances in Cryptology-Eurocrypt. 85*, New York, NY: Springer-Verlag, 1986.
123. C. Pomerance, "The quadratic sieve factoring algorithm," *Lect. Notes Comput. Sci.*, No. 209, 169–182 (1985).
124. M. O. Rabin, "Digital signatures and public key functions as intractable as factorization," In: *Technical Memo TM-212*, Lab. Comput. Sci., MIT (1979).
125. T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," In: *Proc. 21st Annu. ACM Symp. on Theory of Comput.*, 1989, pp. 73–85.
126. C. Rackoff, "Relativized questions involving probabilistic algorithms," *J. ACM*, **29**, No. 1, 261–268 (1982).
127. H. Riesel, "Modern factorization methods," *BIT*, **25**, No. 1, 205–222 (1985).
128. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Comm. ACM*, **21**, No. 2, 120–126 (1978).
129. J. Rompel, "One-way functions are necessary and sufficient for secure signatures," In: *Proc. 22nd Annu. ACM Symp. on Theory of Comput.*, 1990, pp. 387-394.
130. G. Rosenberg, "Some recent developments in formal language theory," In: *Proc. Int. Congress Math. Helsinki, 15-23 Aug., 1978*, Vol. 2, Helsinki, 1980, pp. 973-978.
131. A. Salomaa, "The formal languages column," *Bull. Eur. Assoc. Theor. Comput. Sci.*, No. 33, 42–53 (1987).
132. A. Salomaa and S. Yu, "On a public-key cryptosystem based on iterated morphisms and substitutions," *Theor. Comput. Sci.*, **48**, 283–296 (1986).
133. C. P. Schnorr, "Efficient indentification and signatures for smart cards," *Kryptographie Tagungsber. Math. Forschungsinst., Oberwolfach*, No. 41, 18–19 (1989).
134. A. W. Schrift and A. Shamir, "The discrete log is very discreet," In: *Proc. 22nd Annu. ACM Symp. on Theory of Computing*, 1990, pp. 405-415.
135. A. L. Selman, "Complexity issues in cryptography," In: *Comput. Complexity Theory*, Providence, Rhode Island, 1989, pp. 92-107.
136. M. A. Seysen, "Probabilistic factorization algorithm with quadratic forms of negative discriminant," *Math. Comput.*, **48**, No. 178, 757–780 (1987).
137. A. Shamir, "How to share a secret," *Commun. ACM*, **29**, 612–613 (1979).
138. A. Shamir, "Identity-based cryptosystems and signature schemes," In: *Proc. Crypto-84*, Santa Barbara, California, 1984, pp. 47–53.
139. A. Shamir, "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem," *IEEE Trans. Inform. Theory*, **IT-30**, No. 5, 699–704 (1984).
140. A. Shamir, "The search for provably secure identification schemes," In: *Proc. of the Internat. Congress of Math.*, Berkeley, California, 1986, pp. 1488-1495.
141. A. Shamir, "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem," In: *Proc. 23rd Annu. IEEE Symp. on Found. of Computer Sci.*, 1982, pp. 145–152.
142. A. Shamir, "IP=PSPACE," In: *Proc. 31st Annu. IEEE Symp. on Found. of Computer Sci.*, 1990, pp. 11-15.
143. A. Shamir and R. E. Zippel, "On the security of the Merkle-Hellman cryptographic scheme," *IEEE Trans. Informat. Theory*, **IT-26**, 339–340 (1980).
144. C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Techn. J.*, **27**, No. 3, 379–423, (1948); **27**, No. 4, 623-656, (1948).
145. C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, **28**, No. 4, 656–715, (1949).
146. V. M. Sidelnikov and S. O. Shestakov, "On insecurity of cryptosystems based on generalized Reed-Solomon codes," *Discrete Math. Appl.*, **2**, No. 4, 439-444, (1992).

147. G. J. Simmons, "Authentication theory/coding theory," In: *Advances in Cryptology, Proc. CRYPTO 84, Lecture Notes in Computer Science*, No. 196, G. R. Blakley and D. Chaum, eds., New York: Springer, 1986, pp. 411-431.
148. G. J. Simmons, "A survey of information authentication," *Proc. IEEE*, **76**, No. 5, 603–620 (1988).
149. J. Stern, "Using error correcting codes in cryptography," *Cryptography Tagungsber. Math. Forschungs-inst., Oberwolfach.*, No. 41, 11-12 (1989).
150. I. Stewart, "Factorizing large numbers," *Math. Spectrum*, **20**, No. 3, 74-77 (1987-1988).
151. K. G. Subramanian, R. Siromoney, and P. J. Abisha, "A D0L-T0L public key cryptosystem," *Inf. Process. Lett.*, **26**, No. 2, 95–97 (1987).
152. R. Sugarman et al., "On foiling computer crime," *IEEE Spect.*, No. 16, 31–41 (July 1979).
153. P. L. E. Valerio, *De la Cryptographie* , Part 1, Paris (1893); Part 2, Paris (1896).
154. B. Vallee, "Factorization entière par generation quasi-uniforme de petits residus quadratiques," *C. R. Acad. Sci. Ser. 1*, **308**, No. 3, 59–62 (1989).
155. B. L. Van Der Waerden, *Algebra I*, Springer-Verlag, Berlin (1971).
156. H. O. Vardlay, *The American Black Chamber*, Indianapolis (1931).
157. O. N. Vasilenko, "Current methods for testing for primality of numbers. A survey," *Kybernet. Sb.*, No. 25, 162-188 (1988).
158. U. V. Vazirani and V. V. Vazirani, "RSA bits are $0,732+\varepsilon$ secure," In: *Sec. Adv. Cryptol. Proc. Crypto 83, Proc. Workshop Theory and Appl. Cryptogr. Techn. Santa Barbara, California, Aug. 21-24, 1983*, New York–London, 1984, pp. 369–375.
159. G. S. Vernam, "Cipher printing telegraph system for secret wire and radio telegraphic communications," *AIEE*, **45**, 109–115 (1926).
160. S. S. Wagstaff and J. W. Smith, "Methods of factoring large integers," *Lect. Notes Math.*, **1240**, 281–303 (1987).
161. M. J. Wiener, "Cryptoanalysis of short RSA secret exponents," *IEEE Trans. Inform. Theory*, **36**. No. 3, 553–558 (1990).
162. H. C. Williams, "A modification of the RSA public-key encryption procedure," *IEEE Trans. Inform. Theory*, **26**, No. 6, 726–729 (1980).
163. S. V. Yablonskii, *Introduction to Discrete Mathematics* [translated from Russian], Mir, Moscow (1989).
164. A. C. Yao, "Some complexity questions related to distributive computing," In: *Proc. 11th Annu. ACM Symp. on Theory of Computing*, 1979, pp. 209-213.
165. A. Yao, "Protocols for secure computations," In: *Proc. 23rd Annu. IEEE Symp. on Found. of Comput. Sci.*, 1982, pp. 160-164.
166. A. Yao, "Theory and applications of trapdoor functions," In: *Proc. 23rd Annu. IEEE Symp. on Found. of Comput. Sci.*, 1982, pp. 80-91.
167. A. Yao, "How to generate and exchange secrets," In: *Proc. 27th Annu. IEEE Symp. on Found. of Comput. Sci.*, 1986, pp. 162-167.
168. P. Young, "Some structural properties of polynomial reducibilities and sets in NP," In: *Proc. 15th Annu. ACM Symp. on Theory of Comput.*, 1983, pp. 392-401.
169. G. Yuval, "How to swindle Rabin," *Cryptologia*, **3**, 187–189 (1979).