МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2007
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2007
Proceedings of the Thirty Sixth Spring Conference of
the Union of Bulgarian Mathematicians
St. Konstantin & Elena resort, Varna, April 2–6, 2007

CRYPTOGRAPHIC PROTOCOL*

Ivan Landjev

This paper is a concise introduction to the vast field of cryptographic protocol. It contains a compressed description of some basic cryptographic protocols: oblivious transfer schemes, commitment schemes, zero knowledge proofs and interactive bit generation. A digital voting scheme and a digital cash scheme are also given as examples of more complex protocols.

1. Introduction. In the past 25 years dramatic changes took place in the field of cryptographic research. While not so long ago cryptography was considered as "the science and art" of secret writing, nowadays it is customary to put a much broader meaning in this word. One possible definition is that cryptography is the science concerned with "the construction of (communication) schemes that are robust against malicious attempts to make these schemes deviate from their prescribed functionality" [8]. A cryptographer should design a scheme that not only satisfies a desired functionality, but also maintains this functionality in face of adversarial attempts that are devised after the cryptographer has completed his task. As a rule, an adversary will devise his attack after the scheme has been specified and will try to take actions other than the ones envisioned by the cryptographer. So, it makes no sense to make assumptions regarding the specific startegy the adversary may use. This might be the case if the cryptographer has a very good idea about the environment in which the scheme is to operate. Yet, a cryptographic scheme has to operate in a maliciously selected environment that transcends the designer's view. So, the only assumptions that can be justified refer to the computational abilities of the adversary (that include the available computing power, but also the state-of-the art in the algorithms for specific problems).

There is no unified definition of cryptographic protocol. In [15] it is defined as a distributed algorithm described by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective (functionality). In this paper, we consider several basic cryptographic protocols: oblivious transfer, commitment schemes, zero-knowledge proofs, random bit generation. Due to space limitations we skip the vast area of secret sharing protocols (cf., for instance, [14, 15]). Further, we describe in some detail two applications of the basic protocols: digital voting and digital cash.

*This research has been supported by the Bulgarian NSRF under Contract MM-1405/2005.

Key words: cryptographic protocol, oblivious transfer, commitment schemes, bit commitment, zero knowledge, zero knowledge proofs, random bit generation, digital voting, digital cash

2000 Mathematics Subject Classification: 94A60, 68P25, 11T71

2. Some basic cryptographic protocols.

2.1. Oblivious transfer. A party A possessing a secret wants to transfer it to another party B in such a way that, after the protocol, A does not know whether B got the secret, but B knows it. Another version of this problem is the following: A possesses several secrets and wants to transfer one of them to B. The transfer should be done in such way, that only B knows which of the secrets was transferred. All these problems belong to a subfield of cryptography known as *oblivious transfer* (cf. [13]).

Let us note that all constraints involved in oblivious transfer are easily met with the help of a trusted referee. The importance of the protocols for oblivious transfer lies in the fact that they do not assume the existence of such trusted party.

Example 2.1. Consider the case where A wants to transfer to B obliviously the factorization of an integer n into two primes. This is no loss of generality since the secret can be anything encrypted using an RSA cryptosystem. The protocol below is based on the fact that the knowledge of two square roots modulo n enables us to find a factorization of n.

(1) B chooses an integer x, $1 \leq x \leq n-1$, and tells A the number $x^2 \pmod{n}$;
(2) A, who knows the factorization of $n = pq$, computes the four square roots $\pm x$, $\pm y$ of $x^2 \pmod{n}$ and tells one of them to B.
(3) B checks whether the square root obtained is congruent to $\pm x \pmod{n}$. If this is the case, then B has no new information and the factorization of n is infeasible. Otherwise, B knows two different square roots of the same number and is able to factor n. In both cases A does not know whether B has the factorization or not.

Clearly, the probability of B's obtaining the factorization is $\frac{1}{2}$.

Example 2.2. Now, we present another protocol for oblivious transfer. The setup is more general: A has two secrets s_0 and s_1, and wants to transfer one of them to B. This time the protocol is non-interactive – B sends nothing to A.

The transfer can be carried out between any two users of some system. All users in such system share a large prime number p, a generator $g \in \mathbb{Z}_p^*$ and a number c, $1 \leq c \leq p-1$. The prime p is chosen in such a way, that it is infeasible to compute $g^{xy} \pmod{p}$ from $g^x \pmod{p}$ and $g^y \pmod{p}$. Every user, say B, computes his public and secret keys as follows:

- B chooses randomly an $i \in \{0, 1\}$ and a number x, $0 \leq x \leq p-2$;
- B computes $\beta_i = g^x$ and $\beta_{1-i} = cg^{-x}$;
- B's public key is the pair (β_0, β_1); B's secret key is the pair (i, x).

Let us note at this point that B does not know the discrete logarithm of both β_0 and β_1, since he can not compute the discrete logarithm of c modulo p (cf. [16, 17]). It is not clear, however, which one of the two discrete logarithms, $\log_g \beta_0$ and $\log_g \beta_1$, is known to B. Every user in the system can check that the encryption key has been formed correctly by checking the identity $c = \beta_0 \beta_1$.

Assume that s_0 and s_1 are bit sequences of the same length. This is no restriction, since we can always complete the shorter one by inserting leading zeros. Then, we can compute the bitwise sum $s_0 \oplus s_1$.

(1) A picks randomly y_0 and y_1, $0 \leq y_0, y_1 \leq p-2$, and computes

$$\alpha_0 = g^{y_0}, \quad \gamma_0 = \beta_0^{y_0}, \quad r_0 = s_0 \oplus \gamma_0,$$
$$\alpha_1 = g^{y_1}, \quad \gamma_1 = \beta_1^{y_1}, \quad r_1 = s_1 \oplus \gamma_1.$$

A sends α_0, α_1, r_0 and r_1 to B.

(2) B computes $\alpha_i^x = g^{xy_i} = \beta_i^{y_i} = \gamma_i$ and $s_i = \gamma_i \oplus r_i$. For the other index $1-i$, one has $\alpha_{1-i}^x = g^{xy_{1-i}} = \beta_i^{y_{1-i}} \neq \gamma_{1-i}$.

Example 2.3. Now, consider the situation, when A has several secrets $s_1, \ldots, s_k$, where each of the s_i's is a sequence of bits. A protocol for oblivious transfer which transmits to B one of the secrets runs as follows.

(1) A tells B a one-way function f but keeps f^{-1} to himself.

(2) B decides to buy the secret s_i. He chooses k random values $x_1, \ldots, x_k$ from the domain of f and tells A the k-tuple $(y_1, \ldots, y_k)$, where

$$y_j = \begin{cases} x_j & \text{for } j \neq i; \\ f(x_j) & \text{for } j = i. \end{cases}$$

(3) A computes the numbers $z_j = f^{-1}(y_j)$ and tells B the numbers $z_j \oplus s_j$, $j = 1, \ldots, k$.

(4) B knows $z_i = f^{-1}(f(x_i)) = x_i$ and can compute s_i.

If B is an active cheater and deviates from the protocol, then he can learn more secrets by presenting several of the numbers y_j in the form $f(x_j)$.

Now, consider the case where active cheating is possible. We assume that not both parties are active cheaters. It is customary to assume in such protocols that at most half of the parties are active cheaters. Let the active cheater be B. To prevent himself from cheating, he has to commit himself to specific action, i.e. to specify which one of the secrets he is about to buy. The commitment can be concealed using a one-way function. This he should do without disclosing any information about the action itself (e.g. by a zero-knowledge proof). We also use the notion of flipping a number in the same sense as in section 2.3. As before, the secrets are the strings $s_1, \ldots, s_k$.

(1) A flips to B k numbers $x_1 \ldots, x_k$. The number of bits in the x_i's has been agreed upon in advance, e.g. it coincides with the number of bits in the s_i's.

(2) A tells B an one-way function f, but keeps f^{-1} to himself.

(3) If B has decided to buy s_i, then he computes $f(x_i)$. Some bits in $f(x_i)$ coincide with the corresponding bits in x_i. Let the positions of coincidence be $u_1, \ldots, u_t$.

(4) B tells the bit x_α in a position u_β, for all $1 \leq \alpha \leq k$, $1 \leq \beta \leq t$. He does this in such a way that A can verify the information (cf. section 2.3).

(5) B tells A the k-tuple $(y_1, \ldots, y_k)$, where

$$y_j = \begin{cases} x_j & \text{for } j \neq i; \\ f(x_j) & \text{for } j = i. \end{cases}$$

A verifies that the information is in accordance with step (4).

(6) A tells B the numbers $f^{-1}(y_j) \oplus s_j$, $j = 1 \ldots, k$.

Note that $f(x_i)$ has the same bits with x_i in positions $u_1, \ldots, u_t$, but $f(f(x_j))$ has the same bits as $f(x_j)$ in these positions with probability 2^{-t}. If t is big enough, then this probability is negligible. The protocol is based on the fact that the number t in step (3) is approximately half of the bits in x_j (if f has a reasonably random behavior). If

B would choose two exceptional y_j's in step (5), then the stable bits for both such y_j's would be different which would not go unnoticed by A.

A may cheat by computing, for each j, the positions of the bits for which y_j and $f^{-1}(y_j)$ coincide. For $j = i$ the bits are precisely those communicated to A in step (4), whereas for $j \neq i$ the positions are different with an overwhelming probability. A simple way to overcome this difficulty is to consider two buyers B and C. We assume that the parties A, B, C do not form coalitions.

(1) A tells to B and C one-way functions f and g, but keeps the inverses to himself.
(2) B tells C (resp. C tells B) random numbers $x_1, \ldots, x_k$ (resp. $x'_1, \ldots, x'_k$). The numbers need not be flipped and they have as many bits as the s_j's.
(3) Assume B wants to buy s_b and C wants to buy s_c. B (resp. C) computes $f(x'_b)$ (resp. $g(x_c)$) for his chosen index. The function and argument values are compared with respect to fixed bits, i.e. bits remaining invariant in the transition from x'_b to $f(x'_b)$ (resp. from x_c to $g(x_c)$).
(4) B tells C (resp. C tells B) the indices of the fixed points stable for B (resp. stable for C).
(5) B (resp. C) tells A the numbers $y_1, \ldots, y_k$ (resp. $y'_1, \ldots, y'_k$) where the y_j's and y'_j's result from the corresponding x's by changing every bit, whose index is not stable for C (resp. for B), to its complement.
(6) A tells to B the numbers $f^{-1}(y'_j) \oplus s_j$, and to C the numbers $g^{-1}(y_j) \oplus s_j$, $j = 1, \ldots, k$.

Attempts to choose more than one secret fail with an overwhelming probability because of step (5), provided the number of bits in the s_j's is not very small.

2.2. Commitment schemes. B wants A to commit to a value from some set X (say a bid) so that he cannot change this at a later moment. On the other hand, A does not want B to know, at this time, what is the value he is committing to, but will open it later. This is a typical commitment problem which appears at various places in cryptography. A rigorous definition of commitment schemes is the following: let X, Y and Z be finite sets, then a *commitment scheme* is a function $f : X \times Y \to Z$ which has the following properties:

- *concealing property* – for any $x \in X$, B cannot determine the value of x from $f(x, y)$;
- *binding property* – A is not able to find a z such that $f(x, y) = f(x', z)$, where $x' \in X$.

We say that a commitment scheme is:

information-theoretically binding if A cannot change the value he is committed to, even if he has unlimited computing power;

computationally binding if A cannot change the value he is committed to by polynomial time computations (in the security parameter);

information-theoretically concealing if B is not able to find that the value A is committed to, even with unlimited computer power;

computationally concealing if B cannot find that the value A is committed to performing polynomial time computations (in the security parameter).

An important open question is whether it is possible to construct a commitment scheme which is information theoretically secure with respect to both properties.

Let $G = \langle g \rangle$ be a finite abelian group of prime order q (i.e. a cyclic group). Let b be an element of G with an unknown discrete logarithm $\log_g b$ in G. For instance, we can take q to be a (large) prime divisor of $p-1$, where p is large prime ($p \approx 2^{2^{10}}$). Then, G is the unique subgroup of order q in $\mathbb{F}_p^*$, the multiplicative group of the finite field with p elements. The pair $(g, b) \in G \times G$ is made public to all users. Now, we define two simple commitment schemes:

(1) In order to commit to the value x, $0 \leq x \leq q-1$, A transmits to B: $D(x) = g^x$.

(2) In order to commit to the value x, $0 \leq x \leq q-1$, A generates a random integer a, $0 \leq a \leq q-1$, and transmits to B: $D_a(x) = b^x g^a$.

To reveal the value he is committed to, A reveals x in scheme (1) and the pair (a, x) in scheme (2). It is easily checked that the commitment scheme (1) is information theoretically binding and computationally concealing, while (2) is computationally binding and information theoretically concealing.

When the message to be committed to is a single bit, we speak of *bit commitment schemes*. So, bit commitment schemes are functions $f : \{0,1\} \times Y \to Z$ that have the concealing and binding properties defined above. One way to perform bit commitment is to use the Golwasser-Micali probabilistic cryptosystem [10]. In this system $n = pq$, where p, q are large primes, and m is a quadratic non-residue modulo n. The integers n and m are public while the factorization of n is known only to A. Set $Y = Z = \mathbb{Z}_n^*$ and

$$f(b, y) = m^b y^2 \pmod{n}. \tag{1}$$

A encrypts a value b by choosing a random y and computing $z = f(b, y)$. Later, when A wants to open the commitment, he reveals the values b and y. Then, B can verify that

$$z \equiv m^b y^2 \pmod{n}.$$

Let us note that this scheme reveals no information whether the bit A is committed to, provided the problem of finding a solution to $x^2 \equiv a \pmod{n}$, $n = pq$, is infeasible. Hence, the scheme is computationally concealing. On the other hand, the scheme is information theoretically binding, since it is impossible to find integers x_1 and x_2 from $\mathbb{Z}_n^*$ such that

$$m x_1^2 \equiv x_2^2 \pmod{n}.$$

This would contradict the fact that m is non-residue modulo n.

2.3. Interactive random bit generation (or coin flipping over the telephone). In some cryptographic protocols it is required that the parties generate together a random sequence without the assistance of a trusted referee. If the number of parties is two, then this amounts to a problem known as "flipping a coin over the telephone". The result of a protocol should be a random bit which takes with equal probability the values 0 and 1. None of the parties should be able to influence the result, e.g by changing the probabilities.

A general idea for the algorithm is the following: A picks a random bit a and sends it to B and B picks a random bit b and sends it to A. The problem is that the party who moves second, is able to manipulate the result. So, the party who moves first must commit to its choice. He sends $y = f(a)$ to B so that it is impossible for him to make his choice b a function of a. He sends back b in the clear. Now, A cannot make a a function of b since he is committed to the value a.

The following protocol shows how A can flip a coin to B over the telephone:

(1) A chooses two large integers p and q and tells their product $n = pq$ to B.
(2) B chooses a random number u from the interval $\left(1, \frac{n}{2}\right)$ and tells A the square $z = u^2 \pmod{n}$.
(3) A computes the four square roots of z modulo n. This is possible because he knows the factorization of n. Let us call the roots $\pm x$ and $\pm y$. Denote by x' the smaller of the numbers $x \pmod{n}$ and $-x \pmod{n}$ and, similarly, let y' be the smaller of $y \pmod{n}$ and $-y \pmod{n}$. Now, A knows that $u = x'$ or $u = y'$.
(4) A guesses whether $u = x'$ or $u = y'$ and sends one bit of his guess (e.g. the least significant bit where x' and y' differ). He tells B one of the two guesses "the i-th bit of your number is 0" or "the i-th bit of your number is 1".
(5) B tells whether the guess was correct (the generated bit is 1), or wrong (the generated bit is 0).
(6) B tells A the number u.
(7) A tells B the factorization of n.

This scheme can be easily generalized provided one-way functions do exist.

(1) A and B know a one-way function f.
(2) B chooses a random x and tells A the value $f(x)$.
(3) A makes a guess about some property of x which holds with probability $\frac{1}{2}$ (e.g. whether x is even or odd).
(4) B tells A whether the guess is correct.
(5) Later B discloses x to A.

2.4. Zero-knowledge proofs. Zero-knowledge proofs are proofs which yield nothing beyond the validity of a given assertion. In other words, a verifier obtaining such a proof only gains conviction in the validity of the assertion. In this section, we refer to proofs as to interactive and randomized processes. So, a proof is a multi-round two-party protocol (the parties being called prover and verifier) in which the prover P wishes to convince the verifier V in the validity of a given assertion. Such a proof should have the following properties:

(1) it should allow the prover to convince the verifier of the validity of any true assertion (completeness condition);
(2) no prover strategy may fool the verifier to accept false assertions (soundness condition).

Both the completeness and soundness conditions should hold with a high probability (i.e. a negligible error probability is allowed). In addition, the verifier strategy is required to be efficient. No such requirements are made for the prover strategy, yet we consider "relatively efficient" prover strategies. In what follows, we confine ourselves to a more simple problem: how can P prove to V that x is in a language L in such way that no more knowledge than the fact that $x \in L$ is revealed.

2.4.1. Interactive proof-systems. We start by introducing the notion of an interactive Turing machine.

Definition 2.4. *An* interactive Turing machine (*ITM*) *is a Turing machine with a read-only input tape, a read-only random tape, a read/write work tape, a read-only communication tape, a write-only communication tape and a write-only output tape. The random tape contains an infinite sequence of bits that can be thought of as the outcome of*

unbiased coin tosses; it can be scanned only from left to right. We say that an interactive machine flips a coin, when it reads from its random tape. The contents of the write-only communication tape can be thought of as messages sent by the machine, while the contents of the read-only communication tape can be thought of as messages received by the machine.

An *interactive protocol* is an ordered pair (A, B) of ITM's which share the same input tape; B's write-only communication tape is A's read-only communication tape and *vice versa*. The machines take turns in being active with B being active first. During its active stage, the machine first performs some internal computations based on the contents of its tapes, and second writes a string on its write-only communication tape. The i-th message of A (resp. B) is the string A (resp. B) writes in its write-only communication tape in the i-th stage. At this point the machine is deactivated and the other machine becomes active, unless the protocol is terminated. Either machine can terminate the protocol by not sending any message in its active stage. Machine B accepts or rejects the input by entering an accept or reject state and terminating the protocol.

Machine A is assumed to be computationally unbounded Turing machine. The computation time of machine B is defined as the sum of B's computations during its active stages, and is taken to be bounded by a polynomial in the length of the input string.

Definition 2.5. *Let $L \subseteq \{0,1\}^*$. We say that L has an* interactive proof-system *if there exists an interactive Turing machine V such that:*

1) *$\exists$ ITM P such that (P,V) is an interactive protocol and for every $x \in L$, $|x|$ sufficiently large,* $\Pr(V \text{ accepts }) > 1-\varepsilon$ *for some positive constant ε. (Probabilities are taken over coin tosses of V and P.)*
2) *$\forall$ ITM P such that (P,V) is an interactive protocol and for every $x \notin L$, $|x|$ sufficiently large,* $\Pr(V \text{ accepts }) < \varepsilon$ *for some positive constant ε.*

Example 2.6. In the following examples $B \to A$ denotes an active stage for machine B at the end of which B sends a message. Similarly, $A \to B$ denotes an active stage of A.

1) Set

$$\begin{aligned} \mathbb{Z}_n^* &= \{x \mid x < n, (x,n) = 1\}, \\ QR &= \{\{x,n\} \mid (x,n) = 1, \exists\, y : y^2 \equiv x \pmod{n}\}, \\ QNR &= \{\{x,n\} \mid (x,n) = 1, \nexists\, y : y^2 \equiv x \pmod{n}\}. \end{aligned}$$

We demonstrate an interactive proof-system for QNR. On input x, n to interactive protocol (A, B):

(1) $B \to A$: B sends to A the list $w_1, \ldots, w_k$, $k = |n|$, and

$$w_i = \begin{cases} z_i^2 \pmod{n} & \text{if } b_i = 1, \\ x z_i^2 \pmod{n} & \text{if } b_i = 0, \end{cases}$$

where B selects $z_i \in \mathbb{Z}_n^*$, $b_i \in \{0,1\}$ at random.

(2) $A \to B$: A sends to B the list $c_1, \ldots, c_k$:

$$c_i = \begin{cases} 1 & \text{if } w_i \text{ is a quadratic residue modulo } n, \\ 0 & \text{otherwise .} \end{cases}$$

B accepts if for all $i = 1, \ldots, k$, $c_i = b_i$.

B interprets $b_i = c_i$ as evidence that $\{x, n\} \in QRN$, while $b_i \neq c_i$ leads him to reject. (A, B) is an interactive proof system for QNR. If $\{x, n\} \in QNR$, then w_i is a quadratic residue modulo n iff $b_i = 1$. The all-powerful A can compute whether w_i is a quadratic residue or not, compute c_i correctly and make B accept with probability 1. If$\{x, n\} \notin QNR$ and $\{x, n\} \in QR$ then w_i is a random quadratic residue modulo n regardless of whether $b_i = 0$ or 1. Thus, the probability that A can send c_i such that $c_i = b_i$ is bounded by $1/2$ for each i, and the probability that B accepts is at most $(1/2)^k$.

2) Now, we present an interactive proof system for the problem GRAPH-NON-ISOMORPHISM. The input is a graph pair G_1, G_2. One is required to prove that there is no isomorphism between G_1 and G_2. GRAPH-NON-ISOMORPHISM is known to be in NP.

The interactive proof (A, B) on input (G_1, G_2) proceeds as follows:

(1) $B \to A$: B chooses at random one of the input graphs G_{α_i}, $\alpha_i \in \{1, 2\}$. B creates a random isomorphic copy of G_{α_i} and sends it to A. This is repeated k times for $i = 1, \ldots, k$, with independent random choices.

(2) $A \to B$: A sends B $\beta_i \in \{1, 2\}$ for all $i = 1, \ldots, k$. B accepts, iff $\beta_i = \alpha_i$ for all $i = 1, \ldots, k$. B interprets $\beta_i = \alpha_i$ as evidence that the graphs are not isomorphic, while $\beta_i \neq \alpha_i$ leads him to reject.

If the two graphs are not isomorphic, then the prover can always answer correctly, and the verifier will accept. If the two graphs are isomorphic, then it is impossible to distinguish a random isomorphic copy of the first from a random isomorphic copy of the second and the probability that the prover answers correctly to one query is at most $1/2$. The probability that the prover answers correctly all queries is $\leq (1/2)^k$.

2.4.2. Zero-Knowledge. Now, we address the question of how much knowledge needs to be transferred in order to convince a polynomial-bounded verifier of the truth of a proposition. What is meant here by knowledge? Consider SAT, the NP-complete language of the satisfiable sentences of propositional calculus. The most obvious proof-system is one in which on logical formulae f the prover gives the verifier a satisfying argument I, which the verifier can check in polynomial time. If finding this assignment I by himself would take the verifier more than polynomial time (if $P \neq NP$), then we say that the verifier gains additional knowledge to the mere fact that $f \in$SAT.

This notion is made precise by Goldwasser, Micali and Rackoff in [11]. They call an interactive proof-system for the language L *zero knowledge* if for every $x \in L$ whatever the verifier can compute after participating in the interaction with the prover, it could have been computed in polynomial time on the input x alone by a probabilistic polynomial time Turing machine.

The most important result about zero-knowledge is obtained by Goldreich, Micali and Wigderson [9].

Theorem 2.7. *If there exists a (non-uniform) indistinguishable encryption scheme, then every NP-language has a computational zero-knowledge interactive proof-system.*

Example 2.8. Below we describe a zero-knowledge proof of a discrete logarithm from [4]. The prover P wants to prove to the verifier V that he knows an x that satisfies $a^x \equiv b \pmod{p}$ when p is a (large) prime and $(x, p-1) = 1$. The numbers a, b and p are public and x is randomly chosen.

(1) P generates t random numbers $r_1, \ldots, r_t$ that are less than $p-1$.
(2) P computes $h_i = a^{r_i} \pmod p$, $i = 1, \ldots, t$, and sends the h_i's to V.
(3) P and V engage a coin-flipping protocol to generate t bits $b_1, \ldots, b_t$.
(4) For all t bits P does one of the following:
    (a) if $b_i = 0$, then P sends to V the integer r_i,
    (b) if $b_i = 1$, then P sends to V $s_i = r_i - r_j \pmod{p-1}$, where j is the lowest value of i for which $b_i = 1$.
(5) For all t bits V confirms one of the following:
    (a) if $b_i = 0$ that $a^{r_i} = h_i \pmod p$,
    (b) if $b_i = 1$ that $a^{s_i} = h_i h_j^{-1} \pmod p$.
(6) P sends $Z = x - r_j \pmod{p-1}$ to V.
(7) V checks that $a^z = b h_j^{-1} \pmod p$.

The probability of P's cheating is 2^{-t}.

Example 2.9. Below we give another zero-knowledge proof of discrete logarithm [3]. Again the integers a, b, and the prime p are public. P wants to prove to V that he knows the discrete logarithm modulo p of b on the base a.

Repeat t times the following steps:
(1) P chooses a random integer r, $1 \le r < p-1$. P computes $h = a^r \pmod p$ and sends h to V.
(2) V sends to P a random bit β.
(3) P computes and sends to V $s = r + bx \pmod{p-1}$.
(4) V checks that $a^s \equiv h b^\beta \pmod p$

Again the probability of P's cheating is 2^{-t}.

Zero-knowledge proofs provide a revolutionary new way to realize passwords [7, 12]. The idea is for every user to store a statement of a theorem in his publicly readable directory, the proof of which only he knows. Upon login, the user engages on a zero-knowledge proof of the correctness of the theorem. If the proof is convincing, then access permission is granted. This guarantees that even an adversary that overhears the zero-knowledge proof cannot learn enough to gain unauthorized access. This is a new property that cannot be achieved with traditional password mechanisms. Fiat and Shamir [7] have developed variations on some of the previously proposed zero-knowledge protocols which are quite efficient and particularly useful for user identification and passwords.

3. Some applications of cryptographic protocol.

3.1. Digital voting. In this section, we describe the electronic voting scheme by Cramer-Franklin-Schoenmakers-Yung [6]. The design of a good voting scheme must satisfy a number of criteria that are sometimes competing and even contradictory. Below we list the most obvious of them:
(1) Only legitimate voters must be allowed to vote.
(2) No one can vote more than once.
(3) Each vote is secret; no one can find out the vote of another voter.
(4) No one should be able to repeat the vote of another voter.
(5) The final result should be computed correctly.
(6) Each of the participants should be able to check that his vote is computed correctly.
(7) The protocol works correctly even if some of the users are dishonest.

In this note, we do not discuss such requirements for the system as to be comprehensible and usable by the entire voting population. This is mainly an engineering problem which, when done well, would provide a great improvement over current paper systems.

We assume that the participants in the scheme are m legitimate voters and n panels (or poll workers) counting the results of the elections. The use of many panels guarantees the anonymity of the vote and prevents manipulation of the votes being cast. Further, we assume that each voter can give his vote for one of two candidates. Of course, this is not a real restriction. The current scheme can be generalized for use in more complicated elections.

(1) **Setting up the scheme**

Each of the panels has a public enciphering transformation, E_i say, $i = 1, \ldots, n$, within an asymmetric cryptosystem (such as RSA). A cyclic group G of prime order q is fixed for all voters. In the group a pair of elements b, g is chosen, $b, g \neq 1$. The order of G is so large that the problem of finding the logarithm of b on the base g is intractable. Further, we assume that each voter has his own digital signature generated *via* some asymmetric digital signature scheme (such as DSA, for example). For the sake of simplicity, we assume that the two possible votes to be cast are $+1$ and -1.

(2) **Vote generation**

In order to generate a vote, the j-th voter picks up a vote $v_j \in \{+1, -1\}$, chooses a blinding number $a_j \in \mathbb{Z}_q$ and publishes

$$d_j = d_{a_j}(v_j) = b^{v_j} g^{a_j}.$$

The element $d_j \in G$ is made known to all participants in the election: voters and panels. Together with the element d_j the voter publishes a transcript of a protocol which certifies that the vote has been selected from $\{+1, -1\}$. The vote and the transcript are signed with the digital signature of the j-th voter.

(3) **Casting a vote**

In order to submit a_j and v_j to the panels each voter uses a Shamir secret sharing scheme. To this end, he selects randomly two polynomials over $\mathbb{F}_q$ of degree $t < n$ each:

$$\begin{aligned} R_j(x) &= v_j + r_{1,j}x + \ldots + r_{t,j}x^t, \\ S_j(x) &= a_j + s_{1,j}x + \ldots + s_{t,j}x^t. \end{aligned}$$

Every voter computes $(u_{i,j}, w_{i,j}) = (R_j(i), S_j(i)), \quad 1 \leq i \leq n$. Every voter enciphers the pair $(u_{i,j}, w_{i,j})$ using the algorithm E_i of the i-th panel and sends the result to this panel. After that he registers the polynomial $R_j(x)$ by publishing the group element $d_{l,j} = d_{s_l,j}(r_{l,j}), \quad 1 \leq l \leq t$.

(4) **Checking the correctness of the information**

Every panel checks whether the pair $(u_{i,j}, w_{i,j})$ has been obtained from the j-th voter and whether it corresponds to the commitment d_j. This is obtained by checking the validity of the following identity:

$$\begin{aligned} d_j \prod_{l=1}^{t} d_{l,j}^{i^l} &= d_{a_j}(v_j) \prod_{l=1}^{t} d_{s_{l,j}}(r_{l,j})^{i^l} \\ &= b^{v_j} g^{a_j} \prod_{l=1}^{t} (b^{r_{l,j}} g^{s_{l,j}})^{i^l} \\ &= b^{(v_j + \sum_{l=1}^{t} r_{l,j} i^l)} g^{(a_j + \sum_{l=1}^{t} s_{l,j} i^l)} \\ &= b^{u_{i,j}} g^{w_{i,j}}. \end{aligned}$$

(5) **Counting the votes**

Each of the panels computes and publishes as results of the election the elements $U_i = \sum_{j=1}^{m} u_{i,j}$. Apart from this, each panel computes and publishes the sums of the blinding numbers $W_i = \sum_{j=1}^{m} w_{i,j}$. Every other participant in the scheme – voter or panel – can convince himself in the correctness of the results by checking that

$$\prod_{j=1}^{m} \left(d_j \prod_{l=1}^{d} b_{l,j}^{j^l} \right) = \prod_{j=1}^{m} b^{u_{i,j}} g^{w_{i,j}} = b^{U_i} g^{W_i}.$$

Each participant can determine the result of the election by taking t values U_i and interpolating by them the final result. In fact, U_i is the value of a certain polynomial at point i:

$$\begin{aligned} U_i &= \sum_{j=1}^{m} u_{i,j} \\ &= \left(\sum_{j=1}^{m} v_j \right) + \left(\sum_{j=1}^{m} r_{1,j} \right) i + \cdots + \left(\sum_{j=1}^{m} r_{t,j} \right) i^t. \end{aligned}$$

If the result is in $\left\{1, \ldots, \dfrac{q-1}{2}\right\}$, then the majority of the voters has put $+1$ as their vote; if the result is in $\left\{\dfrac{q+1}{2}, \ldots, q-1\right\}$, then the majority of the voters has put -1.

3.2. Digital cash. One major argument against credit Internet shopping is that it is not anonymous. The identity of the user is established each time he makes a purchase. In the real life, we have the possibility to use cash whenever we want to buy something without establishing our identity. The solution to this problem would be to create untraceable digital money or digital cash. The idea has been developed in [1, 2, 5].

By the term *digital cash*, we denote cryptographic techniques that aim at creating a paying scheme with the following functionality:

- forgery is hard;
- duplication is prevented, or at least detected;
- customers anonymity is preserved;
- the on-line operations on large databases are minimized.

A digital cash scheme consist of three protocols:

- a withdrawal protocol which allows the user U to obtain digital cash from the bank B;

- a payment protocol which allows the user to buy goods from the vendor V in exchange for the digital cash;
- a deposit protocol where the vendor deposits the cash to his account at the bank.

Henceforth it is assumed that B has a public key-private key pair (P_B, S_B). By $\{M\}_B$ we denote the message M together with its signature created by B's private key. The general scheme for the three protocols is given below.

Withdrawal Protocol

(1) U tells to B he would like to withdraw a dollars.
(2) B returns to U randomized bill with the following format: $\{a, r\}_B$. B withdraws a dollars from U's account.
(3) U checks the signature for validity and accepts the bill.

Payment Protocol

(1) U pays to V the bill.
(2) V checks the signature and if it is valid, accepts the bill.

Deposit Protocol

(1) V gives the bill to B.
(2) B checks the signature; if it is valid, then B credits V's account.

If the signature scheme is assumed to be (reasonably) secure, then the digital cash will be hard to forge. However, it is easy to duplicate and double spend the same digital cash. Anonymity is also not preserved, since the bank can link the name of U with the random number r and determine where U spent his cash. Now, we are going to address these problems.

The anonymity problem is solved by using blind signatures. By definition, these are signatures that the signer signs without knowing their content. The bank signs the bill without seeing the content of the bill. In particular, the bank cannot determine where the bill comes from. There is a problem with this. The user can ask the bank make a small withdrawal, say for \$1, and then ask it to sign a \$1000 bill.

Recall that in the RSA-signature scheme, the signature to the message M is $s = m^d \pmod{n}$, where $n = pq$ for some primes p, q, and $ed \equiv 1 \pmod{\varphi(n)}$. Here n and e are publicly known values. Verification consists in comparing $s^e \pmod{n}$ and m. In the case of blind signatures, U wants B to provide a signature s to m without revealing m. Here is a possible anonymous withdrawal protocol.

Withdrawal Protocol

(1) U chooses a random number r, $0 \leq r \leq n$.
(2) U calculates $M' = M \cdot r^e \pmod{n}$.
(3) U sends M to the bank.
(4) B returns a signature s' to M': $s' = (M')^d \pmod{n}$. We have:
$$s' = (M')^d = M^d \cdot r^{ed} = M^d \cdot r \pmod{n}.$$
(5) B debits U's account with \$$a$.
(6) U obtains $s = M^d \pmod{n}$ by computing $M' \cdot r^{-1} \pmod{n}$.

There are two possible solutions of the problem of letting the bank sign higher bills. The first is to have one possible denomination per public key. So the bank provides several public keys and each one of the is used sign one fixed dollar amount only. The second possibility is the so-called "cut-and-choose" procedure. In this case U makes N

bills of $\$a$ each, blinds them all and gives them to the bank picks and signs one bill at random and asks U to open the remaining $N-1$. The signature is returned only if all the unblinded bills were correct. The probability of successful cheating is $\frac{1}{N}$ and can be made sufficiently small by taking N sufficiently large.

The problem of copying and double spending has different solutions in the on-line and off-line versions of the protocol. In the on-line version, the bank is required to record all the bills it receives in a database. During the payment protocol, V would transmit the bill to the bank and ask if the bill was already received. If this is the first time the bill is being used, then it is received; otherwise it is rejected. This simple solution resembles very much to the solution with credit cards when V waits for authorization to finish the transaction. This causes a communication overhead which might cause problems. Also the size of the database to be managed could be problematic.

Now we are going to introduce a protocol from [5] which detects double-spending and does not require an on-line identification. The idea behind off-line digital cash is that during the payment U is forced to write a random identity string on the bill which has the following properties:

- it is different for every two payments;
- only U can create a valid random identity string;
- two different random identity strings on the same coin should allow the bank to retrieve the identity of U.

If the bank receives two identical bills with different random identity strings, then U has cheated and the bank can identify him. If the bank receives two identical bills with the same random identity strings, then V has cheated. The protocols make use of a one-way hash function which we denote by h.

Withdrawal Protocol

(1) U prepares N bills of a dollars each having the following format:

$$M_i = (a, r, y_{i,1}, y'_{i,1}, \ldots, y_{i,K}, y'_{i,K}).$$

Here $y_{i,j} = h(x_{i,j})$, $y_{i,j} = h(x'_{i,j})$, where $x_{i,j}$ are randomly chosen and $x'_{i,j}$ are computed from

$$x_{i,j} \oplus x'_{i,j} = \langle \text{username} \rangle, \quad \forall i = 1, \ldots, N, j = 1, \ldots, K.$$

(2) U blinds all the M_i to random M'_i's (using the blinding protocol above) and sends them to B.
(3) B asks U to open $N-1$ of the N blinded bills.
(4) When U unblinds them, then he also reveals the appropriate $x_{i,j}$ and $x'_{i,j}$.
(5) B checks that these are a dollar bills, that $y_{i,j} = h(x_{i,j}$, $y'_{i,j} = h(x'_{i,j})$, and that $x_{i,j} \oplus x'_{i,j}$ equals U's username.
(6) B returns a signature s of the only unrevealed blind message $M = M_{i_0}$, say.
(7) U checks the signature s on M.

The payment protocol forces U to produce a random identity string on the coin. It is one of $x_{i,j}$ or $x'_{i,j}$ for each $j = 1, \ldots, K$. Which one of the two is to be produced depends on a random challenge from V.

Payment Protocol

(1) U gives (M, s) to V.

(2) V checks the bank's signature on the bill. If it is valid, then V sends U a random bit-string of length K: $b_1 b_2 \dots b_K$.
(3) If $b_i = 0$, then U reveals $x_{i_0,j}$, otherwise he reveals $x'_{i_0,j}$.
(4) V checks that $y_{i_0,j} = h(x_{i_0,j})$ or $y'_{i_0,j} = h(x'_{i_0,j})$ depending on the value of b_j. If the above equalities hold, then he accepts the bill.

The probability that in a different payment of the same random identity string is produced is 2^{-K}, since V chooses his challenge at random. Only U can produce a valid random identity string since the hash function is one-way. Finally, two different random identity strings on the same coin leak the name of U since in this case there is an index j for which we know $x_{i_0,j}$ and $x'_{i_0,j}$.

Deposit Protocol

(1) V brings the coin (M, s) as well as the random identity string to B.
(2) B verifies the signature and checks whether the coin (M, s) has already been returned to the bank.
If the coin is already in the database (of returned coins), then B compares the random identity strings of the two coins. If the random identity strings are different, then U double spent the coin; otherwise, U is trying to deposit the same coin twice.

REFERENCES

[1] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonims. *Communications of the ACM* **28** (1981), 84–88.

[2] D. Chaum. Multiparty unconditionally secure protocols. Pros CRYPTO'87 (ed. C. Pomerance), Lecture Notes in Comp. Science, vol. **293**, 1988, 462.

[3] D. Chaum, J.-H. Evertse, J. van de Graff. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. Advances in Cryptology – EUROCRYPT'87, Springer Verlag, Berlin, 1988, 127–141.

[4] D. Chaum, J.-H. Evertse, J. van de Graff, R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. Advances in Cryptology – CRYPTO'86, Springer Verlag, Berlin, 1987, 200–212.

[5] D. Chaum, A. Fiat, M. Naor. Untraceable electronic cash. Proc. CRYPTO'88 (ed. S. Goldwasser), Lecture Notes in Comp. Science, vol. **403**, 1988, 319–327.

[6] R. Cramer, M. Franklin, B. Schoenmakers, M. Yung. Multi-authority secret-ballot elections with linear work. Advances in Cryptology, EuroCrypt'96, Springer Verlag, Lect. Notes in Compute Science, vol. **1070** (1996), 72–83.

[7] A. Fiat, A. Shamir. How to prove yourself: practical solutions to identification and signature problems. Proc. CRYPTO'86 (ed. A. Odlyzko), Lect. Notes in Comp. Science, vol. **263**, 1987, 186–194.

[8] O. Goldreich. Modern Cryptography, Probabilistic Proofs and Pseudorandomness. Springer Verlag, Berlin-Heidelberg-New York, 1999.

[9] O. Goldreich, S. Micali, A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. Proc. 27th IEEE Symp. on Foundations of Comp. Science, Toronto, 1986, 174–187.

[10] S. Goldwasser, A. Micali. Probabilistic encryption. *Journal of Computer Systems and Science* **28** (1984), 270–299.

[11] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof-systems. Proc. 17th ACM Symp. on Theory of Complexity, 1985, 291–304.

[12] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM J. Computing* **18** (1989), No 1, 186–208.
[13] J. Kilian. Use of Randomness in Algorithms and Protocols. Cambridge, MIT Press, 1990.
[14] I. Landjev. Secret sharing schemes and linear codes over finite fields. *Math. and Education in Math.*, **26** (1997), 13–27.
[15] A. Menezes, P. van Oorschot, S. Vanstone. Handbook of Applied Cryptography. CRC Press Inc., 1997.
[16] A. M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. Advances in Cryptology: Proc. Eurocrypt '84 (eds Th. Beth, N. Cot, I. Ingemarsson) Lecture Notes in Computer Science, vol. **209**, 1985, 224–314.
[17] A. M. Odlyzko. Discrete logarithms: the past and the future. *Designs, Codes and Cryptography* **19** (2000), 129–145.

Ivan Landjev
Institute of Mathematics and Informatics
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: ivan@moi.math.bas.bg

New Bulgarian University
21, Montevideo Str.
1618 Sofia, Bulgaria
e-mail: i.landjev@nbu.bg

КРИПТОГРАФСКИ ПРОТОКОЛ

Иван Ланджев

Настоящата статия е кратко въведение в областта на криптографските протоколи. Тя съдържа кратко описание на някои фундаментални протоколи, като доказателство с нулево разгласяване, интерактивно генериране на случаен бит, схеми за поемане на задължения и др. Като примери за по-сложни протоколи са представени една схема за цифрово гласуване и схема за цифров кеш.