International Journal of Pure and Applied Mathematics

Volume 31 No. 3 2006, 345-360

CRYPTOGRAPHY AND ZERO KNOWLEDGE

R.A. Mollin

Department of Mathematics
University of Calgary
Calgary, Alberta, T2N 1N4, CANADA
e-mail: ramollin@math.ucalgary.ca
url: http://www.math.ucalgary.ca/ř amollin

Abstract: This article is an overview designed to inform the reader concerning the importance of the notion of "zero-knowledge" in cryptography. There are seriously sound reasons for wanting to study aspects of cryptography such as this. As we begin a new millennium, the effects of cryptography on our everyday lives will only increase since we are in the midst of a revolution in information processing and telecommunications. To ever increasing depths, our lives are impacted on a daily basis by interactions that require our sending of digital messages through cyberspace. This may involve the electronic transfer of digital dollars, the sending of personal "e-mail" messages, or the sending of military secrets. What is common to all these types of message-sending is the need to keep these messages secret, and ensure that nobody tampers with the message. Hence, the importance of cryptography to our information-based society will only deepen in the new millennium. It is essential that we are equipped with the knowledge to understand and deal more effectively with the new reality.

AMS Subject Classification: 94A60, 11T71, 68P25
Key Words: zero-knowledge, cryptography

1. Introduction

Let us begin with a description of the terminology to be used. The science of *cryptography* refers to the study of methods for sending messages in *secret* (namely, in *enciphered* or *disguised* form) so that only the intended recipient can

Received: July 22, 2006

remove the disguise and read the message (or *decipher* it). The original message is called the *plaintext*, and the disguised message is called the *ciphertext*. The final message, encapsulated and sent, is called a *cryptogram*. The process of transforming plaintext into ciphertext is called *encryption* or *enciphering*. The reverse process of turning ciphertext into plaintext, which is accomplished by the recipient who has the knowledge to remove the disguise, is called *decryption* or *deciphering*. Anyone who engages in cryptography is called a *cryptographer*. On the other hand, the study of mathematical techniques for attempting to defeat cryptographic methods is called *cryptanalysis*. Those practicing cryptanalysis (usually termed the "enemy") are called *cryptanalysts*.

In what follows, an *entity* will mean any person or thing, such as a computer terminal, which sends, receives, or manipulates information. Inherent in cryptography, is the problem of supplying mutually mistrustful entities with a means of interchanging messages, where each entity desires to reveal as little as possible about its own secrets. For instance, suppose that entity A wants to convince entity B that it has knowledge of the factorization of an integer $n = pq$, which is a product of two distinct large primes p and q. It is certainly simple to do this, since A can merely reveal p and q. However, A has cryptographically sound reasons for not wanting to do this. So how does A proceed to convince B without revealing the factorization? In other words, is it possible for A to prove to B the assertion that it has knowledge of the factorization of n without yielding anything beyond the validity of the assertion? If such a proof exists, it is called a *zero-knowledge* proof. Zero-knowledge protocols (where a *cryptographic protocol* or simply a *protocol* means an algorithm that specifies the actions of two or more entities whose goal is to achieve a given security objective) such as the ones we describe below are designed to address some concerns over password protocols such as the situation where entity A gives password to entity B who can thereafter impersonate A. A practicing cryptographer might argue that zero-knowledge protocols are not in significant use, but this should not detract from their interest since their status is changing. As noted by Crescenzo and Ostrovsky in [8, p. 484]: "Since their introduction, zero knowledge proofs have proven to be very useful as a building block in the construction of cryptographic protocols... Due to their importance, the efficiency of zero-knowledge protocols has received considerable attention."

2. Zero-Knowledge

In order to formally define "zero-knowledge" and related phenomena in mathematical terms, one must first become acquainted with the following definitions.

Definition 1. (Interactive (Minimal Disclosure) Proof Systems) An interactive proof system is a game involving an exchange of messages between an entity A, called a prover who executes a computationally unbounded algorithm (strategy), and an entity B, called a verifier, executing a probabilistic polynomial time strategy. Here A's objective is to convince (or prove to) B the truth of some fact, such as claimed knowledge of some secret. B either accepts or rejects the proof. The interactive proof system is called minimal disclosure also called a proof of knowledge if the following two properties are satisfied.

(1) If the prover does not know the proof, then the chances of convincing the verifier of knowledge of the proof are negligible. In other words, with probability close to 1, B accepts A's proof whenever S is true. This is called the completeness property.

(2) A dishonest prover C (impersonating A), who has a non-negligible probability of successfully executing the protocol must be able to compute S in polynomial time. In other words, any entity C, capable of successfully impersonating A, must essentially "know" S (be able to compute it in polynomial time). This is called the soundness property.

The completeness property is the standard requirement for a protocol to function properly given honest entities, whereas the soundness property prevents a dishonest prover from convincing an honest verifier. Thus, the soundness property ensures that the only aspect of the proof that the verifier gets is that the prover knows the proof. In particular, the verifier cannot convey the proof to anyone else without actually reconstructing the proof from scratch. However, it should be noted that this does not, in itself, guarantee that the protocol is cryptographically secure. In other words, it does not guarantee that the probability of the protocol being defeated is negligible. In order to be cryptographically secure, the underlying mathematical problem faced by an enemy must be computationally hard.

What Definition 1 says is that protocols which are minimum disclosure proofs yield no insights into the proof of an an assertion simply from being convinced that the assertion is true. Whatever one learns from a minimum disclosure proof, one can learn from the assertion itself. Interactive proofs used for *identification* (meaning the corroboration of the identity of an entity) may be reformulated as *proofs of knowledge*. For instance, if entity A, who knows a

secret S, wants to establish its identity to entity B, then A convinces B, via a proof of knowledge protocol, of knowledge of S. Since only A knows S, then A has been *identified* to B.

In order to present an example of an interactive (minimal disclosure) protocol, the following is required. In what follows, the term *computationally infeasible* which means that, given the enormous amount of computer time that would be required to solve the problem, this task cannot be carried out in *realistic* computational time. Thus, "computationally infeasible" means that, although there (theoretically) exists a unique answer to our problem, we cannot find it even if we devoted every scintilla of the time and resources available. This is distinct from a problem that is unsolvable in any amount of time or resources. For example, an unsolvable problem would be to cryptanalyze XYZ assuming that it was enciphered using a monoalphabetic substitution. There is simply no unique verifiable answer without more information. However, it should be stressed here that there is no proved example of a computationally infeasible problem.

Definition 2. (One-Way Functions) A one-to-one function f from a set $\mathcal{M}$ to a set $\mathcal{C}$ is called one-way if $f(m)$ is "easy" to compute for all $m \in \mathcal{M}$, but for a randomly selected $c \in \text{img}(f)$, the image of f, finding an $m \in \mathcal{M}$ such that $c = f(m)$ is computationally infeasible. In other words, one can easily compute f, but it is computationally infeasible to compute f^{-1}.

An example of a one-way function is obtained by randomly selecting large primes p and q and calculating $f(m) \equiv m^2 \pmod{n}$, which is computationally "easy", whereas computing the square root modulo n can be shoown to be computationally equivalent to factoring n, which is deemed to be intractable for an appropriately chosen such modulus.

Based upon Definition 2, an example of Definition 1 is for A to use a one-way function to prove to B that it has a specific morsel of information, but it does not yield any way of determining what that morsel happens to be. If B does not learn *anything at all* other than that A knows a proof, then this is an instance of the next definition, first formalized by Goldwasser, Micali, and Rackoff in [17]–[18] and by Goldreich, Micali, and Wigderson in [14]–[16]

Within the theory of languages (in the complexity theory sense), what Goldwasser, Micali, and Rackoff did was to formalize zero-knowledge proofs in the context of an interactive *proof of membership* of a string x in a language $\mathcal{L}$. They did this by showing that the languages of quadratic residues and quadratic nonresidues each have zero-knowledge interactive proof systems revealing only that $x \in \mathcal{L}$, called *revealing only a single bit of knowledge.*

For the balance of the discussion, a "gain of knowledge" will mean the following. Entity B will be said to have *gained knowledge* from entity A about a fact F if B's computational ability concerning F has increased. In other words, B can compute something about F after interaction with A that was not possible before that interaction. Note that "knowledge" and "information" are not the same. Knowledge pertains primarily to publicly known facts, whereas *information* pertains to facts on which only partial data are publicly known. For instance, if A answers B's queries by telling B the outcome of a coin toss, then B gets from A information concerning the event (from the perspective of information theory). However, B gains *no knowledge* from A, since B can flip coins without A.

Definition 3. A zero-knowledge proof of knowledge is a proof of knowledge that satisfies the additional property:

(3) The verifier learns nothing from the prover that could not have been learned without the prover's participation — called the zero-knowledge property.

A protocol that is a zero-knowledge proof is called an zero-knowledge interactive protocol.

The definition given here reflects the formal notions elucidated in [10], although the literature is not entirely consistent on the notion of zero-knowlege.

Definition 3 tells us that a zero-knowledge interactive protocol allows a proof of the truth of a statement (such as knowledge of a secret), without leaking any information about the statement aside from its validity. A cryptographically similar notion is an answer provided by an *oracle*, which may be an algorithm, or a subroutine in a program, or a trusted entity that can give an answer to any question that a presupposed algorithm is able to settle, without revealing any details of that algorithm (note that an "oracle" may be defined quite generally (without reference to cryptography) as any entity that can provide an answer in negligible time. However, as noted in [19, p. 406], a zero-knowledge proof (in the sense given above) is similar to an answer given by a trusted oracle).

Another way of looking at property (3) in Definition 3 is that B is able to simulate the protocol as if A were actually participating although A, in fact, is not. Often in the literature, a protocol having the zero-knowledge property is defined as a protocol that is simulatable in the sense that there exists an expected polynomial time algorithm, called a *simulator* which takes the assertion to be proven as input, and produces, without interacting with the real prover, output which is indistinguishable from that which would occur from interacting with the real prover. The output, comprising a collection of

messages, resulting from the execution of a protocol is often called a *transcript*. This brings us to another variant of zero-knowledge protocols.

Definition 4. (Computational and Perfect Zero-Knowledge) A protocol is computationally zero-knowledge if an observer (who witnesses an interactive zero-knowledge proof), restricted to probabilistic polynomial-time tests, cannot distinguish simulated from real transcripts. A protocol is perfect zero-knowledge if the probability distributions of the aforementioned transcripts are identical.

Roughly speaking, computational zero-knowledge means that there does not exist an efficient method to distinguish the transcripts mentioned in Definition 4. With perfect zero-knowledge, transcripts are identically distributed rather than computationally indistinguishable. There also exists a notion of *almost-perfect* or *statistical* zero-knowledge. For this notion, the probability distributions of the transcripts must be "statistically indistinguishable" (see [12]–[13]).

In the above discussion was mentioned one application of zero-knowledge proofs, namely *identification*. The following was introduced in [11] by Fiat and Shamir in 1987 as an authentication and digital signature scheme. It was modified in [9]–[10] to a zero-knowledge proof of identity. Also, it is based on the Goldwasser-Micali-Rackoff zero-knowledge proof system for quadratic residuacity (see [17]–[18]). The following is probably the best-known zero-knowledge protocol for identity. Note that zero-knowledge proofs are ideally suited for identification of an owner A (who has a PIN number or password S) of a credit card, ID card, or computer account, by allowing A to convince a merchant B of knowledge of S without revealing even a single bit of S.

In many identification schemes a "judge" is needed to resolve disputes. This judge is sometimes called a "trusted third party" upon whom all parties agree in advance. Another name often used for this disinterested third party is an *arbitrator*, who is needed to complete a protocol. It is essential that this "disinterested" arbitrator have no allegiance to any entity involved in the protocol and has no particular reason to complete the protocol. In this way, all entities participating in the protocol can accept that what is done is not only correct, but also that their portion of the protocol will be complete. In what follows, a TTP will mean a *trusted third party*.

2.1. Feige-Fiat-Shamir Identification Protocol

The following protocol involves a prover A who wants to prove knowledge of a secret s_A to a verifier B.

Setup Stage

(1) An integer $n = pq$, where $p \neq q$ are large primes, is chosen by the TTP and made public, but the p and q are kept secret. Also, a parameter $a \in \mathbb{N}$ is chosen.

(2) A and B select, respectively, a secret $s_A, s_B \in \mathbb{N}$ relatively prime to n with $s_A, s_B \leq n-1$. Then A and B compute $t_A \equiv s_A^2 \pmod{n}$, and $t_B \equiv s_B^2 \pmod{n}$, respectively, and register s_A, s_B, respectively, with the TTP as private keys, where t_A, t_B are their public keys.

Protocol

Execute the following steps:

(1) A selects an $m \in \mathbb{N}$, called a *commitment* such that $m \leq n-1$ and sends $w \equiv m^2 \pmod{n}$, called the *witness*, to B.

(2) B chooses $c \in \{0, 1\}$, called a *challenge*, and sends it to A.

(3) A computes $r \equiv ms_A^c \pmod{n}$, called the *response*, and sends it to B.

(4) B computes r^2 modulo n. If $r^2 \equiv wt_A^c \pmod{n}$, then reset a's value to $a-1$ and go to step (1) if $a > 0$. If $a = 0$, then terminate the protocol with B accepting the proof. If $r^2 \not\equiv wt_A^c \pmod{n}$, then terminate the protocol with B rejecting the proof.

Suppose that an enemy E tries to impersonate A. Then E could try to cheat by selecting any $m \leq n-1$ and sending the witness $w \equiv m^2 t_A^{-1} \pmod{n}$ to B. If B then sends the challenge $c = 0$, E would send the correct answer $r = m$, which passes step (4). However, if $c = 1$ is sent as a challenge, then E , who does not know the square root s_A of t_A, cannot respond correctly. Hence, E has a probability $1/2$ of escaping detection in the first iteration, $1/4$ in the second, and so on. To diminish this kind of arbitrary cheating, to an acceptably small probability of 2^{-a}, one performs iterations of the protocol for a sufficiently high value of a. Thus, the Feige-Fiat-Shamir protocol is a zero-knowledge proof of knowledge of a modular square root of t_A, namely the secret s_A. Given that A is the only entity in possession of knowledge of the square root of t_A, then success in the above protocol is taken as verification that the prover is A. The zero-knowledge property ensures that interacting with A as delineated in the

protocol does not leak information that can be used to impersonate A. The following is an illustration of the above protocol with small parameters.

Example 1. Let $a = 2$, and $n = pq = 101 \cdot 757 = 76457$. The prover A selects $s_A = 3$, so $t_A = 9$, and B selects $s_B = 7$, so $t_B = 49$. In the first round A chooses a commitment $m = 98$ and sends the witness $w = m^2 = 9604$ to B. B chooses the challenge $c = 0$, and A responds with $r = 98 = m \cdot t_A^c$. Then B computes $98^2 = 9604 = w \cdot t_A^c$, so B accepts, and sets $a = 1$. In the second round A selects $m = 747$ and sends $w \equiv m^2 \equiv 22810 \pmod{n}$ to B. B sends $c = 1$ to A, who computes $r \equiv 747 \cdot 3 \equiv 2241 \pmod{n}$, which gets sent to B. Lastly, B computes $r^2 \equiv 52376 \pmod{n}$, checks that $52376 \equiv w \cdot t_A^c \pmod{n}$, and sets $a = 0$. Hence, B accepts A's proof.

The Feige-Fiat-Shamir protocol is representative of a general structure for a larger class of what are called *three-move, zero-knowledge protocols*. In such protocols, A sends a witness, B replies with a challenge, and A replies with a response. One iteration of these "moves" is called a *round*. There is a hidden randomization factor built into the protocol when prover A selects a (private) commitment from some pre-determined set. Then A computes the associated (public) witness. This initial randomness guarantees that the iterations in the protocol are sequential and independent, namely that there is a variation from one protocol round to another. Also, this establishes a set of "questions", which A lays claim to being able to answer, so naturally A's subsequent responses are restricted. The given design of the protocol means that only the prover A with knowledge of the secret S is capable of answering all of the challenges with correct responses, none of which provide information about S. B's challenge selects a question from the set, A provides a response, and B checks that it is valid. The iteration of rounds necessarily restricts the probability of cheating to arbitrarily chosen limits specified by the bound given in the choice of the parameter a.

The process by which A determines the set of questions and B chooses the question, (1)–(2) in the Feige-Fiat-Shamir protocol for example, is an instance of what is called a *cut and choose protocol*. This terminology arises from the classic technique for dividing anything fairly between two people, namely A cuts the object (say a piece of fruit) in "half", and B chooses one of the halves, leaving the other half for A. Also, the process involved in (2)–(3) of the Feige-Fiat-Shamir protocol, given a witness from (1), is an instance of *challenge-response protocols*. In this type of protocol it is essential, for security reasons, that A responds to at most one challenge for a given witness, and should never reuse a witness. Hence, interactive zero-knowledge protocols combine the ideas of

both cut and choose and challenge-response protocols. Other zero-knowledge interactive proofs of identity may be found in [5], [7], [23]–[24], and [28]. See also [19].

In 1978, before the aforementioned formalization by Goldwasser, Micali, and Rackoff of zero-knowledge proofs, Rabin [27] was using the cut and choose protocol for cryptographic applications. Also, it is worth noting that at about the same time as the aforementioned formalization was being developed, Babai and Moran [1]–[2], were trying to formalize the notion of efficient provability. They did this by developing a theory of randomized interactive games, called *Arthur-Merlin games* in an effort to provide a statistical foundation for their notion of efficient provability.

There is a rather informal and informative description of the cut and choose protocol given by Quisquater, Guillou, and Berson in [25] using an analogy of a cave.

2.2. Cut and Choose Protocol — Cave Analogy

The following protocol is illustrated by Figure 1 below.

The setup is that A knows the secret magic words that can open the secret door deep in a cave. A seeks to prove knowledge of the secret to B, but does not want to reveal the magic words. This is how A proceeds. A given parameter $t \in \mathbb{N}$ is chosen and the following steps are executed:

(1) B stands at point P_0 out of sight of point P_1.

(2) A walks all the way into the cave, either to point P_2 or P_3.

(3) After A has disappeared into the cave, B walks to point P_1.

(4) B shouts to A with one of two instructions:

  (a) Leave out of passage number 1, or

  (b) Leave out of passage number 2.

(5) If A obeys, using the magic words to open the door if necessary, then go to step (6). Otherwise, terminate the protocol with B rejecting the proof.

(6) Reset the value of t to $t - 1$ and go to step (1) if $t > 0$. Otherwise, terminate the protocol with B accepting the proof.

In the cave analogy, A cannot repeatedly guess which passage B will choose in step (4). If A did not know the secret words, then A would be forced to come out via the passage entered, but not the other passage. Thus, A has a one in two chance of guessing which passage B will choose in round one, and a one in four chance in round two. Hence, after t rounds, the chance of A fooling B is one in 2^t. Thus, for sufficiently large choice of the parameter t, A has a relatively insignificant chance of fooling B in all t rounds. For instance,

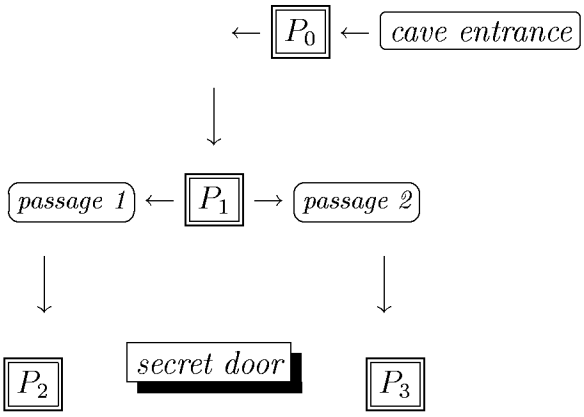

Figure 1: Schematic for the cut and choose cave analogy

if $t = 30$, then A has one chance in $1,073,741,824$ of fooling B for all thirty rounds. (A is more likely to win the lottery!) The attentive reader will note that it was not necessary to go through all the rounds since to convince B, all A has to do is let B witness entrance through one passage and return through the other. This just shows that the cave analogy is imperfect to describe the mathematical reasoning underlying it. Nevertheless, the protocol as illustrated by the constraints in the cave analogy are valid. The following application will give a more mathematical formulation.

First, recall that a *graph* is a set of points, called *vertices* or *nodes* together with a set of lines, called *edges* joining pairs of distinct vertices such that at most one edge joins any pair of vertices. Two vertices are called *adjacent* if there exists an edge joining them. Given vertices V_j for $1 \leq j \leq n \in \mathbb{N}$ on a graph, $(V_1, \ldots, V_n)$ is a *path* from V_1 to V_n if V_k is adjacent to V_{k+1} for all $k = 1, 2, \ldots, n-1$. A *circuit* is a path from a vertex back to itself. A *cycle* is a circuit that visits each node at most once, in other words a circuit without repeated nodes. A *Hamiltonian* cycle in a graph G is a cycle that contains every vertex of G, namely that visits each vertex of G exactly once. There does not exist an efficient method for determining when a given general graph has a Hamiltonian cycle. Thus, for a sufficiently large graph, finding such a cycle in H is computationally infeasible.

2.3. Zero-Knowledge Proofs via Hamiltonian Cycles

Suppose that a prover A has knowledge of a Hamiltonian cycle in a large graph G, and A wants to convince a verifier B of this knowledge without leaking knowledge of the cycle. Then A proceeds as follows. Let $t \in \mathbb{N}$ be a chosen parameter and execute the following steps:

(1) A randomly permutes the vertices and edges of G to create a graph H, which is isomorphic to G (note that determining, for general graphs, an isomorphism between them is a problem for which there is no known polynomial-time algorithm to solve it. Thus, if A did not construct the isomorphism, finding the Hamiltonian cycle would be computationally infeasible. However, A can easily find this cycle in H since the isomorphism that maps the vertices and edges was A's construction).

(2) A gives a copy of H to B.

(3) B asks A to perform one of the following tasks:

(a) Prove that G and H are isomorphic.

(b) Provide a Hamiltonian cycle in H.

(4) Either A responds by either performing one of (a) or (b):

(a) Proving that G and H are isomorphic, but without providing any Hamiltonian cycle in either G or H.

(b) Providing a Hamiltonian cycle in H, without proving that G and H are isomorphic.

or A fails in which case B rejects the proof.

(5) Reset the value t to $t-1$ and go to step (1) if $t > 0$. Otherwise, terminate the protocol with B accepting the proof.

The above protocol is a zero-knowledge proof since B gets no information which would allow the determination of a Hamiltonian cycle for G. If A shows, in step (4), that G and H are isomorphic, this does not assist B since the Hamiltonian cycle is equally difficult to find in either graph. If A provides the Hamiltonian cycle in H, this does not help B either since the problem of determining an isomorphism between the graphs is not much easier than finding the Hamiltonian cycle. Given that, at each round, A generates a new H, then B gets no new information, no matter what the size of the parameter t happens to be.

Determining, for general graphs, an isomorphism between them is a problem for which there is no known polynomial-time algorithm to solve it (however, it is not known to be **NP**-complete, whereas the problem of determining a Hamiltonian cycle in a graph *is* known to be **NP**-complete). Thus, if A did not construct the isomorphism, finding the Hamiltonian cycle would be com-

putationally infeasible. However, A can easily find this cycle in H since the isomorphism that maps the vertices and edges was A's construction.

Suppose that someone, not involved in the above interactive protocols, needs to be convinced of some fact. To do this, one needs a *noninteractive protocol*

2.4. Basic Zero-Knowledge Noninteractive Protocol

Suppose that an entity A has a secret S which is the solution to some hard mathematical problem H (such as the Hamiltonian cycle problem discussed above), and A must convince an entity C of this knowledge without interaction and without revealing any aspect of the secret. This is accomplished when entity C verifies the following steps.

In what follows, a *hash function* is a computationally efficient function that maps bitstrings of arbitrary length to bitstrings of fixed length, called *hash values*. A *one-way hash function* is a hash function that satisfies the property in Definition 2.

(1) A uses S and a random number generator R to transform H into $n \in \mathbb{N}$ isomorphic problems P_j for $j = 1, 2, \ldots, n$. Using R and S, A solves P_j for each $j = 1, 2, \ldots, n$.

(2) A commits to the solutions S_j of the P_j. Call the commitments C_j, which are saved as bitstrings.

(3) A uses the C_j as inputs for a one-way hash function F and saves the first n bits B_j of the output of F for $j = 1, 2, \ldots, n$.

(4) For each $j = 1, 2, \ldots, n$, A takes B_j and:

(a) If $B_j = 0$, then A proves that P_j is isomorphic to H.

(b) If $B_j = 1$, then A demonstrates that S_j is a solution of P_j.

(5) A publishes the C_j from step (2) and the B_j from step 4(b).

The reason that the above protocol is a noninteractive zero-knowledge proof is that A publishes data in step (5) that contains no information about S. Also, A cannot cheat since there is no way of knowing (or forcing) the outcome of the bits from F. In essence, F replaces B (from our interactive protocols) in step (3) of the above noninteractive protocol. The notion of noninteractive zero-knowledge proofs was introduced in 1988 by Blum, Feldman, and Micali in [6].

The last example of a proof of knowledge based upon the following concept. Given an odd prime p, and natural numbers m and c, solving

$$e \equiv \log_m(c) \pmod{p}$$

for the unique integer e with $0 \leq e \leq p - 2$, is called the *discrete logarithm*

problem, or *discrete log*. Another way to think of the discrete log problem based upon the notion of a finite field is the following. Given a prime p, a generator α of $\mathbb{F}_p^*$ and an element $\beta \in \mathbb{F}_p^*$, find the unique nonnegative integer $x \leq p-2$ such that $\alpha^x \equiv \beta \pmod{p}$. If p is "properly chosen", then this is a very difficult problem, for which there is no known polynomial-time algorithm to solve it.

2.5. Zero-Knowledge Proof of Discrete Log

Suppose that entity A has knowledge of x in $a^x \equiv c \pmod{n}$, where $a, c, n \in \mathbb{N}$ are public, $n = pq$ is a product of two distinct large primes, and x is randomly chosen with $\gcd(x, \phi(n)) = \gcd(x, (p-1)(q-1)) = 1$. The following are the steps for convincing entity B of this knowledge without revealing x. Let $t \in \mathbb{N}$ be a chosen parameter and execute the following:

(1) A chooses $y \in \mathbb{N}$ at random such that $y < (p-1)(q-1)$ and computes $z \equiv a^y \pmod{n}$, then sends z to B.

(2) B sends a random bit b to A.

(3) A computes $w \equiv y + bx \pmod{(p-1)(q-1)}$ and sends w to B.

(4) If B confirms that $a^w \equiv zc^b \pmod{n}$, then go to step (5). Otherwise, terminate the protocol with B rejecting the proof.

(5) Reset the value of t to $t-1$, and go to step (1) if $t > 0$. Otherwise, terminate the protocol with B accepting the proof.

The above discussion barely touches upon the complex issues involved in the rapidly-developing area of zero-knowledge. The reader may go to any number of the references given herein to expand horizons in this regard. There are numerous other aspects not mentioned herein such as *multi-prover proof systems* (see [4] for instance), and a variety of other deep and important connections that the reader may touch upon by further reading in this area.

Acknowledgments

The author's research is supported by NSERC Canada grant # A8484.

References

[1] L. Babai, Trading group theory for randomness, In: *Proceed. 17-th ACM Symp. on Theory of Computing* (1985), 421-429.

[2] L. Babai, S. Moran, Arthur Merlin games: a randomized proof system and a hierarchy of complexity classes, *Journal of Computer and System Sciences*, **36** (1988), 254-276.

[3] M. Bellare, O. Goldreich, On defining proofs of knowledge, In: *Advances in Cryptology – CRYPTO '92*, Springer-Verlag, Berlin, LNCS, **740** (1993), 390-420.

[4] M. Ben-Or, S. Goldwasser, J. Kilian, A. Wigderson, Multi-prover interactive proofs: How to remove intractability, In: *Proceed. 20-th ACM Symp. on Theory of Computing* (1988), 113-131.

[5] T. Beth, Efficient zero-knowledge identification scheme for smart cards, In: *Advances in Cryptology – Eurocrypt '88*, Springer-Verlag, Berlin, LNCS, **330** (1988), 77-84.

[6] M. Blum, P. Feldman, S. Micali, Noninteractive zero-knowledge and its applications, In: *Proceed. 20-th ACM Symp. on Theory of Computing* (1988), 103-112.

[7] E.F. Brickell, K.S. McCurley, An interactive scheme based on discrete logarithms and factoring, *Journal of Cryptology*, **5** (1992), 29-39.

[8] G. Di Crescenzo, R. Ostrovsky, On concurrent zero-knowledge with preprocessing, In: *Advances in Cryptology – CRYPTO '99*, Springer-Verlag, Berlin, LNCS, **1666** (1999), 485-502.

[9] U. Feige, A. Fiat, A. Shamir, Zero-knowledge proofs of identity, In: *Proceed. 19-th ACM Symp. on Theory of Computing* (1987), 210-217.

[10] U. Feige, A. Fiat, A. Shamir, Zero-knowledge proofs of identity, *Journal of Cryptology*, **1** (1988), 77-94.

[11] A. Fiat, A. Shamir, How to prove yourself: Practical solution to identification and signature problems, In: *Advances in Cryptology – CRYPTO '86*, Springer-Verlag, Berlin, LNCS, **263** (1987), 186-194.

[12] O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudo-Randomness*, Springer-Verlag, Berlin (1999).

[13] O. Goldreich, A. Sahai, S. Vadhan, Can statistical zero knowledge made non-interactive? Or on the relationship of SZK and NISK, In: *Advances in Cryptology – CRYPTO '99*, Springer-Verlag, Berlin, LNCS, **1666** (1999), 467-484.

[14] O. Goldreich, S. Micali, A. Wigderson, Proofs that yield nothing but their validity and a methodology of cryptographic protocol design, In: *Proceed. of 27-th IEEE Annual Symp. on the Foundations of Computer Science* (1986), 174-187.

[15] O. Goldreich, S. Micali, A. Wigderson, How to prove all NP statements in zero-knowledge, and a methodology of cryptographic protocol design, In: *Advances in Cryptology – CRYPTO '86*, Springer-Verlag, Berlin, LNCS, **263** (1987), 171-185.

[16] O. Goldreich, S. Micali, A. Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems, *J. Assoc. Comp. Mach.*, **38** (1991), 691-729.

[17] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, In: *Proceed. 17-th ACM Symp. on Theory of Computing* (1985), 291-304.

[18] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM Journal Computing*, **18** (1989), 186-208.

[19] A.J. Menezes, P.C. Van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, New York, London, Tokyo (1997).

[20] R.A. Mollin, *Quadratics*, CRC Press, Boca Raton, New York, London, Tokyo (1996).

[21] R.A. Mollin, *Fundamental Number Theory with Applications*, CRC Press, Boca Raton, New York, London, Tokyo (1998).

[22] R.A. Mollin, *Algebraic Number Theory*, Chapman and Hall/CRC Press, Boca Raton, New York, London, Tokyo (1999).

[23] K. Ohta, T. Okamoto, Practical extension of Fiat-Shamir scheme, *Electronics Letters*, **24** (1988), 955-956.

[24] K. Ohta, T. Okamoto, A modification of the Fiat-Shamir scheme, In: *Advances in Cryptology – CRYPTO '88*, Springer-Verlag, Berlin, LNCS **403** (1990), 232-243.

[25] J.-J. Quisquater, M. Guillou, T. Berson, How to explain zero-knowledge to your children, In: *Advances in Cryptology – CRYPTO '89*, Springer-Verlag, Berlin, LNCS **435** (1990), 628-631.

[26] J.-J. Quisquater, M. Guillou, A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory, In: *Advances in Cryptology – Eurocrypt '88*, Springer-Verlag, Berlin, LNCS, **330** (1988), 123-128.

[27] M.O. Rabin, Digital signatures, In: *Foundations of Secure Computation*, Academic Press, New York (1978), 155-168.

[28] C.P. Schnorr, Efficient identification and signatures for smart cards, In: *Advances in Cryptology – CRYPTO '89*, Springer-Verlag, Berlin, LNCS, **435** (1990), 239-252.