ZERO-KNOWLEDGE AUTHENTICATION BY THE SHERLOCK HOLMES METHOD

DIMA GRIGORIEV AND VLADIMIR SHPILRAIN

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth."
Arthur Conan Doyle, The Sign of Four

Abstract. We propose a class of authentication schemes that are zero-knowledge by design, which means they are *actually* zero-knowledge, as compared to "provably" zero-knowledge, where the latter usually means, in the cryptographic lingo, that retrieving any information about the prover's long-term private key from an authentication session is at least as hard as solving some problem that is "believed" to be hard. The principal idea behind our schemes is: the verifier challenges the prover with a question that has only a small number of possible answers (say, just 2), and such that *the verifier himself knows the right answer.* The prover then responds with one of the possible answers, and the verifier compares it to the answer he already knew. We prove that no information about the prover's long-term private key can possibly be leaked during such an authentication session.

The schemes proposed in this paper can also be used for encryption.

1. Introduction

For a general theory of public-key authentication (a.k.a. identification) as well as early examples of authentication protocols, the reader is referred to [8]. In this paper, we propose a class of authentication schemes that are zero-knowledge by design, because the prover gives to the verifier a "yes" or "no" answer without giving any proof of the validity of the answer. Moreover, the verifier knows the correct answer *before* communicating his challenge to the prover, which means he cannot possibly obtain any new information from the prover's response.

The reason why we call our idea of authentication the "Sherlock Holmes method" is the following. It is the case with most natural decision problems in algebra (such as the identity problem, the conjugacy problem, the membership problem, etc.) that, for a generic input, the "no" answer can be obtained much more efficiently than the "yes" answer. Thus, if the prover "eliminates the impossible" by (efficiently) getting the "no" answers whenever she can, she is left with the only remaining possibility for a "yes" answer. We note that in a typical concrete realization of this idea, the prover will not be able to give a "yes" answer by any other method than "eliminating the impossible", which is why we call it the "Sherlock Holmes method."

To conclude the Introduction, we summarize what we think are the most interesting features of our proposal:

(1) Malicious verifier cannot possibly obtain from the prover any information that he does not already know.
(2) There is no "concrete" problem for the adversary to solve in order to obtain the prover's long-term private key. The problem he/she faces is to obtain a test for non-membership in a set that he/she does not know.

Finally, we note that our general scheme can also be used for encryption, see Remark 1 at the end of section 2.

2. The meta-protocol

In this section, we give a description of our general idea of "authentication by the Sherlock Holmes method", leaving particular realizations to the next sections. Here Alice is the prover and Bob the verifier.

Alice's private key consists of: (a) a subset S_0 of some "universal" set S; (b) an efficient test telling that a given element of S does *not* belong to S_0; (c) a way to disguise S_0 to some S_0'.

Alice's public key is a pair of sets S_0', S_1 that either are disjoint or have negligible intersection. Both sets are given to the public in such a way that it is possible to efficiently select a random element from either set.

We note that the idea of a "non-membership test" for S_0 can be expressed in a more precise language. Alice should have her private *separator* T, such that $S_0 \subset T$, the intersection $T \cap S_1$ is negligible, and T is a "nice" set in the sense that the problem of membership in T is efficiently solvable. Thus, Alice can efficiently check membership of an element x in question in the set T; then, if $x \notin T$, she knows for sure that $x \notin S_0$. If $x \in T$, then she *assumes* that $x \in S_0$; the smaller T is, the more chances this assumption has to be correct. Thus, even though there might be many different separators for a given pair of sets, a good separator is hard for the adversary to find without knowing the set S_0.

The protocol itself is the following sequence of steps.

(1) Bob selects a random element x from either S_0 or S_1 and sends it to Alice.
(2) Alice checks, using her private test, whether $x \notin S_0$. If, indeed, $x \notin S_0$, she sends "1" to Bob, meaning that $x \in S_1$. If her test fails, Alice assumes that $x \in S_0$, and sends "0" to Bob.
(3) Bob, who knows the right answer, simply compares it to Alice's response and accepts or rejects authentication accordingly.

To prevent the adversary from guessing the right answer with non-negligible probability, several rounds of this protocol have to be run; this is similar to the Feige-Fiat-Shamir scheme [1].

Since Bob does not obtain from Alice any information that he does not already know, the following is obvious:

Proposition 1. *No information about the prover's private key is leaked during any authentication session in the above authentication scheme.*

Our construction is therefore perfectly zero-knowledge.

To conclude this section, we make a couple of remarks.

Remark 1. *The protocol in this section can also be used for encryption. Namely, if Bob wants to transmit an encrypted bit to Alice, he sends her a random element from S_0 in case he wants to transmit "0", and a random element from S_1 in case he wants to transmit "1".*

Remark 2. *The protocol in this section admits the following modification. Instead of having the private key of just 2 sets, Alice (the prover) can have several private sets $S_1, \ldots, S_k$, together with private tests, for each i, detecting that a given element does not belong to S_i. This will increase the size of Alice's private and public keys, but at the same time this will decrease the adversary's chances to guess the right answer in a single round of the protocol.*

In the following sections, we showcase three particular realizations of our meta-protocol to illustrate the diversity of possible applications of our main idea.

3. A particular realization: subset sum

In this section, we offer a particular realization of the meta-protocol from Section 2, exploiting the hardness of the subset sum problem, see e.g. [2]. We note that the complexity of this particular problem was previously used in [5] in different cryptographic contexts, namely for constructing a pseudo-random generator and a universal one-way hash function.

The "universal" set S in this section is the set of all m-tuples of m-dimensional vectors over $\mathbf{Q}$.

Alice's private key is a set $S_0 = \{a_1, \ldots, a_m\}$ of m random linearly independent (over $\mathbf{Q}$) m-dimensional vectors *with integer coordinates*, which is therefore a basis of $\mathbf{Q}^m$. However, the vector a_1 is special: the g.c.d. of its coordinates is $2r$ for some positive integer r.

Alice's public key includes the vector a_1 and a set of $k > m$ random vectors $c_1, \ldots, c_k$ from the $\mathbf{Z}_+$-span of S_0.

Now we give a description of the authentication protocol.

(1) Bob selects, with equal probabilities, either a random vector $c \in Span_{\mathbf{Z}+}(a_1, c_1, \ldots, c_k)$ or a random vector $c \in Span_{\mathbf{Z}+}(\frac{1}{2}a_1, c_1, \ldots, c_k)$ and sends the vector c to Alice. Here $Span_{\mathbf{Z}+}$ denotes the set of all linear combinations of given vectors *with nonnegative integer coefficients.*
(2) Alice, using standard linear algebra, finds (rational) coordinates of c in the basis S_0. If at least one of these coordinates is not a nonnegative integer, she knows that $c \notin Span_{\mathbf{Z}+}(a_1, c_1, \ldots, c_k)$; therefore, she sends "1" to Bob. If all coordinates are nonnegative integers, Alice assumes that $c \in Span_{\mathbf{Z}+}(a_1, c_1, \ldots, c_k)$, and sends "0" to Bob.
(3) Bob, who knows the right answer, simply compares it to Alice's response and accepts or rejects authentication accordingly.

We note that there is a negligible probability for Bob to reject a legitimate Alice because it may happen that all coordinates of c in the basis S_0 are nonnegative integers, but $c \notin Span_{\mathbf{Z}+}(a_1, c_1, \ldots, c_k)$. It may, in fact, even happen (again, with negligible probability) that $c \in Span_{\mathbf{Z}+}(a_1, c_1, \ldots, c_k)$, but Bob expected Alice to respond with a "1" because he selected his $c \in Span_{\mathbf{Z}+}(\frac{1}{2}a_1, c_1, \ldots, c_k)$.

We also note that the reason for using a public vector a_1 with g.c.d. of coordinates equal to $2r$ is to have Bob's vector c in $Span_{\mathbf{Q}+}(a_1, c_1, \ldots, c_k)$ in either case, because there is a polynomial-time test detecting whether or not a given vector belongs to the $\mathbf{Q}_+$-span of other given vectors (cf. linear programming problem), see [6] or [10].

Finally, we note that the problem that the adversary who wants to impersonate the prover faces is the following: find out whether or not the matrix equation $Bx = c$ has a solution for x as a vector with nonnegative integer coordinates. Here B is the matrix made up of coordinates of the vectors $a_1, c_1, \ldots, c_k$, c is the challenge vector selected by Bob, and x is the vector unknown to both the prover and the adversary. A special case of this problem, where B is just a vector with integer coordinates, x is a 0-1 vector, and c is just an integer, is known as the *subset sum problem* and is NP-complete, see e.g. [2]. Moreover, as pointed out, for example, in [3, p.41], it appears that the subset sum problem might be hard on random instances, not just on some carefully selected ones.

3.1. **Suggested parameters and key generation.** Suggested parameter values for the protocol above are:

(1) The dimension of vectors is $m = 20$.
(2) Coordinates of the vectors a_i: random nonnegative integers ≤ 10. We note that m random m-dimensional vectors like that are going to be linearly independent with overwhelming probability.
(3) Vectors c_i are constructed by Alice as random linear combinations of the vectors a_i with nonnegative integer coefficients ≤ 10. The number of vectors c_i is $k = 2m$.
(4) Bob constructs his vector c as a random linear combination of the public vectors with nonnegative integer coefficients ≤ 10, with one exception: according to the protocol description, he may choose the coefficient at a_1 to be of the form $\frac{n}{2}$, where n is odd, $1 \leq n \leq 19$.

4. A particular realization: polynomial equations

In this section, we offer another particular realization of the meta-protocol from Section 2.

Alice's private key consists of: (i) a polynomial $h(x_1, \ldots, x_k)$ over $\mathbf{Z}$; (ii) a large prime p.

Alice's public key includes: (i) polynomial $f(x_1, \ldots, x_k) = (h(x_1, \ldots, x_k))^2 - c \ (mod\ p)$. Thus, for any $x_1, \ldots, x_k \in \mathbf{Z}$, there is $u \in \mathbf{Z}$ such that $f(x_1, \ldots, x_k) + c = u^2 (mod\ p)$; (ii) a random polynomial $g(x_1, \ldots, x_k)$ with the same collection of monomials as f.

Now we give a description of the authentication protocol.

(1) Bob selects random integers $x_1, \ldots, x_k$ and plugs them, with equal probabilities, into either f or g. He then sends the result, call it $Bob(x_1, \ldots, x_k)$, to Alice.
(2) Alice computes $a = Bob(x_1, \ldots, x_k) + c \ (mod\ p)$ and checks whether or not a is a square modulo p. If not, she knows that $Bob(x_1, \ldots, x_k) \neq f(x_1, \ldots, x_k)$ and sends "1" to Bob. If it is, Alice assumes that $Bob(x_1, \ldots, x_k) = f(x_1, \ldots, x_k)$ and sends "0" to Bob.
(3) Bob, who knows the right answer, simply compares it to Alice's response and accepts or rejects authentication accordingly.

The way Alice checks whether or not a is a square modulo p is as follows. She raises a to the power of $\frac{p-1}{2}$. If the result is equal to 1 modulo p, then a is a square modulo p; if not, then not.

Again, we note that there is a negligible probability for Bob to reject a legitimate Alice because it may happen that $Bob(x_1, \ldots, x_k) + c$ is a square modulo p, but $Bob(x_1, \ldots, x_k) = g(x)$.

4.1. **Suggested parameters and key generation.** Suggested parameter values for the protocol above are:

(1) The number k of variables: between 3 and 5.
(2) The value of p: on the order of 2^t, where t is the security parameter.
(3) The degree of Alice's private polynomial h: between 2 and 3. The magnitude of its coefficients: at least $\frac{p}{2}$.
(4) Bob generates his integers $x_1, \ldots, x_k$ uniformly randomly from the interval $[1, 2^{\frac{t}{k}}]$.

Remark 3. *The adversary may try to attack Bob's challenge by solving one of the equations $f(x_1, \ldots, x_k) = Bob(x_1, \ldots, x_k)$ or $g(x_1, \ldots, x_k) = Bob(x_1, \ldots, x_k)$ for integers $x_1, \ldots, x_k$, or just try to find out whether either of these equations has integer solutions. The corresponding decision problem (the Diophantine problem, or Hilbert's 10th problem) is known to be undecidable, see* [7]. *In our situation, however, adversary actually faces a promise problem since he/she knows that at least one of the equations has integer solutions. Furthermore, in our situation the range for the unknowns is bounded. Still, the "bounded" Diophantine problem is known to be NP-hard, see e.g.* [2], *which makes this kind of attack look infeasible.*

5. A particular realization: matrices

In this section, we offer yet another particular realization of the meta-protocol from Section 2 using a ring of matrices as the platform. Matrices have been occasionally used in the literature as platforms for various cryptographic primitives, see e. g. [4], [11], or [9] for a general discussion.

The "universal" set S here is the semigroup of all 2×2 matrices over R, the ring of truncated k-variable polynomials over the ring $\mathbf{Z}_2$. Truncated (more precisely, N-truncated) k-variable polynomials over $\mathbf{Z}_2$ are elements of the factor algebra of the algebra $\mathbf{Z}_2[x_1, \ldots, x_k]$ of k-variable polynomials over $\mathbf{Z}_2$ by the ideal generated by all monomials of degree N. In other words, N-truncated k-variable polynomials are expressions of the form $\sum_{0 \le s \le N-1} a_{j_1 \ldots j_s} \cdot x_{j_1} \cdots x_{j_s}$, where $a_{j_1 \ldots j_s}$ are elements of $\mathbf{Z}_2$, and x_{j_s} are variables.

To make computation efficient for legitimate parties, we suggest to use *sparse* polynomials as entries in participating matrices. This means that there is an additional parameter d specifying the maximum number of non-zero coefficients in polynomials randomly generated by Alice or Bob. Note that the number of different monomials of degree N in k variables is $M(N,k) = \binom{N+k}{k}$. This number grows exponentially in k (assuming that N is greater than k). The number of different collections of d monomials (with non-zero coefficients) of degree $< N$ is more than $\binom{M(N,k)}{d}$, which grows exponentially in both d and k. Concrete suggested values for parameters are given below; right now we just say that, if we denote the *security parameter* by t, we suggest that the number $M(N,k) = \binom{N+k}{k}$ is at least t. At the same time, neither N nor k should exceed t.

Now we get to describing Alice's private key. The set S_0 is a finitely generated (by matrices $G_1, \ldots, G_n$) ring of 2×2 matrices over a subring $R_0 \subset R$ that consists of all N-truncated k-variable polynomials over $\mathbf{Z}_2$ with one particular variable x_j missing. This particular variable x_j is selected by Alice uniformly randomly from the set of k variables.

Alice's way to disguise S_0 is conjugation by an invertible 2×2 matrix X.

Alice's private test for a given matrix *not* to belong to S_0 is simple: if at least one entry of the given matrix depends on x_j, then the matrix does not belong to S_0.

Now we get to Alice's public key. She publishes two sets of matrices: (1) the set $\varphi(S_0)$ is given by n generating matrices $X^{-1}G_1X, \ldots, X^{-1}G_nX$, and (2) the set S_1 is given by n generating matrices $H_1, \ldots, H_n$.

In the following subsection, we are going to describe how the matrices G_i, H_i and X are selected. Here we give a description of the authentication protocol:

(1) Bob selects a matrix M, from either S_0 or S_1, as a sum of random products of published matrices in the relevant set.
(2) Alice computes $M_1 = XMX^{-1}$ and then checks, by inspection, whether there is an entry of the matrix M_1 that depends on x_j. If there is such an entry, she sends "1" to Bob, meaning that $M \in S_1$. If not, Alice assumes that $x \in S_0$, and sends "0" to Bob.
(3) Bob, who knows the right answer, simply compares it to Alice's response and accepts or rejects authentication accordingly.

We note that there is a negligible probability for Bob to reject a legitimate Alice because it may happen that all entries of a matrix M_1 from S_1 do not depend on x_j.

5.1. **Generating matrices.**

Our notation here follows that of Section 5. First we describe how to generate the matrices $G_1, \ldots, G_n$. To prevent the adversary from obtaining any information about these matrices from their conjugates by computing the trace or the determinant, we want the trace and the determinant to be equal to

0. To that effect, each matrix G_i is selected in the following form: $\begin{pmatrix} p_iq_i & q_i^2 \\ -p_i^2 & -p_iq_i \end{pmatrix}$, where p_i and q_i are random polynomials that do not depend on a particular, randomly selected, variable x_j, the same for all matrices G_i. Clearly, any matrix G_i of this form has both the determinant and the trace equal to 0.

For efficiency reasons, we require that polynomials p_i and q_i are $\sqrt{d}$-sparse N-truncated k-variable polynomial over $\mathbf{Z}_2$, which are generated the obvious way. Namely, one first chooses $\sqrt{d}$ random monomials of degree at most $N-1$, then randomly chooses non-zero coefficients from $\mathbf{Z}_2$ for these monomials.

Then, Alice generates her private matrices $H_1, \ldots, H_n$ in the same form as she generated the matrices G_i, only this time, after the polynomials p_i and q_i are selected, one monomial involving x_j is introduced either in p_i or in q_i.

An invertible matrix X can be generated as a random product of m *elementary* matrices. A square matrix is called elementary if it differs from the identity matrix by exactly one non-zero element outside the diagonal. This single non-zero element is generated as described above. Denote by $E_{ij}(u)$ the elementary matrix that has $u \neq 0$ in the (i,j)th place, $i \neq j$.

We note that multiplying m elementary matrices may result in the number of non-zero coefficients in some of the entries growing exponentially in m. More precisely, when we multiply $E_{ij}(u)$ by $E_{jk}(v)$, one of the entries in the product involves uv, and the polynomial uv is no longer d-sparse, but d^2-sparse. However, this phenomenon is limited to products of elementary matrices of the form $E_{ij}(u) \cdot E_{jk}(v)$, and the expected maximum length of such "matching" chains in a product of m elementary 2×2 matrices is $\frac{m}{2}$. We therefore require that $d^{\frac{m}{2}} \cdot k \cdot \log N \cdot 4 < t$, where t is the efficiency parameter.

5.2. **Suggested parameters.** Suggested values for parameters of our scheme in this section are:

(1) Presently, $N = 1000$, $d = 25$, and $k = 10$ should be quite enough to guarantee security against "brute force" attacks. In particular, with these values of parameters, the number $M(N,k)$ of different monomials is greater than 10^{20}.
(2) The suggested number n of matrices published by Alice is 5.
(3) The matrix X is generated by Alice as a product of m random elementary matrices, where the value for m is randomly selected from the interval $8 \leq m \leq 16$.
(4) Bob's challenge M is selected as a sum of random products of published matrices in the relevant set S_i. The suggested number of products in such a sum is between 3 and 5, and the suggested number of factors in a product is between 5 and 10.

Acknowledgement. Both authors are grateful to Max Planck Institut für Mathematik, Bonn for its hospitality during the work on this paper.

References

[1] U. Feige, A. Fiat and A. Shamir, *Zero knowledge proofs of identity,* Journal of Cryptology **1** (1987), 77–94.

[2] M. Garey, J. Johnson, *Computers and Intractability, A Guide to NP-Completeness*, W. H. Freeman, 1979.

[3] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*, Cambridge University Press, 2007.

[4] D. Grigoriev, I. Ponomarenko, *Constructions in public-key cryptography over matrix groups*, Contemp. Math., Amer. Math. Soc. **418** (2006), 103–119.

[5] R. Impagliazzo, M. Naor, *Efficient cryptographic schemes provably as secure as subset sum*, J. Cryptology **9** (1996), 199–216.

[6] L. G. Khatchyian, *A polynomial algorithm in linear programming*, Doklady Akad. Nauk USSR, **244** (1979), 1093–1096 (Russian). [Translated as Soviet Math. Doklady, 20, 191–194.]

[7] Yu. Matiyasevich, *Hilbert's 10th Problem (Foundations of Computing)*, The MIT Press, 1993.

[8] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC-Press 1996.

[9] A. G. Myasnikov, V. Shpilrain, and A. Ushakov, *Group-based cryptography*, Birkhäuser 2008.

[10] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley 1998.

[11] V. Shpilrain and A. Ushakov, *An authentication scheme based on the twisted conjugacy problem*, in: ACNS 2008, Lecture Notes Comp. Sc. **5037** (2008), 366-372.

CNRS, Mathématiques, Université de Lille, 59655, Villeneuve d'Ascq, France
E-mail address: `dmitry.grigoryev@math.univ-lille1.fr`

Department of Mathematics, The City College of New York, New York, NY 10031
E-mail address: `shpil@groups.sci.ccny.cuny.edu`