Zero-Knowledge for Multivariate Polynomials

Valérie Nachef[1], Jacques Patarin[2], Emmanuel Volte[1]

[1] Department of Mathematics, University of Cergy-Pontoise, CNRS UMR 8088
2 avenue Adolphe Chauvin, 95011 Cergy-Pontoise Cedex, France
[2] PRISM, University of Versailles
45 avenue des Etats-Unis, 78035 Versailles Cedex, France
valerie.nachef@u-cergy.fr, emmanuel.volte@u-cergy.fr
jacques.patarin@prism.uvsq.fr

Abstract. In [15] a protocol $ZK(2)$ using zero-knowledge argument of knowledge was designed from a solution of a set of multivariate quadratic equations over a finite field (i.e. from MQ problem). In this paper, we propose a new scheme $ZK(d)$ which is a generalization of $ZK(2)$, i.e. we consider systems of polynomials of degree d. The key idea of the scheme $ZK(d)$ is to use a polarization identity that allows to get a d-linear function and then use a cut-and-choose technique. We also observe that the scheme $\tilde{ZK}(d)$, which is the natural generalization of the protocol based on the MQ problem to higher degree, is more efficient in terms of computations whereas the $ZK(d)$ scheme is better in terms of bits to be sent. Moreover these properties are still true for all kinds of polynomials: for example if the polynomials are sparse or dense. Finally, we will present two examples of applications: with Brent equations, or with morphisms of polynomials.

Key words: Authentication scheme, Zero-Knowledge, Multivariate polynomials.

1 Introduction

The first protocols using zero-knowledge arguments of knowledge were based on the factorization problem, for example Fischer-Micali-Rackoff in 1984, or Fiat-Shamir in 1986, or the Graph Isomorphism Problem. However the factorization problem is not expected to be an NP complete problem (since it is in NP and Co NP) and it has sub-exponential algorithms (such as NFS) and even polynomial algorithms on quantum computers (Shor algorithm). Then, it was proved in 1991 by O. Goldreich, S. Micali and A. Widgerson that any problem of NP has a zero-knowledge proof [6]. But the general construction (cf [6]) of zero-knowledge proofs of knowledge from any problem of NP is usually not very efficient. This is why various zero-knowledge proof of knowledge have been specifically designed from some well suited and well chosen NP complete schemes based on simple combinatorial problems expected to be exponentially difficult, such as PKP of Adi Shamir [16], PP of David Pointcheval [13] or CLE [18] or SD [17] of Jacques Stern for example. In [15] such a scheme was designed from the MQ problem, i.e.

the problem of finding a solution from a set of multivariate quadratic equations over a finite field. This MQ problem is related to various primitives in cryptography [2, 8, 10, 11], and is NP-complete over any finite field [4, 12].
However, in some cases, it is really interesting to manipulate higher degree systems. For example, when the systems are sparse, or when non random equations with high degree arise naturally like with Brent equations as we will see in this paper. Moreover, the evolution of the algorithms that solve multivariate polynomial equations (we will define this problem as the "MPol" problem) for example with Gröbner basis might imply that we need in the future to make different choices regarding the number of variables, the degree, the field or the type of equation.
This problem was also studied independently of us, and recently in [14] the author designs a public-key identification scheme based on multivariate cubic polynomials. But his technique can not be generalized to multivariate polynomials of degree $d \geq 4$. For example in [14] p. 187 it is written: "*An open problem.* Efficient constructions based on multivariate polynomials of degree ≥ 4 remains as an open problem. However it might be difficult to construct them by using techniques similar to those of [15] or of ours".
In this paper, we will generalize the construction of [15] in order to design a zero-knowledge argument of knowledge from a solution of any set of multivariate polynomials of degree d over a finite field (i.e. not only $d = 2$ or $d = 3$). We consider two schemes extending the results of [15] and [14]. First we define the scheme called $ZK(d)$, for which the key new idea is to use a polarization identity that allows to consider any degree d. In [14] the author uses a linearization of one of the variables. Our polarization technique is more general. Our second scheme, called $\tilde{ZK}(d)$, is the natural generalization of the protocol based on the MQ problem to higher degree. This scheme is "quasi-optimal" in terms of computations, i.e. the number of computations is just proportional to the time to compute the polynomials on the given point. However this scheme was mentioned independently of us in [14] and it was pointed out that it is not efficient (in terms of bits to be sent).
For practical applications the case $d = 3$ (i.e. cubic equations) is particularly important, since from these polynomials we will be able to design zero-knowledge argument of knowledge based on the (NP-complete) Morphism problem (MP) or from the Brent equations related to the optimal way to solve sets of linear equations (i.e. improvements of the Gauss elimination). We will explain in this paper why these two problems are really interesting for cryptography. We can notice that MP (morphism of Polynomial) is NP hard while IP (isomorphism of Polynomials) is expected not to be NP hard (since it has an Arthur-Merlin game for yes or no answers).

2 Zero-Knowledge Arguments of Knowledge and Commitments

In an interactive protocol, there are two entities: the prover and the verifier. The prover wants to convince the verifier that she knows a secret. Both interact and at the end, the verifier accepts or refuses. In zero-knowledge argument of knowledge, there is a possibility of fraud. A cheater will be able to answer some of the questions (but not all of them). The zero-knowledge arguments of knowledge must be designed such that an answer to one of the questions does not give any indication on the secret but if someone is able to answer all the questions then this will reveal the prover's secret. Informally, the security requirements of a zero-knowledge argument of knowledge are the following (for a more formal reference, see [5]):

1. The zero-knowledge argument of knowledge has **perfect correctness** if a legitimate prover is always accepted.
2. The zero-knowledge argument of knowledge is **statistically zero knowledge** if there exists an efficient simulating algorithm U such that for every feasible Verifier strategy V, the distributions produced by the simulator and the proof protocol are statistically indistinguishable.
3. The zero-knowledge argument of knowledge has **knowledge error** α if there is a knowledge extractor K and a polynomial Q such that if p denotes the probability that K finds a valid witness for x using its access to a prover P^* and p_x denotes the probability that P^* convinces the honest verifier on x, and $p_x > \alpha$, then we have $p \geq Q(p_x - \alpha)$. We emphasize that our scheme has only computational soundness, i.e. the existence of the knowledge extractor will be based on some computational assumption.

In our zero-knowledge arguments of knowledge, we will need string commitment schemes. A string commitment function is denoted by Com. The commitment scheme runs in two phases. In the first phase, the sender computes a commitment value $c = Com(s; \rho)$ and sends c to the receiver, where s is the committed string and ρ is a random string. In the second phase, the sender gives (s, ρ) and the receiver verifies if $c = Com(s; \rho)$. We require the two security properties, statistically hiding and computationally binding. Informally, the first property means that, at the end of the first phase, no receiver can distinguish two commitments values generated from two different strings even if the receiver is computationally unbounded. The second property means that no polynomial-time sender can change the committed string after the first phase. The formal definition and a practical construction of such a commitment is given in [7]. We will assume the existence of such a commitment scheme which can be constructed from a collision resistant hash function [7].

3 Systems of Multivariate equations of degree d and the family of functions $\mathcal{MP}ol(d, n, m, \mathbb{F}_q)$.

We denote by MPol(d) the problem of finding at least one solution for a system of multivariate polynomials of degree d over finite field. Then we have MPol(2)=MQ for the quadratic case and MPol(3)=MC for the cubic case. More generally, we will denote simply by "MPol", the problem of finding at least one solution for a system of multivariate polynomials over finite fields. In "MPol" the degree d is not necessary a critical parameter, for example if there is only one monomial of degree d.
We consider the following function of degree d from $\mathbb{F}_q^n$ to $\mathbb{F}_q^m$:

$$\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$$

where $\forall \ell,\ 1 \leq \ell \leq m$, and $\boldsymbol{x} = (x_1, \ldots, x_n)$:

$$f_\ell(\boldsymbol{x}) = \sum_{1 \leq i_1 \leq \ldots \leq i_d \leq n} \gamma^\ell_{i_1 \ldots i_d} x_{i_1} x_{i_2} \ldots x_{i_d} + \sum_{1 \leq i_1 \leq \ldots \leq i_{d-1} \leq n} \gamma^\ell_{i_1 \ldots i_{d-1}} x_{i_1} x_{i_2} \ldots x_{i_{d-1}}$$

$$+ \ldots + \sum_{1 \leq i_1 \leq i_2 \leq n} \gamma^\ell_{i_1 i_2} x_{i_1} x_{i_2} + \sum_{1 \leq i_1 \leq n} \gamma^\ell_{i_1} x_{i_1}.$$

We omit the constant term since we are going to deal with a system of the form $\boldsymbol{F}(\boldsymbol{v}) = \boldsymbol{s}$.
We denote by $\mathcal{MP}ol(d, n, m, \mathbb{F}_q)$, the family of the functions $\boldsymbol{F}$ defined above. For example, for quadratic polynomials, the family is denoted by $\mathcal{MP}ol(2, n, m, \mathbb{F}_q)$. In [15], the authors used the notation $\mathcal{MQ}(n, m, \mathbb{F}_q)$.

Let the function $\boldsymbol{G}$ defined from $(\mathbb{F}_q^n)^d$ to $\mathbb{F}_q^m$ by:

$$\boldsymbol{G}(\boldsymbol{r_0}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}) = \sum_{i=1}^{d} (-1)^{d-i} \sum_{\substack{S \subset \{0, \ldots, d-1\} \\ |S| = i}} \boldsymbol{F}(\sum_{j \in S} \boldsymbol{r_j}). \qquad (*)$$

This polarization identity $(*)$ which gives the function $\boldsymbol{G}$ is the keystone of the generalization to any degree d.

For example, when $d = 3$ we have: $\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ where $\forall \ell,\ 1 \leq \ell \leq m$ and $\boldsymbol{x} = (x_1, \ldots, x_n)$

$$f_\ell(\boldsymbol{x}) = \sum_{1 \leq i \leq j \leq k \leq n} \gamma^\ell_{ijk} x_i x_j x_k + \sum_{1 \leq i \leq j \leq n} \gamma^\ell_{ij} x_i x_j + \sum_{1 \leq i \leq n} \gamma^\ell_i x_i.$$

And $\boldsymbol{G}$ is defined by:

$$\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{F}(\boldsymbol{x}+\boldsymbol{y}+\boldsymbol{z}) - \boldsymbol{F}(\boldsymbol{x}+\boldsymbol{y}) - \boldsymbol{F}(\boldsymbol{x}+\boldsymbol{z}) - \boldsymbol{F}(\boldsymbol{y}+\boldsymbol{z}) + \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{F}(\boldsymbol{y}) + \boldsymbol{F}(\boldsymbol{z}).$$

If: $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = (g_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}), g_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}), \ldots, g_m(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$
where $\forall \ell,\ 1 \leq \ell \leq m$, $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$

then:

$$g_\ell(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \sum_{1\le i\le j\le k\le n} \gamma^\ell_{ijk}(x_i y_j z_k + x_i y_k z_j + x_j y_i z_k + x_j y_k z_i + x_k y_i z_j + x_k y_j z_i).$$

It's easy to check that $g_\ell(\boldsymbol{x}+\boldsymbol{x'},\boldsymbol{y},\boldsymbol{z}) = g_\ell(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) + g_\ell(\boldsymbol{x'},\boldsymbol{y},\boldsymbol{z})$. We obtain that $\boldsymbol{G}$ is trilinear. For any d we can prove that $\boldsymbol{G}$ is d-linear (see also [20]).

An intractability assumption for a random instance of $\mathcal{MP}ol(d,n,m,\mathbb{F}_q)$ is defined as follows:

Definition 1. *For polynomially bounded functions $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$, it is said that $\mathcal{MP}ol(d,n,m,\mathbb{F}_q)$ is intractable if there is no polynomial-time algorithm that take $(\boldsymbol{F},\boldsymbol{v})$ generated via $R \in_R \mathcal{MP}ol(d,n,m,\mathbb{F}_q)$, $\boldsymbol{s} \in \mathbb{F}_q^n$, and $\boldsymbol{v} \longleftarrow \boldsymbol{F}(\boldsymbol{s})$ and finds a preimage $\boldsymbol{s'} \in \mathbb{F}_q^n$ such that $\boldsymbol{F}(\boldsymbol{s'}) = \boldsymbol{v}$ with non-negligible probability $\epsilon(\lambda)$.*

For $\boldsymbol{F}$, a multivariate function of degree d, we define a binary relation $R_{\boldsymbol{F}} = \{(\boldsymbol{v},\boldsymbol{x}) \in \mathbb{F}_q^m \times \mathbb{F}_q^n;\ \boldsymbol{v} = \boldsymbol{F}(\boldsymbol{x})\}$. Given an instance $\boldsymbol{F} \in \mathcal{MP}ol(d,n,m,\mathbb{F}_q)$ and a vector $\boldsymbol{v} \in \mathbb{F}_q^m$ the MPol(d) problem is finding $\boldsymbol{s} \in \mathbb{F}_q^n$ such that $\boldsymbol{F}(\boldsymbol{s}) = \boldsymbol{v}$, i.e. $\boldsymbol{s} \in R_{\boldsymbol{F}}(\boldsymbol{v})$.

4 $ZK(3)$ Schemes

Here we consider the MPol(3) problem which is denoted MC in [14].

4.1 Techniques for our constructions

Our constructions employ the cut-and-choose approach, where a prover first divides her secret into shares and then proves the correctness of some shares depending on the choice of a verifier without revealing the secret itself. The shared secrets are the following: $\boldsymbol{s} = \boldsymbol{r_0} + \boldsymbol{r_1} + \boldsymbol{r_2}$, $\boldsymbol{r_0} = \boldsymbol{t_0} + \boldsymbol{t_1}$, $\boldsymbol{F}(\boldsymbol{r_0}) = \boldsymbol{e_0} + \boldsymbol{e_1}$, $\boldsymbol{F}(\boldsymbol{r_0}+\boldsymbol{r_1}) = \boldsymbol{f_0} + \boldsymbol{f_1}$ and $\boldsymbol{F}(\boldsymbol{r_0}+\boldsymbol{r_2}) = \boldsymbol{h_0} + \boldsymbol{h_1}$.
Since $\boldsymbol{G}$ is trilinear we have: $\boldsymbol{G}(\boldsymbol{r_0},\boldsymbol{r_1},\boldsymbol{r_2}) = \boldsymbol{G}(\boldsymbol{t_0},\boldsymbol{r_1},\boldsymbol{r_2}) + \boldsymbol{G}(\boldsymbol{t_1},\boldsymbol{r_1},\boldsymbol{r_2})$. Moreover:

$$\begin{aligned}\boldsymbol{G}(\boldsymbol{r_0},\boldsymbol{r_1},\boldsymbol{r_2}) &= \boldsymbol{F}(\boldsymbol{s}) - \boldsymbol{F}(\boldsymbol{r_0}+\boldsymbol{r_1}) - \boldsymbol{F}(\boldsymbol{r_0}+\boldsymbol{r_2}) - \boldsymbol{F}(\boldsymbol{r_1}+\boldsymbol{r_2}) + \boldsymbol{F}(\boldsymbol{r_0}) + \boldsymbol{F}(\boldsymbol{r_1}) + \boldsymbol{F}(\boldsymbol{r_2}) \\ &= \boldsymbol{v} - \boldsymbol{f_0} - \boldsymbol{f_1} - \boldsymbol{h_0} - \boldsymbol{h_1} - \boldsymbol{F}(\boldsymbol{r_1}+\boldsymbol{r_2}) + \boldsymbol{e_0} + \boldsymbol{e_1} + \boldsymbol{F}(\boldsymbol{r_1}) + \boldsymbol{F}(\boldsymbol{r_2}).\end{aligned}$$

So the prover can express $\boldsymbol{v} = \boldsymbol{F}(\boldsymbol{s})$ in terms of $\boldsymbol{G}(\boldsymbol{t_0},\boldsymbol{r_1},\boldsymbol{r_2})$, $\boldsymbol{G}(\boldsymbol{t_1},\boldsymbol{r_1},\boldsymbol{r_2})$, $\boldsymbol{t_0}$, $\boldsymbol{t_1}$, $\boldsymbol{e_0}$, $\boldsymbol{e_1}$, $\boldsymbol{f_0}$, $\boldsymbol{f_1}$, $\boldsymbol{h_0}$, $\boldsymbol{h_1}$ and reveals it piece by piece so that $\boldsymbol{r_0}$, $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are never revealed together. We must care that if, for example, $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are revealed together, we should reveal no information on $\boldsymbol{F}(\boldsymbol{r_0}+\boldsymbol{r_1})$ nor on $\boldsymbol{F}(\boldsymbol{r_0}+\boldsymbol{r_2})$ since it could reveal something about $\boldsymbol{r_0}$.

The basic idea is that a prover proves that she has a tuple $(\boldsymbol{r_0}, \boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{t_0}, \boldsymbol{t_1}, \boldsymbol{e_0}, \boldsymbol{e_1}, \boldsymbol{f_0}, \boldsymbol{f_1}, \boldsymbol{h_0}, \boldsymbol{h_1})$ satisfying the five equalities:

$$\boldsymbol{t_0} = \boldsymbol{r_0} - \boldsymbol{t_1} \tag{1}$$
$$\boldsymbol{e_0} = \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{e_1} \tag{2}$$
$$\boldsymbol{f_0} = \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_1}) - \boldsymbol{f_1} \tag{3}$$
$$\boldsymbol{h_0} = F(\boldsymbol{r_0} + \boldsymbol{r_2}) - \boldsymbol{h_1} \tag{4}$$
$$\begin{aligned}&\boldsymbol{v} - \boldsymbol{G}(\boldsymbol{t_1}, \boldsymbol{r_1}, \boldsymbol{r_2}) + \boldsymbol{e_1} - \boldsymbol{f_1} - \boldsymbol{h_1} \\ &\quad -\boldsymbol{F}(\boldsymbol{r_1} + \boldsymbol{r_2}) + \boldsymbol{F}(\boldsymbol{r_1}) + \boldsymbol{F}(\boldsymbol{r_2}) = \boldsymbol{G}(\boldsymbol{t_0}, \boldsymbol{r_1}, \boldsymbol{r_2}) - \boldsymbol{e_0} + \boldsymbol{f_0} + \boldsymbol{h_0}\end{aligned} \tag{5}$$

4.2 3-pass scheme

For simplicity, the random string in Com is not written explicitly. If X is a set, $x \in_R X$ means that x is randomly chosen in X with the uniform distribution.

1. The Prover picks up $\boldsymbol{r_0}, \boldsymbol{r_1}, \boldsymbol{t_0} \in_R \mathbb{F}_q^n$ and $\boldsymbol{e_0}, \boldsymbol{f_0}, \boldsymbol{h_0} \in_R \mathbb{F}_q^m$. Then she computes

   $$\begin{aligned}&\boldsymbol{r_2} = \boldsymbol{s} - \boldsymbol{r_1} - \boldsymbol{r_0}, \quad \boldsymbol{t_1} = \boldsymbol{r_0} - \boldsymbol{t_0} \\ &\boldsymbol{e_1} = \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{e_0}, \quad \boldsymbol{f_1} = \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_1}) - \boldsymbol{f_0}, \quad \boldsymbol{h_1} = \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_2}) - \boldsymbol{h_0}\end{aligned}$$

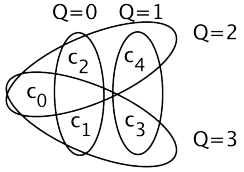
   The Prover sends to the Verifier

   $$\begin{aligned}&c_0 = Com(\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{G}(\boldsymbol{t_0}, \boldsymbol{r_1}, \boldsymbol{r_2}) - \boldsymbol{e_0} + \boldsymbol{f_0} + \boldsymbol{h_0}) \\ &c_1 = Com(\boldsymbol{r_1}, \boldsymbol{t_0}, \boldsymbol{e_0}, \boldsymbol{f_0}), \quad c_2 = Com(\boldsymbol{r_1}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{f_1}) \\ &c_3 = Com(\boldsymbol{r_2}, \boldsymbol{t_0}, \boldsymbol{e_0}, \boldsymbol{h_0}), \quad c_4 = Com(\boldsymbol{r_2}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{h_1})\end{aligned}$$

2. The verifier chooses a query $\mathcal{Q} \in_R \{0, 1, 2, 3\}$ and sends $\mathcal{Q}$ to the prover.
3. The figure 1 summarizes how the verifier deals with the commitments.
   (a) If $\mathcal{Q} = 0$ then the Prover sends $(\boldsymbol{r_0}, \boldsymbol{r_1}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{f_1})$. The Verifier checks if $c_1 = Com(\boldsymbol{r_1}, \boldsymbol{r_0} - \boldsymbol{t_1}, \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{e_1}, \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_1}) - \boldsymbol{f_1})$, $c_2 = Com(\boldsymbol{r_1}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{f_1})$. Here we use (1), (2) and (3).
   (b) If $\mathcal{Q} = 1$ then the Prover sends $(\boldsymbol{r_0}, \boldsymbol{r_2}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{h_1})$. The Verifier checks if $c_3 = Com(\boldsymbol{r_2}, \boldsymbol{r_0} - \boldsymbol{t_1}, \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{e_1}, \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_2}) - \boldsymbol{h_1})$, $c_4 = Com(\boldsymbol{r_2}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{h_1})$. Here we use (1), (2) and (4).
   (c) If $\mathcal{Q} = 2$ then the Prover sends $(\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{f_1}, \boldsymbol{h_1})$. The Verifier checks if $c_0 = Com(\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{v} - \boldsymbol{G}(\boldsymbol{t_1}, \boldsymbol{r_1}, \boldsymbol{r_2}) + \boldsymbol{e_1} - \boldsymbol{f_1} - \boldsymbol{h_1} - \boldsymbol{F}(\boldsymbol{r_1} + \boldsymbol{r_2}) + \boldsymbol{F}(\boldsymbol{r_1}) + \boldsymbol{F}(\boldsymbol{r_2}))$, $c_2 = Com(\boldsymbol{r_1}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{f_1})$, $c_4 = Com(\boldsymbol{r_2}, \boldsymbol{t_1}, \boldsymbol{e_1}, \boldsymbol{h_1})$. Here we use (5).
   (d) If $\mathcal{Q} = 3$ then the Prover sends $(\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{t_0}, \boldsymbol{e_0}, \boldsymbol{f_0}, \boldsymbol{h_0})$. The Verifier checks if $c_0 = Com(\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{G}(\boldsymbol{t_0}, \boldsymbol{r_1}, \boldsymbol{r_2}) - \boldsymbol{e_0} + \boldsymbol{f_0} + \boldsymbol{h_0})$, $c_1 = Com(\boldsymbol{r_1}, \boldsymbol{t_0}, \boldsymbol{e_0}, \boldsymbol{f_0})$, $c_3 = Com(\boldsymbol{r_2}, \boldsymbol{t_0}, \boldsymbol{e_0}, \boldsymbol{h_0})$.

The verifier outputs 1 if the she gets the correct value in the commitments, 0 otherwise.

Fig. 1. Check of the commitments



4.3 Properties of the 3-pass scheme

It is easy to see that the verifier always accepts an interaction with the honest prover. Thus the 3-pass scheme has perfect correctness.

Theorem 1 *The 3-pass protocol is statistically zero knowledge when the commitment scheme Com is statistically hiding.*

Proof. We construct a black-box simulator $\mathcal{S}$ which have oracle access to a cheating verifier $\mathcal{CV}$ takes $\boldsymbol{F}$ and $\boldsymbol{v}$, and outputs a simulated transcripts with probability 3/4 as follows. The simulator randomly chooses a value $\mathcal{Q}^* \in_R \{0,1,2,3\}$ and vectors $\boldsymbol{s'}, \boldsymbol{r'_0}, \boldsymbol{r'_1}, \boldsymbol{t'_0} \in_R \mathbb{F}_q^n$ and $\boldsymbol{e'_0}, \boldsymbol{f'_0}, \boldsymbol{h'_0} \in_R \mathbb{F}_q^m$, where $\mathcal{Q}^*$ is a prediction what value the cheating verifier $\mathcal{CV}$ will not choose. Then it computes

$$\boldsymbol{r'_2} = \boldsymbol{s'} - (\boldsymbol{r'_0} + \boldsymbol{r'_1}),\ \boldsymbol{t'_1} \leftarrow \boldsymbol{r'_0} - \boldsymbol{t'_0},\ \boldsymbol{e'_1} \leftarrow \boldsymbol{F}(\boldsymbol{r'_0}) - \boldsymbol{e'_0}.$$

Moreover it sets in order to simulate the transcripts:

1. If $\mathcal{Q}^* = 0$, $\boldsymbol{f'_1} = \boldsymbol{v} - \boldsymbol{F}(\boldsymbol{s'}) + \boldsymbol{F}(\boldsymbol{r'_0} + \boldsymbol{r'_1}) - \boldsymbol{f'_0}$, else $\boldsymbol{f'_1} = \boldsymbol{F}(\boldsymbol{r'_0} + \boldsymbol{r'_1}) - \boldsymbol{f'_0}$.
2. If $\mathcal{Q}^* = 1$, $\boldsymbol{h'_1} = \boldsymbol{v} - \boldsymbol{F}(\boldsymbol{s'}) + \boldsymbol{F}(\boldsymbol{r'_0} + \boldsymbol{r'_2}) - \boldsymbol{f'_0}$, else $\boldsymbol{h'_1} = \boldsymbol{F}(\boldsymbol{r'_0} + \boldsymbol{r'_2}) - \boldsymbol{h'_0}$.
3. If $\mathcal{Q}^* = 3$, $c'_0 = Com(\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{v} - \boldsymbol{G}(\boldsymbol{t'_1}, \boldsymbol{r'_1}, \boldsymbol{r'_2}) - \boldsymbol{f'_1} - \boldsymbol{h'_1} + \boldsymbol{e'_1} - \boldsymbol{F}(\boldsymbol{r'_1} + \boldsymbol{r'_2}) + \boldsymbol{F}(\boldsymbol{r'_1}) + \boldsymbol{F}(\boldsymbol{r'_2}))$, else $c'_0 = Com(\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{G}(\boldsymbol{t'_0}, \boldsymbol{r'_1}, \boldsymbol{r'_2}) - \boldsymbol{f'_0} - \boldsymbol{h_0} + \boldsymbol{e'_0})$

It also computes:

$$c'_1 = Com(\boldsymbol{r'_1}, \boldsymbol{t'_0}, \boldsymbol{e'_0}, \boldsymbol{f'_0}),\ \ c'_2 = Com(\boldsymbol{r'_1}, \boldsymbol{t'_1}, \boldsymbol{e'_1}, \boldsymbol{f'_1})$$
$$c'_3 = Com(\boldsymbol{r'_2}, \boldsymbol{t'_0}, \boldsymbol{e'_0}, \boldsymbol{h'_0}),\ \ c'_4 = Com(\boldsymbol{r'_2}, \boldsymbol{t'_1}, \boldsymbol{e'_1}, \boldsymbol{h'_1})$$

and sends $(c'_0, c'_1, c'_2, c'_3, c'_4)$ to $\mathcal{CV}$.
Receiving a query $\mathcal{Q}$ from $\mathcal{CV}$ the simulator outputs $\perp$ if $\mathcal{Q} = \mathcal{Q}^*$ and stops. If $\mathcal{S}$ does not output $\perp$, it produces a transcript as follows:

- If $\mathcal{Q} = 0$, it outputs $((c'_0, c'_1, c'_2, c'_3, c'_4), 0, (\boldsymbol{r'_0}, \boldsymbol{r'_1}, \boldsymbol{t'_1}, \boldsymbol{e'_1}, \boldsymbol{f'_1}))$
- If $\mathcal{Q} = 1$, it outputs $((c'_0, c'_1, c'_2, c'_3, c'_4), 1, (\boldsymbol{r'_0}, \boldsymbol{r'_2}, \boldsymbol{t'_1}, \boldsymbol{e'_1}, \boldsymbol{h'_1}))$
- If $\mathcal{Q} = 2$, it outputs $((c'_0, c'_1, c'_2, c'_3, c'_4), 2, (\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{t'_1}, \boldsymbol{e'_1}, \boldsymbol{f'_1}, \boldsymbol{h'_1}))$
- If $\mathcal{Q} = 3$, it outputs $((c'_0, c'_1, c'_2, c'_3, c'_4), 3, (\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{t'_0}, \boldsymbol{e'_0}, \boldsymbol{f'_0}, \boldsymbol{h'_0}))$

We can check that if $\mathcal{S}$ does not output $\perp$, the transcript is accepted. For example, we consider the case where $\mathcal{Q}^* = 0$ and $\mathcal{Q} = 2$. The output is $((c'_0, c'_1, c'_2, c'_3, c'_4), 2, (\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{t'_1}, \boldsymbol{e'_1}, \boldsymbol{f'_1}, \boldsymbol{h'_1}))$. Thus, we have the right values for c'_2 and c'_4. Now,

c'_0 is computed as follows: $c'_0 = Com(\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{v} - \boldsymbol{G}(\boldsymbol{t'_1}, \boldsymbol{r'_1}, \boldsymbol{r'_2}) + \boldsymbol{e'_1} - \boldsymbol{f'_1} - \boldsymbol{h'_1} - \boldsymbol{F}(\boldsymbol{r'_1} + \boldsymbol{r'_2}) + \boldsymbol{F}(\boldsymbol{r'_1}) + \boldsymbol{F}(\boldsymbol{r'_2}))$. Here $\boldsymbol{f'_1} = \boldsymbol{v} - \boldsymbol{F}(\boldsymbol{s'}) + \boldsymbol{F}(\boldsymbol{r'_0} + \boldsymbol{r'_1}) - \boldsymbol{f'_0}$. Thus we obtain $c'_0 = Com(\boldsymbol{r'_1}, \boldsymbol{r'_2}, \boldsymbol{G}(\boldsymbol{t'_0}, \boldsymbol{r'_1}, \boldsymbol{r'_2}) - \boldsymbol{f'_0} - \boldsymbol{h_0} + \boldsymbol{e'_0})$ and the transcript is accepted. The other cases are checked similarly.

The distribution of the output $\mathcal{S}$ is statistically close to the distribution of a real transcript since the commitment is statistically hiding and the values are randomly chosen in $\mathbb{F}_q^n$ or in $\mathbb{F}_q^m$. The details can be found in the extended version of the paper. □

Theorem 2 *The 3-pass protocol is proof of zero knowledge with zero knowledge error* $3/4$ *when the commitment scheme Com is computationally binding.*

Proof. Suppose that there exists a false prover C that can answer all the questions. Then either C will compute a collision for Com or will extract a solution for $(\boldsymbol{F}, \boldsymbol{v})$. Let $((c_0, c_1, c_2, c_3, c_4), \mathcal{Q}_0, Rsp_0)$, $((c_0, c_1, c_2, c_3, c_4), \mathcal{Q}_1, Rsp_1)$, $((c_0, c_1, c_2, c_3, c_4), \mathcal{Q}_2, Rsp_2)$, $((c_0, c_1, c_2, c_3, c_4), \mathcal{Q}_3, Rsp_3)$, be four transcripts such that $\mathcal{Q}_i = i$ and all the responses are accepted. Consider the situation where the responses are parsed as $Rsp_0 = (\boldsymbol{r_0}^{(0)}, \boldsymbol{r_1}^{(0)}, \boldsymbol{t_1}^{(0)}, \boldsymbol{e_1}^{(0)}, \boldsymbol{f_1}^{(0)})$, $Rsp_1 = (\boldsymbol{r_0}^{(1)}, \boldsymbol{r_2}^{(1)}, \boldsymbol{t_1}^{(1)}, \boldsymbol{e_1}^{(1)}, \boldsymbol{h_1}^{(1)})$, $Rsp_2 = (\boldsymbol{r_1}^{(2)}, \boldsymbol{r_2}^{(2)}, \boldsymbol{t_1}^{(2)}, \boldsymbol{e_1}^{(2)}, \boldsymbol{f_1}^{(2)}, \boldsymbol{h_1}^{(2)})$, $Rsp_3 = (\boldsymbol{r_1}^{(3)}, \boldsymbol{r_2}^{(3)}, \boldsymbol{t_0}, \boldsymbol{e_0}, \boldsymbol{f_0}, \boldsymbol{h_0})$. We obtain:

$$\begin{aligned} c_0 &= Com(\ \boldsymbol{r_1}^{(2)}, \boldsymbol{r_2}^{(2)}, \boldsymbol{v} - \boldsymbol{G}(\boldsymbol{t_1}^{(2)}, \boldsymbol{r_1}^{(2)}, \boldsymbol{r_2}^{(2)}) - \boldsymbol{f_1}^{(2)} - \boldsymbol{h_1}^{(2)} + \boldsymbol{e_1}^{(2)} \\ &\qquad\qquad -\boldsymbol{F}(\boldsymbol{r_1}^{(2)} + \boldsymbol{r_2}^{(0)}) + \boldsymbol{F}(\boldsymbol{r_1}^{(2)}) + \boldsymbol{F}(\boldsymbol{r_2}^{(2)}) \quad) \\ &= Com(\ \boldsymbol{r_1}^{(3)}, \boldsymbol{r_2}^{(3)}, \boldsymbol{G}(\boldsymbol{t_0}, \boldsymbol{r_1}^{(3)}, \boldsymbol{r_2}^{(3)}) + \boldsymbol{f_0} + \boldsymbol{h_0} - \boldsymbol{e_0} \quad) \qquad (a) \end{aligned}$$

$$\begin{aligned} c_1 &= Com(\ \boldsymbol{r_1}^{(0)}, \boldsymbol{r_0}^{(0)} - \boldsymbol{t_1}^{(0)}, \boldsymbol{F}(\boldsymbol{r_0}^{(0)}) - \boldsymbol{e_1}^{(0)}, \boldsymbol{F}(\boldsymbol{r_0}^{(0)} + \boldsymbol{r_1}^{(0)}) - \boldsymbol{f_1}^{(0)}\ ) \\ &= Com(\ \boldsymbol{r_1}^{(3)}, \boldsymbol{t_0}, \quad \boldsymbol{e_0}, \quad \boldsymbol{f_0} \quad) \qquad (b) \end{aligned}$$

$$c_2 = Com(\boldsymbol{r_1}^{(0)}, \boldsymbol{t_1}^{(0)}, \boldsymbol{e_1}^{(0)}, \boldsymbol{f_1}^{(0)}) = Com(\boldsymbol{r_1}^{(2)}, \boldsymbol{t_1}^{(2)}, \boldsymbol{e_1}^{(2)}, \boldsymbol{f_1}^{(2)}) \qquad (c)$$

$$\begin{aligned} c_3 &= Com(\ \boldsymbol{r_2}^{(1)}, \boldsymbol{r_0}^{(1)} - \boldsymbol{t_1}^{(1)}, \boldsymbol{F}(\boldsymbol{r_0}^{(1)}) - \boldsymbol{e_1}^{(1)}, \boldsymbol{F}(\boldsymbol{r_0}^{(1)} + \boldsymbol{r_2}^{(1)}) - \boldsymbol{h_1}^{(1)}\ ) \\ &= Com(\ \boldsymbol{r_2}^{(3)}, \boldsymbol{t_0}, \quad \boldsymbol{e_0}, \quad \boldsymbol{h_0} \quad) \qquad (d) \end{aligned}$$

$$c_4 = Com(\boldsymbol{r_2}^{(1)}, \boldsymbol{t_1}^{(1)}, \boldsymbol{e_1}^{(1)}, \boldsymbol{h_1}^{(1)}) = Com(\boldsymbol{r_2}^{(2)}, \boldsymbol{t_1}^{(2)}, \boldsymbol{e_1}^{(2)}, \boldsymbol{h_1}^{(2)}) \qquad (e)$$

If two tuples of the arguments of Com are are distinct on either of the above equations, then we have a collision for Com. Otherwise, these equalities give:

$$\boldsymbol{h_1}^{(1)} \overset{(e)}{=} \boldsymbol{h_1}^{(2)}, \boldsymbol{r_1}^{(0)} \overset{(b)}{=} \boldsymbol{r_1}^{(3)} \overset{(a)}{=} \boldsymbol{r_1}^{(2)}, \boldsymbol{t_1}^{(0)} \overset{(c)}{=} \boldsymbol{t_1}^{(1)} \overset{(e)}{=} \boldsymbol{t_1}^{(2)}, \boldsymbol{r_0}^{(0)} \overset{(c,d)}{=} \boldsymbol{r_0}^{(1)}$$

$$\text{and} \quad \boldsymbol{r_2}^{(2)} \overset{(a)}{=} \boldsymbol{r_2}^{(3)} \overset{(d)}{=} \boldsymbol{r_2}^{(1)}, \boldsymbol{e_1}^{(2)} \overset{(c)}{=} \boldsymbol{e_1}^{(0)} \overset{(b,d)}{=} \boldsymbol{e_1}^{(1)}, \boldsymbol{f_1}^{(0)} \overset{(c)}{=} \boldsymbol{f_1}^{(2)}.$$

So, all upper scripts are useless and from (a) we have:

$$\boldsymbol{v} = \boldsymbol{G}(\boldsymbol{t_1} + \boldsymbol{t_0}, \boldsymbol{r_1}, \boldsymbol{r_2}) + \boldsymbol{f_1} + \boldsymbol{h_1} - \boldsymbol{e_1} + \boldsymbol{F}(\boldsymbol{r_1} + \boldsymbol{r_2}) - \boldsymbol{F}(\boldsymbol{r_1}) - \boldsymbol{F}(\boldsymbol{r_2}) + \boldsymbol{f_0} + \boldsymbol{h_0} - \boldsymbol{e_0}.$$

Then, from (b) and (d) we have $\boldsymbol{r_0} = \boldsymbol{t_0} + \boldsymbol{t_1}$, $\boldsymbol{F}(\boldsymbol{r_0}) = \boldsymbol{e_0} + \boldsymbol{e_1}$, $\boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_1}) = \boldsymbol{f_0} + \boldsymbol{f_1}$ and $\boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_2}) = \boldsymbol{h_0} + \boldsymbol{h_1}$, so if we replace these values in the previous equality, we obtain: $\boldsymbol{v} = \boldsymbol{G}(\boldsymbol{r_0}, \boldsymbol{r_1}, \boldsymbol{r_2}) + \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_1}) + \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_2}) + \boldsymbol{F}(\boldsymbol{r_1} + \boldsymbol{r_2}) - \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{F}(\boldsymbol{r_1}) - \boldsymbol{F}(\boldsymbol{r_2}) = \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_1} + \boldsymbol{r_2})$. This means that a solution $\boldsymbol{r_0} + \boldsymbol{r_1} + \boldsymbol{r_2}$ for $\boldsymbol{v}$ is extracted. □

4.4 Computations in the 3-pass scheme

We give the maximum number of computations that have to be done either by the prover or by the receiver in the case of $\mathbb{F}_2$. We must calculate the number of computations for F and for G. Moreover we see that F is computed at most 3 times and G is computed just one time. We only count multiplications. In $\mathbb{F}_2$, we have: $x_i^3 = x_i^2 = x_i$. We can write:

$$f_\ell(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \big[\gamma_i^\ell + \gamma_{ii}^\ell + \gamma_{iii}^\ell + \sum_{j=i+1}^{n} x_j \big(\gamma_{ij}^\ell + \gamma_{ijj}^\ell + \sum_{k=j+1}^{n} \gamma_{ijk}^\ell x_k\big)\big]$$

Let M denotes the number of multiplications needed to compute F. Using the above expression for $\boldsymbol{F}$, we obtain $M \simeq \frac{n^3}{6} m$. We want to set the parameters in order to have 80-bit security. Since even for a quadratic system with 84 variables and 80 equations over $\mathbb{F}_2$ satisfies 80-bits security [15], we will use $\mathcal{MP}ol(3, 84, 80, \mathbb{F}_2)$. In the following table, we give the characteristics of the scheme $ZK(3)$ and the values that we obtain when we choose $n = 84$, $m = 80$, in order to have 80-bit security (cf. [2]). Moreover, if we want an impersonation probability less than 2^{-30}, we need to perform at least 73 rounds. R stands for the number of rounds and C for the maximum number of computations that have to be done either by the prover or by the receiver. Moreover, we will use a standard trick for saving communication data size. We employ a collision resistant hash function H. Let $c_\alpha = H(c_1, c_2)$, $c_\beta = H(c_3, c_4)$ and $c = H(c_0, c_\alpha, c_\beta)$. In the first pass, the sender send the value c, instead of five commitments $(c_0, c_1, c_2, c_3, c_4)$. In the third pass, she will proceed as follows:

- If $\mathcal{Q} = 0$, the pair (c_0, c_β) is appended to the prover's responses.
- If $\mathcal{Q} = 1$, the pair (c_0, c_α) is appended to the prover's responses.
- If $\mathcal{Q} = 2$, the pair (c_1, c_3) is appended to the prover's responses.
- If $\mathcal{Q} = 3$, the pair (c_2, c_4) is appended to the prover's responses.

Finally she checks if $c = H(c_0, c_\alpha, c_\beta)$. She can obtain the needed values from the prover's responses in each case. As a result, the number of hash values sent is reduced from 5 to 3. This modified version of the 3-pass protocol has the same properties. Table 1 summarizes the features of $ZK(3)$.

Remarks.

We may need less computations. It depends on the number of non zero coefficients. This is the case for Brent equations as explain in Section 8.

It is also possible to design a 5-pass scheme. This in done in the extended version of this paper.

Table 1. $ZK(3)$ Scheme

	Formulas	Parameters for 2^{80} security
Public key (Number of bits)	m	80
Secret key (Number of bits)	n	84
M	$\frac{n^3}{6} \times m$	7902720
Communication (bit)	$(3 \times 160 + 2 + 3n + 3m) \times R$	71102
Number of multiplications	C=9MR	2^{33}

5 $ZK(d)$ Scheme for any d

5.1 The $ZK(d)$ Scheme

We will design a 3-pass scheme.
We consider the following function of degree d from $\mathbb{F}_q^n$ to $\mathbb{F}_q^m$:

$$\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$$

where $\forall \ell,\ 1 \leq \ell \leq m$,

$$f_\ell(\boldsymbol{x}) = \sum_{1 \leq i_i \leq \ldots \leq i_d \leq n} \gamma^\ell_{i_1 \ldots i_d} x_{i_1} x_{i_2} \ldots x_{i_d} +$$

$$\sum_{1 \leq i_i \leq \ldots \leq i_{d-1} \leq n} \gamma^\ell_{i_1 \ldots i_{d-1}} x_{i_1} x_{i_2} \ldots x_{i_{d-1}} + \ldots + \sum_{1 \leq i_i \leq i_2 \leq n} \gamma^\ell_{i_1 i_2} x_{i_1} x_{i_2} + \sum_{1 \leq i_i \leq n} \gamma^\ell_{i_1} x_{i_1}$$

and $\boldsymbol{x} = (x_1, \ldots, x_n)$. We omit the constant term. Let

$$\boldsymbol{G}(\boldsymbol{r_0}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}) = \sum_{i=1}^{d} (-1)^{d-i} \sum_{\substack{S \subset \{0, \ldots, d-1\} \\ |S| = i}} \boldsymbol{F}(\sum_{j \in S} \boldsymbol{r_j}).$$

The public key is $(\boldsymbol{F}, \boldsymbol{v})$. The secret is $\boldsymbol{s}$ such that $\boldsymbol{F}(\boldsymbol{s}) = \boldsymbol{v}$.

1. The Prover picks up at random $\boldsymbol{r_0}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-2}}, \boldsymbol{t_0} \in_R \mathbb{F}_q^n$, $\boldsymbol{f_0} \in_R \mathbb{F}_q^m$, and $\forall p,\ 1 \leq p \leq d-2,\ \forall i_1, \ldots i_p,\ 1 \leq i_1 < i_2 < \ldots i_p \leq d-1,\ \boldsymbol{f_0}^{i_1 \ldots i_p} \in_R \mathbb{F}_q^m$. Then she computes

$$\begin{aligned} \boldsymbol{r_{d-1}} &= \boldsymbol{s} - \textstyle\sum_{i=1}^{d-2} \boldsymbol{r_i} \\ \boldsymbol{t_1} &= \boldsymbol{r_0} - \boldsymbol{t_0} \\ \boldsymbol{f_1} &= \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{f_0} \end{aligned}$$

and

$$\forall p,\ 1 \leq p \leq d-2,\ \forall i_1, \ldots i_p,\ 1 \leq i_1 < i_2 < \ldots i_p \leq d-1$$

$$\boldsymbol{f_1}^{i_1 \ldots i_p} = \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_{i_1}} + \ldots + \boldsymbol{r_{i_p}}) - \boldsymbol{f_0}^{i_1 \ldots i_p}.$$

Then the Prover sends to the Verifier

$$c_0 \leftarrow Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{G}(\boldsymbol{t_0}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}) + \sum_{p=1}^{d-2} (-1)^{d-p} \sum_{1 \le i_1 < \ldots < i_p \le d-1} \boldsymbol{f_0}^{i_1 \ldots i_p} + (-1)^d \boldsymbol{f_0} \Big)$$

and she sends also $\forall i,\ 1 \le i \le d-1$:

$$c_{2i-1} \leftarrow Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_0}, \boldsymbol{f_0}, \forall p,\ 1 \le p \le d-2,\ \forall i_1, \ldots i_p,\ 1 \le i_1 < i_2 < \ldots i_p \le d-1, \text{ such that } \forall j,\ i_j \ne i,\ \boldsymbol{f_0}^{i_1 \ldots i_p} \Big)$$

$$c_{2i} \leftarrow Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_1}, \boldsymbol{f_1}, \forall p,\ 1 \le p \le d-2,\ \forall i_1, \ldots i_p,\ 1 \le i_1 < i_2 < \ldots i_p \le d-1, \text{ such that } \forall j\ i_j \ne i,\ \boldsymbol{f_1}^{i_1 \ldots i_p} \Big).$$

The verifier chooses a query $\mathcal{Q} \in_R \{0, 1, \ldots, d\}$ and sends $\mathcal{Q}$ to the prover.

2. (a) If $\mathcal{Q} = 0$, then the Prover sends $(\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_0}, \boldsymbol{f_0}, \forall p,\ 1 \le p \le d-2,\ \forall i_1, \ldots i_p,\ 1 \le i_1 < i_2 < \ldots i_p \le d-1,\ \boldsymbol{f_0}^{i_1 \ldots i_p})$.
The Verifier checks if

$$c_0 = Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{G}(\boldsymbol{t_0}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}) + \sum_{p=1}^{d-2} (-1)^{d-p} \sum_{1 \le i_1 < \ldots < i_p \le d-1} \boldsymbol{f_0}^{i_1 \ldots i_p} + (-1)^d \boldsymbol{f_0} \Big)$$

and $\forall i,\ 1 \le i \le d-1$,

$$c_{2i-1} = Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_0}, \boldsymbol{f_0}, \forall p,\ 1 \le p \le d-2,\ \forall i_1, \ldots i_p,\ 1 \le i_1 < i_2 < \ldots i_p \le d-1, \text{ such that } \forall j,\ i_j \ne i,\ \boldsymbol{f_0}^{i_1 \ldots i_p} \Big).$$

(b) If $\mathcal{Q} = d$, then the Prover sends $(\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_1}, \boldsymbol{f_1}, \forall p,\ 1 \le p \le d-2,\ \forall i_1, \ldots i_p,\ 1 \le i_1 < i_2 < \ldots i_p \le d-1,\ \boldsymbol{f_1}^{i_1 \ldots i_p})$.
$\forall i,\ 1 \le i \le d-1$,
The Verifier checks if

$$c_0 = Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{v} - \boldsymbol{G}(\boldsymbol{t_1}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{d-1}}) - \sum_{p=1}^{d-2} (-1)^{d-p} \sum_{1 \le i_1 < \ldots < i_p \le d-1} \boldsymbol{f_1}^{i_1 \ldots i_p} - (-1)^d \boldsymbol{f_1} + \sum_{i=1}^{d} (-1)^{d-i} \sum_{\substack{S \subset \{1, \ldots, d-1\} \\ |S| = i}} \boldsymbol{F}(\sum_{j \in S} \boldsymbol{r_j}) \Big)$$

and $\forall i,\ 1 \leq i \leq d-1$,

$$c_{2i} = Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_1}, \boldsymbol{f_1}, \\ \forall p,\ 1 \leq p \leq d-2,\ \forall i_1, \ldots i_p,\ 1 \leq i_1 < i_2 < \ldots i_p \leq d-1, \\ \text{such that } \forall j,\ i_j \neq i,\ \boldsymbol{f_1}^{i_1 \ldots i_p} \Big).$$

(c) if $\mathcal{Q} = i$, then the prover sends $(\boldsymbol{r_0}, \boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_1}, \boldsymbol{f_1}$, $\forall p,\ 1 \leq p \leq d-2,\ \forall i_1, \ldots i_p,\ 1 \leq i_1 < i_2 < \ldots i_p \leq d-1$, such that $\forall j,\ i_j \neq i,\ \boldsymbol{f_1}^{i_1 \ldots i_p}$)
The Verifier checks if

$$c_{2i-1} = Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{r_0} - \boldsymbol{t_1}, \boldsymbol{F}(\boldsymbol{r_0}) - \boldsymbol{f_1}, \\ \forall p,\ 1 \leq p \leq d-2,\ \forall i_1, \ldots i_p,\ 1 \leq i_1 < i_2 < \ldots i_p \leq d-1, \\ \text{such that } \forall j,\ i_j \neq i,\ \boldsymbol{F}(\boldsymbol{r_0} + \boldsymbol{r_{i_1}} + \ldots + \boldsymbol{r_{i_p}}) - \boldsymbol{f_1}^{i_1 \ldots i_p} \Big)$$

and

$$c_{2i} = Com\Big(\boldsymbol{r_1}, \ldots, \boldsymbol{r_{i-1}}, \boldsymbol{r_{i+1}}, \ldots, \boldsymbol{r_{d-1}}, \boldsymbol{t_1}, \boldsymbol{f_1}, \\ \forall p,\ 1 \leq p \leq d-2,\ \forall i_1, \ldots i_p,\ 1 \leq i_1 < i_2 < \ldots i_p \leq d-1, \\ \text{such that } \forall j,\ i_j \neq i,\ \boldsymbol{f_1}^{i_1 \ldots i_p} \Big).$$

We now give the properties of the $ZK(d)$ Scheme. The proof are detailed in the extended version of this paper.

Theorem 3 *The 3-pass protocol is statistically zero knowledge when the commitment scheme Com is statistically hiding.*

Theorem 4 *The 3-pass protocol is proof of zero knowledge with knowledge error* $\frac{d}{d+1}$ *when the commitment scheme Com is computationally binding.*

We provide the features of this scheme in table 2. The trick used to compute the number of communication bits for $ZK(3)$ can be generalized to $ZK(d)$.

Remark. As for the case $d = 3$, it is possible to design a 5-pass scheme.

6 The $\tilde{ZK}(d)$ Scheme.

In this section, we propose another scheme inspired from [15] and also mentioned in [14]. The idea is to transform the system of degree d into a quadratic one and the to use the scheme given in [15]. As we will see, the number of computations is smaller, but the number of communication bits is more important.
First, we investigate the transformation of a system with cubic equations to a system with quadratic equations. We will introduce new variables. Once we have obtained a system of equations with quadratic polynomials, we can apply the

Table 2. $ZK(d)$ scheme

	$ZK(d)$ scheme
Public key (Number of bits)	m
Secret key (Number of bits)	n
M	$\frac{n^d}{d!} \times m$
Rounds	$R = \lceil \frac{30 \ln(2)}{\ln(1+1/d)} \rceil$
Number of Communication (bits)	$(160d + \lceil \frac{\ln d}{2} \rceil + 2^{d-1} - 1) \cdot R$
Multiplications	$C = 9(2^{d-1} - 1 + d!)MR$

identification scheme of [15]. We will calculate the number of multiplications in this case. In our system, we have $\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ where $\forall \ell,\ 1 \leq \ell \leq m$,

$$f_\ell(\boldsymbol{x}) = \sum_{1 \leq i \leq j \leq k \leq n} \gamma_{ijk}^\ell x_i x_j x_k + \sum_{1 \leq i \leq j \leq n} \gamma_{ij}^\ell x_i x_j + \sum_{1 \leq i \leq n} \gamma_i^\ell x_i$$

and $\boldsymbol{x} = (x_1, \ldots, x_n)$. We introduce the new variables $\forall i, j,\ 1 \leq i \leq j \leq n,\ X_{ij} = x_i x_j$. The number of new variables is $\frac{n(n-1)}{2}$, if $q = 2$, and $\frac{n(n+1)}{2}$, if $q \neq 2$. In our new system, we have

$$\tilde{F} = (\tilde{f}_1, \ldots, \tilde{f}_m, (\tilde{f}_{ij})_{1 \leq i \leq j \leq n})$$

where for $\tilde{\boldsymbol{x}} = (x_1, \ldots, x_n, (X_{ij})_{1 \leq i \leq j \leq n})$ and $1 \leq \ell \leq m$,

$$f_\ell(\tilde{\boldsymbol{x}}) = \sum_{1 \leq i \leq j \leq k \leq n} \gamma_{ijk}^\ell X_{ij} x_k + \sum_{1 \leq i \leq j \leq n} \gamma_{ij}^\ell X_{ij} + \sum_{1 \leq i \leq n} \gamma_i^\ell x_i$$

and for $1 \leq i \leq j \leq n$, $f_{ij}(\tilde{x}) = X_{ij} - x_i x_j$. Here the number of variables is $\tilde{n} \simeq n + \frac{n^2}{2}$ and the number of equations is $\tilde{m} = m + \frac{n^2}{2}$. As before, M denotes the number of multiplications needed to compute F. $\tilde{M}$ denotes the number of multiplications for the computation of $\tilde{F}$. We choose $q = 2$. Then $\tilde{M} = M + \frac{n(n-1)}{2}$. Thus $\tilde{M} \simeq M \simeq \frac{n^3}{6} m$. $\tilde{C}$ stands for the maximum number of computations that have to be done either by the prover or by the receiver. If $\tilde{R}$ denotes the number of rounds performed in order to have an impersonation probability less than 2^{-30}, then $\tilde{R} = 52$ (cf. [15]).

More generally, for $d \geq 4$, we introduce new variables and new equations in order to get a system of quadratic equations. Again there are two functions F and $\tilde{F}$. If M (resp. $\tilde{M}$) denotes the number of computations for F (resp. $\tilde{F}$), then $M \simeq n^d/d!$ and $\tilde{M} \simeq \frac{n^d}{d!} + n^{\lceil \frac{d}{2} \rceil} / \lceil \frac{d}{2} \rceil! \simeq M$. For $ZK(d)$, there are n variables and m equations. For $\tilde{ZK}(d)$, there are $\tilde{n} \simeq n + n^{\lceil \frac{d}{2} \rceil} / \lceil \frac{d}{2} \rceil!$ variables

and $\tilde{m} \simeq m + n^{\lceil \frac{d}{2} \rceil}/\lceil \frac{d}{2} \rceil!$ equations. The following table gives the characteristics of the $\tilde{ZK}(d)$ scheme and the values we get when $n = 84$ and $m = 80$.

Table 3. $\tilde{ZK}(d)$ scheme

	$\tilde{ZK}(3)$	$\tilde{ZK}(d)$	$\tilde{ZK}(3)$
	Formulas		2^{-80} security
Public key (bit)	$\tilde{m}$	$\simeq m + n^{\lceil \frac{d}{2} \rceil}/\lceil \frac{d}{2} \rceil!$	3483
Secret key (bit)	n		84
M	$\frac{n^3}{6} \times m$	$\frac{n^d}{d!} \times m$	7902720
Rounds	$\tilde{R} = 52$		
Communication (bits)	$2 \times 160 + 2 + 2\tilde{n} + \tilde{m}$ $\times \tilde{R}$	$322 + 2n + m + 3n^{\lceil \frac{d}{2} \rceil}/\lceil \frac{d}{2} \rceil!$ $\times \tilde{R}$	560508
Multiplications	$\tilde{C} = 3\tilde{M}\tilde{R}$		2^{31}

7 Relations between the number of computations and the number of coefficients

In the previous computations, we considered the MPol(d) problem since we supposed that we have the maximum number of coefficients. Then we obtained that in both cases, $M \simeq \tilde{M} \simeq \frac{n^d}{d!}$. Then the total number of multiplications is a function of M or $\tilde{M}$ and the number of rounds. For sparse systems, M or $\tilde{M}$ will be smaller but we will still have the same relations between C, M and R (and similarly $\tilde{C}$, $\tilde{M}$ and $\tilde{R}$). Here again, we can see that there are more variables and more communications bits in the $\tilde{ZK}(d)$ schemes and more computations in the $ZK(d)$ schemes. This is related to the "MPol" problem.

More precisely with the $ZK(d)$ scheme, we have:

$$f_\ell(x_1, x_2, \ldots, x_n) = \sum_{(i_1,\ldots,i_d)\in S_d^\ell} \gamma^\ell_{i_1\ldots i_d} x_{i_1} x_{i_2} \ldots x_{i_d} + \sum_{i_i,\ldots,i_{d-1}\in S_{d-1}^\ell} \gamma^\ell_{i_1\ldots i_{d-1}} x_{i_1} x_{i_2} \ldots x_{i_{d-1}}$$

$$+ \ldots + \sum_{i_i,i_2\in S_2^\ell} \gamma^\ell_{i_1 i_2} x_{i_1} x_{i_2} + \sum_{i_i\in S_2^\ell} \gamma^\ell_{i_1} x_{i_1}$$

where each S_u^ℓ is a subset of $\{1, 2, \ldots, n\}^u$.
The number of multiplications for f_ℓ is given by

$$d|S_d^\ell| + (d-1)|S_{d-1}^\ell| + (d-2)|S_{d-2}^\ell| + \ldots + 2|S_2^\ell| + |S_1^\ell|$$

and for F the number M of multiplications is

$$M = \sum_{\ell=1}^{m} \Big[d|S_d^\ell| + (d-1)|S_{d-1}^\ell| + (d-2)|S_{d-2}^\ell| + \ldots + 2|S_2^\ell| + |S_1^\ell| \Big].$$

Moreover, F is computed at most $2^{d-1}-1$ times during the process. For g_l we have $\big(d!(d-1)+1\big)|S_d^\ell|$ multiplications and for G, $\sum_{\ell=1}^{m}\big(d!(d-1)+1\big)|S_d^\ell|$ multiplications and G is computed one time. Finally, for one round, the number of multiplications is given by

$$\Big(\sum_{\ell=1}^{m}\Big[\Big(d(2^{d-1}-1)+d!(d-1)+1\Big)|S_d^\ell|+(d-1)|S_{d-1}^\ell|+(d-2)|S_{d-2}^\ell|+\ldots+$$

$$2|S_2^\ell|+|S_1^\ell|\Big]\Big)\quad(\sharp)$$

and then we have to multiply by the number of rounds R to get C.

For the $\tilde{Z}K(d)$ scheme, we have $\tilde{M}=M+n^{\lceil\frac{d}{2}\rceil}/\lceil\frac{d}{2}\rceil!$.
Then $\tilde{C}=3\tilde{M}\tilde{R}$.

8 An application of $ZK(3)$ and $\tilde{ZK}(3)$ to Brent equations

8.1 Brent equations

These equations arise when we want to compute efficiently the product of two $N\times N$ matrices in only s multiplications of entries. Of course, we require s to be less that N^3, since the naive method uses N^3 multiplications. For $N=2$, Strassen's algorithm [19] requires 7 multiplications instead of 8 multiplications and Laderman showed that for $N=3$, it is possible to use 23 multiplications instead of $3^3=27$ [9]. For $N=2$, the least number we can obtain is seven. For $N=3$, it is not known if 23 is the least number in the non-commutative case. For the commutative case, only 22 multiplications are needed. However we still do not know if it is the least. But the non-commutative case is more important since it allows to consider the product of matrices whose entries are themselves matrices and not only scalars. So the same technique can be generalized for larger matrices.

For sake of completeness, we recall how we get Brent equations (cf [1]). Suppose that we want to compute the product of the two matrices $A=(A_{ij})_{\substack{1\le i\le N\\1\le j\le N}}$ and $B=(B_{ij})_{\substack{1\le i\le N\\1\le j\le N}}$. We set $C=AB$ with $C=(C_{ij})_{\substack{1\le i\le N\\1\le j\le N}}$. We want to compute AB using only s multiplications. We will use the following method. We have to find coefficient $(\alpha_{abk}),(\beta_{cdk}),(\gamma_{ijk})$, with $1\le k\le s$ and $a,b,c,d,i,j\in\{1,2,\ldots,n\}$. These coefficients do not depend on the choice of the matrices. Now for $1\le k\le s$, we set $L_k=\sum_{a,b}\alpha_{abk}A_{ab}$, $R_k=\sum_{cd}\beta_{cdk}B_{cd}$ and $P_k=L_kR_k$. Then we have $C_{ij}=\sum_{k=1}^{s}\gamma_{ijk}P_k$. Since matrix multiplication is a bilinear map and the space of matrices is a linear space, we can apply the process to each element of the basis and we are led to the so-called "Brent equations":

$\sum_{k=1}^{s}\gamma_{ijk}\alpha_{abk}\beta_{cdk}=\delta_{bc}\delta_{ia}\delta_{jd}$ where $a,b,c,d,i,j\in\{1,2,\ldots,n\}$. This is a system

of cubic equations and with the previous notations, we have: $n = 3sN^2$ and $m = N^6$. It is still unknown if it is possible to multiply 3×3 matrices with less than 23 multiplications. The best known algorithms so far for solving the Brent equations for the 3×3 matrix multiplication require more than 2^{80} computations. Generally the work done consists in looking for solutions in a restricted area (with more symmetries for example), or to try some algorithms such as SAT Solver for some time (as Nicolas Courtois proposed in [3]).

8.2 A zero-knowledge arguments of knowledge based on Brent equations

The idea is to be as close as we can of the Brent equations in order to have informal arguments to show that if somebody is able to break our authentication scheme, then there is "probably a high probability" that he will able to break the Brent equations as well. Thus we will not use exactly Brent equations to construct a zero-knowledge arguments of knowledge. In order to be very near the Brent equations, but with a solution, we have essentially made 2 changes:
a) We consider the equations mod 2. This is justified in the sense that if a solution of the Brent equation would be found mod 2, it would some insight about the real solution and allow us to progress in finding a solution (it would cut the problem in easier independent steps).
b) We changed only the constant terms in the Brent equations, i.e. all the multivariate polynomials are exactly the same, except the constant terms. Then, we have generated some random input in the equations, so our constant terms will look random. Of course, we cannot exclude the possibility that somebody may found one day an algorithm that would be able to solve the Brent equations with its very specific constant terms, and not with random constant terms. However, with all the known algorithms on multivariate polynomials this seems to be at present rather unlikely since the complexity of these known algorithms do not really depend of the specific constant terms (as soon as they are not all 0 for example). More precisely, we proceed as follows:

1. We consider the finite field $\mathbb{F}_2$.
2. We take the left part of the Brent equations with $s = 22$ in order obtain an open problem in the non-commutative case.
3. We pick randomly variables $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ in ${\mathbb{F}_2}^{N^2 s}$.
4. We deduce the values $\sum_{k=1}^{s} \gamma_{ijk}\alpha_{abk}\beta_{cdk}$ where $a, b, c, d, i, j \in \{1, 2, \ldots, n\}$. This gives the value $\boldsymbol{v} \in {\mathbb{F}_2}^M$.
5. We then use either $ZK(3)$ (or $\tilde{ZK}(3)$) to have a zero-knowledge arguments of knowledge with the system $\boldsymbol{F}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \boldsymbol{v}$. Here $\boldsymbol{\alpha} = (\alpha_{abk})$, $1 \leq k \leq s$, $a, b \in \{1, 2, \ldots, n\}$, $\boldsymbol{\beta} = (\beta_{cdk})$, $1 \leq k \leq s$, $c, d \in \{1, 2, \ldots, n\}$ and $\boldsymbol{\gamma} = (\gamma_{ijk})$, $1 \leq k \leq s$, $i, j \in \{1, 2, \ldots, n\}$. The system is sparse and $\boldsymbol{F} \in MC(n, m, \mathbb{F}_2)$ where $n = 794$ and $m = 729$.

8.3 More comments about Brent equations

Solving Brent equation would be a real progress for the scientific community, and would accelerate many algorithms. Therefore we think that it is really interesting to have a security based on them or as near as possible of them. The problem is mathematically and for computer science interesting and valuable. Similarly, it is interesting to have a security based on factorization for example instead of based on a new an unknown problem, since this problem has been studied for years. Indeed, a devastating progress in factoring (as for Brent) would be scientifically interesting, i.e. it would not be just sad news for the crypto-systems, but also good news for the scientific community.

9 Morphisms of polynomials and systems of cubic equations

9.1 The IP and MP Problems

We explain the IP and MP problem in the case of quadratic forms. Let u and n two integers. On $\mathbb{F}_q$, we consider a public set $(\mathcal{A})$ of public quadratic equations:

$$(\mathcal{A}) \quad c_k = \sum_{1\leq i\leq n,\, 1\leq j\leq n} \gamma_{ij}^k a_i a_j + \sum_{i=1}^{n} \mu_i^k a_i + \delta_k, \quad 1 \leq k \leq u.$$

Let S be a bijective affine transformation of the variables $a_i,\ 1 \leq i \leq n$ and T be a bijective affine transformation of the variables $c_k,\ 1 \leq k \leq u$. We have $S(a_1, \ldots, a_n) = (x_1, \ldots, x_n)$ and $T(c_1, \ldots, c_u) = (z_1, \ldots, z_u)$. Then from $(\mathcal{A})$, we obtain another set $(\mathcal{B})$ of u equations:

$$(\mathcal{B}) \quad z_t = \sum_{1\leq i\leq n,\, 1\leq j\leq n} \alpha_{ij}^t x_i x_j + \sum_{i=1}^{n} \beta_i^t x_i + \omega_t, \quad 1 \leq t \leq u.$$

We say that (S, T) is an isomorphism from $(\mathcal{A})$ to $(\mathcal{B})$ and that $(\mathcal{A})$ and $(\mathcal{B})$ are isomorphic.
The IP (isomorphism of polynomials) problem is the following: if $(\mathcal{A})$ and $(\mathcal{B})$ are two public sets of u quadratic equations, find an isomorphism (S, T) from $(\mathcal{A})$ to $(\mathcal{B})$.
When S and T are not bijective the corresponding problem is the MP (morphism of polynomials) problem.
The IP problem (Isomorphism of Polynomials) has been used to construct pubic key schemes (cf [11]). On one hand, this is not a NP-complete problem since it admits an Arthur-Merlin game when the answer is yes and when the answer is no. On the other hand, the MP problem (morphisms of polynomials) where matrices are not supposed to be invertible is proved to be NP-complete [4, 12] and thus is much more difficult. So it is interesting to design a public key authentication scheme based on MP. We explain briefly below how it is possible to construct such a scheme by transforming MP very efficiently into a system of equations of degree 3 and then applying our $ZK(3)$ or $\tilde{ZK}(3)$ protocols.

9.2 A zero-knowledge arguments of knowledge based on the MP problem

We consider the two following systems:

$$(\mathcal{A}) \quad c_k = \sum_{1\leq i\leq n,\, 1\leq j\leq n} \gamma_{ij}^k a_i a_j + \sum_{i=1}^{n} \mu_i^k a_i, \quad 1 \leq k \leq u$$

$$(\mathcal{B}) \quad z_t = \sum_{1\leq i\leq p,\, 1\leq j\leq p} \alpha_{ij}^t x_i x_j + \sum_{i=1}^{p} \beta_i^t x_i, \quad 1 \leq t \leq v$$

We want to find 2 matrices $M = (m_{rs})_{\substack{1\leq r\leq v\\ 1\leq s\leq u}}$ and $H = (h_{df})_{\substack{1\leq d\leq n\\ 1\leq f\leq p}}$ such that

$$M \begin{pmatrix} c_1 \\ \vdots \\ c_u \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_v \end{pmatrix} \quad \text{and} \quad H \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}.$$

For all $t,\ 1 \leq t \leq v$, on one hand, we have: $z_t = \sum_{s=1}^{u} m_{ts} \Big(\sum_{\substack{1\leq i\leq n\\ 1\leq j\leq n}} \gamma_{ij}^s a_i a_j + \sum_{i=1}^{n} \mu_i^s a_i \Big)$

$z_t = \sum_{s=1}^{u} m_{ts} \Big(\sum_{\substack{1\leq i\leq n\\ 1\leq j\leq n}} \gamma_{ij}^s (\sum_{f=1}^{p} h_{if} x_f)(\sum_{b=1}^{p} h_{jb} x_b) + \sum_{i=1}^{n} \mu_i^s (\sum_{f=1}^{p} h_{if} x_f) \Big)$

$z_t = \sum_{f=1}^{p}\sum_{b=1}^{p} \Big[\sum_{s=1}^{u} \sum_{\substack{1\leq i\leq n\\ 1\leq j\leq n}} \gamma_{ij}^s m_{ts} h_{if} h_{jb} \Big] x_f x_b + \sum_{f=1}^{p} \Big[\sum_{s=1}^{u}\sum_{i=1}^{n} \mu_i^s m_{ts} h_{if} \Big] x_f$. On the other hand, we have: $z_t = \sum_{\substack{1\leq i\leq p\\ 1\leq j\leq p}} \alpha_{ij}^t x_i x_j + \sum_{i=1}^{p} \beta_i^t x_i$. This gives $\forall t\, 1 \leq t \leq v,\ \forall f, 1 \leq f \leq p,\ \forall b, 1 \leq b \leq p,\ \alpha_{fb}^t + \beta_f^t = \sum_{s=1}^{u} \sum_{\substack{1\leq i\leq n\\ 1\leq j\leq n}} \gamma_{ij}^s m_{ts} h_{if} h_{jb} + \sum_{s=1}^{u}\sum_{i=1}^{n} \mu_i^s m_{ts} h_{if}$. Thus we obtain a system of vp^2 cubic equations with $np + vu$ unknowns. Then it is possible to construct zero-knowledge arguments of knowledge with $ZK(3)$, (or $\tilde{ZK}(3)$).

10 Conclusion

In [15], a very efficient zero-knowledge proof based on the MQ problem (multivariate quadratic polynomials) is given. In this paper we proved that this construction can be generalized to polynomials of degree d for any $d \geq 3$ (unlike in [14] where a construction valid only for $d = 3$ was given) . We studied several constructions and we presented here the two most efficient ones denoted by

$ZK(d)$ and $\tilde{ZK}(d)$. $ZK(d)$ is more efficient in terms of number of communication bits, and $\tilde{KZ}(d)$ in terms of computations. In table 4, we can compare our schemes for $d = 3$ from the recent scheme [14]. It is interesting to notice that if the polynomials are sparse, our schemes will be able to use this fact in order to be more efficient (for $\tilde{ZK}(d)$ for example, the numbers of computations is still proportional to the time to compute the polynomial on a given point). Finally, we also presented two important specific problems (Brent equations and morphisms of polynomials) that can be transformed into efficient public key schemes using $ZK(d)$ and $\tilde{ZK}(d)$.

Table 4. Comparison of schemes on 80-bit security against key-recovery attack when the impersonation probability is less than 2^{-30}

	MC	$ZK(3)$	$\tilde{ZK}(3)$
round	73	73	52
communication (bit)	53,290	74,022	560,000
arithmetic operations (times/field)	$2^{32}/\mathbb{F}_2$	$2^{33}/\mathbb{F}_2$	$2^{31}/\mathbb{F}_2$
generalization for $d \geq 4$	NO	YES	YES

References

1. Gregory V. Bard. New Practical Strassen-like Approximate Matrix-multiplication Algorithms found via solving a system of cubic equations. *http://www.users.math.umd.edu/~bardg/*.
2. Come Berbain, Henri Gilbert, and Jacques Patarin. QUAD: A Practical Cipher with Provable Security . In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4404 of *Lecture Notes in Computer Science*, pages 109–128. Springer-Verlag, 2006.
3. Nicolas Courtois, Gregory V. Bard, and Daniel Hulme. A new general-purpose method to multiply 3x3 matrices using only 23 multiplications. *CoRR*, abs/1108.2830, 2011.
4. Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completness*. W.H. Freeman and Co, 1979.
5. Oded Goldreich. *Foundations of Cryptography: Volume I. Basic Tools*. Cambridge University Press, Cambridge, 2001.
6. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38:690–728, July 1991.
7. Shai Halevi and Silvio Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing . In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer-Verlag, 1996.
8. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*, volume 1492 of *Lecture Notes in Computer Science*, pages 206–222. Springer-Verlag, 1999.

9. J. Laderman. A noncomuutative algorithm for multiplying 3 x3 matrices using 23 multiplication. *Bulletin of the American Mathematical Society*, 82:126–128, 1976.
10. Tsutomu Matsumo and Hideki Imai. Public Quadratic polynomial-Tuples for Efficient Signature-Verification and Message-Encryption . In C.G. Gunther, editor, *Advances in Cryptology – EUROCRYPT 1988*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer-Verlag, 1988.
11. Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphims of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer-Verlag, 1996.
12. Jacques Patarin and Louis Goubin. Trapdoor one-way permutations and multivariate polynomials. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *ICICS*, volume 1334 of *LNCS*, pages 356–368. Springer, 1997.
13. David Poincheval. A New Identification Scheme based on the Perceptrons Problem. In Alfredo de Santis, editor, *Advances in Cryptology – EUROCRYPT 1995*, volume 950 of *Lecture Notes in Computer Science*, pages 319–328. Springer-Verlag, 1995.
14. Koichi Sakumuto. Public-Key Identification Schemes based on Multivariate Cubic Polynomials. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography– PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 172–189. Springer-Verlag, 2012.
15. Koichi Sakumuto, Taizo Shirai, and Harunaga Hiwatari. Public-Key Identification Schemes based on Multivariate Quadratic Polynomials. In Philip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 706–723. Springer-Verlag, 2011.
16. Adi Shamir. An Efficient Identification Scheme Based on Permuted Kernels (Extended Abstract). In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO 1989*, volume 435 of *LNCS*, pages 606–609. Springer-Verlag, 1989.
17. Jacques Stern. A New Identification Scheme based on Syndrome Decoding. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.
18. Jacques Stern. Designing Identification Schemes with Keys of Short Size. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 164–173. Springer, 1994.
19. Volker Strassen. Gaussion Elimination is not optimal. *Numerische Mathematik*, 13(3):354–356, 1969.
20. Erik G.F. Thomas. A Polarizatin Identity for Multilinear Maps. *University of Groningen - Preprint*, (1997).