Multivariates Polynomials for Hashing

Jintai Ding[1] and Bo-Yin Yang[2]

[1] University of Cincinnati and Technische Universität Darmstadt
ding@math.uc.edu
[2] Institute of Information Science, Academia Sinica
by@moscito.org

Abstract. We propose the idea of building a secure hash using quadratic or higher degree multivariate polynomials over a finite field as the compression function. We analyze some security properties and potential feasibility, where the compression functions are randomly chosen high-degree polynomials, and show that under some plausible assumptions, high-degree polynomials as compression functions has good properties. Next, we propose to improve on the efficiency of the system by using some specially designed polynomials generated by a small number of random parameters, where the security of the system would then relies on stronger assumptions, and we give empirical evidence for the validity of using such polynomials.

Keywords: hash function, multivariate polynomials, sparse.

1 Introduction

There is a rather pressing need to find new hash functions as of today, after the work of Wang et al culminated in collisions of some well-known and standard hash functions [13,14]. One common feature of the currently used hash functions is that it is more an area of art, where the design of the system is based on certain procedures and the security of the system is very very difficult to study from the theoretical point of view. For example, even the work of Wang et al is still not well understood and people are still trying to analyze the methods used in some systematical way. [12]

Naturally, one direction is to look for provably secure hash functions, whose security relies on well-understood hard computational problems. These hashes tend to be slower, although they have a saving grace in terms of having some measure of provable security.

In these formulations, the designer seeks a reduction of the security of the compression function to some other intractable problem. Of course, we should be careful about these "provably secure" constructions. There are two pitfalls:

– Often, the security reduction has a large looseness factor. The practical result is that these reductions end up "proving" a very low security level — the complexity of the "underlying hard problem" divided by the looseness factor — which in a practically-sized instance will be insufficient to satisfy security requirements [10,9,17].

It should be mentioned, however, that often times this kind of reductions shows that if hard problem A is exponential in a power of n, then so is cryptosystem B under suitable assumptions. So having a proof still beats not having one because it implies that there are no real serious gotchas.
- The other possibility is that the security of the hash function may only loosely depend on the hard problem. The case of NTRU's signature schemes exemplify this possibility. One can only say in this case that the scheme is inspired by the hard problem.

In this paper, we propose our own version, which is inspired by the $\mathcal{MQ}$ problem, and study how well it can work. Our paper is arranged as follows. We first study the case of random systems. Then we study the case of sparse construction and present the main theoretical challenges and its practical applications. We will then discuss other new ideas and the future research in the conclusion. Much work remains in this area in terms of security reductions and speed-ups.

2 The MQ Frame Work

Problem $\mathcal{MQ}$: Solve a polynomial system, with coefficients and variables in $K = \mathrm{GF}(q)$, where each p_i is randomly chosen and quadratic in $\mathbf{x} = (x_1, \ldots, x_n)$.

$$p_1(x_1, \ldots, x_n) = 0,\ p_2(x_1, \ldots, x_n) = 0,\ \ldots,\ p_m(x_1, ..., x_n) = 0, \qquad (1)$$

$\mathcal{MQ}$ is NP-hard generically [7]. If instead of quadratics, each p_i is of degree $d_i > 1$, the problem may be termed called $\mathcal{MP}$, which is of course no easier than $\mathcal{MQ}$, so must also be NP-hard. That a problem is NP-hard generically need not mean that its solution is of exponential time complexity in its parameters *on average*, or imply anything about the coefficients. However, today, as m and n increases to infinity, we believe that the following holds.

Conjecture 2.1. $\forall \epsilon > 0$, $\Pr(n/m = \Theta(1)$, $\mathcal{MQ}$ can be solved in $\mathrm{poly}(n)) < \epsilon$.

The conjecture above or something similar to it is the basis of multivariate public-key cryptosystems [6,15] as well as some symmetric cryptosystems [3]. The latest way of tackling such systems involve guessing at some optimal number of variables (depending on the method in the closing stage) then use a Lazard-Faugère solver (XL or $\mathbf{F_5}$, [2]). We also believe, and it is commonly accepted that the complexity to solve the set of equations is indeed exponential, if $p_i(x_1, \ldots, x_n)$ are polynomials of a fixed degree whose coefficients are randomly chosen — say that $p_i(x_1, ..., x_n)$ are randomly chosen quadratic polynomials. We will assume this statement as the security foundation.

Under this assumption, the first straightforward idea is to build an iterated hash using completely random quadratic polynomials as compression functions, namely the one way function $F : K^{2n} \to K^n$ is given by

$$F(x_1, \ldots, x_n, y_1, \ldots, y_n) = (f_1(x_1, \ldots, x_n, y_1, \ldots, y_n), ..., f_n(x_1, ..., x_n, y_1, \ldots, y_n)),$$

where each f_i is a randomly chosen quadratic polynomial over GF(q). All the coefficients are chosen randomly. We will use this as a compressor with state $\mathbf{x} = (x_1, \ldots x_n)$, and each input message block is $\mathbf{y} = (y_1, \ldots, y_n)$. In the following, we show that this cannot work and suggest the next improvements, particularly cubic and stacked (composed) quadratics, and the idea of using the least number of parameters to determine our maps.

2.1 General Remark on Solvability

A rough estimate of effort to solve these $\mathcal{MQ}$ problems: 20 quadratic equations in 20 GF(256) variables: 2^{80} cycles; 24 quadratic in 24 GF(256) variables: 2^{92} cycles. It is harder to solve higher-order equations: for reference, 20 cubic equations in 20 GF(256) variables takes about 2^{100} cycles, 24 cubics in 24 GF(256) variables about 2^{126} cycles, and 20 quartics in in 20 GF(256) variables about 2^{128} cycles.

Clearly, the problem for our hash schemes is not going to be the direct solution (algebraic attack) using a polynomial system solver. The above shows that any multivariate polynomial systems are not really susceptible to preimage or second preimage attacks.

Note: There is a special multivariate attack to solve under-defined systems of equations [5] that applies to this situation where there is a lot many more variables than equations, but

– the number of variables that it reduces is proportional to the square root of n, and
– for $q > 2$ under most optimistic estimates it has proved to be rather useless.

We would then, of course, try multivariate quadratics as the obvious idea. However, it is not secure because collisions are easy to find:

Proposition 2.1. *With randomly chosen $F := F(\mathbf{x}, \mathbf{y})$, it is easy to solve the equation.*

$$F(\mathbf{x}_1, \mathbf{y}_1) = F(\mathbf{x}_2, \mathbf{y}_2)$$

Proof

$$F(\mathbf{x} + \mathbf{b}, \mathbf{y} + \mathbf{c}) - F(\mathbf{x}, \mathbf{y}) = 0$$

is a linear equation in $2n$ variables $(\mathbf{x}, \mathbf{y})$ and n equations, so must be solvable.

However, we this points out a better way to building a secure hash.

3 The Applications of Sparsity

In Section 3 we show that the compressor and hence the hash function is likely to be collision-free. In the following sections, we propose some naive instances, which has the security level at 2^{80} or 2^{128}, but those system are very slow for practical applications. A natural idea is to use sparse polynomials. This is as in the case of the central maps of TTS signature schemes, namely we will choose each of the above polynomial to be a sparse polynomial. However, the security relies on the following stronger assumption:

Conjecture 3.1. As $n \to \infty$ and $m/n \to k \in \mathbb{R}^+$, for any fixed $0 < \epsilon < 1$ a random sparse polynomial system of any fixed degree with a randomly picked ϵ proportion of the coefficients being non-zero (and still random, if $q > 2$) will still take time exponential in n to solve.

This can be termed a version of the conjecture S$\mathcal{M}P$ (sparse multivariate polynomial). S$\mathcal{M}P$ is only one of the many conjectures that we can make, because as long as we are not dealing with packed bits, the speed of the implementation depends more on the size of the parameter set than how we use the parameters. If a set of polynomials is determined from a relatively small set of parameters, we can call them "sparsely determined".

There is no real reduction known today that reduces S$\mathcal{M}P$ to a known hard problem. However, we can justify this assumption somewhat by trying to solve these sparse systems, and by extension sparsely generated systems, using the best tools we have.

– If we solve the system with an XL-with-Wiedemann like attack [16,17], it is clear that the running time will simply be ϵ times that of the corresponding dense problem.
– There is no commercially available implementation of $\mathbf{F_5}$, however, it runs at the same degree as $\mathbf{F_4}$ and mostly differ in the fact that $\mathbf{F_5}$ avoids generating extraneous equations that are bound to be eliminated.
– The most sophisticated solver commercially available is MAGMA [4]. We ran many tests on random systems, random sparse systems, and not necessarily sparse but sparsely generated systems. In every case, solving the sparse determined systems takes roughly the same amount of time and memory as the non-sparse ones.
– We know of some specialized methods that solves sparse systems, but some have a different definition of sparsity than we have here [11], and some are not well-quantified.

Please see the appendix for the tests that we ran. Clearly, the key is to choose wisely a correct proportion ϵ of the nonzero terms.

1. How many we choose such that it is fast?
   Our answer: no more than maybe 0.1% (see below).
2. How many we choose such that it is secure?
   Our answer: a predetermined fixed percentage.
3. How do we choose the sparse terms?
   Our answer: probably randomly.

4 Cubic Construction

Suppose that $q = 2^k$ and the F above is changed to a random *cubic* $K^{2n} \to K^n$, then the following argument that this compression function is secure.

Let $\bar{K}$ be a degree n extension of K. We have a map ϕ which identifies $\bar{K}$ as K^n.

Then it is clear from the work of Kipnis and Shamir [8] we can show that F can be lifted to be a map from $\bar{K} \times \bar{K}$ to $\bar{K}$. Then we have

$$\bar{F}(X,Y) = \sum A_{ijk} X^{q^i+q^j+q^k} + \sum B_{ijk} X^{q^i+q^j} Y^{q^k} +$$
$$\sum C_{ijk} X^{q^i} Y^{q^j+q^k} + \sum D_{ijk} Y^{q^i} X^{q^j+q^k} +$$
$$\sum E_{ij} X^{q^i} Y^{q^j} + \sum F_{ij} X^{q^i} X^{q^j} + \sum G_{ij} Y^{q^i} Y^{q^j} +$$
$$\sum H_i X^{q^I} + \sum I_i Y^{q^i} + J.$$

In this case, we can view all the coefficients as random variables.

We can show that no matter how one the choose the difference of the (X,Y) coordinates that the coefficients of the difference, namely

$$\bar{F}(X+X', Y+Y') - \bar{F}(X,Y)$$

are still linearly independet random variables as a quadratic function.

By mathematical induction, we can prove the following theorem.

Proposition 4.1. *Define a function in the Kipnis-Shamir form*

$$\bar{F}(X,Y) = \sum A_{ijk} X^{q^i+q^j+q^k} + \sum B_{ijk} X^{q^i+q^j} Y^{q^k} \sum C_{ijk} X^{q^i} Y^{q^j+q^k} + \sum D_{ijk} Y^{q^i+q^j+q^k}$$
$$+ \sum E_{ij} X^{q^i} Y^{q^j} + \sum F_{ij} X^{q^i} X^{q^j} + \sum G_{ij} Y^{q^i} Y^{q^j} + \sum H_i X^{q^I} + \sum I_i Y^{q^i} + J.$$

where each coeffcients are linear independent random variables and i, j, k are less than a fixed integer r, the nonzero coefficients of the function

$$\bar{F}(X+X', Y+Y') - \bar{F}(X,Y)$$

are also linearly independ random variables.

Proof. Let us first assume that r is zero, we have that

$$\begin{aligned}
&\bar{F}(X+X', Y+Y') - \bar{F}(X,Y) \\
&= (3A_{000}X' + B_{000}Y')X^2 + (C_{000}X' + 3D_{000}Y')Y^2 + (2X'B_{000} + 2Y'C_{000})XY \\
&+ (3A_{000}X'^2 + 2X'Y'B_{000} + Y'^2C_{000} + E_{00}Y' + 2F_{00}X')X \\
&+ (3D_{000}Y'^2 + 2X'Y'C_{000} + X'^2B_{000} + 2G_{00}Y' + E_{00}X')Y \\
&+ A_{000}X'^3 + B_{000}X'^2Y' + C_{000}X'Y'^2 + D_{000}Y'^3 + E_{00}X'Y' + F_{00}X'^2 + G_{00}Y'^2 + H_0X' + I_0Y'
\end{aligned}$$

The first three terms' coefficients are clearly independent, and involves only the $A_{000}, B_{000}, C_{000}, D_{000}$, as long as (X', Y') are not both zero. Then in the fourth and the fifth term we have $E_{00}Y' + 2F_{00}X'$ in X term and $2G_{00}Y' + E_{00}X'$ in the Y term. Therefore the initial 5 coefficients are linearly independent. Finally, we know that $H_0X' + I_0Y'$ in the constant term ensures that all the coefficients are linearly independent.

Clearly we can see that we write the difference above in the form of

$$\bar{F}(X+X', Y+Y') - F(X,Y) = \bar{A} \times Q \times Z^t,$$

according to the order of degree of the (X, Y) terms, where

$$\bar{A} = (A_{000}, ..., I_0),$$

and Q is the coefficient matrix (which may depend on X', Y') and

$$Z = (X^2, Y^2, XY, X, Y, 1)$$

Then Q clearly have a blockwise triangular structure, which makes the proof possible.

Then suppose the theorem is proved when $i, j, k < r$, we can use the same trick to proceed to $i, j, k \leq r$, namely we will write down

$$\bar{F}(X + X', Y + Y') - \bar{F}(X, Y) = \bar{A} \times Q \times Z^t,$$

where Z is arranged in the form that the first block are the terms involves terms that either have X^{q^r} or Y^{q^r}, then the rest. The $\bar{A}$ is arranged in the same way that the first block contains the terms whose subindices must have r. Then we can write down the expresssion explicitly and show the independence, just as in the case $r = 0$. We will omit the tedious detail here.

Corollary 4.1. *For a quadratic map $M : K^{2n} \to K^n$ with uniformly chosen random coefficients, we can construct a cubic map $\bar{F} : K^{2n} \to K^n$ and a differential X', Y' such that $M(X, Y) = \bar{F}(X + X', Y + Y') - \bar{F}(X, Y)$, such that X', Y' and $\bar{F}$ have uniformly random coefficients or components.*

From this proposition, we can infer directly the following important conclusions.

Proposition 4.2. *If $\bar{F}$ is randomly chosen,the function $\bar{F}(X + X', Y + Y') - \bar{F}(X, Y)$ is also random as long as X' and Y' are not both zero.*

Therefore we have

Proposition 4.3. *If F is randomly chosen,the function $F(\mathbf{x}+\mathbf{b}, \mathbf{y}+\mathbf{c})-F(\mathbf{x}, \mathbf{y})$ is also random as long as $\mathbf{b}$ and $\mathbf{c}$ are not both zero.*

Theorem 4.1. *A random cubic $F : K^{2n} \to K^n$ (written as $F := F(\mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \in K^n$) is*

1. *impossible to invert or to find a second image for, in polynomial time.*
2. *is impossible find a collision for, in polynomial time.*

Proof. From the proposition above, we can assume a attacker knows the difference of the collision he or she intens to find. In this case, it means that the attacker have the equation $F(\mathbf{x} + \mathbf{b}, \mathbf{y} + \mathbf{c}) - F(\mathbf{x}, \mathbf{y}) = 0$ that he or she must solve if he or she can find the collision. However, we just showed that no matter how one chooses $\mathbf{b}$ and $\mathbf{c}$, the equation can be viewed as a random equation. Therefore it is impossible to solve in polynomial time.

5 Random Cubic Polynomial

We use completely random cubic polynomials, namely the one way compression function is given by

$$F(\mathbf{x}, \mathbf{y}) = (f_1(\mathbf{x}, \mathbf{y}), ..., f_n(\mathbf{x}, \mathbf{y})),$$

where f_i is a randomly chosen cubic polynomial over GF(q). Here $\mathbf{x}, \mathbf{y} \in K^n$ as before. All the coefficients are chosen randomly. We will use this as a compressor in a Merkle-Damgård iterated compression hash function. For example we may use the following Merkle-Damgård like structure:

State: $\mathbf{x} := (x_1, x_2, \ldots, x_n)$.
Incoming Data Block: $\mathbf{y} := (x_{n+1}, \ldots, x_{2n})$.
Compression: $F : (\mathrm{GF}(q))^{2n} \to (\mathrm{GF}(q))^n$.
Initial State: $F(P(a_1, \ldots, a_{n/2}), P(a_{n/2+1}, \ldots, a_n))$, P is a random given quadratic $K^{n/2} \to K^n$.
According to [3], the output of P is impossible to predict or distinguish from random.
Final Output: in $(\mathrm{GF}(q))^n$, possibly with some post-treatment.

Assuming 160-bit hashes (SHA-1), preliminary runs with a generic F:

– 40 GF(256) variables into 20: 67300 cycles/byte (6.0 cycles/mult)
– 80 GF(16) variables into 40: 4233000 cycles/byte (3.2 cycles/mult)

Assuming 256-bit hashes (SHA-2), preliminary runs with a generic F:

– 32 GF(2^{16} variables into 16: 48200 cycles/byte (19 cycles/mult)
– 64 GF(256) variables into 32: 3040000 cycles/byte (6.0 cycles/mult)
– 128 GF(16) variables into 64: 9533000 cycles/byte (3.2 cycles/mult)

Some implementation details:

– The coefficients of the map F is taken from the binary expansion of π.
– Multiplication in GF(16) and GF(256) are implemented with tables. In fact, in GF(16), we implement a 4kBytes table with a where we can multiply simultaneously one field element by two others.
– Multiplication in GF(2^{16}) is implemented via Karatsuba multiplication over GF(256).

But using a random cubic system is not very efficient in general, the new idea is the next one namely we will use sparse polynomials.

5.1 Sparse Cubic Polynomial

The idea is the same as above but we choose each of the above cubic polynomial to be sparse. Note that due to the new result by Aumasson and Meier [1], we now advise that only the *non-affine* portion of the polynomials be sparse.

The key point is to pick a the ratio ϵ of the nonzero terms. In general, storing the formula in a sparse form takes extra space and time to unscramble the storage, so it is never worthwhile to have $\epsilon > 1/10$ or so in practice. In the examples in the following, less than 0.2% of the coefficients are non-zero (one in 500). To give one example, there is around 30 terms per equation in a 40-variable cubic form.

Assuming 160-bit hashes (SHA-1), preliminary runs with a generic sparse F:

- 40 GF(256) variables into 20, 0.2%: 215 cycles/byte
- 80 GF(16) variables into 40, 0.1%: 4920 cycles/byte

Assuming 256-bit hashes (SHA-2), preliminary runs with a generic sparse F:

- 32 GF(2^{16} variables into 16, 0.2%: 144 cycles/byte
- 64 GF(256) variables into 32, 0.1%: 3560 cycles/byte
- 128 GF(16) variables into 64, 0.1%: 9442 cycles/byte

6 Stacked (Composed) Quadratics

Having noted that cubics don't work so well, another way is to have quartics which are composed of quadratics. The first quadratic maps $2n$ variables to $3n$, the second one then maps the $3n$ to n. This avoids the problem by using a set of quartic that can still be computed relatively quickly.

The first question is, whether this is a good idea.

Proposition 6.1. *Let the compression function F be equal to $F_2 \circ F_1$, with generic $F_1 : K^{2n} \to K^{3n}$, $F_2 : K^{3n} \to K^n$.*

Proof. Schematically, $F_2^{-1} = F_1 \circ F^{-1}$. Hence If we can invert F, then we can invert F_2. This is hard by assumption.

Proposition 6.2. *F is collision-resistant.*

Proof (Heuristic). We note that F_1 has no collisions on average, and if it has a pair it is hard to find. Thus, a collision on F means a collision on F_2, which poses no problem, but then we will have to solve F_1 twice, which is difficult by assumption.

Now, let us consider a direct differential attack and compute $F^{(\mathbf{b},\mathbf{c})}$. Using the chain rule, all the polynomials belong to the ideal generated by the polynomials of F_1. Since for generic polynomials we should not see a reduction to zero under the degree of regularity [2], the solution is at as hard as inverting F_1.

Assuming 160-bit hashes (SHA-1), preliminary runs with a function $F = F_2 \circ F_1$, with generic $F_1 : K^{2n} \to K^{3n}$, $F_2 : K^{3n} \to K^n$

- 40 GF(256) variables into 20: 27400 cycles/byte
- 80 GF(16) variables into 40: 101200 cycles/byte

Assuming 256-bit hashes (SHA-2), preliminary runs with $F = F_2 \circ F_1$, generic F_1, F_2:

– 32 GF(2^{16} variables into 16: 24100 cycles/byte
– 64 GF(256) variables into 32: 64200 cycles/byte
– 128 GF(16) variables into 64: 247000 cycles/byte

In this way we can explore another form of sparsity, which relies on the idea is that any quadratic map can be written as $f \circ L$, where f is a certain standard form and L is an invertible linear map. Now we will instead choose L to be sparse.

In this instance, the key question are still the same:

1. How many we choose such that it is fast?
2. How many we choose such that it is secure?
3. How do we choose the sparse terms?

In this area, we note that it is not necessary that L be sparse, but only that it be decided by a relatively small number of parameters, and that the evaluation is fast. Along these lines, a new construction is the continued sparse compositions, where we use composition of sparse random linear maps. We propose some instances of these hash functions for practical applications. There are furthermore several new ideas that should be further studied.

6.1 Sparse Composition Factor: "Rotated" Quadratic Sets

The idea is that any quadratic map can be written as $f \circ L$, where f is a certain standard form and L is an invertible affine map. Now we will choose the linear part of L to be sparse. The obvious standard form for characteristic 2 fields is to start with the standard form ("rank form").

$$f_1(\mathbf{x}) = x_1x_2 + x_3x_4 + \cdots x_{n-1}x_n.$$

Let us explain a little why. Let k be a field of characteristic 2. A quadratic form in n variables over k is defined by $Q = \sum_{1\leq i\leq j\leq n} p_{ij}x_ix_j$, $p_{ij} \in F$.

Theorem 6.1. *Any quadratic form over k is equivalent to*

$$Q' = \sum_{i=1}^{\nu} x_iy_i + \sum_{j=\nu+1}^{\tau+\nu} \left(a_jx_j^2 + x_jy_j + b_jy_j^2\right) + \sum_{k=1}^{d} c_kz_k^2$$

with $c_k \neq 0$ and $2\nu + 2\tau + d \leq n$.

When $2\nu + 2\tau + d = n$, the form Q' is regular. The number d is the deficiency of the form and the form q' is completely regular, if Q' is regular and its deficiency is zero, which corresponding the case that the corresponding symmetric form is non-degenerate. A randomly chosen is in general expected to completely regular.

Furthermore, we have

Theorem 6.2. *If two quadratic forms over k, $\psi + q_1$ and $\psi + q_2$, are equivalent and ψ is completely regular, then q_2 and q_1 are equivalent over F.*

We will use this to give a general characterization of a quadratic function. Any quadratic function $f(x_1, ..., x_{2n})$ can be written in the form

$$f(x_1, .., x_n) = \sum_{1 \le i \le j \le 2n} a_{ij} x_i x_j + \sum_{1 \le i \le 2n} b_i x_i + c$$

where a_{ij}, b_i, c are in k. We know that through a linear transformation of the form $L(x_i) = x_i + d_i$, if the quadratic part is non degenerate, we can have that

$$f(L_1(x_1, .., x_n)) = \sum_{1 \le i \le j \le 2n} a'_{ij} x_i x_j + c'.$$

From the theorem above we know that there is a linear map L_2 such that

$$f((L_2 \circ L_1)(x_1, .., x_n)) = \sum_{1 \le i \le n} x_{2i-1} x_{2i} + \sum_{i \in S} x_i^2 + c',$$

where S is a set consisting of pairs in the form of $(2j - 1, 2j)$. The simplest form this kind of function is surely

$$f((L_2 \circ L_1)(x_1, .., x_n)) = \sum_{1 \le i \le n} x_{2i-1} x_{2i} + c',$$

and its difference from others in some sense are something of the linear nature, which can be neglected in some way. From this we conclude that a general quadratic function can be represented as: $F \circ L$, where

$$F = \sum_{1 \le i \le n} x_{2i-1} x_{2i} + c',$$

which is the basis we will use to build our hash function.

Knowing that any quadratic functions from $\mathrm{GF}(q)^k \to \mathrm{GF}(q)$ can be written this way. The key question are similar now: *How do we choose L such that it is fast? How many we choose such that it is secure? How do we choose the sparse terms?*

In this particular instance, there is something that leaps out at us. starting with $\mathbf{x}_1 := \mathbf{x}$, we compute $f_1(\mathbf{x}_1)$, then transform $\mathbf{x}_1 \mapsto \mathbf{x}_2 := L_2\mathbf{x}_1 + \mathbf{c}_2$, where L_2 has three randomly chosen entries in each row, and $f_2(\mathbf{x}) := f_1(\mathbf{x}_2)$. Continue in this vein and do $\mathbf{x}_2 \mapsto \mathbf{x}_3 := L_3\mathbf{x}_2 + \mathbf{c}_3$, $f_3(\mathbf{x}) := f_1(\mathbf{x}_3)$, and so on and so forth.

A set of quadratic polynomials like this we call "rotated". "Rotated" quadratics are obviously a kind of sparsely generated polynomials, and they behave like random ones under $\mathbf{F_4}$. *In Fig. 1 in the appendix, data points denoted "non-random sparse" polynomials are rotated.*

Assuming 160-bit hashes (SHA-1), preliminary runs with a composed rotated F:

- 40 GF(256) variables into 20: 1720 cycles/byte
- 80 GF(16) variables into 40: 3220 cycles/byte

Assuming 256-bit hashes (SHA-2), preliminary runs with a composed rotated F:

- 64 GF(256) variables into 32: 8100 cycles/byte
- 128 GF(16) variables into 64: 24100 cycles/byte

Again, we note that the transformation L has random constant terms. This leads to random affine parts in the structure throughout, to defend against the attack below.

7 Further Discussion: Variant Ideas and Other Attacks

We see that our hash schemes are roughly on a par speedwise with other schemes that depends on hard problems. It is true that there may be better formulations of the same idea, but $\mathcal{M}Q$ is a known hard problem, which should lend a measure of confidence.

7.1 Other Attacks

There are many specialized attacks in multivariate public key cryptography, especially the true multivariates, that one may think to use to attack our systems. But one should realize that due to the property of random polynomials, it is hard to imagine the usual attacks of linear and differential cryptanalysis working since cubic and quartic equations are so far removed from linearity. From what we can see, all but two related ones are now inapplicable to attack our hash.

These attacks are proposed by Aumasson and Meier [1]. The points are:

- If the affine part is as sparse as the rest of the polynomials, then there is a high probability to construct a collision or partial collision.
- There is a way to solve the differential of the hash if we use a cubic construction where the cubic terms are sparse, but this involves searching for a short vector in a code, and the exact time complexity is unknown.

Both of these ideas apply over GF(2) only, hence our use of GF(16) and larger. Of course, the specific points still bear more investigation.

7.2 Other Constructions

The key idea here is once we have a sparse construction, we would like to add some kind of internal perturbation to make system even more complicated. The key question is about how we add the perturbation. Another idea is to use a Feistel structure, which might speed up evaluation a lot with random maps, but requires more set-up and makes any putative pre-image resistance harder to show. Since the principle is not much different we don't go in that direction.

8 Conclusion

In this paper, we present the idea of using randomly polynomials, and randomly polynomials with sparse property to build hash functions. What is new here are: the cubic construction, the amplify-compress composed quadratic construction, and the specially construced systems using compositions of sparse linear functions.

We present arguments for the security of the system and for the case of sparse construction. We can improve our ideas with internal perturbation by adding additional noise into our sparse system. One more idea is to use composite field, where we explore further field structure to make our system more secure.

It is clear that some of these programming is very preliminary. The idea is mainly to point out a new direction in developing hash function whose security relies on a clear hard problems and therefore easier to study and understand, our work is just the beginning of this new direction and much work need to be done. We believe the multivariate hash has a very strong potential in practical applications. Much more work is needed in finding the right constructions and optimal parameters, and in rigorous proofs of security.

Acknowledgements

The authors would like to thank the Taiwan Information Security Center at Nat'l Taiwan U. of Sci. and Tech. location (NSC Project 95-2218-E-011-015) in 2006, the discussions following which led to this work.

In addition, BY and JD would like to thank the Taiwan's National Science Council and the Humboldt Foundation respectively for support (NSC 95-2115-M-001-021).

A version of this paper is also on http://eprint.iacr.org/2007/137.

References

1. Aumasson, J.-P., Meier, W.: Analysis of multivariate hash functions. In: Proc. ICISC. LNCS (to appear, 2007), cf.`http://www.131002.net/files/pub/AM07.pdf`
2. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving, pp. 71–74 (2004); Previously INRIA report RR-5049
3. Berbain, C., Gilbert, H., Patarin, J.: QUAD: A Practical Stream Cipher with Provable Security. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 109–128. Springer, Heidelberg (2006)
4. Computational Algebra Group, University of Sydney. The MAGMA Computational Algebra System for Algebra, Number Theory and Geometry, `http://magma.maths.usyd.edu.au/magma/`
5. Courtois, N., Goubin, L., Meier, W., Tacier, J.-D.: Solving Underdefined Systems of Multivariate Quadratic Equations. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, Springer, Heidelberg (2002)

6. Ding, J., Gower, J., Schmidt, D.: Multivariate Public-Key Cryptosystems. In: Advances in Information Security, Springer, Heidelberg (2006)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability — A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
8. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, Springer, Heidelberg (1999), `http://www.minrank.org/hfesubreg.ps` or `http://citeseer.nj.nec.com/kipnis99cryptanalysis.html`
9. Menezes, A., Koblitz, N.: Another Look at "Provable Security". II. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 148–175. Springer, Heidelberg (2006)
10. Koblitz, N., Menezes, A.: Another look at "provable security". Journal of Cryptology 20, 3–37 (2004)
11. Raddum, H., Semaev, I.: New technique for solving sparse equation systems. Cryptology ePrint Archive, Report 2006/475 (2006), `http://eprint.iacr.org/`
12. Sugita, M., Kawazoe, M., Imai, H.: Gröbner basis based cryptanalysis of sha-1. Cryptology ePrint Archive, Report 2006/098(2006), `http://eprint.iacr.org/`
13. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
14. Wang, X., Yu, H.: How to break md5 and other hash functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
15. Wolf, C., Preneel, B.: Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 64 (May, 2005) `http://eprint.iacr.org/2005/077/`
16. Yang, B.-Y., Chen, J.-M.: All in the XL Family: Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
17. Yang, B.-Y., Chen, O.C.-H., Bernstein, D.J., Chen, J.-M.: Analysis of QUAD. In: Biryukov, A. (ed.) Fast Software Encryption — FSE 2007. volume to appear of Lecture Notes in Computer Science, pp. 302–319. Springer, Heidelberg (2007) workshop record available

A Testing

We depict in Fig. 1 some MAGMA-2.13.6 tests we ran, with $K = \mathrm{GF}(2)$, $2n$ variables and $3n$ equations. At $n = 16$ the system started to thrash due to lack of memory.

Despite the fact that we had a very good workstation with 64 GB of main memory, this thrashing is perfectly in line with theory. I.e., it should run out of memory when $n = 16$.

The tests we did included quadratic, cubic, and a few quartic systems with number of variable n, number of equations m, and $n : m$ being roughly equal to 1, 0.8, 0.75, 2/3, and 0.5. We also did systems over GF(2), GF(16), GF(256), and we went up for as long as the memory allowed.

– Systems with every coefficient randomly chosen.
– Systems with ϵ proportion of coefficients randomly picked to be non-zero, then randomly chosen from non-zero field elements. We tried ϵ being two percent, one percent, half a percent, and 1 mil (that is 1/1000).

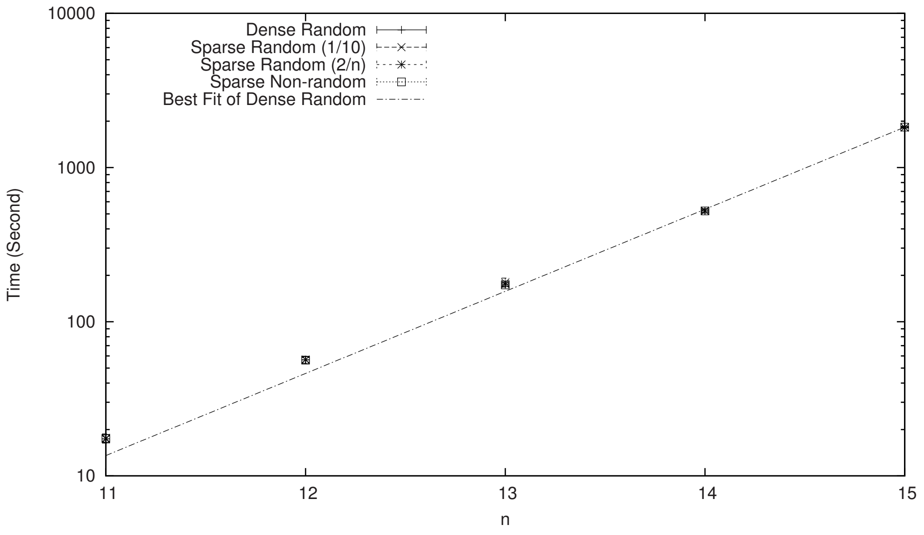

Fig. 1. “Sparsely determined random quadratics” in MAGMA

– We tried systems that have about n and $2n$ randomly picked non-zero non-linear terms in every equation (this is very sparse).
– Rotated sets (see Section 6.1) which had 3, 4, and 5 non-zero terms in every row (or every column, when we made a mistake in the program) of the transition matrices.

In every test using magma, the running time and memory used was close to each other which verifies the observation made by several people that the matrices become reasonably uniformly dense in $\mathbf{F_4}$ as we go up in degree.