Multivariates Polynomials for Hashing

Jintai Ding, Bo-yin Yang*
University of Cincinnati, Technical University of Darmstadt
*Academica Sinica**

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, whose security relies on simple hard questions. We analyze some security properties and potential feasibility, where the compression functions are randomly chosen high-degree polynomials. Next, we propose to improve on the efficiency of the system by using some specially designed polynomials using composition of maps and certain sparsity property, where the security of the system would then relies on stronger assumptions.

1 Introduction: the Dilemma

It is well-circulated joke that hash functions are the only common cryptological primitives that are getting slower as chips evolve. This is becoming rather pressing as of today, after the work of Wang et al culminated in the collision of some well-known and standard hash functions [7, 8]. 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. [6]

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 provable security. In these formulations, the designer seeks a reduction of the security of the compression function to some other intractable problem. In this paper, we propose another method, which is inspired by the $\mathcal{MQ}$ problem, and study how well it works.

The paper is arranged as follows. We first study the case of random systems. Then we study the case of composition construction and 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 the improvement of the computation efficiency of the systems.

2 The MQ frame work

Problem $\mathcal{MQ}$: Solve the polynomial system with coefficients and variables in $K = \mathrm{GF}(q)$

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

where each p_i is generic and quadratic in $\mathbf{x} = (x_1, \ldots, x_n)$.

$\mathcal{MQ}$ is NP-hard generically [5] and is the basis of multivariate public-key cryptosystems [4, 9]. If 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, we believe that the following holds.

Conjecture 1 *$\mathcal{MQ}$ is exponential in n unless $n/m = o(1)$ in the generic case.*

The latest way of tackling such methods involve guessing at some optimal number of variables (depending on the method in the closing stage) then use a Lazard-Faugère solver (XL or $\mathbf{F_5}$, [1]). 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 quadratic polynomials whose coefficients are

randomly chosen or 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 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 sovler. 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 [3] that applies to this situation where there is a lot many more variables than equations, but for fields other than $q = 2$ it has proved to be rather useless if we just plug in the numbers into the formulas.

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

Proposition 1 *With randomly chosen $F = F(\mathbf{x}, \mathbf{y})$, it is easy to find at least one solution for the equation.*

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

Proof. The equation above can be written as

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

which is a set of linear equations in $2n$ variables $(\mathbf{x},\mathbf{y})$ and n equations. If we randomly choose the values of $\mathbf{b},\mathbf{c}$, then in general, it should have none trivial solutions, because the number of variables is twice the number of equations; even if it does not, we can choose another set of random values of $\mathbf{b},\mathbf{c}$ and find a solution. □

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

3 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 heuristic argument implies that this compression function is secure.

Proposition 2 *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 second solution for in polynomial time.*

The first item is more or less $\mathcal{MP}$ hypothesis for cubics, the second one requires that

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

be impossible to solve in polynomial time, with F as above.

Proof. [Heuristic] First, let us consider that we are using a randomly generated set of equations, so it should be approximately equal opportunity or probability in collisions of any pair of points. In other words, in terms of the difference of the collision pairs, there should not be any difference among all the possibilities for the difference except the zero point. Thus, we can assume that the adversary knows the differential, or choose any differential at random. Consider now

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

This is a set of quadratic equations in $\mathbf{x}$ and $\mathbf{y}$. We can show that this is the only way to obtain quadratic equations associated with a collision by intersectiong our equations with an additional hyperplan of codimension n, otherwise it is impossible to eliminate all the cubic terms. But this differential construction is hard to solve, because we can show that as long as $\mathbf{b}$ and $\mathbf{c}$ are not both zero, $F^{(\mathbf{b},\mathbf{c})}$ is essential random using an argument on the Jacobian of F. Since $F^{(\mathbf{b},\mathbf{c})}$ still random and quadratic, and is hence hard to solve. □

This shows that the compressor and hence the hash function should be collision-free with proper chosen parameters. We propose some 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 2 *As n and $m = kn$ goes to infinity, for any fixed $0 < \epsilon < 1$, a random sparse quadratic system with a randomly picked ϵ proportion of the coefficients being non-zero (and still random) will still take time exponential in n to solve.*

The key problem is that the ratio ϵ of the nonzero terms.

1. How many we choose such that it is fast?

   Answer: no more than maybe 0.1% (see below).

2. How many we choose such that it is secure?

   Answer: a predetermined fixed percentage.

3. How do we choose the sparse terms?

   Answer: probably randomly.

4 Random Cubic Polynomial

We use completely random cubic polynomials, namely the one way function is gives 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. Suppose we have 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 quadratic $K^{n/2} \to K^n$.

According to [2], 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 function 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 function 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.

4.1 Sparse Polynomial

The idea is the same as above but we choose each of the above polynomial to be sparse.

Conjecture 3 *As n and $m = kn$ goes to infinity, for any fixed $0 < \epsilon < 1$, a random sparse nonlinear polynomial system with a randomly picked ϵ proportion of the coefficients being non-zero (and still random) will still take time exponential in n to solve.*

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 $\epsilon > 1/6$ 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

5 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 3 *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 4 *F is collision-resistant.*

Proof. [Heuristic] Frst, if we find a collision of F_1, we can find collision of F But it is easy to see that F_1 has no chance of collision once n is big enough. This can be seen using the differential to find the collision of F_1, where one gets a set of linear equation with $3n$ equations and $2n$ variables; and the probability for it to have a nontrivial solution is essentially zero if n is large enough; and even if it has a pair, it is hard to find because we need to know the right difference of the pair.

Thus, to find a collision on F means we need to find a collision on F_2, which poses no problem, but then we need to make sure that the pair produced by a collision on F_2 must have inverse under F_1, which has a essentially zero probability, further more we will have to invert F_1 twice, which is difficult by assumption.

Now, let us consider a direct differential attack and compute $F^{(\mathbf{b},\mathbf{c})}$. First, the differential is a set of degree 3 polynomial. Using the chain rule and the rule of expansion, we can see that the polynomial of the differential belong to the ideal generated the polynomials of F_1. Since for generic polynomials we should not see a reduction to zero under the degree of regularity [1], the solution is at as hard as inverting F_1. □

We would like to note that the using of expansion function like F_1 is also used in the construction of QUAD [2].

Another easy way to improve the security of the differential attack is to build a new function F, where

$$F = F_2 \circ \bar{F}_1,$$

where

$$\bar{F}_1 = (F_{11}^{-1}, F_{22}^{-1}, ..., F_{3n,3n}^{-1}),$$

and $F_1 = (F_{11}, F_{22}, ..., F_{3n,3n})$

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 type hash functions for practical applications. There are furthermore several new ideas that should be further studied.

5.1 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 linear map. Now we will choose

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 $Q = \sum_{1\le i\le j\le n} p_{ij}x_ix_j$, $p_{ij} \in F$.

Theorem 5 (A) *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 $2\nu + 2\tau + d \le n$.

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 (A) *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

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

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

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

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

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

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

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

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

Knowing that any quadratic functions from $\mathrm{GF}(q)^k \rightarrow \mathrm{GF}(q)$ can be written this way. The key question are similar now:

1. How do we choose L 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 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 $\mathbf{x}_1 \mapsto \mathbf{x}_2 := L_2\mathbf{x}_1$, 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 $\mathbf{x}_2 \mapsto \mathbf{x}_3 := L_3\mathbf{x}_2$, $f_3(\mathbf{x}) := f_1(\mathbf{x}_3)$, and so on and so forth.

A set of quadratic polynomials like this we call "rotated".

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

6 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 $\mathcal{MQ}$ is a known hard problem, which should lend a measure of confidence.

6.1 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, from what we can see, all but one of them is now inapplicable to attack our hash.

There are the usual attacks of linear and differential cryptanalysis to think of, but cubic and quartic equations are so far removed from linearity that it is hard to imagine such attacks working.

6.2 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 preimage resistance harder to show. Since the principle is not much different we don't go in that direction.

7 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 constructed 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, which is something we are pursuing next.

References

[1] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004. Previously appeared as INRIA report RR-5049.

[2] C. Berbain, H. Gilbert, and J. Patarin. QUAD: A practical stream cipher with provable security. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2006.

[3] N. Courtois, L. Goubin, W. Meier, and J.-D. Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography — PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 211–227. David Naccache and Pascal Paillier, editors, Springer, 2002.

[4] J. Ding, J. Gower, and D. Schmidt. *Multivariate Public-Key Cryptosystems*. Advances in Information Security. Springer, 2006. ISBN 0-387-32229-9.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979. ISBN 0-7167-1044-7 or 0-7167-1045-5.

[6] H. I. Makoto Sugita, Mitsuru Kawazoe. Gröbner basis based cryptanalysis of sha-1. Cryptology ePrint Archive, Report 2006/098, 2006. `http://eprint.iacr.org/`.

[7] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full sha-1. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Victor Shoup, editor, Springer, 2005. ISBN 3-540-28114-2.

[8] X. Wang and H. Yu. How to break md5 and other hash functions. In *Advances in Cryptology — EUROCRYPT 2005*, Lecture Notes in Computer Science, pages 19–35. Ronald Cramer, editor, Springer, 2005. ISBN 3-540-25910-4.

[9] C. Wolf and B. Preneel. Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 12$^{\text{th}}$ of May 2005. `http://eprint.iacr.org/2005/077/`, 64 pages.