On Building Hash Functions from Multivariate Quadratic Equations[*]

Olivier Billet, Matt J.B. Robshaw, and Thomas Peyrin

France Telecom R&D, Issy-les-Moulineaux, France
{forename.name}@orange-ftgroup.com

Abstract. Recent advances in hash functions cryptanalysis provide a strong impetus to explore new designs. This paper describes a new hash function MQ-HASH that depends for its security on the difficulty of solving randomly drawn systems of multivariate equations over a finite field. While provably achieving pre-image resistance for a hash function based on multivariate equations is relatively easy, naïve constructions using multivariate equations are susceptible to collision attacks. In this paper, therefore, we describe a mechanism—also using multivariate quadratic polynomials—yielding the collision-free property we seek while retaining provable pre-image resistance. Therefore, MQ-HASH offers an intriguing companion proposal to the provably collision-free hash function VSH.

1 Introduction

Cryptographic hash functions are essential components within the information security infrastructure. A cryptographic hash function HASH($\cdot$) is a function that takes an arbitrary length input and generates a fixed length output of n bits. Classically, there are three main properties of such functions which can be loosely described in the following way [19]:

1. *Pre-image resistance.* Given an output y it is computationally hard to find any input x such that HASH(x) = y;
2. *Second pre-image resistance.* Given an input and output pair (x, y) so that HASH(x) = y, it is computationally hard to find an input x' distinct from x such that HASH(x') = y;
3. *Collision resistance.* It is computationally hard to find any two inputs x and x' such that HASH(x) = HASH(x').

While there have been a variety of different hash function proposals over the years, most currently deployed hash functions are closely built around design principles which go back to MD4 [28]. Probably the most popular hash functions in use today are MD5 [29] and SHA-1 [22]. However recent cryptanalytic advances [32,33] have shown weaknesses that allow collisions to be computed

---

for these hash functions much faster than by brute force. A more recent family of hash functions [22] has been standardised by NIST in the U.S., but these are closely related to SHA-1 and confidence in their construction is somewhat undermined by recent cryptanalytic work on MD5 and SHA-1. An alternative approach has been to build hash functions around a secure block cipher, see for example [19,27] and more recently [26], and the widespread deployment of the AES [21] may well provide new opportunities in this direction.

In this paper we consider a third approach and we base the security of a hash function on a hard mathematical problem. Until recently this approach was very limited, but there has been considerable success with VSH [8] which relates the difficulty of finding collisions to a hard problem built around the mechanics of current factoring techniques. A variant based on the discrete logarithm problem VSH-DL has also been proposed [17] and other proposals include the FSB hash function [2] that is related to fast syndrome-based decoding problems and LASH [6] which is based on problems in lattice theory.

The new hash function in this paper, MQ-HASH, is built around the problem of solving a system of multivariate quadratic equations. This is the same problem that underpins the security of the stream cipher QUAD [4] and we will find it useful to appeal to some of the results that feature in that design.

Designing a pre-image resistant compression function from a set of multivariate quadratic polynomials without a care for collision-resistance is quite easy, because it is enough to rely on the hard problem MQ. However, such naïve constructions give rise to collision-full hash functions. So the difficult part in building a hash function based on sets of multivariate quadratic polynomials is in providing collision-resistance in a plausible way, but without sacrificing any pre-image resistance or its proof. This is the purpose of the current paper.

Our paper is structured as follows. In the next section we provide some background on hash function design and the use of systems of quadratic polynomials. We illustrate the difficulty of using quadratic polynomials directly by demonstrating an intrinsic weakness. In Section 3 we describe our construction MQ-HASH while we prove its structural security in Section 4 and its instantiated security with explicit parameter values in Section 5. The performance of our proposal and some variants are considered in Section 5.3 and we close by highlighting open problems and drawing our conclusions in Section 6.

2   Hash Functions and Quadratic Equations

In this paper we consider the problem of building a hash function from one particular hard problem, namely that of solving a system of multivariate equations over a finite field **F**. The natural one-wayness of this primitive, together with its computational efficiency, provides an interesting starting point for a new hash function proposal.

While evaluating a random set of m multivariate polynomials in n variables is of polynomial complexity with respect to n, finding a common root of this set

of polynomials is well known to be an NP-hard problem. This remains true even when restricted to quadratic polynomials or to the case of two equations [12].

The problem of solving multivariate quadratic equations over a finite field $\mathbf{F}$ is known as the MQ-problem. It is a hard problem, but also one that permits efficient schemes. Consequently it has been used in the design of several cryptographic applications. See [34] for an overview along with some additional information that can be found in [4].

A tuple of multivariate quadratic polynomials consists of a finite ordered set of polynomials of the form:

$$q(x_1, \ldots, x_n) = \sum_{1 \le i \le j \le n} a_{i,j} \, x_i x_j + \sum_{1 \le k \le n} b_k \, x_k + c \; ,$$

where the constants are in a finite field $\mathbf{F}$. The MQ problem can then be stated as follows:

Problem 1 (MQ). Given a tuple $q = (q_1, \ldots, q_m)$ of m multivariate quadratic polynomials in n unknowns defined over $\mathbf{F}$, and the image $y = (q_1(z), \ldots, q_m(z))$ of an element z randomly chosen from $\mathbf{F}^n$ through q, find an element x of $\mathbf{F}^n$ such that $y = (q_1(x), \ldots, q_m(x))$.

Solving a set of randomly chosen quadratic equations in several variables over a finite field is a well-known hard problem [13]. That it is conjectured to be very difficult not only asymptotically and in worst case, but already for well chosen practical values of m and n makes it very attractive as a cryptographic building block. Apart from degenerate parameters like $n \ll m$ or $n \gg m$ (see [31]) or low rank polynomials, the complexity of the best known algorithms for solving the problem are exponential in $\min(m, n)$, (see [3,31]).

This leads to the following naïve construction for a compression function based on the evaluation of multivariate quadratic polynomials:

Attempt 1 (naïve and flawed). Let $\mathbf{F}$ be a finite field of size q and, assume that we wish to compress a fixed-length input of $\rho = r \log_2 q$ bits to give an output of $\nu = n \log_2 q$ bits. A compression function g can be obtained by randomly choosing a tuple $(g_1, \ldots, g_n)$ of n quadratic polynomials in r variables defined over $\mathbf{F}$:

$$\mathbf{F}^r \longrightarrow \mathbf{F}^n$$
$$x = (x_1, \ldots, x_\rho) \longmapsto g(x) = \big(g_1(x), \ldots, g_\nu(x)\big) \; .$$

While the one-way property of Attempt 1 is straightforward to establish, it is very easy to find collision and it would not be, in itself, an acceptable way to build a cryptographic hash function. In the next section, therefore, we investigate more closely the problem of collision resistance in the setting of multivariate quadratic polynomials.

2.1   About Collision Resistance

Unfortunately there is no collision-resistance when using a system of quadratic equations directly and it is hard to achieve this property in a simple way for

the following reason. For polynomial equations of degree d, any differential of order $d-1$ is an affine application. Thus, in the special case of sets of quadratic polynomials, this amounts to saying that the set of first order differentials of any quadratic polynomial in the original set is a set of affine mappings. This simple fact has previously been used for instance by Fouque, Granboulan, and Stern to attack an asymmetric multivariate scheme [11].

Theorem 1. *Let Q be a tuple of e quadratic equations $f_1, \ldots, f_e$ in u variables over a finite field $\mathbf{F}$. For every value $\delta = (\delta_1, \ldots, \delta_u)$, it is possible to give, with time complexity $O(eu^2)$, a parametrized description of the set of inputs $x = (x_1, \ldots, x_u)$ and $y = (y_1, \ldots, y_u)$ colliding through Q and such that $y-x = \delta$, if any.*

Proof. Given δ, one computes a linear system $L_\delta(z) = 0$ in the indeterminate z where L_δ is the affine mapping defined by $L_\delta : z \mapsto Q(z + \delta) - Q(z)$. Thus, any colliding pair $(x, y) = (x, x+\delta)$ for Q with prescribed difference δ translates into a solution x of a linear system, and any standard algorithm for solving linear system recovers the set of solutions of the collision equation $Q(z) = Q(z+\delta)$. □

Theorem 1 thus basically implies that collisions can be easily constructed for any naïve design like the one described in Attempt 1. Further, a hash function design facilitating the analysis of differences might be subject to attack. It is therefore reasonable to ask whether there is any way to plausibly achieve collision resistance when using sets of multivariate quadratic polynomials, and yet to provably retain the original one wayness property? The following sections answer this question positively.

3   Construction of MQ-HASH

We now present one particular approach to using multivariate quadratic polynomials in the design of a hash function. While we do so with general parameter sets, we propose some concrete values in Section 5.3.

3.1   Preliminaries

While recent analysis [14,15] has provided new insight into the Merkle-Damgård paradigm [9,20], our goal has been to design a secure compression function for use in this familiar way. The Merkle-Damgård construction requires the specification of an $(\mu+\nu)$-bit to ν-bit compression function COMPRESS. The compression function will be used repeatedly to hash the input message M in a component-wise manner. We denote the block of a message being hashed at the i-th iteration by B_i, where each block is of constant length μ bits. Clearly this requires padding and the standard Merkle-Damgård or MD-hardening is assumed. Thus we append a single bit '1' followed by as many '0' as required to leave the message

64 bits short of a multiple of μ. The remaining 64 bits are then used for a representation of the length of the input message M in bits[1]. We will assume that the message M requires t blocks after padding and so $M = B_1||\cdots||B_t$.

3.2   The Compression Function of MQ-HASH

At iteration i, for $1 \leq i \leq t$, the compression function is used to update the value v_{i-1} of an ν-bit *chaining variable* to v_i. The initial value of the chaining variable v_0 is specified and fixed. Thus, at iteration i of the compression function we have that $v_i = \text{COMPRESS}(v_{i-1}, B_i)$. At the end of the iteration process, the last chaining variable is used as the output of the hash function.

Figure 1 shows the compression function MQ-HASH. It uses two non-invertible components with the first component f providing a stretching function while the second, g, provides a shrinking function. These are embodied by randomly chosen tuples of multivariate quadratic polynomials. Thus, in the process of proving the necessary security properties, we have a construction that shares features with the work of Aiello, Haber, and Venkatesan [1].
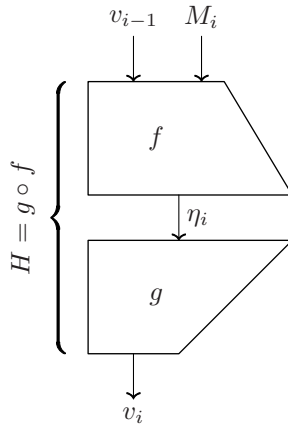

Fig. 1. Schematic description of the compression function of MQ-HASH, where v_i denotes the chaining variable and B_i the message block being hashed at iteration i

The compression function of MQ-HASH takes as input a message block B_i of μ bits and a chaining variable v_{i-1} of ν bits. Let $\mathbf{F}$ be a finite field of size q so that μ and ν are multiples of $\log_2 q$, say $\mu = m \log_2 q$ and $\nu = n \log_2 q$. We also fix another integer $\rho = r \log_2 q$ so that $r \geq m + n$. Then the stretching function consists of a randomly chosen tuple $(f_1, \ldots, f_r)$ of r quadratic polynomials in $n + m$ variables defined over $\mathbf{F}$. That is, f is given by:

---

[1] Thus the maximum length of a message that can be hashed using MQ-HASH is 2^{64} as for many other hash functions [29,22].

$$\mathbf{F}^{n+m} \longrightarrow \mathbf{F}^r$$
$$x = (c_1, \ldots, c_n, b_1, \ldots, b_m) \longmapsto f(x) = \big(f_1(x), \ldots, f_r(x)\big) \;,$$

where $(b_1, \ldots, b_m)$ stands for the μ-bit message block split into m elements of $\mathbf{F}$, and $(c_1, \ldots, c_n)$ stands for the ν-bit chaining variable split into n elements of $\mathbf{F}$. The shrinking stage is, in turn, defined by a randomly chosen tuple $(g_1, \ldots, g_n)$ of n quadratic polynomials in r variables. That is, g is given by:

$$\mathbf{F}^r \longrightarrow \mathbf{F}^n$$
$$\eta = (\eta_1, \ldots, \eta_r) \longmapsto g(\eta) = \big(g_1(\eta), \ldots, g_n(\eta)\big) \;.$$

The final compression function is then defined to be the composition of f and g and $v_i = g \circ f(v_{i-1}, B_i)$. For the construction to be secure, we will show that:

- the two functions f and g must be hard to invert;
- the *stretch factor* $\frac{r}{m+n}$ in the first step must lie within a certain range.

The value of the stretch factor will be discussed in Section 5 and it will depend on the number of bits hashed at each compression function iteration as well as the length of the chaining variable.

In order to ease the exposition, we will assume for the rest of the paper that the ground field $\mathbf{F}$ is the binary field GF(2) and it follows that $m = \mu$, $n = \nu$, and $r = \rho$.

4 The Security of MQ-HASH

The work of Merkle and Damgård allows us to concentrate on the properties of the compression function $g \circ f$. We first give elements of provable security for the first pre-image resistance of MQ-HASH. Then we discuss the collision resistance of MQ-HASH.

4.1 Preliminaries to the Study of Pre-image Resistance

In what follows, U_k denotes the uniform distribution over $\{0,1\}^k$. We say that two distributions X and Y over the binary strings of size k are distinguishable in time T with advantage ε if there exists a probabilistic algorithm D running in time less than T such that:

$$\left| \Pr_{x \in X}\big[D(x) = 1\big] - \Pr_{y \in Y}\big[D(y) = 1\big] \right| \geq \varepsilon \;.$$

We describe a pseudo-random number generator as a deterministic polynomial-time algorithm G from $\{0,1\}^l$ to $\{0,1\}^k$ with $k > l$ such that $G(U_l)$ cannot be distinguished from U_k in reasonable time (for instance with a time complexity lower than 2^s for some security level s) and with a non-negligible advantage.

We say that a function g is non-invertible in time T with probability ε if for any probabilistic algorithm $\mathcal{B}$ running in time less than T:

$$\left| \Pr_{z \in U_r} \left[g\Big(\mathcal{B}\big(g(z)\big)\Big) = g(z) \right] \right| < \varepsilon \ .$$

An important aspect to our proofs will be the fact that a tuple of multivariate quadratic equations with a small stretching factor is in effect acting as a pseudo-random number generator. This ensures that the outputs from the stretching function f does not have noticeable specific properties. Since this is a property that underpins the design of QUAD, it is not surprising to find some of the fundamental components for our work covered in [4].

Theorem 2. *Let $\mathcal{A}$ be an algorithm that, on input a randomly chosen tuple f of r multivariate quadratic equations in $n + m$ binary unknowns distinguishes the distribution $\{f_1(x)|| \cdots ||f_r(x)\}_{x \in U_{n+m}}$ over the binary strings of length r from the uniform distribution U_r in time T and with advantage ε. Then $\mathcal{A}$ can be converted into an algorithm $\mathcal{B}$ that inverts a tuple g of r randomly chosen multivariate quadratic equations in $n + m$ binary unknowns with probability $\varepsilon/2$ (over both g and the inputs) in time less than:*

$$\tilde{T}(T, n, m, \varepsilon) = \frac{128(n + m)^2}{\varepsilon^2} \left(T + \log\left(\frac{128(n + m)}{\varepsilon^2} \right) + r(n + m) + 2 \right) \ .$$

Proof. The proof is a direct application of Theorems 2 and 3 from [4].   □

The above theorem gives rise to two comments. First, the choice of the base field is GF(2). However no obstacles to generalisations over other fields are anticipated, though the reduction would obviously lead to another value of $\tilde{T}$. Second, the reduction achieved by Theorem 2 is not very tight. However this is enough for us to derive secure parameters in Section 5.

4.2   Pre-image Resistance of MQ-HASH

The next theorem reduces the pre-image resistance of MQ-HASH's compression function to the problem of inverting random multivariate quadratic systems. Let T_f (resp. g) be the time required to evaluate f (resp. g) on its input.

Theorem 3. *Let $\mathcal{A}$ be an algorithm inverting $g \circ f$ in time T with probability ε, where f is a randomly chosen tuple of r multivariate quadratic polynomials in $n + m$ binary unknowns and g is a randomly chosen tuple of m multivariate quadratic polynomials in n binary unknowns. Then $\mathcal{A}$ can be either converted into an algorithm inverting g in time $T + T_f + T_g$ with probability ε or into an algorithm that can invert randomly chosen tuples of r multivariate quadratic polynomials in $n + m$ binary unknowns in time $\tilde{T}(T + T_f + 3T_g, n, m, \varepsilon)$ with probability $\varepsilon/2$.*

Proof. Let us define $\tilde{\mathcal{A}}(x) = f\big(\mathcal{A}(g(x))\big)$. By the assumption on algorithm $\mathcal{A}$

$$\left| \Pr_{x \in U_{n+m}} \big[g \circ f \left(\mathcal{A}\big(g \circ f(x) \big) \right) = g \circ f(x) \big] \right| \geq \varepsilon \ .$$

Thus g can be inverted by $\tilde{\mathcal{A}}$ in time $T + T_f + T_g$ with probability ε when queried with the distribution $f(U_{n+m})$. So either g can be inverted in time $T + T_f + T_g$ with probability ε or $f(U_{n+m})$ can be distinguished from U_r in time $T + T_f + 3T_g$. The theorem then follows from a direct application of Theorem 2. $\qquad\square$

Thus, assuming that g and f are hard to invert and that f is a pseudo-random number generator, we deduce that their composition $g \circ f$, that is MQ-HASH's compression function, is pre-image resistant.

4.3   Collision and Second Pre-image Resistance of MQ-HASH

There are two sets of multivariate quadratic polynomials corresponding to functions f and g and it is their composition that gives the compression function in MQ-HASH. Intuitively, the function g provides the actual compression. However, Theorem 1 demonstrated the potential ease of finding collisions when using g on its own. So in a first step we use a non-invertible function f. This ensures that lifting collisions in g to yield pre-images for f is hard. However, finding collisions for f must not be easy, or even better, f must be an injection. This will be the rational behind what we term the *stretch* requirement for f.

The construction used in MQ-HASH is a close analogue to the construction of Aiello, Haber, and Venkatesan [1]. Their claims for collision-resistance apply equally to our own construction. Consider the compression function $g \circ f$. We know that g has collisions since it compresses but there can be no collisions in f if f is an injection. For any collisions across g to be useful for the entire compression function, they must (a) lie in the range of f and (b) be invertible through f. The choice of stretch factor ensures that (a) is unlikely while the choice of hard problem prevents (b).

Of course, this is not a proof, and a proof for the collision resistance in the standard model remains an open problem. Nevertheless, it is possible to prove this conjecture in the *random oracle model*, which, while less appealing than the standard model, provides some evidence that the overall construction is not completely flawed. It is interesting to note that this is one difference between MQ-HASH and VSH. While both proposals are able to provide a classical hash function property in a provable manner, the remaining classical properties are still conjectured to hold for MQ-HASH.

5   Establishing Parameters for MQ-HASH

While several elements of provable security for MQ-HASH were given in Section 4, the limitations of such proofs are exposed when we instantiate the general constructions in practice. In this section, therefore, we study the security and performance of MQ-HASH and illustrate the different trade-offs possible.

Our proofs in Section 4 required that f be an injection; this was the basis for the stretching role of f. But our construction also requires that solving a random system of multivariate quadratic equations is a hard problem. Thus, we observe that there are two conflicting practical constraints:

- A sufficiently large stretch is needed to ensure (to a degree of certainty that is consistent with the intended security level) that there are no collisions in the first part of the compression function.
- The system of equations that results, which will have more equations than variables, must remain computationally non-invertible.

5.1   On the Injectivity of f

The following theorem provides a bound on the stretch factor needed for the first stage f, embodied by a system of quadratic equations, to ensure its injectiveness.

Proposition 1. *The probability that a tuple f of e randomly chosen quadratic polynomials in u unknowns over a finite field $\mathbf{F}$ of size q, with $e > u$, is not an injection is lower than q^{2u-e}.*

Proof. The linear part of the affine application $A_\delta(z) = f(z + \delta) - f(z)$ is a matrix of size $e \times u$ and is defined over a finite field $\mathbf{F}$ of size q. So A_s is of rank less than u. But the probability that any matrix of size $e \times u$ and of rank u has a uniformly randomly chosen element in its image is less than q^{u-e}. Writing the tuple f as $f = f^{(2)} + f^{(1)} + f^{(0)}$ where $f^{(i)}$ denotes the homogeneous part of degree i, we see that for a randomly drawn value δ the constant $f^{(2)}(\delta) + f^{(1)}(\delta)$ is uniformly randomly distributed in $\mathbf{F}^e$, independently of the coefficients of $f^{(2)}$. Expanding the expression of A_δ as $A_\delta(z) = \beta_{f^{(2)}}(\delta, z) + f^{(2)}(\delta) + f^{(1)}(\delta)$ where $\beta_{f^{(2)}}$ is the bilinear form associated to f, one see that:

$$\Pr_{\delta \in U_u}\big[\mathrm{Ker}(A_\delta) \neq \{0\}\big] = \Pr_{\delta \in U_u, c \in U_e}\big[c \in \mathrm{Im}(A_\delta)\big] \leq q^{u-e} \ .$$

The corresponding tuple f thus has less than q^{u-e} chances of providing a collision pair of the form $(x, x+\delta)$ for any randomly chosen δ. Running through all possible values for δ, we have that the probability of f being an injection is greater than $(1 - q^{u-e})^{q^u}$ and thus the probability of f not being an injection is lower than q^{2u-e}.   □

Interpreting this result and assuming that we seek a security level s, we have the constraint $2^{2u-e} < 2^{-s}$, or $e > 2u + s$ over the binary field as ground field $\mathbf{F}$. Hence, our construction will asymptotically show a stretch factor of two in the case of the binary field.

5.2   On the Hardness of Inverting f

The hardness of the system solving problem is closely related to the ratio between the number of equations and the number of variables. So we need to study the

complexity of solving randomly generated quadratic equation systems over the field GF(2) when there are more equations than variables. This has been studied in detail [3] and we summarize the results in our very special case.

Theorem 4. *Solving a random system of e quadratic equations in u unknowns over the field GF(2) by the best Gröbner basis algorithm requires $\binom{u}{d}^{\omega}$ operations where $\omega \approx 2.3$ and*

$$d = \frac{u}{2} - e + \frac{e}{2}\sqrt{2 - \left(\frac{u}{e}\right)^2 - 10\frac{u}{e} + 2\sqrt{8\left(\frac{u}{e}\right)^3 + 12\left(\frac{u}{e}\right)^2 + 6\frac{u}{e} + 1}} \ .$$

Proof. The proof is available in [3].   □

Since we expect to use a stretch factor slightly bigger than two for our construction, the complexity of solving with the best Gröbner basis methods will be about $\binom{u}{u/20}^{\omega}$. For the values proposed in Section 5.3 this complexity is much higher than the security level.

# variables	80	128	160	256	512
time complexity	2^{47}	2^{74}	2^{99}	2^{153}	2^{323}

5.3   Performance Considerations

In this section, we investigate how the security requirements impact the performances of MQ-HASH. For conservative settings and aiming at 80-bit security, the use of the base field $\mathbf{F} = GF(2)$ seems mandatory. In this case, the chaining variable could be 160 bits in length, the message block at each iteration could be 32 bits in length, and the intermediate output from f should be around 464 bits. This would leave us with the parameter set $n = \nu = 160$, $m = \mu = 32$, and $r = \rho = 464$ which are consistent with the security levels implied by Theorem 4. The storage requirements for the first part of the computation, the evaluation of f, is about 1 MB while the storage for the evaluation of g is less than 2.2 MB. The total amount of storage is more than 3 MB of memory, so it will not fit in the cache of contemporary processors, incurring a big performance penalty that will severely restrict its practical use.

As usual, the property of provable security comes at a price. Crude estimates for the performance of MQ-HASH show that it might be expected to run thousands of times slower than SHA-1. However, we foresee that various modifications can be made to the design of MQ-HASH so as to lower the gap of performance with usual hash functions like SHA-1. We leave this question as an open research subject.

5.4   Deploying Random Systems

One issue with using quadratic systems might be a concern about weak instances. This is of special interest in the case of multivariate quadratic systems since

trapdoors for this environment have been proposed as a fundamental feature of several asymmetric schemes [16,23,24,25]. However, this is not such an unusual issue in cryptographic deployment and shared equation systems can be generated using a variety of techniques so as to allay suspicion. See, for example, the case of DSS [30].

5.5   Alternative Approaches

Our proposal MQ-HASH might be viewed as a first attempt to build a practical hash function that relies for its security on multivariate quadratic equations. There are several ways the work might be extended.

For instance, we might consider some slight variants to the structure of MQ-HASH. It would be very natural to replace the *fixed* tuple f of multivariate polynomials with tuples that are randomly re-generated at each iteration of the compression function. Such a variant, outlined below, allows the tuple f to be modified via some transformation of the chaining variable.
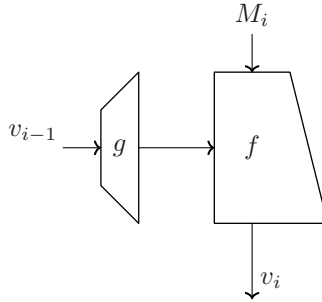
It is interesting to observe that this approach, that we denote RMQ-HASH, can be viewed as bringing us closer to some established block-cipher constructions such as *Matyas-Meyer-Oseas* [7,27]: the one-way function f would be akin to a block cipher with feed-forward and the treatment of the chaining variable would be analogous to a (very) unusual key-schedule. While there are some intriguing challenges in this approach, early analysis suggests that such a scheme would allow for a more compact system of equations with accompanying performance advantages. This may well be an interesting structure to consider in future work.

6   Conclusions

In this paper we have introduced a new hash function MQ-HASH. The security of this hash function is based on the difficulty of solving systems of multivariate quadratic equations, a problem that is well-studied and used elsewhere in

cryptography. The hash function MQ-HASH is provably pre-image resistant in the standard model, and there is good evidence to support the conjecture that MQ-HASH is collision-free and second pre-image resistant. However a proof in the standard model remains an area of open research.

We believe there to be considerable promise in using multivariate quadratic equations as a hard problem in symmetric cryptography. This is something that has been pioneered with QUAD, and we anticipate similar success in the design of other primitives. With regards to hash functions, however, there are some particular challenges in using multivariate quadratic equations. In particular one is forced to adopt a more complex construction than one might initially like, and one must act carefully so as retain provable pre-image resistance. This may well result in a wide variety of alternative constructions. In this paper we have considered one particular approach and establishing a broader range of designs with alternative security/performance trade-offs remains a topic of ongoing research.

References

1. Aiello, W., Haber, S., Venkatesan, R.: New Constructions for Secure Hash Functions. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 150–167. Springer, Heidelberg (1998)
2. Augot, D., Finiasz, M., Sendrier, N.: A Family of Fast Syndrome Based Cryptographic Hash Functions. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 64–83. Springer, Heidelberg (2005)
3. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: ICPSS, pp. 71–74 (2004)
4. 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)
5. Berbain, C.: Personal communication (November 21, 2006)
6. Bentahar, K., Page, D., Silverman, J.H., Saarinen, M.-J.O., Smart, N., LASH (2006) Available from: `http://csrc.nist.gov/pki/HashWorkshop/2006/`
7. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
8. Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an Efficient and Provable Collision-Resistant Hash Function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)
9. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
10. Ding, J., Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
11. Fouque, P.-A., Granboulan, L., Stern, J.: Differential cryptanalysis for multivariate schemes. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)

12. Fraenkel, A.S., Yesha, Y.: Complexity of Problems in Games, Graphs, and Algebraic Equations. Discr. Appl. Math. 1, 15–30 (1979)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co. New York (1979)
14. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M.k. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
15. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
16. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
17. Lenstra, A.K., Page, D., Stam, M.: Discrete logarithm variants of VSH. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 229–242. Springer, Heidelberg (2006)
18. Lidl, R., Niederreiter, H.: Finite Fields. Cambridge University Press, Cambridge (1997)
19. Menezes, A.J., Vanstone, S.A., Van Oorschot, P.C.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
20. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1989)
21. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard (November 2001) Available from: `http://csrc.nist.gov`
22. National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard (August 2002) `http://csrc.nist.gov`
23. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
24. Patarin, J., Courtois, N.T., Goubin, L.: QUARTZ, 128-Bit Long Digital Signatures. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 282–297. Springer, Heidelberg (2001)
25. Patarin, J., Courtois, N.T., Goubin, L.: FLASH, a Fast Multivariate Signature Algorithm. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 298–307. Springer, Heidelberg (2001)
26. Peyrin, T., Gilbert, H., Muller, F., Robshaw, M.J.B.: Combining Compression Functions and Block Cipher-based Hash Functions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 315–331. Springer, Heidelberg (2006)
27. Preneel, B.: Analysis and design of cryptographic hash functions. Ph.D. thesis. Katholieke Universiteit Leuven (1993)
28. Ronald, L.: Rivest. RFC 1320: The MD4 Message-Digest Algorithm (April 1992)`http://www.ietf.org/rfc/rfc1320.txt`
29. Rivest, R.L.: RFC 1321: The MD5 Message-Digest Algorithm (April 1992) `http://www.ietf.org/rfc/rfc1321.txt`
30. Smid, M.E., Branstad, D.K.: Response to Comments of the NIST Proposed Digital Signature Standard. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 76–88. Springer, Heidelberg (1993)

31. Courtois, N., Goubin, L., Meier, W., Tacier, J.-D.: Solving Underdefined Systems of Multivariate Quadratic Equations. Public Key Cryptography, 211–227 (2002)
32. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Ziarko, W., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 17–36. Springer, Heidelberg (2001)
33. 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)
34. Wolf, C., Preneel, B.: Taxonomy of Public Key Schemes based on the problem of Multivariate Quadratic equations. `http://eprint.iacr.org/`