On Building Hash Functions From Multivariate Quadratic Equations

Olivier Billet, Thomas Peyrin, and Matt Robshaw

Orange Labs
France

02.07.07

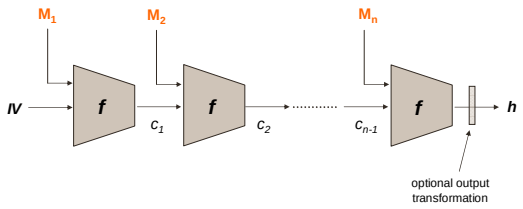
Overview

- Hash Functions
- Multivariate quadratic equations
- Hash functions and multivariate quadratic equations
- Pro's and con's
- Conclusions

Hash Functions

- We want a fixed-length output from an arbitrary length input
- Classically, good hash functions satisfy three properties
  - Pre-image resistant
  - Second pre-image resistant
  - Collision-free
  - However, it is not always clear what we want or what we need
- Typical designs are built around a compression function
  - These compress a fixed-length string
  - Multiple calls to the compression function allow inputs of (close to) arbitrary length to be hashed (Merkle-Damgård)

Compression Functions



Compression Functions (I)

- Typically built around a block cipher
- Sometimes it's a block cipher of dedicated design
  - *e.g.* MD4, MD5, SHA, SHA-1, etc.
  - The underlying construct is an (unusual) block cipher
- Sometimes it's an established block cipher (DES or AES)
  - *e.g.* MDC-2, MDC-4

Compression Functions (II)

- There is much interest in number-theoretic approaches
  - Primarily due to the success of VSH
  - Other examples include LASH, FSB, …
- Here we try and get good (or reasonable) performance coupled with an element of "provable security"

In This Paper

- We consider efforts to build a compression function based on *Multivariate Quadratic Equations (MQE)*
- Can we get some "provable" security with reasonable performance ?

Multivariate Quadratic Equations

- Solving a random system of multivariate quadratic equations over a field **F** is (in general) difficult

$$q_1(x_1, \ldots, x_n) = \sum_{1 \le i \le j \le n} a_{ij} x_i x_j + \sum_{1 \le k \le n} b_k x_k + c$$
$$q_2(x_1, \ldots, x_n) = \ldots$$
$$\downarrow$$
$$q_m(x_1, \ldots, x_n) = \ldots$$

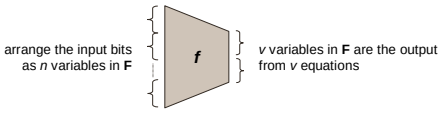
Given $y_1, \ldots, y_m$ find some $x_1, \ldots, x_n$ such that $y_1 = q_1(x_1, \ldots, x_n), \ldots, y_m = q_m(x_1, \ldots, x_n)$

Multivariate Quadratic Equations

- However, evaluating a set of polynomials is very easy
  - There is a very appealing natural *one-way* quality
- There has been mixed success using this in public key cryptography
  - We need to embed a trapdoor which is not always easy
- But some success in symmetric cryptography (QUAD)

Starting Out

- It is natural to try and build a hash function from MQE
  - We get one-way properties for free



compress: $\mathbf{F}^n \rightarrow \mathbf{F}^v$

compress$(x_1, \ldots, x_n) = (q_1(x_1, \ldots, x_n), \ldots, q_v(x_1, \ldots, x_n))$

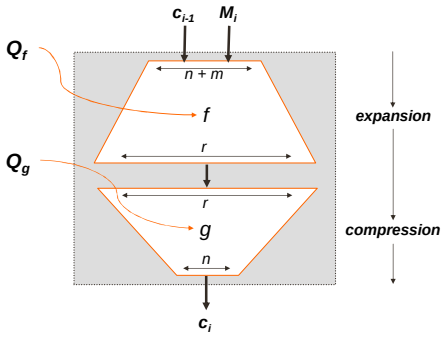
Pre-image Resistant, but …

- If there are collisions they will be easy to find
  - First order differential of quadratic polynomials is affine
- Our challenge is to find a different way of using MQE
  - Provably maintain pre-image resistance property
  - Provide (at least) plausible collision-free property

A Two-Step Approach

- We build a two-step compression function **MQ-*hash***
  - Use MQE in both steps
- Use MQE to give some "compression" but apply some pre-processing
  - Pre-processing appears in several guises, but our work is somewhat related to Aiello, Haber, and Venketasen (FSE 1998)
  - **Intuition**: Collisions might be obvious in the second component but they hard to extend to the full compression function

MQ-*hash* = $Q_g \cdot Q_f$



Outline of Reasoning

- For MQ-*hash* to be one-way
  - Q_g is one-way (this is our starting point)
  - The MQE in Q_f are "well-behaved"
    - We borrow from QUAD for this
- For MQ-*hash* to be (plausibly) collision-free
  - Collisions in Q_g cannot be *lifted* to the compression function
    - Q_f should be one-way
  - Q_f should not induce collisions
    - Q_f *stretches* the input

Parameters and Performance

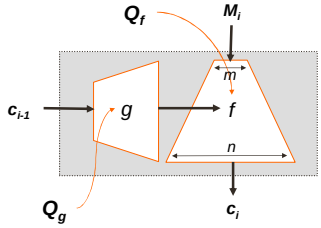
- Q_f consists of ***r*** equations in ***n+m*** variables
- Q_g consists of ***n*** equations in ***r*** variables
- Suppose we seek a security level of 2^k operations
  - We require that $n \geq 2k$
  - We can bound the probability that Q_f is not an injection
  - We require that $r \approx 2(n+m) + k$
  - At each iteration we hash ***m*** bits of message
  - For GF(2) we might chose $n = 160$, $m = 32$, and $r = 464$
- … but the performance is (very) poor

The MQ-*hash* Proposal

- Pro's:
  - Provably pre-image resistant construction
  - Conjectured collision-free and second pre-image resistant
- Con's:
  - Terrible performance (storage and hashing rate)

An Alternative Construction

- However MQE are very versatile building blocks
  - Perhaps this construction could be of some interest



Conclusions

- We have explored the use of multivariate quadratic equations in designing a hash function
  - We (successfully) tackled some intricate issues
  - Gained additional insight into using MQE
- However, our feeling is that this isn't the way to go
  - New research might uncover better ways of using ME
  - But we doubt random MQE systems are a practical building block for a hash function