Multivariate hash functions:
constructions and security

Jean-Philippe Aumasson

University of Applied Sciences Northwestern Switzerland
School of Engineering

Lightweight cryptology (Introduction)

Jean-Philippe Aumasson

University of Applied Sciences Northwestern Switzerland
School of Engineering

WHAT IS IT ?

Lightweight $\equiv$
Dedicated to environments (HW or SW) with **limited resources**, be it in

- power/energy
- size (e.g. code, gate count, storage)
- communication bandwith
- time (throughput)
- physical protection

WHAT IS IT ?

As in conventional cryptography, covers

- **primitives** (e.g. stream ciphers)
- **modes** of operations (e.g. authentication)
- **protocols** (e.g. group key agreement, secret sharing, broadcast encryption)

WHAT IS IT ?

Applies to heterogeneous wireless networks, sensors arrays, smartcards, etc., and includes items as

WHY SHOULD WE CARE ?

Economics:

- growing market for RFID, non-desktop applications, ubicomp
- $>$ 95% of CPU's are embedded
- variety of wireless networks (WLAN, GSM, PCS, etc.)
- many applications in defense and space industry

WHY SHOULD WE CARE ?

Research interest: new adversarial models and scenarios, *cf.* constraints as

- devices that might be stolen by an adversary
- data processing not necessarily protected
- devices with a very short lifetime
- unreliable network connectivity
- dynamic changes in topology
- multitude of side channels

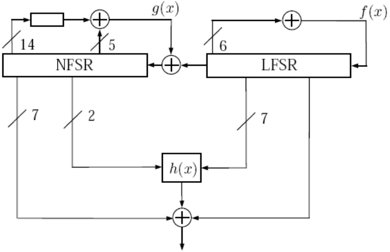
+ cryptanalysis of existing schemes.

HOW TO BUILD EFFICIENT PRIMITIVES ?

For hardware ciphers:

- avoid complex arithmetic (e.g. integer multiplication, exponentiation)
- use simple bitwise components (e.g. shift-registers, boolean functions)
- reduce wiring, internal state size.

Example: the stream cipher **Grain** [Hell-Johansson-Meier 05]



Alternative for fast hardware crypto: **multivariate** schemes.

Multivariate hash functions:
constructions and security

Jean-Philippe Aumasson

n|w University of Applied Sciences Northwestern Switzerland
School of Engineering

$$\left\{ \begin{array}{rcl} 3xy^2 + zt & = & 0 \\ x^2z + 5xyt & = & 0 \\ y^3 + 7z + 11t & = & 0 \\ x^2t + 13yz & = & 0 \end{array} \right.$$

Some characteristics of **multivariate systems**:

- Base field: typically an extension of GF(2) for crypto.
- Nb. of unknowns n, nb. of equations m, ratio n/m.

For any field, if $n \approx m$, solving a random quadratic system is **NP-hard** (problem $\mathcal{MQ}$).
But easier for **sparse systems**.

SOLVING MULTIVARIATE SYSTEMS

- **Linearization**: needs $\#$equations $\geq \#$monomials.
- Variants of Buchberger's algorithm for Groebner bases:
  - $\mathbf{F_4}$ and $\mathbf{F_5}$ [Faugère 99, 02],
  - **XL** & co [Lazard 83, Courtois-Klimov-Patarin-Shamir 99],
- **SAT-solvers** with ANF↔SAT conversion [Massaci-Marraro 00, Courtois-Bard 06],
- Dedicated methods for under-/over-defined or sparse systems.

Ex: GF(256) system with 40 eq. and 20 unknowns, solved by **XL-Wiedemann** within $< 2^{45}$ Opteron cycles ("a few hours") [Yang-Chen-Bernstein-Chen 07].

MULTIVARIATE CRYPTOGRAPHY

Mainly **signature, asym. encryption, authentication** schemes.

Pioneering works with C* [Matsumoto-Imai 88] and HFE [Patarin 96].
Subsequent variants (PMI, QUARTZ, SFLASH, TTS, etc.), and a stream cipher (QUAD).

Advantages

- **Fast** in cheap hardware and smart-cards, short signatures.
- **Reduction** to a hard problem ($\mathcal{MQ}$, IP, Minrank, etc.).

But many designs and/or instances broken with differential attacks, rank attacks, system solvers, etc.

MULTIVARIATE HASH FUNCTIONS

Merkle-Damgård construction with m-field-element message blocks and n-field-element chaining value.

Compression function

$$h : \mathbb{K}^{m+n} \mapsto \mathbb{K}^n,\ \ m \in \mathbb{Z}$$

explicitly defined as n algebraic equations (the components)

$$\{h_i : \mathbb{K}^{m+n} \mapsto \mathbb{K}\}_{0 \le i < n}.$$

For a given set of parameters (m, n, degree, density,...) we consider **families** indexed by the equation system.

Security reduction for **preimage** only, for a **random** instance h.

(We'll also call h a "hash function".)

QUADRATIC HASH (DEGREE 2)

Quadratic components ($\deg(h_i) = 2$, $0 \leq i < n$).

Can find **collisions efficiently** by solving the linear system

$$h(x) - h(x - \Delta) = 0$$

for an arbitrary fixed and known difference $\Delta \neq 0$.
Time in $\mathcal{O}(m^3)$.

Generally, finding collisions in a degree-d system essentially reduces to solving a degree-$(d-1)$ system.

SPARSE CUBIC HASH (DEGREE 3)

[Ding-Yang 07]

Cubic components ($\deg(h_i) = 3$, $0 \leq i < n$), with

$$h : \mathbb{K}^{2n} \mapsto \mathbb{K}^n$$

of fixed density $\delta = 0.1\%$ (*vs.* expected density 50% for a random system).

Low density $\Rightarrow$ less storage requirements, faster, etc., but no longer reduction to a NP-hard problem.

QUARTIC HASH (DEGREE 4)

[Billet-Robshaw-Peyrin 07]

Two composed quadratic systems:

$$h = g \circ f$$

with

$$f : \mathbb{K}^{m+n} \mapsto \mathbb{K}^{r},\ g : \mathbb{K}^{r} \mapsto \mathbb{K}^{n}, r > m + n.$$

Security reduction to $\mathcal{MQ}$ for preimage.

Large memory requirements ($\approx$ 3 Mb for SHA-1 param. over GF(2)).

HOW SECURE IS IT ? [Aumasson-Meier 07]

1. Universality and collisions of sparse systems
2. Collisions in semi-sparse systems
3. Pseudo-randomness and unpredictability
4. HMAC and NMAC

COLLISIONS IN SPARSE SYSTEMS

Key fact: for a random h of low density δ, there exists with high probability a collision of the form

$$h(0, \ldots, 0) = h(0, \ldots, 0, x_i \neq 0, 0, \ldots, 0).$$

$\Rightarrow$ **universality** and **collision resistance** broken for sparse systems. (degree-independent.)

Solution: don't apply δ to linear terms (**semi-sparse** systems).

COLLISIONS IN SEMI-SPARSE SYSTEMS

Consider **cubic hash** over GF(2), low density for **cubic** monomials only.
Algorithm for **collision search**:

1. Compute the quadratic differential system $h(x) - h(x - \Delta)$
2. Remove quadratic terms, get the system $h'(x) = 0$
   Consider the linear system $h'(x) = 0$ where quadratic terms have been deleted.
3. Compute the generating matrix of the corresponding linear code.
4. Compute a low-weight word of this code (i.e. a solution of $h(x) = 0$).
5. Plug this solution into $h(x) - h(x - \Delta)$: sums of quadratic terms vanish with non-negligible probability: a collision is found.

COLLISIONS IN SEMI-SPARSE SYSTEMS

Bottleneck: find **low-weight words** in a random linear code; fastest algorithm in [Canteaut-Chabaud 98].

For a cubic system over GF(2) with 160 equations and 320 unknowns, density 0.1% for cubic monomials only:

$$\text{Ratio time/success } \approx 2^{52},$$

against $\approx 2^{80}$ for a birthday attack.

DISTRIBUTIONS' PROPERTIES

Definitions for function families [Naor-Reingold 98], for a black-box random instance h:

- **Pseudo-randomness**: hard to distinguish from a random function.
- **Unpredictability**: for all x, hard to compute $h(x)$ without a box query.

Multivariate hash over GF(2): because of the **low degree**, can recover the full ANF of the components within

$$\sum_{i=0}^{d} \binom{m+n}{i} \text{ queries to the box.}$$

For parameters proposed, $< 2^{26}$ queries for both schemes.

Can fix this with some padding rule and/or output filter ?

KEY RECOVERY IN HMAC AND NMAC

$\mathrm{HMAC}_k(x) = h\,(k \oplus \mathrm{OPAD} \| h(k \oplus \mathrm{IPAD} \| x))$
$\Rightarrow$ can get equations of **degree $\mathbf{d^3}$** ($d = \deg(h)$).

$\mathrm{NMAC}_{k_1,k_2}(x) = h_{k_1}\,(h_{k_2}(x))$
$\Rightarrow$ can get equations of **degree $\mathbf{d^2}$**.

Depending on parameters, linearization and/or system solvers can outperform brute force...

Ex: **NMAC** with sparse cubics over GF(256) with 20 equations and 40 variables. 2^{23} queries are sufficient to run linearization (time cost $C \cdot 2^{74}$ vs. 2^{160} by brute force).

SUMMARY

Multivariate hash provide

- **speed** in HW (presumably, need benchmarks),
- **security reduction** for preimage,

but

- give no argument for **collision resistance**,
- do not provide **pseudo-random** function families,
- **sparse** equations can lead to trivial collisions,
- NMAC significantly weaker than HMAC,

However,

- we studied **compression functions**: can a smart operating mode strengthen the (iterated) hash functions ?

THE END

Full references in [Aumasson-Meier 07].

Paper & slides online at `www.131002.net`.

QUESTIONS ?