Secure PRNGs from Specialized Polynomial Maps over Any $\mathbb{F}_q$

Michael Feng-Hao Liu[*] Chi-Jen Lu[†] Bo-Yin Yang[†‡] Jintai Ding[‡]

October 23, 2007

Abstract

We prove that a random map drawn from any class $\mathfrak{C}$ of polynomial maps from $(\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^{n+r}$ that is (i) totally random in the affine terms, and (ii) has a negligible chance of being not strongly one-way, provides a secure PRNG (hence a secure stream cipher) for any q. Plausible choices for $\mathfrak{C}$ are semi-sparse (i.e., the affine terms are truly random) systems and other systems that are easy to evaluate from a small (compared to a generic map) number of parameters.

To our knowledge, there are no other positive results for provable security of specialized polynomial systems, in particular sparse ones (which are natural candidates to investigate for speed). We can build a family of provably secure stream ciphers a rough implementation of which *at the same security level* can be more than twice faster than an optimized QUAD (and any other provably secure stream ciphers proposed so far), and uses much less storage. This may also help build faster provably secure hashes.

We also examine the effects of recent results on specialization on security, e.g., Aumasson-Meier (ICISC 2007), which precludes Merkle-Damgård compression using polynomials systems uniformly *very* sparse in every degree from being universally collision-free. We conclude that our ideas are consistent with and complements these new results. We think that we can build secure primitives based on specialized (versus generic) polynomial maps which are more efficient.

Keywords: sparse multivariate polynomial map, PRNG, hash function, provable security

1 Introduction

Cryptographers have used multivariate polynomial maps for primitives since Matsumoto-Imai [22] but there is a dearth of results proving security based on plausible hardness assumptions.

Berbain-Gilbert-Patarin presented a seminal result at Eurocrypt 2006 [6] that if an overdetermined generic random multivariate quadratic map over $\mathbb{F}_2$ is probabilistically one-way, then it provides a provablely secure pseudo-random number generator (PRNG). However it is not very efficient for practical applications, and it was shown that tweaking the system for speed may lead to a [nearly] brute-force attack as in [28].

In the following we investigate one way to ameliorate this problem, namely by considering non-generic (e.g., sparse) polynomial systems of possibly higher degree, and over any $\mathbb{F}_q$ *not just* $\mathbb{F}_2$.

[*]Department of Computer Science, Brown University, Providence RI, USA, fenghao.liu@gmail.com
[†]Institute of Information Science, Academia Sinica, Taipei, Taiwan, [cjlu,byyang]@iis.sinica.edu.tw
[‡]Department of Mathematical Sciences, University of Cincinnati, Cincinnati OH, USA, ding@math.uc.edu

1.1 Questions

Problem $\mathcal{MQ}(q,n,m)$: Solve $P_1(\mathbf{x}) = P_2(\mathbf{x}) = \cdots = P_m(\mathbf{x}) = 0$ where P_i's are quadratic in $\mathbf{x} = (x_1, \ldots, x_n)$. All coefficients and variables are in $\mathbb{F}_q$.

The "multivariate quadratic" problem, often claimed as basis for multivariate quadratic public-key cryptosystems, is long known to be NP-hard [16]. We could take instead of quadratic P_i's polynomials of degree d and have a "multivariate polynomial system problem" $\mathcal{M}P(q,d,n,m)$, clearly also NP-hard. However, it is not easy to base a proof on worst-case hardness; the premise used in [6] is the following average-case hardness assumption:

Conjecture $\mathcal{MQ}$: No algorithm can solve (say in poly(n)-time or, practically, 2^{80} operations) any fixed $\varepsilon > 0$ proportion of $\mathcal{MQ}(q,n,m)$ instances with random coefficients in $\mathbb{F}_q$ and $n \propto m$.

If Conjecture $\mathcal{MQ}$ is true, then an iterative stream cipher with randomly chosen multivariate quadratics in $\mathbb{F}_2$ as update and output filter functions is secure [6]. However, a looseness factor in the theorem means that despite being faster than other provably secure stream ciphers at the same guaranteed cryptanalytic complexity, it is much, much slower than AES.

We can increase the difficulty of solving a system of nonlinear polynomial equations, by increasing (a) the field size q, (b) the number of variables n, or (c) the degree d of the system (for analysis see [3, 4, 26]). Alas, each costs time. Indeed cubic or higher-degree generic polynomials would be inhibitively time-consuming. Also, fields larger than $\mathbb{F}_2$ are not covered by the [6] security proof.

A logical next step is to consider specialized polynomial systems that are easier to evaluate, such that we can compensate for the extra complexity incurred in strengthening the cipher. A natural candidate is sparsity in the chosen polynomials. To our knowledge, however, there are no prior positive results for provable security of specialized polynomial systems, and specifically sparse ones.

So the questions we are trying to answer are:

- *Can we prove a similar result to [6] allowing for more efficiently evaluated specialized systems?*
- *What do we know about how these specializations affect complexity of system-solving?*

1.2 Our New Ideas and Main Results

We consider instead of $\mathcal{MQ}$ maps, a different class $\mathfrak{C}$ of multivariate polynomial maps from $(\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^{n+r}$ that is random in the affine terms (i.e., for any $f \in \mathfrak{C}$, $f + L \in \mathfrak{C}$ for any affine L). We further assume that an adversary has negligible chance to solve an inversion problem instance randomly drawn from $\mathfrak{C}$ within some resource constraint R. This is usually poly($n+r$) [for theory purposes] or 2^{80} elementary operations [when discussing a practical instance]. We will call such a class POWFRAT ("probabilistically one-way functions, random affine terms") in $\mathbb{F}_q$.

We will show in Secs. 2–3 that a random map from a POWFRAT class $\mathfrak{C}$ has a high probability to provide a secure PRNG (and hence a probably secure stream cipher), ***for any q.*** The key to the extension from $\mathbb{F}_2$ to general $\mathbb{F}_q$ involves a reconstruction over linear polynomials, which is a generalization of the Goldreich-Levin hard core bit by Goldreich-Rubinfeld-Sudan [18].

The following two classes of inversion problems are hard to solve to the best of our knowledge, and we conjecture that these represent POWFRAT classes:

Problem $\mathcal{SMP}(q, d, n, m, (\eta_2, \ldots, \eta_d))$**:** Solve $P_1(\mathbf{x}) = P_2(\mathbf{x}) = \cdots = P_m(\mathbf{x}) = 0$, where each P_i is a randomly chosen degree-d equation, such that $\eta_i = \eta_i(n)$ nonzero degree-i terms for each $i \geq 2$ are present. The affine terms (coefficients) are totally randomly chosen.

Problem SRQ(q, n, m, h)**:** The P_i are quadratics formed from sequence of rotations. Start with $P_0 = x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n$ (where n is even), and obtain successive P_j by performing sparse affine maps on $\mathbf{x}$. I.e., $\mathbf{x}^{(0)} := \mathbf{x}$, $\mathbf{x}^{(i)} := M^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}^{(i)}$, $y_i := P_i(\mathbf{x}) := P_0(\mathbf{x}^{(i)}) + c_i$, $\forall i$. Matrices $M^{(i)}$ are randomly chosen, invertible and sparse with h entries per row.

$\mathcal{SMP}$ and SRQ mean respectively "Sparse Multivariate Polynomials" and "Sparse-Rotated Quadratics", whose instances we tested to be hard to solve via generic solvers (Sec. 4). We try to analyze their use for symmetric primitives (stream ciphers and hash functions). In preliminary implementations, we are able to achieve 5541 and 11744 cycles per byte for a $\mathcal{SMP}$-based *secure* stream cipher over $\mathbb{F}_{16}$ (quartic, 108 variables) and $\mathbb{F}_2$ (cubic, 208 variables) respectively. The former is at least twice as fast as any other stream ciphers provably secure at the same parameters (cf. Sec. 5.2).

1.3 Previous Work

There had been known "provably secure" PRNGs. Those based on discrete log [17], or on RSA (as in Blum, Blum, and Shub [8]) or a modification thereof [25], or $\mathcal{MQ}$ [6]. However, some security proofs for cryptosystems require impractically high parameters for "provable security", which limit their utility. For example:

- The BBS stream generator at commonly used parameters is not provably secure [19, Sec. 6.1].
- With [25], the specified security level was 2^{70}, today's cryptographers usually aim for 2^{80}.
- Similarly with `QUAD` there is a gap between the "recommended" instances and the provably secure instances (i.e., the tested instances were unprovable or unproven [28]).

Our work is the first positive result on using specialized polynomial systems which apparently achieves a faster provably-secure PRNG/stream cipher than anything previously known.

The generic types of methods for solving polynomial systems — Faugère's $\mathbf{F_4}$-$\mathbf{F_5}$ and XL-derivatives — are not affected drastically by sparsity. In the former, sparsity is quickly lost and tests show that there is little difference in timing when solving $\mathcal{SMP}$ or SRQ instances. Recent versions of XL [28] speeds up proportionally to sparsity. We therefore surveyed the literature for recent results on solving or attacking specialized systems in crypto, listed below.

- Aumasson-Meier (ICISC 2007) [1] shows that in some cases sparsity in primarily underdefined systems leads to improved attacks. Results do not apply to overdetermined systems in general.
- Bard-Courtois-Jefferson [2] tests SAT solvers on uniformly sparse $\mathbb{F}_2$ equations; give numbers.
- Raddum-Samaev [23, 24] attacks "clumped" systems (even though the title says "sparse"). Similarly the Courtois-Pieprzyk XSL attack [11] requires a lot of structure (i.e., "clumping").

Our results are consistent with the above results and complements them.

1.4 Future Work

We think that this represents only a first step in using specialized polynomials for secure primitives. There are many obvious directions for future work in this area: Can we tighten our theorem to get a smaller "looseness factor"? Can we get an upper and lower bound on the order of ϵ that is required? Can we quantify exactly how small would our η have to be before specialized attacks mentioned below becomes a serious problem? Much remains to be done.

2 PRNG Based on Specialized Polynomial Map in $\mathbb{F}_2$

This section both provide a recap of past results and extend them to specialized maps over $\mathbb{F}_2$. We will start with definitions and models, then give the key results on the provable security level.

Computational Distinguishability: Probability distributions D_1 and D_2 over a finite set Ω are **computationally distinguishable** with computing resources R and advantage ϵ if there exist a probabilistic algorithm A which on any input $x \in \Omega$ outputs answer 1 (accept) or 0 (reject) using computing resources at most R and satisfies $|\Pr_{x\in D_1}(A(x)=1) - \Pr_{x\in D_2}(A(x)=1)| > \epsilon$. The above probabilities are not only taken over x values distributed according to D_1 or D_2, but also over the random choices that are used by algorithm A. Algorithm A is called a distinguisher with advantage ϵ.

If no such algorithm exists, then we say that D_1 and D_2 are computationally indistinguishable with advantage better than ϵ. If R is not specified, we implicitly mean feasible computing resources (e.g., $< 2^{80}$ simple operations, and reasonable limits in sampling from D_1 and D_2).

PRNG: Let $n < L$ be two integers and $K = \mathbb{F}_q$ be a finite field. The map $G : K^n \to K^L$ is said to be a Pseudorandom Number Generator (PRNG) if the probability distribution of the random variable $G(\mathbf{x})$, where the vector $\mathbf{x}$ is uniformly random in K^n, is computationally indistinguishable from a uniformly random vector in K^L. *Usually $q = 2$ but it is not required.*

Given any PRNG, there is a standard way to stretch it into a secure iterative stream cipher. Thus, to build a PRNG from any POWFRAT family of map $\mathfrak{C}$ from $\mathbb{F}_2^n \to \mathbb{F}_2^m$, we need to

1. show that if a random instance $\mathbf{S}$ drawn from $\mathfrak{C}$ is *not* a PRNG, then we can predict, with the help of information from the image $\mathbf{S}(\mathbf{x})$, any linear function $R(\mathbf{x})$ with strictly larger than $1/2$ probability; then
2. use the Goldreich-Levin theorem, which states that any linear function R is a hardcore bit of any $\mathbb{F}_2^n \to \mathbb{F}_2^m$ one-way function $\mathbf{S}$. I.e., *being able to guess with strictly larger than $1/2$ probability $R(\mathbf{x})$ from $\mathbf{S}(\mathbf{x})$ means that we can compute $\mathbf{x}$ from $\mathbf{S}(\mathbf{x})$ with non-negligible probability.*

2.1 From Distinguisher to Predictor

The following two results are valid for any $K = \mathbb{F}_q$. First, a standard result:

Proposition 1. *Take an old-fashioned iterated stream cipher with $\mathbf{Q} : K^n \to K^n$ and $\mathbf{P} : K^n \to K^r$ as the update and output filter functions and random initial state $\mathbf{x}_0$, that is, starting from the initial state $\mathbf{x}_0$, at each step we update with $\mathbf{x}_{i+1} = \mathbf{Q}(\mathbf{x}_i)$ and output $\mathbf{y}_i = \mathbf{P}(\mathbf{x}_i)$.*

$$\begin{tikzcd}
\mathbf{x}_0 \arrow[r] \arrow[d] & \mathbf{x}_1 = \mathbf{Q}(\mathbf{x}_0) \arrow[r] \arrow[d] & \mathbf{x}_2 = \mathbf{Q}(\mathbf{x}_1) \arrow[r] \arrow[d] & \mathbf{x}_3 = \mathbf{Q}(\mathbf{x}_2) \arrow[r] \arrow[d] & \cdots & \textit{(state)} \\
\mathbf{y}_0 = \mathbf{P}(\mathbf{x}_0) & \mathbf{y}_1 = \mathbf{P}(\mathbf{x}_1) & \mathbf{y}_2 = \mathbf{P}(\mathbf{x}_2) & \mathbf{y}_3 = \mathbf{P}(\mathbf{x}_3) & \cdots & \textit{(output)}
\end{tikzcd}$$

If we can distinguish between its first λ blocks of output $(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{\lambda-1})$ with advantage ϵ in time T, then we can distinguish between the output of a true random vector in K^{n+r} and the output of $\mathbf{S} = (P, Q)$ in time $T + \lambda T_{\mathbf{S}}$ with advantage ϵ/λ. [We put a proof in the appendix.]

The following extends [6, Theorem 2] to work for any $\mathbb{F}_q$, linear (only) R and any POWFRAT class $\mathfrak{C}$. *Also from this proof we see why the restrictions on a POWFRAT class $\mathfrak{C}$ exist.*

Proposition 2. *Let $K = \mathbb{F}_q$ and $\mathfrak{C}$ be a POWFRAT class of polynomial functions. Suppose further there is an algorithm A that given a randomly chosen known system $\mathbf{S}(: K^n \to K^m) \in \mathfrak{C}$, and distinguishes $\mathbf{S}(\mathbf{x})$, where $\mathbf{x} \in K^n$ is an unknown random input value, from a random element in K^m with advantage at least ϵ in time T. Then there is an algorithm B that, given a randomly chosen system $\mathbf{S} : K^n \to K^m$ in $\mathfrak{C}$ and, any $K^n \to K$ linear form R, and $\mathbf{y} = \mathbf{S}(\mathbf{x})$ where $\mathbf{x}$ is a random input value, predicts $R(\mathbf{x})$ with success probability at least $(1+\epsilon/2)/q$ using at most $T + 2T_{\mathbf{S}}$ operations.*

Proof. Without loss of generality, we may suppose that A has probability at least ϵ higher to return 1 on an input $(\mathbf{S}, \mathbf{S}(\mathbf{x}))$ than on a random $(\mathbf{S}, \mathbf{y})$. Define a recentered distinguisher

$$A'(\mathbf{S}, \mathbf{w}) := \begin{cases} A(\mathbf{S}, \mathbf{w}), & \text{probability } \frac{1}{2} \\ 1 - A(\mathbf{S}, \mathbf{u}), & \mathbf{u} \in K^m \text{ uniform random, probabilty } \frac{1}{2} \end{cases}$$

then A' returns 1 with probability $\frac{1+\epsilon}{2}$ on input $(\mathbf{S}, \mathbf{S}(\mathbf{x}))$ and with probability $\frac{1}{2}$ on input $(\mathbf{S}, \mathbf{u})$ for random $\mathbf{u}$.

Now, given an input $\mathbf{S}$ and $\mathbf{y} \in K^m$, the algorithm B first guesses a value $v \in K$ (representing a guess for $R(\mathbf{x})$), then a random vector $\mathbf{u} \in K^m$, and form $\mathbf{S}' := \mathbf{S} + R\mathbf{u} : K^n \to K^m$. This is equal to S plus a random affine function, and is hence random and in $\mathfrak{C}$.

$$B(\mathbf{S}, \mathbf{y}, R) := \begin{cases} v, & \text{if } A'(\mathbf{S}', \mathbf{y} + v\mathbf{u}) = 1; \\ \text{uniform random from } K\backslash\{v\}, & \text{if } A'(\mathbf{S}', \mathbf{y} + v\mathbf{u}) = 0. \end{cases}$$

If $v = R(\mathbf{x})$, $\mathbf{y} + v\mathbf{u} = \mathbf{S}'(\mathbf{x})$, else $\mathbf{y} + v\mathbf{u}$ is equal to $\mathbf{S}'(\mathbf{x})$ plus a nonzero multiple of the random vector u, hence is equivalent to being uniformly random. The probability that $B = B(\mathbf{S}, \mathbf{S}(\mathbf{x}), R)$ is the correct guess is hence

$$\begin{aligned} \Pr(B = R(\mathbf{x})) &= \Pr(B = v | v = R(\mathbf{x})) \Pr(v = R(\mathbf{x})) + \Pr(B = R(\mathbf{x}) | v \neq R(\mathbf{x})) \Pr(v \neq R(\mathbf{x})) \\ &= \frac{1}{q}\left(\frac{1}{2} + \frac{\epsilon}{2}\right) + \left(\frac{q-1}{q}\right)\frac{1}{2}\left(\frac{1}{q-1}\right) = \frac{1}{q}\left(1 + \frac{\epsilon}{2}\right). \end{aligned}$$

□

We see that the reasoning can work this way if and only if $\mathbf{S} + R\mathbf{u}$ have the same distribution as $\mathbf{S}$.

2.2 Constructing a PRNG from POWFRAT maps ($\mathbb{F}_2$ Case)

Proposition 3 ([21]). *Suppose there is an algorithm B that given a system $\mathbf{S}(: \mathbb{F}_2^n \to \mathbb{F}_2^m)$, a random n-bit to one-bit linear form R and the image $\mathbf{S}(\mathbf{x})$ of a randomly chosen unknown $\mathbf{x}$, predicts $R(\mathbf{x})$ with probability at least $\frac{1}{2} + \epsilon$ over all possible inputs $(\mathbf{S}, \mathbf{S}(\mathbf{x}), R)$ using time T, then there is an algorithm C that given $\mathbf{S}$ and the m-bit image $\mathbf{S}(\mathbf{x})$ of a randomly chosen n-bit vector $\mathbf{x}$ produces a preimage of $\mathbf{S}(\mathbf{x})$ with probability (over all $\mathbf{x}$ and $\mathbf{S}$) at least $\epsilon/2$ in time*

$$T' = \frac{8n^2}{\epsilon^2}\left(T + \log\left(\frac{8n}{\epsilon^2}\right) + \frac{8n}{\epsilon^2}T_{\mathbf{S}}\right)$$

Note: This is really the Goldreich-Levin theorem of which we omit the proof here. This essentially states that linear forms are hard-core bits of any one-way function. In fact, the tighter form [6, Proof of Theorem 3] (using a fast Walsh transform) can be simply followed word-for-word.

This above result (which only holds for $\mathbb{F}_2$) with Prop. 2 shows that any POWFRAT family of maps induces PRNGs over $\mathbb{F}_2$. To get a useful stream cipher, we can combine Props. 1–3:

Proposition 4. *If $\mathbf{S} = (P, Q)$ is an instance drawn from a POWFRAT class $\mathfrak{C}$, where $\mathbf{P} : \mathbb{F}_2^n \to \mathbb{F}_2^r$, $\mathbf{Q} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are the output filter and update functions of a stream cipher as in Prop. 1, then if we can distinguish between λ output blocks of the stream cipher in T time, we can find $\mathbf{x}$ from $\mathbf{S}(\mathbf{x})$ with probability at least $\frac{\epsilon}{8\lambda}$ in time*

$$T' = \frac{2^7 n^2 \lambda^2}{\epsilon^2}\left(T + (\lambda+2)T_{\mathbf{S}} + \log\left(\frac{2^7 n \lambda^2}{\epsilon^2}\right) + 2\right) + \frac{2^7 n \lambda^2}{\epsilon^2}T_{\mathbf{S}} \qquad (1)$$

Note: Roughly this means that if we let $r = n$, want to establish a safety level of 2^{80} multiplications, want $L = \lambda r = 2^{40}$ bits between key refreshes, and can accept $\epsilon = 10^{-2}$, then $T' \lesssim 2^{230}/n$. All we need now is to find a map from $\mathbb{F}_2^n \to \mathbb{F}_2^{2n}$ which takes this amount of time to invert.

As we see below, unless equation-solving improves greatly for sparse systems, this implies that a handful of cubic terms added to a `QUAD` system with $n = r = 208$, $q = 2$ can be deemed secure to 2^{80}. There is no sense in going any lower than that, because solving a system with n bit-variables can never take much more effort than $2^n\times$ whatever time it takes to evaluate one equation.

3 PRNG Based on Specialized Polynomial Map in $\mathbb{F}_q$

In this section, we show a way to extend the main results of the last section to $\mathbb{F}_q$, by using an extension of the Goldreich-Levin hard-core bit theorem.

Proposition 5. *Let $K = \mathbb{F}_q$. Suppose there is an algorithm B that given a system $\mathbf{S}(: K^n \to K^m)$, a random $K^n \to K$ linear form R and the image $\mathbf{S}(\mathbf{x})$ of a randomly chosen unknown $\mathbf{x}$, predicts $R(\mathbf{x})$ with probability at least $\frac{1}{q} + \epsilon$ over all possible inputs $(\mathbf{S}, \mathbf{S}(\mathbf{x}), R)$ using time T, then there is an algorithm C that given $\mathbf{S}$ and the m-bit image $\mathbf{S}(\mathbf{x})$ of a randomly chosen vector $\mathbf{x} \in K^n$ produces a preimage of $\mathbf{S}(\mathbf{x})$ with probability (over all $\mathbf{x}$ and $\mathbf{S}$) at least $\epsilon/2$ in time*

$$T' \le 2^{10}\left(\frac{nq}{\epsilon^5}\right)\log^2\left(\frac{n}{\epsilon}\right)T + \left(1 - \frac{1}{q}\right)^2 \epsilon^{-2}\, T_{\mathbf{S}}$$

If we know that one out of two exclusive possibilities takes place with probability strictly larger than 50%, then the other one must happen strictly less often 50%. If we know that one of q possibilities takes place with probability strictly greater than $1/q$, we cannot be sure that another possibility does not occur with even higher possibility. Therefore, we can only treat this as a case of learning a linear functional with queries to a highly noisy oracle. **Due to this difference, the order of ϵ in T'/T is as high as ϵ^{-5} in Prop. 5, but only ϵ^{-2} in Prop. 3.**

Proposition 6. *If $\mathbf{S} = (P, Q)$ is an instance drawn from a POWFRAT class $\mathfrak{C}$, where $\mathbf{P} : \mathbb{F}_q^n \to \mathbb{F}_q^r$, $\mathbf{Q} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ are the output filter and update functions of a stream cipher as in Prop. 1, then if we can distinguish between λ output blocks of the stream cipher in T time, we can find $\mathbf{x}$ from $\mathbf{S}(\mathbf{x})$ with probability at least $\frac{\epsilon}{4q\lambda}$ in time*

$$T' = 2^{15}\frac{nq^6\lambda^5}{\epsilon^5}\log^2\left(\frac{2qn\lambda}{\epsilon}\right)(T + (\lambda+2)T_{\mathbf{S}}) + \left(1 - \frac{1}{q}\right)^2 \frac{4q^2\lambda^2}{\epsilon^2}T_{\mathbf{S}} \tag{2}$$

This is a straightforward combination of Props. 1, 2, and 5. In the remainder of this section, we give a proof to Prop. 5 by a variation of the procedure used by Goldreich-Rubinfeld-Sudan [18, Secs. 2 and 4], to give it concrete values that we can derive security proofs from.

3.1 Hardcore Predicate and Learning Polynomials

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{b} = (b_1, b_2, \ldots, b_n)$, and x_i, b_i are elements in a finite field $K = \mathbb{F}_q$. Given an arbitrary strong one way function $h(\mathbf{x})$, then $F(\mathbf{x}, \mathbf{b}) = (h(\mathbf{x}), \mathbf{b})$ is also a one way function. Claim $\mathbf{x} \cdot \mathbf{b}$ is the hard-core bit of $F(\mathbf{x}, \mathbf{b})$, where $\mathbf{x} \cdot \mathbf{b}$ means their inner product.

Supposed we have a predictor P which predicts its hardcore $\mathbf{x} \cdot \mathbf{b}$ given $(h(\mathbf{x}), \mathbf{b})$ with probability more than $\frac{1}{q} + \epsilon$, then we can write in the math form:

$$\Pr_{\mathbf{b},\mathbf{x}}[P(h(\mathbf{x}), \mathbf{b}) = \mathbf{x} \cdot \mathbf{b}] > \frac{1}{q} + \epsilon.$$

By Markov inequality, we know there must be more than $\epsilon/2$ fraction of x such that $\Pr_{\mathbf{b}}[P(h(\mathbf{x}), \mathbf{b}) = \mathbf{x} \cdot \mathbf{b}] > \frac{1}{q} + \frac{\epsilon}{2}$. For this fraction of x, we are trying to find the inverse of $h(\mathbf{x})$ ($F(\mathbf{x})$ as well) through the predictor. Also $\mathbf{x} \cdot \mathbf{b}$ can be written as $\sum b_i x_i$, then

$$\Pr_{\mathbf{b}}\left[P(h(\mathbf{x}), \mathbf{b}) = \sum b_i x_i\right] > \frac{1}{q} + \frac{\epsilon}{2}.$$

What this means in English is that, *if we can find a polynomial which almost matches an arbitrary function P, a predictor function, then we can eventually invert $\mathbf{x}$ from $F(\mathbf{x})$ a non-negligible portion of the time.* Now we try to reconstruct such linear polynomials through the access of the predictor, largely following the footsteps of [18].

3.2 Intuition of Reconstructing Linear Polynomials

Now we are given some oracle accesses to a function $f : K^n \to K$, where K is a finite field and $|K| = q$. We need to find all linear polynomials which match f with at least $\frac{1}{q} + \epsilon$ fraction of inputs x. Let $p(x_1, x_2, \ldots, x_n) = \sum_1^n p_i x_i$, and i-th prefix of p is $\sum_1^i p_j x_j$. The algorithm runs n rounds, and in the i-th round, it extends all possible candidates from the $(i-1)$-th round with all elements in

K and screens them, filtering out most bad prefixes. The pseudocode of the algorithm is presented in Algorithm 2. Since we want the algorithm to be efficient, we must efficiently screen possible prefixes from all extensions. We now introduce a screening algorithm to be called TestPrefix.

Algorithm 1 TestPrefix(f,ϵ,n,$(c_1, c_2 \dots, c_i)$)

```
Repeat poly$_1(\frac{n}{\epsilon})$ times:
Pick $\vec{s} = s_{i+1}, \dots, s_n \in_R$ GF$(q)$
Let $t =$ poly$_2$ $(\frac{n}{\epsilon})$
for $k = 1$ to $t$ do
    Pick $\vec{r} = r_1, r_2 \dots, r_i \in_R$ GF$(q)$
    $\sigma^{(k)} = f(\vec{r}, \vec{s}) - \sum_{j=1}^{i} c_j r_j$
end for
If there is $\sigma^{(k)} = \sigma$ for at least $\frac{1}{q} + \frac{\epsilon}{3}$ fraction of the $k$'s then ouput accept and halt
endRepeat
If all iterations were completed without accepting, then reject
```

Algorithm 2 Find All Polynomials(f, ϵ)

```
set a candidate queue Q[$i$] which stores all the candidates ($c_1$, $c_2$, $c_3$, ..., $c_i$) in the $i$-th round
for $i = 1$ to $n$ do
    Pick all elements in Q[$i$]
    TestPrefix($f$,$\epsilon$,$n$,$(c_1, c_2 \dots, c_i, \alpha)$) for all $\alpha \in$ F
    If TestPrefix accepts, then push $(c_1, c_2 \dots, c_i, \alpha)$ into Q[$i + 1$] i.e. it is a candidate in the
    $(i + 1)$-th round
end for
```

Supposed we are testing the i-th prefix $(c_1, c_2, \dots, c_i)$, we are going to evaluate the quantity of:

$$P_{\vec{s}}(\sigma) := \Pr_{r_1, r_2 \dots, r_i \in K} \left[f(\vec{r}, \vec{s}) = \sum_{j=1}^{i} c_j r_j + \sigma \right]$$

where $\vec{r} = (r_1, r_2, \dots, r_i)$. The value of σ can be thought as a guess of $\sum_{i+1}^{n} p_j s_j$. For every $\vec{s}$, we can estimate the probability by a sample of several $\vec{r}$'s, and the error rate can be controlled by the times of sampling. If such $\vec{s}$ makes the probability significantly larger than $1/q$, then we accept. If no such $\vec{s}$ exists, we reject. The detailed algorithm is stated in the Algorithm 1: TestPrefix.

If a candidate $(c_1, c_2, \dots, c_i)$ passes through the Algorithm 1 for at least one suffix $\vec{s}$, there is a σ such that the estimate of $P_s(\sigma)$ is greater than $\frac{1}{q} + \frac{\epsilon}{3}$. For a correct candidate $(c_1, c_2, \dots, c_i)$, i.e. $(c_1, c_2, \dots, c_i)$ is the prefix of $p = (p_1, p_2, \dots, p_n)$ which matches f for at least $\frac{1}{q} + \epsilon$, and an arbitrary $\sigma = \sum_{i+1}^{n} p_j s_j$, it satisfies that $E_s[P_s(\sigma)] \geq \frac{1}{q} + \epsilon$. By Markov's inequality, for at least $\epsilon/2$ fraction of $\vec{s}$ and some corresponding σ, it holds that $P_s(\sigma) \geq \frac{1}{q} + \frac{\epsilon}{2}$. In Algorithm 1, we set $\frac{1}{q} + \frac{\epsilon}{3}$ as the passing criteria; thus the correct candidate will pass though the Algorithm 1 with great probability. However, [18, Sec. 4] shows that the total passing number of candidates in each round is limited. In fact, only a small number of candidates will pass the test. This maximum (also given by [18, Sec. 4]) number of prefixes that pass the test is $\leq (1 - \frac{1}{q})^2 \epsilon^{-2}$.

3.3 Giving Concrete Values to "Order of Polynomially Many"

Since there are $\epsilon/2$ fraction of suffix $\vec{s}$ such that $P_s(\sigma) \geq \frac{1}{q} + \frac{\epsilon}{2}$, we can randomly choose the suffix polynomially many times (k_1 times) to ensure that we would select such $\vec{s}$ with high probability. Also, for such $\vec{s}$, if we choose polynomially many times (k_2 times) of $\vec{r}$, there would be high probability that we would find some α for at least $\frac{1}{q}+\frac{\epsilon}{3}$ fraction. We are estimating how the polynomially many should be as the following:

$$\Pr[\text{ TestPrefix fails }] \leq$$

$$\Pr[\text{no such } \vec{s} \text{ is chosen }] + \Pr\left[\text{ no single element exists more than } \frac{1}{q} + \frac{\epsilon}{3} \text{ fraction }\right]$$

$$\Pr[\text{ no such } \vec{s} \text{ is chosen}] \leq (1-\epsilon/2)^{k_1} \leq e^{-\frac{k_1\epsilon}{2}} \leq \frac{1}{2}\frac{\epsilon}{\left(1-\frac{1}{q}\right)^2 \epsilon^{-2}\, nq}$$

So, we take k_1 as $O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})) \approx 3\frac{1}{\epsilon}\log(\frac{n}{\epsilon})$. On the other hand, we want to estimate the probability of there are no σ's with fraction at least $\frac{1}{q}+\frac{\epsilon}{3}$. For a correct suffix $\vec{s}$, we know for uniform $\vec{r}$, we get that σ with probability more than $\frac{1}{q}+\frac{\epsilon}{2}$. Let X_i be the random variable with value 1 if the i-th trial of $\vec{r}$ gets the correct σ, 0 otherwise. Then we have $\Pr[X_i = 1] \geq \frac{1}{q}+\frac{\epsilon}{2}$. Suppose we do k_2 trials:

$$\Pr\left[\text{ no single element exists more than } \frac{1}{q} + \frac{\epsilon}{3} \text{ fraction}\right] \leq \Pr\left[\sum_1^{k_2} X_i < (\frac{1}{q}+\frac{\epsilon}{3})k_2\right]$$

$$\leq \Pr\left[\left|\frac{\sum_{i=1}^{k_2} X_i}{k_2} - \left(\frac{1}{q}+\frac{\epsilon}{2}\right)\right| \geq \frac{\epsilon}{6}\right],$$

since these X_i's are independent, then by Chernoff's bound we have

$$\Pr\left[\left|\frac{\sum_{i=1}^{k_2} X_i}{t} - (\frac{1}{q}+\frac{\epsilon}{2})\right| \geq \frac{\epsilon}{6}\right] \leq 2e^{-\frac{k_2\epsilon^2}{2\times 36}} \leq \frac{1}{2}\frac{\epsilon}{\left(1-\frac{1}{q}\right)^2 \epsilon^{-2}\, nq},$$

$k_2 = O(\frac{\log(n/\epsilon)}{\epsilon^2}) \approx 216\frac{\log(n/\epsilon)}{\epsilon^2}$ is sufficient to make the inequality hold. Thus, we have

$$\Pr[\text{ TestPrefix fails }] \leq \frac{\epsilon}{\left(1-\frac{1}{q}\right)^2 \epsilon^{-2}\, nq}.$$

Also,

$$\Pr[\text{ Algorithm 2 fails }] \leq \Pr[\text{ one TestPrefix fails }] \leq \sum_{\text{all TestPrefix run}} \Pr[\text{ TestPrefix fails }]$$

$$\leq \left(\left(1-\frac{1}{q}\right)^2 \epsilon^{-2}\, nq\right)\left(\frac{\epsilon}{\left(1-\frac{1}{q}\right)^2 \epsilon^{-2}\, nq}\right) = \epsilon$$

Therefore, the algorithm will work with high probability. The worst case running time of algorithm 2 should be: $k_1k_2(1-\frac{1}{q})^2\frac{1}{\epsilon^2}nq = O(\frac{n}{\epsilon^5}\log^2(\frac{n}{\epsilon})) \lesssim 2^{10}\left(\frac{nq}{\epsilon^5}\right)\log^2(\frac{n}{\epsilon})$.

Note: $\left(1-\frac{1}{q}\right)^2 \epsilon^{-2}$ is the maximum number of candidates which pass in each round.

4 On $\mathcal{SMP}$ and SRQ under Generic Solvers

To verify that $\mathcal{SMP}$ and SRQ represent POWFRAT classes we need to show that

1. generic system-solvers do not run substantially faster on them; and
2. there are no specialized solvers that can take advantage of the sparsity.

In this section we look at generic solvers.

In general, there are two generic types of methods for solving polynomial systems, both related to the original Buchberger's algorithm. One is Faugère's $\mathbf{F_4}$-$\mathbf{F_5}$ and the other is XL-derivatives. In the former, sparsity is quickly lost and tests show that there are little difference in timing when solving POWFRAT (e.g., $\mathcal{SMP}$ or SRQ) instances. With recent versions of XL [28], the sparsity results in a proportional decrease in complexity. The effect of sparsity on such generic methods should be predictable and not very drastic, as shown by some testing (cf. Sec. 4.3).

4.1 Why SRQ can be a POWFRAT Class

The idea behind SRQ is that any quadratic map can be written as $f \circ L$, where f is a standard form and L is an invertible linear map. Now we will choose L to be sparse. A standard form for characteristic 2 fields is the "rank form" which for full-rank quadratics is

$$P_0(\mathbf{x}) = x_1x_2 + x_3x_4 + \cdots x_{n-1}x_n.$$

Clearly, by taking a random $\mathbf{c}$ and b, we can give $P_0(\mathbf{x}+\mathbf{c}) + b$ *any* random affine part. Since each $\mathbf{x}^{(i)}$ is related to $\mathbf{x} = \mathbf{x}^{(0)}$ by an invertible affine map, this holds for every component P_i. This means that results pertaining to sparsity of the linear terms such as [1] (cf. Sec. 4.5) never apply.

4.2 XL and $\mathbf{F_4}$-$\mathbf{F_5}$ Families for System-Solving

The XL and $\mathbf{F_4}$-$\mathbf{F_5}$ families of algorithms are spiritual descendants of Lazard's idea [20]: run an elimination on an extended Macaulay matrix (i.e., extending the resultant concept to many variables) as an improvement to Buchberger's algorithm for computing Gröbner bases [9].

Since we cannot discuss these methods in detail, we try to describe them briefly along with their projected complexities. Again, suppose we have the system $P_1(\mathbf{x}) = P_2(\mathbf{x}) = \cdots = P_m(\mathbf{x}) = 0$, where P_i is a degree-d_i polynomial in $\mathbf{x} = (x_1, \ldots, x_n)$, coefficients and variables in $K = \mathbb{F}_q$.

Method XL [10]: Fix a degree $D(\geq \max P_i)$. The set of degree-D-or-lower monomials is denoted $\mathcal{T} = \mathcal{T}^{(D)}$. $|\mathcal{T}^{(D)}|$ is the number of degree $\leq D$ monomials and will be denoted $T = T^{(D)}$. We now take each equation $P_i = 0$ and multiply it by every monomial up to $D - d_i$ to get an equation that is at most degree D. Collect all such equations in the set $\mathcal{R} = \mathcal{R}^{(D)} := \bigcup_{i=1}^{m}\{(uP_i = 0) : u \in \mathcal{T}^{(D-d_i)}\}$. We treat every monomial in $\mathcal{T}$ as independent and try to solve $\mathcal{R}$ as a linear system of equations.

The critical parameter is the difference between $I = \dim(\text{span}\mathcal{R})$, the rank of the space of equations $\mathcal{R}$, and T. If $T - I = 0$, the original system cannot be satisfied; if $T - I = 1$, then we should find a unique solution (with very high probability). Also, if $T - I < \min(D, q-1)$, we can reduce to a univariate equation [10]. We would like to predict D_0, the smallest D enabling resolution.

Note: For any pair of indices $i, j \leq m$, among linear combinations of the multiples of $P_j = 0$ will be $P_iP_j = 0$, and among linear combinations of the multiples of $P_i = 0$ will be $P_iP_j = 0$ — i.e., one dependency in span$\mathcal{R}$. In $\mathbb{F}_q$, $(P_i)^q = P_i$ which generates a similar type of dependency.

Proposition 7 ([27]). *Denote by $[u]s$ the coefficient of the monomial u in the expansion of s, then:*

1. *$T = [t^D]\,\dfrac{(1-t^q)^n}{(1-t)^{n+1}}$ which reduces to $\binom{n+D}{D}$ when $q > D$, and $\sum_{j=0}^{D}\binom{n}{j}$ when $q = 2$.*

2. *If the system is regular up to degree D, i.e., if* ***the relations $\mathcal{R}^{(D)}$ has no other dependencies than the obvious ones generated by $P_iP_j = P_jP_i$ and $P_i^q = P_i$****, then*

$$T - I = [t^D]\,G(t),\ \text{where } G(t) := G(t;n;d_1,d_2,\ldots,d_m) = \frac{(1-t^q)^n}{(1-t)^{n+1}}\prod_{j=1}^{m}\left(\frac{1-t^{d_j}}{1-t^{q\,d_j}}\right). \quad (3)$$

3. *For overdefined systems, Eq. 3 cannot hold when $D > D_{XL} = \min\{D : [t^D]G(t) \le 0\}$. If Eq. 3 holds up for every $D < D_{XL}$ and resolves at D_{XL}, we say that the system is* ***q-semiregular****. It is generally believed [3, 12] that* ***for random systems it is overwhelmingly likely that $D_0 = D_{XL}$, and indeed the system is not q-semiregular with very small probability.***

4. *When it resolves, XL takes $C_{XL} \lesssim (c_0 + c_1 \lg T)\,\tau\,T^2$ multiplications in $\mathbb{F}_q$, using a sparse solver like Wiedemann [26]. Here τ is the average number of terms per equation.*

We cannot describe methods $\mathbf{F_4}$-$\mathbf{F_5}$ [14, 15], which are just too sophisticated and complex to present here. Instead, we simply sketch a result that yields their complexities:

Proposition 8. *[3] For q-semiregular systems, $\mathbf{F_4}$ or $\mathbf{F_5}$ operates at the degree*

$$D = D_{reg} := \min\left\{D : [t^D]\left(\frac{(1-t^q)^n}{(1-t)^n}\prod_{j=1}^{m}\left(\frac{1-t^{d_j}}{1-t^{q\,d_j}}\right)\right) < 0\right\},$$

and take $\lesssim (c_0' + c_1' \lg \bar{T})\,\bar{T}^{\omega}$ multiplications, where $\bar{T} = [t^{D_{reg}}]\ ((1-t^q)^n(1-t)^{-n})$ counts the monomials of degree exactly D_{reg}, and $2 < \omega \le 3$ is the order of matrix multiplication used.

We do not know what works best under various resource limitations. We take the position of [28], e.g., XL with a sparse solver represents the best way to solve large and more or less random overdetermined systems when *the size of main memory space* is the critical restraint.

4.3 Testing the One-Wayness with Generic Solvers

We conducted numerous tests on $\mathcal{SMP}$ maps at various degrees and sparsity over the fields $\mathbb{F}_2$, $\mathbb{F}_{16}$, and $\mathbb{F}_{256}$. For example, Table 1 lists our tests in solving random $\mathcal{MQ}(256,n,m)$ instances where each polynomial only has n quadratic terms [we call these instances $\mathcal{SMQ}(256,n,m,n)$] with $\mathbf{F_4}$ over GF(256). It takes almost the same time as solving an $\mathcal{MQ}$ instance of the same size.

$m-n$	D_{XL}	D_{reg}	$n=9$	$n=10$	$n=11$	$n=12$	$n=13$
0	2^m	m	6.03	46.69	350.38	3322.21	sigmem
1	m	$\lceil\frac{m+1}{2}\rceil$	1.19	8.91	53.64	413.34	2535.32
2	$\lceil\frac{m+1}{2}\rceil$	$\lceil\frac{m+2-\sqrt{m+2}}{2}\rceil$	0.31	2.20	12.40	88.09	436.10

Table 1: $\mathcal{SMQ}(256,n,m,n)$ timing (sec): MAGMA 2.12, 2GB RAM, Athlon64x2 2.2GHz

n	7	8	9	10	11	12	13
D	5	6	6	7	7	8	8
$\mathcal{SMQ}(256, n, n+2, n)$	$9.34 \cdot 10^{-2}$	$1.17 \cdot 10^{0}$	$4.04 \cdot 10^{0}$	$6.02 \cdot 10^{1}$	$1.51 \cdot 10^{2}$	$2.34 \cdot 10^{3}$	$5.97 \cdot 10^{3}$
$\mathcal{MQ}(256, n, n+2)$	$2.06 \cdot 10^{-1}$	$2.92 \cdot 10^{0}$	$1.10 \cdot 10$	$1.81 \cdot 10^{2}$	$4.94 \cdot 10^{2}$	$8.20 \cdot 10^{3}$	$2, 22 \cdot 10^{4}$
ratio	2.20	2.49	2.73	3.00	3.27	3.50	3.72

Table 2: XL/Wiedemann timing (sec) on Core2Quad 2.4GHz, icc, 4-thread OpenMP, 8GB RAM

For XL variants that use sparse solvers as the last step [28] test results (one of which is shown in Table 2) confirms the natural guess: For $\mathcal{SMP}$ instances where the number of non-linear terms is not overly small, the solution degree of XL is unchanged, and the speed naturally goes down as the number of terms, nearly in direct proportion (in Tab. 2, should be close to $n/4$).

For $\mathbb{F}_2$, there are many special optimizations made for $\mathbf{F_4}$ in MAGMA, so we ran tests at various densities of quadratic terms in version 2.12-20. Typical results in Fig. 1 (Appendix). SRQ samples are labelled "sparse non-random". Most of the time the data points are close to each other. In some tests they overlap each other so closely that no difference in the timing is seen in a diagram.

4.4 A Brief Discussion on Specialization and Security

Since generic system-solvers show no unexpected improvement on our specializations, it remains for us to check that there are no other big improvements in solving specialized systems for. We list below what we know of recent new attempts on solving or attacking specialized systems in crypto, and show that *our results are consistent with these new results and somewhat complements them.*

- Aumasson-Meier [1] presented several ideas to attack primitives built on sparse polynomials systems, which we sketch separately in Sec. 4.5 below.

- Raddum-Samaev [23, 24] attacks what they term "sparse" systems, where each equation depend on a small number of variables. Essentially, the authors state that for systems of equations in n bit variables such that each equation depends on only k variables, we can solve the system in time roughly proportional to $2^{(1-\frac{1}{k})n}$ using a relatively small memory footprint. Since XL for cubics and higher degrees over $\mathbb{F}_2$ is more time-consuming than brute-force, this is fairly impressive. However, the "sparsity" defined by the authors is closer to "input locality" and very different from what people usually denote with this term. The attack is hence not applicable to $\mathcal{SMP}$-based stream ciphers.

  In a similar vein is the purported XSL attack on AES [11]. While the S was supposed to stand for Sparse, it really requires Structure – i.e., each equation depending on very few variables. So, whether that attack actually works or not, it does not apply to $\mathcal{SMP}$-based systems.

- Bard-Courtois-Jefferson [2] use SAT solvers on uniformly sparse $\mathbb{F}_2$ equations and give experimental numbers. According to the authors, the methods takes up much less memory than $\mathbf{F_4}$ or derivatives, but is slower than these traditional methods when they have enough memory.

  Some numbers for *very* overdefined and *very* sparse systems shows that converting to a conjunctive normal form and then running a SAT solver can have good results. This seems to be a very intriguing approach, but so far there are no theoretical analysis especially for when the number of equations is a few times the number of variables, which is the case for $\mathcal{SMP}$ or SRQ constructions.

4.5 Solutions and Collisions in Sparse Polynomial Systems

Aumasson-Meier recent published [1]) some quite interesting ideas on finding solutions or collisions for primitives using sparse polynomial systems (e.g., hashes proposed in [13]).

They showed that which implies that using sparse polynomials systems of uniform density *(in every degree)* for Merkle-Damgård compression will not be universally collision-free. Some *under-defined* systems that are sparse in the higher degrees can be solved with lower complexity. Their results do not apply to overdetermined systems in general. We summarize relevant results below.

- Overdetermined higher-degree maps that are sparse of uniform density, or at least sparse in the linear terms, is shown to have high probability of trivial collisions and near-collisions.

  It seems that everyone agrees, that linear terms should be totally random when constructing sparse polynomial systems for symmetric primitives.

- Suppose we have an *underdetermined* higher-degree map sparse in the non-affine part, i.e.,

  $$\mathbf{P} : \mathbb{F}_2^{n+r} \to \mathbb{F}_2^n,\ \mathbf{P}(\mathbf{x}) = \mathbf{b} + M\mathbf{x} + \mathbf{Q}(\mathbf{x})$$

  where $\mathbf{Q}$ has only quadratic or higher terms and is sparse. Aumasson-Meier suggests that we can find $\mathbf{P}^{-1}(\mathbf{y})$ as follows: find a basis for the kernel space of the augmented matrix $[M; \mathbf{b} + \mathbf{y}]$. Collect these basis vectors in a $(n + r + 1) \times (r + 1)$ matrix M' as a linear code. For an arbitrary $\mathbf{w} \in \mathbb{F}_2^{r+1}$, the codeword $\bar{\mathbf{x}} = M'\mathbf{w}$ will represent a solution to $\mathbf{y} = M\mathbf{x} + \mathbf{b}$ if its last component is 1. Use known methods to find relatively low-weight codewords for the code M' and substitute into $\mathbf{Q}(\mathbf{x})$, expecting it to vanish with non-negligible probability.

  Aumasson-Meier proposes to apply this method to construct collisions in Merkle-Damgård hashes with cubic compressor functions. It *does not work* for fields other than $\mathbb{F}_2$ or *overdetermined* systems. Its exact complexity is unknown and requires some further work. Finally, we point out that for something like our SRQ construction, even if $h = 3$ (the rotation matrices $M^{(i)}$ have only three entries per row), the number of cross-terms in each equations still quickly increases to have as many terms as totally random ones.

- Conversely, it has been suggested if we have an *overdetermined* higher-degree map

  $$\mathbf{P} : \mathbb{F}_2^{n} \to \mathbb{F}_2^{n+r},\ \mathbf{P}(\mathbf{x}) = \mathbf{b} + M\mathbf{x} + \mathbf{Q}(\mathbf{x})$$

  where $\mathbf{Q}$ has only quadratic or higher terms and is *extremely* sparse, we can consider $\mathbf{P}(\mathbf{x}) = \mathbf{y}$ as $M\mathbf{x} = (\mathbf{y} + \mathbf{b}) +$ **perturbation**, and use known methods for decoding attacks, i.e., solving overdetermined linear equations with perturbation. However, since a quadratic over $\mathbb{F}_2$ of rank $2k$ has bias 2^{-k-1}, the SRQ form acts exactly like a random quadratic under this kind of attack, and even $\mathcal{SMP}$ maps with a moderate number of quadratic terms will be intractible.

5 Summary of Uses for Specialized Polynomial Systems

The conclusion is that we always use totally random linear terms, no matter what else we do. With that taken into account, specialized systems drawn from a POWFRAT class represent improvements over generic systems in terms of storage and (very likely) speed.

5.1 Secure Stream Ciphers

We build a stream cipher called $\texttt{SPONGE}(q, d, n, r, (\eta_2, \ldots, \eta_d))$, which has form as given in Prop. 1. We specify a prime power q (usually a power of 2), positive integers n and r, a degree d. We have "update function" $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_n) : \mathbb{F}_q^n \to \mathbb{F}_q^n$ and "output filter" $\mathbf{P} = (P_1, P_2, \ldots, P_r) : \mathbb{F}_q^n \to \mathbb{F}_q^r$. We still do $\mathbf{y}_n = \mathbf{P}(\mathbf{x}_n)$ [output]; $\mathbf{x}_{n+1} = \mathbf{Q}(\mathbf{x}_n)$ [transition].

This time, every polynomial is of degree d. affine (constant and linear) term or coefficient are still uniformly random. But terms of each degree are selected according to different densities of terms, such that **but the degree-i terms are sparse to the point of having only η_i terms.**

The difference between Eq. 1 and Eq. 2, which governs the maximum provable security levels we can get, affects our parameter choices quite a bit, as seen below.

By Eq. 2, if $L = \lambda n \lg q$ is the desired keystream length, the looseness factor T'/T is roughly

$$\frac{2^{15} q^6 (L/\epsilon)^5}{n^4 \lg^5 q} \lg^2 \left(\frac{2qL}{\epsilon \lg q}\right)$$

If we let $q = 16$, $r = n$, want a safety level of $T = 2^{80}$ multiplications, $L = 2^{40}$ bits between key refreshes, and can accept $\epsilon = 10^{-2}$, then $T' \lesssim 2^{354}/n^4$. We propose the following instances:

- $\texttt{SPONGE}$ using $q = 2$, $n = r = 208$, $d = 3$ (cubics), with 20 cubic terms each equation. Preliminary tests achieve 11744 cycles/byte. The expected complexity for solving 208 variables and 416 equations is $\sim 2^{224}$ (by brute-force trials, which is much faster than XL here), which translates to a 2^{82} proven security level.
- $\texttt{SPONGE}$ using $q = 16$, $d = 3$ (cubics), $n = r = 160$, 20 quadratic and 15 cubic terms per equation. Projected XL degree is 54, storage requirement is 2^{184} bytes. T' is about 2^{346} multiplications, which guarantees $\gtrsim 2^{88}$ multiplications security. This runs at 6875 cycles/byte.
- $\texttt{SPONGE}$ using $d = 4$ (quartics), $n = r = 108$, 20 quadratic, 15 cubic, and 10 quartic terms per equation. Projected XL degree is 65, storage requirement is 2^{174} bytes. T' is about 2^{339} multiplications guaranteeing $\gtrsim 2^{81}$ multiplications security at a preliminary 5541 cycles/byte.

5.2 Comparisons: A Case for SPONGE

All modern-day microprocessor are capable of doing 64-bit arithmetic at least, and there is a natural way to implement QUAD that runs very fast over $\mathbb{F}_2$, limited only by the ability to stream data. However, as number of variables goes up, the storage needed for QUAD goes up cubically, and for parameter choices that are secure, the dataset overflows the caches of even an Intel Core 2. That seems to be what slows down $\texttt{QUAD}(2, 320, 320)$ — tests on a borrowed ia64 (6MB cache) server shows that it is almost exactly the same speed as the $\texttt{SPONGE}(2, 3, 208, 208, [480, 20])$. Looking at the numbers, it seems that the idea of specializd polynomials is a good complement to the approach of using polynomial maps for symmetric primitives introduced by Berbain-Gilbert-Patarin.

We hasten to add that our programming is quite primitive, and may not match the more polished implementations (e.g., [5]). We are still working to improve our programming and parameter choices. Also, in hardware implementations, the power of sparsity should be even more pronounced.

Stream Cipher	Block	Storage	Cycles/Byte	Security Level
SPONGE (2,3,208,208,[480,20])	208b	0.43 MB	11744	2^{82} Proven
SPONGE (16,4,108,108,[20,15,10])	864b	48 kB	5541	2^{80} Proven
QUAD (2,320,320)	320b	3.92 MB	13646	2^{82} Proven
QUAD (2,160,160)	160b	0.98 MB	2081	2^{140} Best Attack
SPONGE (16,4,32,32,[10,8,5])	128b	8.6 kB	1244	2^{152} Best Attack

Table 3: Point-by-Point, SPONGE vs. QUAD on a K8 or C2

5.3 For Possible Use in Hashes

In [7] Billet *et al* proposes to use two-staged constructions with a random 192-bit to 464-bit expanding quadratic map followed by a 464-bit to 384-bit quadratic contraction. They show that in general a PRNG followed by a one-way compression function is a one-way function.

In [13] the same construction is proposed but with SRQ quadratics and no proof. Now we see that the abovementioned results from [7] and Prop. 6, which justify the design up to a point. This is an area that still takes some study, and perhaps require extra ideas, such as having a hybrid construction with a sparse polynomial expansion stage and a different kind of contraction stage.

Acknowledgements

We would like to thank Jean-Philippe Aumasson for his helpful comments. Also thanks goes to Taiwan Information Security Center and the Taft Foundation for support,

References

[1] J.-P. Aumasson and W. Meier. Analysis of multivariate hash functions. In *Proc. ICISC*, LNCS, 2007. to appear, cf. `http://www.131002.net/files/pub/AM07.pdf`.

[2] G. V. Bard, N. T. Courtois, and C. Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over gf(2) via sat-solvers. Cryptology ePrint Archive, Report 2007/024, 2007. `http://eprint.iacr.org/`.

[3] 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 INRIA report RR-5049.

[4] M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic expansion of the degree of regularity for semi-regular systems of equations. In P. Gianni, editor, *MEGA 2005 Sardinia (Italy)*, 2005.

[5] C. Berbain, O. Billet, and H. Gilbert. Efficient implementations of multivariate quadratic systems. In *Proc. SAC 2006.* Springer, in press, dated 2006-09-15.

[6] 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.

[7] O. Billet, M. J. B. Robshaw, and T. Peyrin. On building hash functions from multivariate quadratic equations. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 82–95. Springer, 2007.

[8] L. Blum, M. Blum, and M. Shub. Comparison of two pseudo-random number generators. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *CRYPTO'82*, pages 61–78, New York, 1983. Plenum Press.

[9] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Innsbruck, 1965.

[10] N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Bart Preneel, editor, Springer, 2000. Extended Version: `http://www.minrank.org/xlfull.pdf`.

[11] N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Yuliang Zheng, editor, Springer, 2002.

[12] C. Diem. The XL-algorithm and a conjecture from commutative algebra. In *Advances in Cryptology — ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 323–337. Pil Joong Lee, editor, Springer, 2004. ISBN 3-540-23975-8.

[13] J. Ding and B.-Y. Yang. Multivariate polynomials for hashing. In *Inscrypt*, LNCS. Springer, 2007. to appear, cf. `http://eprint.iacr.org/2007/137`.

[14] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139:61–88, June 1999.

[15] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *International Symposium on Symbolic and Algebraic Computation — ISSAC 2002*, pages 75–83. ACM Press, July 2002.

[16] 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.

[17] R. Gennaro. An improved pseudo-random generator based on the discrete logarithm problem. *Journal of Cryptology*, 18:91–110, 2000.

[18] O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: The highly noisy case. *SIAM Journal on Discrete Mathematics*, 13(4):535–570, Nov. 2000.

[19] N. Koblitz and A. Menezes. Another look at "provable security". ii. In R. Barua and T. Lange, editors, *INDOCRYPT*, volume 4329 of *Lecture Notes in Computer Science*, pages 148–175. Springer, 2006.

[20] D. Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *EUROCAL 83*, volume 162 of *Lecture Notes in Computer Science*, pages 146–156. Springer, March 1983.

[21] L. Levin and O. Goldreich. A hard-core predicate for all one-way functions. In D. S. Johnson, editor, *21th ACM Symposium on the Theory of Computing — STOC'89*, pages 25–32. ACM Press, 1989.

[22] T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In *Advances in Cryptology — EUROCRYPT 1988*, volume 330 of *Lecture Notes in Computer Science*, pages 419–545. Christoph G. Günther, editor, Springer, 1988.

[23] H. Raddum and I. Semaev. New technique for solving sparse equation systems. Cryptology ePrint Archive, Report 2006/475, 2006. `http://eprint.iacr.org/`.

[24] I. Semaev. On solving sparse algebraic equations over finite fields ii. Cryptology ePrint Archive, Report 2007/280, 2007. `http://eprint.iacr.org/`.

[25] R. Steinfeld, J. Pieprzyk, and H. Wang. On the provable security of an efficient rsa-based pseudorandom generator. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 194–209. Springer, 2006. formerly ePrint 2006/206.

[26] B.-Y. Yang and J.-M. Chen. All in the XL family: Theory and practice. In *ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 67–86. Springer, 2004.

[27] B.-Y. Yang and J.-M. Chen. Theoretical analysis of XL over small fields. In *ACISP 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 277–288. Springer, 2004.

[28] B.-Y. Yang, O. C.-H. Chen, D. J. Bernstein, and J.-M. Chen. Analysis of `QUAD`. In *Fast Software Encryption — FSE 2007*, volume to appear of *Lecture Notes in Computer Science*, pages 302–319. Alex Biryukov, editor, Springer, 2007. workshop record available.

A Proof of Prop. 1

Proof. We introduce hybrid probability distributions $D_i(\mathbf{S})$ over K^L ($L := \lambda r$):
For $0 \le i \le \lambda$ respectively associate with the random variables

$$t^i(\mathbf{S}, \mathbf{x}) := \Big(\mathbf{w}_1,\, \mathbf{w}_2, \ldots, \mathbf{w}_i,\, \mathbf{P}(\mathbf{x}),\, \mathbf{P}(\mathbf{Q}(\mathbf{x})), \ldots,\, \mathbf{P}(\mathbf{Q}^{\lambda-i-1}(\mathbf{x}))\Big)$$

where the $\mathbf{w}_j$ and $\mathbf{x}$ are random independent uniformly distributed vectors in K^n and we use the notational conventions that $(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_i)$ is the null string if $i = 0$, and that

$$\Big(\mathbf{P}(\mathbf{x}),\, \mathbf{P}(\mathbf{Q}(\mathbf{x})), \ldots,\, \mathbf{P}(\mathbf{Q}^{\lambda-i-1}(\mathbf{x}))\Big)$$

is the null string if $i = \lambda$. Consequently $D_0(\mathbf{S})$ is the distribution of the L-unit keystream and $D_\lambda(\mathbf{S})$ is the uniform distribution over K^L. We denote by $p_i(\mathbf{S})$ the probability that A accepts a random L-long sequence distributed according to $D_i(\mathbf{S})$, and p_i the mean value of $p_i(\mathbf{S})$ over the space of sparse polynomial systems $\mathbf{S}$. We have supposed that algorithm A distinguishes between $D_0(\mathbf{S})$ and $D_\lambda(\mathbf{S})$ with advantage , in other words that $|p_0 - p_\lambda| \ge \epsilon$.

Algorithm B works thus: on input $(\mathbf{x}_1, \mathbf{x}_2) \in K^{n+r}$ with $\mathbf{x}_1 \in K^r$, $\mathbf{x}_2 \in K^n$, it selects randomly an i such that $0 \leq i \leq \lambda - 1$ and constructs the L-long vector

$$t(\mathbf{S}, \mathbf{x}_1, \mathbf{x}_2) := (\mathbf{w}_1,\, \mathbf{w}_2, \ldots,\, \mathbf{w}_i, \mathbf{x}_1, \mathbf{P}(\mathbf{x}_2),\, \mathbf{P}(\mathbf{Q}(\mathbf{x}_2)), \ldots,\, \mathbf{P}(\mathbf{Q}^{\lambda-i-1}(\mathbf{x}_2))).$$

If $(\mathbf{x}_1, \mathbf{x}_2)$ is distributed accordingly to the output distribution of $\mathbf{S}$, i.e. $(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{S}(\mathbf{x}) = (\mathbf{P}(\mathbf{x}), \mathbf{Q}(\mathbf{x}))$ for a uniformly distributed value of $\mathbf{x}$, then

$$t(\mathbf{S}, \mathbf{x}_1, \mathbf{x}_2) := \Big(\mathbf{w}_1,\, \mathbf{w}_2, \ldots, \mathbf{w}_i,\, \mathbf{P}(\mathbf{x}),\, \mathbf{P}(\mathbf{Q}(\mathbf{x})), \ldots,\, \mathbf{P}(\mathbf{Q}^{\lambda-i-1}(\mathbf{x}))\Big)$$

is distributed according to $D_i(\mathbf{S})$. Now if $(\mathbf{x}_1, \mathbf{x}_2)$ is distributed according to the uniform distribution, then

$$t(\mathbf{S}, \mathbf{x}_1, \mathbf{x}_2) = \Big(\mathbf{w}_1,\, \mathbf{w}_2, \ldots, \mathbf{w}_i,\, \mathbf{x}_1,\, \mathbf{P}(\mathbf{x}_2),\, \mathbf{P}(\mathbf{Q}(\mathbf{x}_2)), \ldots,\, \mathbf{P}(\mathbf{Q}^{\lambda-i-2}(\mathbf{x}_2))\Big)$$

which is distributed according to $D_{i+1}(\mathbf{S})$. To distinguish between the output of $\mathbf{S}$ from uniform, algorithm B calls A with inputs $(\mathbf{S}, t(\mathbf{S}, \mathbf{x}_1, \mathbf{x}_2))$ and returns that same return value. Hence

$$\begin{aligned} & \left| \Pr_{\mathbf{S},\mathbf{x}}(B(\mathbf{S}, \mathbf{S}(\mathbf{x})) = 1 - \Pr_{\mathbf{S},\mathbf{x}}(B(\mathbf{S}, \mathbf{S}(\mathbf{x}_1, \mathbf{x}_2)) = 1 \right| \\ = & \left| \frac{1}{\lambda} \sum_{i=0}^{\lambda-1} p_i - \frac{1}{\lambda} \sum_{i=0}^{\lambda-1} p_i \right| = \frac{1}{\lambda} \left| p_0 - p_\lambda \right| \geq \frac{\epsilon}{\lambda}. \end{aligned}$$

$\square$

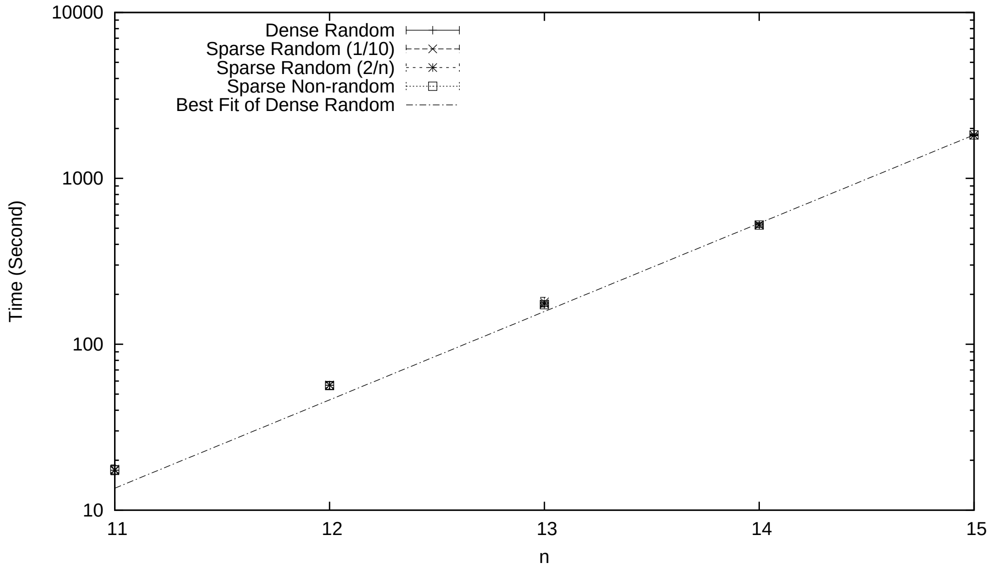

Figure 1: “Sparsely $2n \to 3n$ $\mathbb{F}_2$ quadratics” in MAGMA

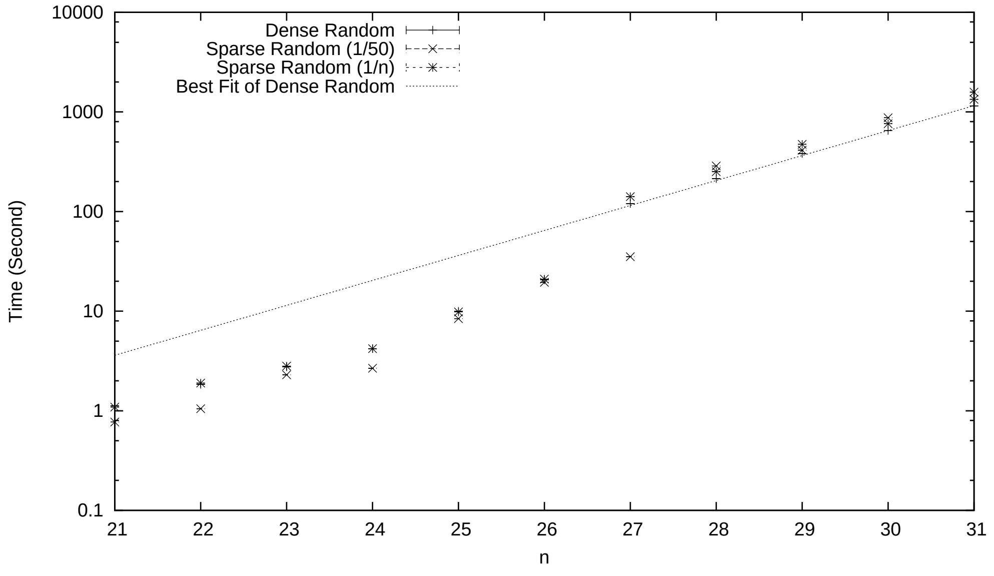

Figure 2: “Sparsely $n \to 2n$ $\mathbb{F}_2$ quadratics” in MAGMA