QUAD: a Practical Stream Cipher with Provable Security

Côme Berbain[1], Henri Gilbert[1], Jacques Patarin[2]

[1] France Telecom Research and Development,
38-40 rue du Général Leclerc, F-92794 Issy-les-Moulineaux, France.
[2] Université de Versailles,
45 avenue des Etats-Unis, F-78035 Versailles cedex, France.

Abstract. We introduce a practical synchronous stream cipher with provable security named QUAD. The cipher relies on the iteration of a multivariate quadratic system of m equations in $n < m$ unknowns over a finite field. The security of QUAD is provably reducible to the conjectured intractability of the MQ problem, namely solving a multivariate system of quadratic equations.

1 Introduction

Stream ciphers represent, together with block ciphers, one of the two main classes of symmetric encryption algorithms. Generally speaking stream ciphers seem to allow faster encryption and to require lower computing resources than block ciphers, and the fastest known stream ciphers (e.g. SEAL, RC4, SNOW 2.0, the Shrinking Generator) are indeed significantly faster in software than an efficient block cipher such as AES [27]. However, the design of secure stream ciphers is not currently as well understood as the design of secure block ciphers. The state of the art of the cryptanalysis of stream ciphers, e.g. LFSR based stream ciphers, has evolved significantly over the last ten years and many recent proposals still suffer from security weaknesses. This is illustrated by the fact that none of the candidate stream ciphers submitted to the call for cryptographic primitives of the European project NESSIE were retained since attacks more efficient than exhaustive search were found for all candidates during the evaluation period. This is also illustrated by the ongoing eSTREAM [11] call for stream ciphers proposals of the European project ECRYPT. Stream ciphers complying with two main profiles have been called for, namely stream ciphers allowing much faster software encryption than existing block ciphers (profile 1) and stream ciphers requiring much lower resources for hardware implementation than existing block ciphers (profile 2). However, more than one third of the 34 submitted stream ciphers, which cover these two profiles, have already been shown to be insecure.

[0] The work described in this paper has been supported by the French Ministry of Research RNRT X-CRYPT project and by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

Our aim is to propose a practical cipher with unusually strong security arguments. The novel stream cipher we propose was designed with another tradeoff between security, speed and computing resources than reflected by the eSTREAM profiles 1 and 2. We slightly relax the requirements on speed and computing resources, i.e. we only require a stream cipher that is sufficiently fast for most practical purposes. But we introduce an unusually strong security requirement for symmetric cryptography (which is out of reach of the current state of the art for block ciphers), namely that the security of the cipher be provably reducible to the conjectured intractability of a well-known and studied mathematical problem. The security of the novel stream cipher is provably reducible to the intractability of the MQ problem [15], which consists of finding a solution (if any) to a multivariate quadratic system of m quadratic equations in n variables over a finite field $GF(q)$, typically $GF(2)$. The MQ problem is conjectured to be difficult for suitably chosen values of n and m. In general the associated decision problem is known to be NP-complete even in the case where the considered field is $GF(2)$, and moreover no efficient algorithm to solve MQ with a significant success probability is known to exist for sufficiently large values of n (say $n > 100$) when the quadratic equations are randomly chosen. The implementation complexity of our stream cipher is reasonable and the encryption speed (4.6 Mbit/s for a software implementation in C on a standard PC), though lower than AES, is more than sufficient for many practical purposes.

Constructing a provably secure stream cipher is not a novel topic. However, designing a practical provably secure stream cipher is an open problem. Following seminal work by Shamir, Blum and Micali [4], Yao [31], Levin and Goldreich [25] in the 80's, considerable research effort has been dedicated to the construction of provably secure pseudo-random number generators (PRNG) that expand a short seed (e.g. a key) into a larger bit string. This can be used as the keystream for encryption purposes. Available security results typically state that if the iterated function underlying the construction of a number generator satisfies suitable one-wayness properties, then the generator is a secure PRNG, i.e. its L-bit output is computationally indistinguishable from the uniform distribution over $\{0,1\}^L$. This research effort has led to remarkable generic results, e.g. the proof by Impagliazzo, Levin, Luby and Håstad [21] that a secure PRNG can be constructed based upon any one way function (OWF). It has also led to provably secure PRNG constructions based on the conjectured intractability of specific problems. The first provably secure PRNG was introduced by Blum and Micali [4] and relates the security of the PRNG to the one-wayness of exponentiation modulo a prime number. The provably secure PRNG proposed by L. Blum, M. Blum and M. Shub [3] exploits the conjectured intractability of quadratic residuosity modulo Blum integers. Alexi, Chor, Goldreich and Schnorr proposed a PRNG construction with security that relies upon the RSA assumption. Impagliazzo and Naor [24] and Fisher and Stern [13] proposed PRNG constructions respectively relying on the difficulty of the subset sum problem and of the syndrome decoding problem. Even in the case of specific constructions, current provably secure PRNGs are too inefficient to provide a practical stream cipher. This is due to the fact that the function iterated by the PRNG is usually too computationally expensive, and that only a restricted number of bits can be produced at each iteration (this number is generally at most proportional to the logarithm of the input length n of the iterated function). However some efforts have been made to improve the constructions. A first idea is to extract more than $\log n$ bits at each round. Constructions based on the discrete logarithm problem makes it possible to extract $n - \log(n)$ bits at each iteration instead of $\log n$. Despite this fact, the fastest generator based on discrete logarithm proposed by Gennaro [16] is still impractical: it requires 350 multiplications of 3000-bit numbers to extract 2775 bits. Another problem for which it is possible to extract more than $\log n$ bits is the syndrome decoding problem. A PRNG has been proposed by Fisher and Stern in [13] but the number of extracted bits, although higher than $\log n$, is still small for practical values of n. Another recently proposed idea is to replace a slow iterated function by some primitive which is much faster to compute. Håstad and Näslund proposed BMGL [30], a stream cipher with security that relies on the difficulty of extracting the key from one plaintext ciphertext couple in AES. Their practical construction consists of iterating AES and extracting $\log n$ bits at each round. This cipher is fast, especially compared to other provably secure ciphers, but its security relies only on the security of the AES and not on a simple and well-studied mathematical problem.

On the contrary, MQ is a simple and well-studied mathematical problem and the values of n for which the problem is difficult are small (around 100 bits), particularly when compared to discrete logarithm or factorisation, where at least 1024 bits are required. Furthermore a large number of bits (e.g. $\frac{n}{2}$) bits or even more can be produced at each iteration.

This paper is organized as follows. We first give some preliminary background on the status of the MQ problem and basic security definitions in a concrete (non asymptotic) security model. Then we describe the new construction and give a formal proof of security for the associated keystream generator. Finally we give the encryption speed of software implementations of our stream cipher.

2 Preliminaries

2.1 Multivariate Quadratic Systems

We consider a finite field $GF(q)$. A multivariate quadratic equation (or equivalently a multivariate quadratic form) in n variables over $GF(q)$ is a polynomial of degree at most 2 in $GF(q)[x_1, \ldots, x_n]$ which can be written as

$$Q(x) = \sum_{1 \le i \le j \le n} \alpha_{i,j} x_i x_j + \sum_{1 \le i \le n} \beta_i x_i + \gamma$$

with all the coefficients $\alpha_{i,j}$, β_i, and γ in $GF(q)$. In the particular case $q = 2$, which will be considered in the sequel, the monomial forms $x_i x_i$ and x_i are equal.

It is easy to see that the set $\mathcal{Q}$ of multivariate quadratic forms in n variables is an N-dimensional vector space over $GF(q)$, where $N = \frac{n(n+3)}{2} + 1$ if $q \neq 2$ and $N = \frac{n(n+1)}{2} + 1$ if $q = 2$. A basis of this vector space is given by the $N - 1$ distinct monomial functions of degree 1 or 2 and the constant form 1. Any element of $\mathcal{Q}$ can be represented by the N-tuple of its $GF(q)$ coefficients in this basis. Throughout the rest of this paper, we mean by a randomly chosen quadratic form in n unknowns the quadratic form represented in the above basis by a uniformly and independently drawn N-tuple of $GF(q)$ coefficients.

A multivariate quadratic system S of m quadratic equations in n variables over $GF(q)$ is a set $(Q_1, \ldots Q_m)$ of m quadratic equations in n variables over $GF(q)$. In the sequel, we mean by a randomly chosen system of m quadratic form in n unknowns, n independently and randomly chosen quadratic forms. Such a system is represented by mN uniformly and independently drawn $GF(q)$ coefficients.

A quadratic form Q over n unknowns over $GF(2)$ is called non degenerate iff Q is not equivalent to a quadratic form in strictly fewer than n linear combinations of the n input variables. There exists a polynomial time algorithm to check whether a given quadratic form is non degenerate and more generally to compute the so-called rank of a quadratic form [26]. The number of solutions of the quadratic equation $Q = 0$ associated with a non degenerate quadratic form Q over n unknowns is either 2^{n-1} or $2^{n-1} + 2^{\frac{n-2}{2}}$ or $2^{n-1} - 2^{\frac{n-2}{2}}$ depending on the parity of n and the value of γ. Thus for sufficient large values of n, say $n > 100$, non degenerate quadratic forms are either perfectly balanced (odd n values) or have an undetectable bias (even n values).

2.2 Status of the MQ problem

We define the problem of solving simultaneous **multivariate quadratic** equations (**MQ problem**) as follows: given a multivariate quadratic system of m quadratic equations over $GF(q)$ $S = (Q_1, \ldots, Q_m)$, find a value $x \in GF(q)^n$, if any, such that $Q_i(x) = 0$ for all $1 \leq i \leq m$.

Depending on the respective values of n and m, instances of MQ can be either easy or very difficult to solve. For $m = 1$ the number of solutions is known [26] and it is quite easy to find one solution. When m is significantly smaller than n, that is for an underdefined quadratic system, finding a solution is easy [6]. In the opposite situation of an overdefined system ($m > n$) providing $N = \frac{n(n+1)}{2} + 1$ ($q = 2$ case) or $\frac{n(n+3)}{2} + 1$ ($q \neq 2$ case) linearly independent quadratic equations, or more generally when nearly N linearly independent quadratic equations are available, solving an MQ problem is easy by linearization. The total complexity is then only $O(n^6)$. However for general values of m and n the MQ problem is known to be NP-hard, even when restricted to quadratic equations over $GF(2)$ [15] [14] or over any finite field [28].

Moreover, what seems to make the MQ problem particularly well suited to cryptographic applications is that it is conjectured to be very difficult not only asymptotically and in worst case, but already for small suitably selected values of m and n and in terms of the average complexity of solving a random instance. The problem seems to be most difficult when m is close to n. For $m = n$ and $q = 2$ the complexity of the best known solving algorithms is $2^{n-O(\sqrt(n))}$ and thus rather close to the 2^n complexity of exhaustive search, and totally out of reach of existing computers for a random instance and n values larger than 100. Even when $q = 2$, $m = kn$ and $k > 1$ is small enough compared with $\frac{n}{2}$, the best known computer algebra algorithms such as XL [10] and improved variants of Buchbergers's Groebner basis computation algorithm such as Faugère's F4 and F5 algorithms [12] are exponential in n for a randomly chosen quadratic system. Much research has been dedicated in the past years to the above problem [9], [7]. Magali Bardet's PHD thesis [1] provides an accurate analysis of the complexity of the most efficient known Groebner basis computation algorithm for solving a random system of $m = kn$ equations in n unknowns. We will use some complexity estimates of [1] when discussing practical recommendations of the parameter values of our cipher.

Though we expect degenerate instances of the systems used in our construction leading to a weak stream cipher to be extremely unlikely, we suggest the following extra precaution when drawing these systems at random to provide some extra guaranties that some of the weakest instances are avoided: check that each quadratic equation is non degenerate or at least has a high rank value close to the one of a non degenerate form, and discard any quadratic equation which would not satisfy this condition. In order to discard a slightly larger subset of weak instance, one can also check that low weight linear combinations of the selected quadratic equations satisfy the above rank conditions. Also check that the obtained quadratic equations are linearly independent in $\mathcal{Q}$.

2.3 Basic Security Notions

All the security definitions used throughout this paper relate to the concrete (non asymptotic) security model. We are using the following basic security notions that we state here informally. Two probability distributions D_1 and D_2 over a finite set Ω are said to be **computationally distinguishable** with computing resources R and advantage ϵ if there exits a probabilistic testing algorithm A which on any input value $x \in \Omega$ outputs a binary answer "1" (accept) or "0" (reject) using computing resources at most R and satisfies

$$|Pr_{x \in D_1}\left(A(x) = 1\right) - Pr_{x \in D_2}\left(A(x) = 1\right)| \geq \epsilon.$$

Though this is not explicitly reflected in our notation, the above probabilities are not only taken over x values distributed according to D_1 or D_2, but also over the random choices of 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. When the computing resources R is not specified, we implicitly mean feasible computing resources (i.e. say less than 2^{80} simple operations).

Let n and L denote integers such that $L > n$. A n-bit to L-bit function G is said to be a **Pseudo Random Number Generator (PRNG)** if for a random n-bit input variable x selected according to the uniform law on $\{0,1\}^n$ the probability distribution of the random variable $G(x)$ is computationally indistinguishable from the uniform law over $\{0,1\}^L$.

3 QUAD: a New Stream Cipher

We now introduce the proposed stream cipher, named QUAD.

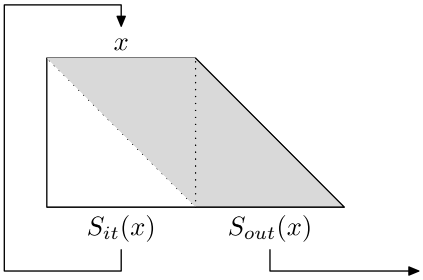
$S = (Q_1, \ldots, Q_{kn})$ denotes a multivariate quadratic system of kn randomly chosen equations in n variables over $GF(q)$, and S_0 and S_1 denote two (k times smaller) additional multivariate systems of n randomly chosen equations in n variables over $GF(q)$. S, S_0 and S_1 are fixed and publicly known. During the key and IV loading and the keystream generation, the internal register state is a $x = (x_1, \ldots, x_n)$ n-tuple of $GF(q)$ values.

3.1 Keystream Generation and Encryption

The keystream generation process simply consists in iterating the three following steps in order to produce $(k-1)n$ $GF(q)$ keystream values at each iteration.

- Compute the kn-tuple of $GF(q)$ values $S(x) = (Q_1(x), \ldots, Q_{kn}(x))$ where x is the current value of the internal state;
- Output the sequence $S_{out}(x) = (Q_{n+1}(x), \ldots, Q_{kn}(x))$ of $(k-1)n$ $GF(q)$ keystream values
- Update the internal state x with the sequence of n $GF(q)$ first generated values $S_{it}(x) = (Q_1(x), \ldots, Q_n(x))$

The maximal keystream sequence that may be generated with a single (key,iv) pair is L $GF(q)$ values. In order to encrypt a plaintext of length $l \leq L$ $GF(q)$ symbols, each of the first l $GF(q)$ values of the keystream sequence is added (using the $GF(q)$ addition) with the corresponding plaintext value.



3.2 Key and IV Setup

Before generating any keystream we need to initialize the internal state x, with the key K and the initialization vector IV, which are respectively represented by a sequence of $GF(q)$ elements of length $|K|$ and a binary sequence of $\{0,1\}$ values of length $|IV|$. We assume for the time being, for simplicity of the subsequent proofs [3] that $|K|$ is chosen exactly equal to n.

The initialization is done as follows : we use two carefully randomly chosen multivariate quadratic systems S_0 and S_1 of n equations over n unknowns. We initially set the internal state value x to the n bit value K. Then for each of the $|IV|$ bits IV_1 to $IV_{|IV|}$ of the IV value the internal state x is updated as follows: if $IV_i = 0$, x is replaced by the $GF(q)^n$ value $S_0(x)$; if $IV_i = 1$, x is replaced by the $GF(q)^n$ value $S_1(x)$. These $|IV|$ steps provide a key and IV dependent internal state value x. We then clock the cipher $|IV|$ additional times as described in section 3.1, but without outputting the keystream in order to further transform the internal state value x, and then enter the keystream generation mode to produce the keystream.

4 Security

We now give a proof that for a randomly chosen multivariate quadratic system our PRNG is secure. For simplicity of the proof we will work over $GF(2)$. The proof can be divided in three parts, which can be informally outlined as follows.

In the first part (Theorem 1), we prove that if the L-bit keystream sequence associated with a known fixed or randomly chosen system S of $m = kn$ quadratic equations and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ is distinguishable from the L-bit output of a perfectly uniform generator, then for a known random quadratic system S of $m = kn$ equations and an unknown randomly chosen input value $x \in \{0,1\}^n$, $S(x)$ is distinguishable from a random kn bit word.

In the second part (Theorem 2), we prove that if for a known randomly chosen quadratic system S and an unknown randomly chosen x, $S(x)$ is distinguishable from a random kn bit word then, for any n-bit to 1-bit quadratic form R (in particular any linear form R), one has the property that for a randomly chosen n bit value x, $R(x)$ can be predicted better than at random given $S(x)$.

In the third part (Theorem 3), we prove that, for a known fixed or randomly chosen S and a randomly chosen linear form R, $R(x)$ can be predicted better than at random given $S(x)$, then with non negligible probability a preimage of $S(x)$ can be efficiently computed given $S(x)$. Thus S is not strongly one way. This part is essentially a proof of Goldreich-Levin's theorem [25], in which a fast Walsh transform computation is used to get a tighter reduction.

[3] Note however that we will consider later on, in section 4.5, an extended key loading method allowing to set the key length to values strictly lower than n, for instance to $|K| = \frac{n}{2}$ if one wishes the key length to reflect the complexity of the best known attack.

4.1 Distinguishing the Keystream Allows to Distinguish the Output of a Random Quadratic System

Theorem 1 states that if one can distinguish the keystream of the generator based on the iteration of a quadratic system S from a random L-bit sequence, then one can distinguish the output of S from a random m-bit sequence. Though we consider a randomly chosen system S because we need distinguishing properties related to a random system for the sequel, the property we prove would also hold if we considered a fixed system S. Our proof is inspired by the proof given in [20] that a similar result holds for the generator based on iteration of any fixed n-bit to m-bit function, where $m > n$, but provides a tighter bound for the advantage.

Theorem 1. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced in time λT_S using λ iterations of our construction. Suppose there is an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm B that for a randomly chosen S distinguishes $S(x)$ corresponding to an unknown random input x, from a random value of size kn in time $T' = T + \lambda T_S$ with advantage $\frac{\epsilon}{\lambda}$.*

Proof. We introduce the hybrid probability distributions $D^i(S)$ over $\{0,1\}^L$xi. For $0 \leq i \leq \lambda$ respectively associated with the random variables

$$t^i(S,x) = (r_1, r_2, \ldots, r_i, S_{out}(x), S_{out}(S_{it}(x)), \ldots, S_{out}(S_{it}^{\lambda-i-1}(x)))$$

where the r_j and x are random independent uniformly distributed values of $\{0,1\}^n$ and the notational conventions that $(r_1, r_2, \ldots, r_i)$ is the null string if $i = 0$ and that $(S_{out}(x), \ldots, S_{out}(S_{it}^{\lambda-i-1)}(x)))$ is the null string if $i = \lambda$. Consequently $D^0(S)$ is the distribution of the L-bit keystream and $D^\lambda(S)$ is the uniform distribution over $\{0,1\}^L$. We denote by $p^i(S)$ the probability that A accepts a random L-bit sequence distributed according to $D^i(S)$, and denote by p^i the average value of $p^i(S)$ over the $(k-1)n(n\frac{(n+1)}{2}+1)$-dimensional vector space of quadratic systems S. We have supposed that algorithm A distinguishes between $D^0(S)$ and $D^\lambda(S)$ with advantage ϵ, in other words that $|p^0 - p^\lambda| \geq \epsilon$. Algorithm B works as follows : on input $(x_1, x_2) \in \{0,1\}^{kn}$ with $x_1 \in \{0,1\}^n$ and $x_2 \in \{0,1\}^{(k-1)n}$, it selects randomly an i such that $0 \leq i \leq \lambda - 1$ and constructs the L-bit vector

$$t(S, x_1, x_2) = (r_1, r_2, \ldots, r_i, x_2, S_{out}(x_1), S_{out}(S_{it}(x_1)), \ldots, S_{out}(S_{it}^{\lambda-i-2}(x_1))).$$

If (x_1, x_2) is distributed accordingly to the output distribution of S, i.e. $(x_1, x_2) = S(x) = (S_{it}(x), S_{out}(x))$ for a uniformly distributed value of x, then

$$t(S, x_1, x_2) = (r_1, r_2, \ldots, r_i, S_{out}(x), S_{out}(S_{it}(x)), \ldots, S_{out}(S_{it}^{\lambda-i-1}(x)))$$

is distributed according to $D^i(S)$. Now if (x_1, x_2) is distributed according to the uniform distribution, then

$$t(S, x_1, x_2) = (r_1, r_2, \ldots, r_i, x_2, S_{out}(x_1), S_{out}(S_{it}(x_1)), \ldots, S_{out}(S_{it}^{\lambda-i-2}(x_1))).$$

Thus $t(S, x_1, x_2)$ is distributed according to $D^{i+1}(S)$. In order to distinguish the output distribution of S from the uniform law, algorithm B calls algorithm A with inputs $(S, t(S, x_1, x_2))$ and returns the value returned by A. Thus

$$|Pr_{S,x}(B(S, S(x)) = 1) - Pr_{S,x_1,x_2}(B(S, (x_1, x_2)) = 1)|$$
$$= |\frac{1}{\lambda}\sum_{i=0}^{\lambda-1} p^i - \frac{1}{\lambda}\sum_{i=1}^{\lambda} p^i| = \frac{1}{\lambda}|p^0 - p^\lambda| \geq \frac{\epsilon}{\lambda}.$$

Thus B distinguishes the output distribution of S from the uniform distribution with probability at least $\frac{\epsilon}{\lambda}$ in time $T + \lambda T_S$.

4.2 Distinguishing the Output of a Random Quadratic System Allows to Predict any Quadratic Equation

Now we prove that if there exists a distinguisher between $S(x)$ and a kn-bit random value such as the one considered in the above theorem, it can be converted into an algorithm that predicts the result of any quadratic polynomial (and in particular any linear polynomial).

Theorem 2. *Suppose there is an algorithm A that, given a randomly chosen known multivariate quadratic system S of kn equations in n unknowns, distinguishes $S(x)$, where x is an unknown random input value, from a random string of length kn with advantage at least ϵ and in time T. Then there is an algorithm B that, given a randomly chosen quadratic system S of kn equations in n unknowns, any n-bit to 1-bit quadratic form R, and $y = S(x)$ where x is a random input value, predicts $R(x)$ with success probability at least $\frac{1}{2} + \frac{\epsilon}{4}$ using at most $T' = T + 2T_S$ operations.*

Proof. We first show that there exists an algorithm A' which returns 1 on input $(S, S(x))$ with probability at least $\frac{1}{2} + \frac{\epsilon}{2}$ and returns 1 on input (S, u) for some random u with probability $\frac{1}{2}$: if the acceptance probability of A is larger (by at least ϵ) on an input $(S, S(x))$ than on a random input. Then it suffices to consider A' which on input (S, r) either returns $A(S, r)$ or draws a random value u and returns $1 - A(S, u)$ with probability $\frac{1}{2}$ for each case. In the opposite situation, it suffices to consider A' which on input (S, r) either returns $1 - A(S, r)$ or draws a random value and returns $A(S, u)$ with probability $\frac{1}{2}$ for each case.

Algorithm B works as follows. On input $S = (Q_1, \ldots Q_{kn})$,R and a kn-bit value y, B selects a random kn-bit vector $a = (a_1, \ldots, a_{kn})$ and a random bit b, which represents an hypothesis for $R(x)$. Then it computes for all i from 1 to kn the quadratic equation $P_i = Q_i + (a_i \cdot R)$. All the equations P_i form the quadratic system S'. Then B invokes the algorithm A' with input the new quadratic system S' and the value $y + (b \cdot a)$. Finally B returns what A' returns.

Now assume that $y = S(x)$ where x is an unknown random value. We have $\forall i, x, P_i(x) = Q_i(x) + (a_i \cdot R(x)) = y_i + (a_i \cdot R(x))$.

Suppose b is really equal to $R(x)$, then $S'(x) = y + (b \cdot a)$ so the distinguisher A' has been fed with the random quadratic system $S' = (P_1, \cdots, P_{kn})$ and $S'(x)$:

$$Pr_{S,x \in U_n}(B(S, S(x), R) = R(x)) = Pr_{S',x \in U_n}(A'(S', S'(x)) = 1) \geq \frac{1}{2} + \frac{\epsilon}{2}.$$

On the contrary, suppose b is not equal to $R(x)$, then $S'(x) = y + ((1 + b) \cdot a) = (y + (b \cdot a)) + a$. Thus there is an error of a on the value furnished to A' as compared with $S'(x)$. Because a is randomly chosen, we have:

$$\begin{aligned} Pr_{S,x \in U_n}(B(S, S(x), R) = R(x)) &= Pr_{S',x \in U_n}(A'(S', S'(x) + a) = 0) \\ &= Pr_{S',t \in U_{kn}}(A'(S', t) = 0) = \frac{1}{2} \end{aligned}$$

Thus we have:

$$Pr_{S,x \in U_n}(B(S, S(x), R) = R(x)) \geq \frac{1}{2}\left(\left(\frac{1}{2} + \frac{\epsilon}{2}\right) + \frac{1}{2}\right) = \frac{1}{2} + \frac{\epsilon}{4}$$

The total running time of B is at most $T + 2T_S$, since computing the kn P_i requires for each i to compute all the $\frac{n(n-1)}{2}$ monomials of Q_i and R, which does not cost more than two evaluations of the system for some entry.

4.3 A Linear Form is a Hard Core Bit for any One Way Function

Now we show that if for a fixed or random quadratic system S and more generally any fixed or random n-bit to m-bit function f there exists a predictor such as the one considered in the former theorem, i.e. a predictor allowing, given an n-bit to 1-bit linear form R, to predict $R(x)$ with a success probability (over all S and x values) strictly larger than $\frac{1}{2}$, then a preimage of $S(x)$ (resp. f(x)) can be efficiently computed, so that S (resp f) is not one way. This result is the Goldreich-Levin theorem [25] that we prove as to get a tight reduction. Before proving the theorem, which relates to the computation, given the image $S(x)$ or $f(x)$ for a random unknown value x and a random system S, of a list containing x, we first establish a lemma representing the technical core of the proof in which a fixed (unknown) value of x is considered. Our proofs are inspired by the simplified treatment of the original Goldreich-Levin proofs developed by Rackoff, Goldreich[18] and Bellare [2], and also by the proofs provided by Håstad and Näslund in their BMGL paper [30].

Lemma 1. *Let us denote by x a fixed unknown n-bit value and denote by f a fixed n-bit to m-bit function. Suppose there exists an algorithm B that given the value of $f(x)$ allows to predict the value of any linear equation R over n unknowns with probability $\frac{1}{2} + \epsilon$ over R, using at most T operations. Then there exists an algorithm C, which given $f(x)$ produces in time at most T' a list of at* *most $4n^2\epsilon^{-2}$ values such that the probability that x appears in this list is at least* $1/2$.

$$T' = \frac{2n^2}{\epsilon^2}\left(T + \log\left(\frac{2n}{\epsilon^2}\right) + 2\right) + \frac{2n}{\epsilon^2}T_f$$

The proof of lemma 1 is given in the Appendix. Lemma 1 applies to a fixed x and a fixed system S (or a fixed n-bit to m-bit function f). However, the success probability of the predictor of Theorem 2 is taken over all $(x,\ S)$ pairs for any linear form R. Consequently, we need a theorem allowing us to exploit the existence of such a predictor to show the applicability of the lemma to a non-negligible fraction of (x, S) pairs.

Theorem 3. *Suppose there is an algorithm B, that given a randomly chosen quadratic system S of m quadratic equations, a randomly chosen n-bit to 1-bit quadratic form R and the image $S(x)$ of a randomly chosen (unknown) n-bit value x, predicts the value of $R(x)$ with probability at least $\frac{1}{2} + \epsilon$ over all possible (x, S, R) triplets using T operations. Then there is an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x produces a preimage of $S(x)$ with probability at least $\epsilon/2$ (over all possible values of x and S) in time T' .*

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

Proof. The assumption about algorithm B can be written as

$$Pr_{(x,S,R)\in\{0,1\}^{n+mN+n}}\left\{B(S, S(x), R) = R(x)\right\} \geq \frac{1}{2} + \epsilon.$$

It results that for a fraction at least ϵ of all the (x, S) pairs one has

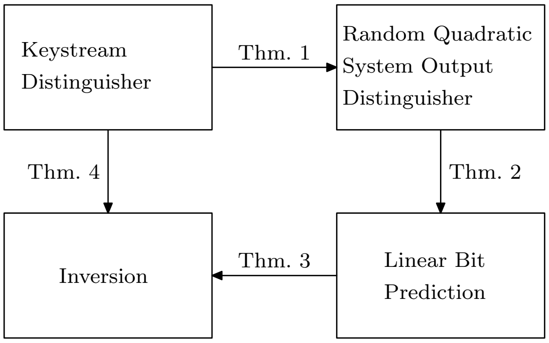
$$Pr_{R\in\{0,1\}^{n}}\left\{B(S, S(x), R) = R(x)\right\} \geq \frac{1}{2} + \frac{\epsilon}{2}.$$

Otherwise, there would exist a fraction at least $1 - \epsilon$ of the (x, S) pairs which associated prediction probability over the R values would be strictly less than $\frac{1}{2} + \frac{\epsilon}{2}$, and therefore $Pr_{(x,S,R)\in\{0,1\}^{n+mN+n}}\left\{B(S, S(x), R) = R(x)\right\}$ would be upper bounded by $(1-\epsilon)(\frac{1}{2}+\frac{\epsilon}{2})+\epsilon = \frac{1}{2}+\epsilon-\epsilon^2$, which contradicts the assumption about Algorithm B.

Thus for a fraction at least ϵ of all the (x, S) pairs the conditions of lemma 1 are met and algorithm C of the lemma provides a preimage of $S(x)$ with probability at least $1/2$.

4.4 A Security Proof for the Proposed PRNG

Now it is easy to see that if we sequentially apply theorems 1, 2, and 3, we obtain the following reduction theorem, which states that if, for a random system and a random initial value, the L-bit keystream sequence was distinguishable from a random L-bit sequence then there would exist an efficient algorithm allowing to find a preimage of the image of a random n-bit input value by a random quadratic n-bit to m-bit system, which for suitably chosen values of n would contradict the assumptions made in Section 2 on the difficulty of solving MQ.



Theorem 4. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced by in time λT_S using λ iterations of our construction. Suppose there exists an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x by a randomly chosen n-bit to m-bit quadratic system S produces a preimage of $S(x)$ with probability at least $\frac{\epsilon}{2^3\lambda}$ over all possible values of x and S in time upper bounded by T'.*

$$T' = \frac{2^7 n^2 \lambda^2}{\epsilon^2}\left(T + (\lambda+2)T_S + \log\left(\frac{2^7 n\lambda^2}{\epsilon^2}\right) + 2\right) + \frac{2^7 n\lambda^2}{\epsilon^2} T_S$$

Proof. Theorems 1 to 3 state that if an algorithm X exists, then another algorithm Y exists. In the case of Theorem 1, the resulting algorithm Y can be directly play the role of algorithm X in Theorem 2. In the case of Theorem 2, the resulting algorithm Y, named algorithm B, has the property

$$\forall R \in \{0,1\}^N Pr_{(x,S)\in\{0,1\}^{n+mN}}\left\{B(S, S(x), R) = R(x)\right\} \geq \frac{1}{2} + \frac{\epsilon}{4}$$

which implies

$$Pr_{(x,S,R)\in\{0,1\}^{n+mN+N}}\left\{B(S, S(x), R) = R(x)\right\} \geq \frac{1}{2} + \frac{\epsilon}{4}$$

Thus algorithm Y can play the role of algorithm X in Theorem 3, and if we compose the distinguishing probability and complexity expressions of the three concatenated theorems, we obtain the claimed distinguishing probability and complexity bounds.

Discussion: Theorem 4 above relates to the keystream generation part of QUAD, i.e. to the expansion of a randomly chosen initial state into the keystream and does not include the key and IV loading for deriving the initial state. Moreover it does not guarantee the strength of a particular instance of QUAD associated with a fixed system S but (informally) it shows that if MQ is intractable then most instances of QUAD are secure.

4.5 Specifying the Parameter Values for QUAD

We now propose concrete parameters n, k, L, $|K|$ and $|IV|$ for our construction. We restrict ourselves to the $GF(2)$ case. We want to ensure a security level of at least 2^{80}. More precisely we want Theorem 4 to ensure that if for a random system and a random initial internal state value at the beginning of the keystream generation there exists a testing algorithm that allows us to distinguish an L-bit keystream produced by QUAD from a uniformly drawn keystream sequence with an advantage of more than $\epsilon = \frac{1}{100}$ in time less than $T = 2^{80}$ this would imply the existence of an inversion algorithm of non negligible success probability $\epsilon' = \frac{\epsilon}{2^3 \lambda}$ allowing, given a random n-bit to kn-bit system of quadratic equations and the $S(x)$ image by S of a random input value x, to find a preimage by S of $S(x)$ in time T' lower by a factor of more than ϵ' than the best known inversion algorithms for the MQ problem, and thus result in the existence of a large set of weak instances of MQ.

Depending on the intended application of the stream cipher, the maximum keystream length L can vary from a few hundreds bits for a mobile phone application to up to 2^{40} bits. Consequently the allowed parameter values for n and k will also vary, since it is much more demanding to get a security argument for $L = 2^{40}$ bits than for $L = 1000$ bits. We will however retain the latter value $L = 2^{40}$ for a first estimate of the corresponding required value of n.

In her thesis, Magali Bardet [1] shows that the best Groebner basis algorithm to solve a system of kn equations in k unknowns has (in the case of a regular system) a complexity of $T(k,n) = \left(\binom{n+1}{D}\right)^{2.37}$, where D is close to $\left(-k + \frac{1}{2} + \frac{1}{2}\sqrt{2k^2 - 10k - 1 + 2(k+2)\sqrt{k(k+2)}}\right) n$. To obtain a contradiction, we need to have T' lower than $\epsilon' T(k,n)$. For $k = 2$ and with the previous values of $L = 2^{40}$, $T = 2^{80}$ and $\epsilon = \frac{1}{100}$, we get $\epsilon' = 2^{-42}$ and we need to have n greater than 350. For $n = 256$ and $k = 2$, we only get a contradiction if we produce less than $L = 2^{22} = 4$ Mbits of keystream for each key and IV pair.

Practical values For practical use of QUAD we recommend an internal state length of $n = 160$ bits and an expansion factor k of 2 and a maximum keystream length $L = 2^{40}$. We further recommend an IV length $|IV|$ of 80 bits. For such n, k and L values, we do not get a contradiction as for the former parameter values. However our proof reduction is not optimal, and we expect that these parameter values suffice to provide the desired security level of about 2^{80}.

If instead of the n-bit key length assumed (for simplicity of the security arguments) in sections 2 and 3, a keylength $|K|$ strictly lower than n is preferred in order for $|K|$ to better reflect the expected security level, we suggest the following extension of the key loading method described in section 3: periodically repeat the $|K|$ bits of K to get an expanded key of length n, and apply the key and IV procedure of section 3 to this expanded key. We suggest, if this extended key loading method option is retained, to select a key length $|K| = 80$. Though the shorter key option weakens the security arguments of section 4 and can thus be considered less conservative than the full length $n = 160$-bit key, we are not aware of any major security weakness resulting from this option.

An indication of the advantages of the use of the MQ problem for constructing a provably secure stream cipher, in terms of the required internal state size, is given by a comparison with the fastest known provably secure stream cipher, namely a discrete log based construction proposed by Gennaro in [16] with internal state length $n = 3000$ bits (to be compared with the $n = 350$ and 256 internal state lengths derived above) and which produces 2775 bits per iteration and applies 335 modular multiplications of 3000-bit numbers at each iteration. Moreover the security argument of [16] does not assume the existence of a keystream sequence distinguishing algorithm in time $T = 2^{80}$ to get a contradiction, but only a distinguishing algorithm in time $T = 3.5 \cdot 10^{10} \simeq 2^{35}$. Another advantage of MQ is that MQ is NP-hard, whereas the Discrete Logarithm Problem is only in NP $\cap$ co-NP. Moreover the best known algorithm to solve the Discrete Logarithm problem are subexponential, while for MQ, those algorithm are exponential.

5 Cryptanalysis

In this section, we consider various attacks and verify whether they are applicable to our construction. We focus on security aspects not covered by the proof of security of the former section, e.g. the protection against resynchronization attacks provided by the key and IV loading mechanism.

Resistance against Algebraic Attacks: QUAD was designed to resist algebraic attack techniques. As a matter of fact, the key and IV loading and keystream generation mechanisms of QUAD are based upon the iteration of quadratic systems whose associated equations are conjectured to be computationally impossible to solve [4]. In more details, recovering the initial state x of the keystream generator from the whole keystream is more difficult than recovering x from $S(x)$, i.e. solving an intractable quadratic system of kn equations. As for the key and IV loading mechanism, it is possible to express any keystream block, as a set of $(k-1)n$ algebraic equations on the $|K| \geq 80$ key bits. However since the key and IV setup consists of $2|K|$ rounds of a quadratic function, this set consists of $(k-1)n$ equations of degree $|K|$ or nearly $|K|$ on the $|K|$ key bits. It is quite natural to conjecture that such a system is highly intractable.

Correlation Attacks and Distinguishing Attacks: we expect QUAD to be immune to such attacks except for extremely unlikely degenerate instances of the quadratic system S, for example if one of the n-bit to 1-bit quadratic

[4] except for a small fraction of degenerate instances of S, S_0 and S_1 whose occurrence is extremely unlikely if these systems are selected as described in section 4.5.

forms of S_{out} or a linear combination of these $(k-1)n$ quadratic forms has an exceptionally low rank and therefore (for even values of n) a detectable bias.

Time-Memory-Data Tradeoffs and other Generic Attacks: the internal state of our construction has a size n of at least 160 bits in order to resist against generic time-data tradeoff, which have a complexity of $2^{\frac{n}{2}}$.

Since QUAD is based upon the iteration of the quadratic system S_{it}, the keystream sequences it produces are ultimately periodic. Moreover, since S_{it} is not one to one, the order of magnitude of the period can be expected to be $2^{\frac{n}{2}}$ $(k-1)n$-bit keystream blocks. One of the consequences of specifying a maximal keystream length $L << 2^{\frac{n}{2}}$ (a typical order of magnitude is $L = 2^{40}$) is that the detection of short cycles is extremely unlikely.

Guess and Determine Attacks: the analysis of attacks of this type allows us to fix an upper bound on k. Let us assume that an adversary is able to guess p bits of the internal state. Then this adversary gets a system of $(k-1)n$ equations in the $(n-p)$ remaining internal state variables. If the number of monomials generated by these $n-p$ variables $n_p = \frac{1}{2}(n-p)(n-p+1)$ is close to $(k-1)n$, the adversary can linearize the system and recover the internal state. Solving $n_p = (k-1)n$ gives us a number $p_0 = n + \frac{1-\sqrt{1+8n(k-1)}}{2}$ such that for $p \geq p_0$ the linearization is possible. The complexity C of the resulting "attack" is about $2^{p_0}((k-1)n)^{\omega}$, where ω is between 2 and 3. If C is lower than $2^{|K|}$, then the attack is better than exhaustive search. Consequently, k has to be chosen such that C be larger than $2^{|K|}$. For instance for $n = 160$ and $|K| = 80$, $k < 21$ implies that $p_0 > 80$, and therefore $C >> 2^{80}$. More conservative (i.e. lower) values of k than the one given by this simple bound are of course recommended.

Unsurprisingly, the attack would become more efficient for unlikely degenerate instances of S, for instance if several quadratic forms of S could be all expressed as quadratic functions of substantially less than n linear combinations of the n state variables.

Resistance to Resynchronization Attacks with Chosen IVs: our proof does not cover the Key and IV setup but only the keystream generation. They provide a strong argument towards the conjecture that the keystream sequence resulting from any single known or chosen IV value cannot be distinguished from a random sequence, but do not provide guarantees regarding the independence of the sequences resulting from several chosen IVs and the resistance of QUAD against resynchronization attacks. However the following informal argument indicates that the key and IV setup construction of QUAD prevents such resynchronization attacks, or more generally any detectable statistical bias on the joint distribution of the keystream sequences resulting from the same key and several chosen IVs. Let us consider any t-tuple $(IV^1, \cdots, IV^t)$ of t distinct IV values and one randomly chosen n-bit initial state value before IV loading x. By applying the security proofs of section 4 to the $S = (S_0, S_1)$ system of $2n$ quadratic equations, the n-bit to $2n$-bit mapping S_0, S_1 is a strong pseudorandom generator. However, the key and IV loading consists of applying a tree-based construction proposed by Goldreich, Goldwasser and Micali [19] to this generator, so that we can expect the distribution of the $(x^1, \cdots, x^t)$ t-tuple of internal state values resulting from the loading of x and IV^1 to IV^t to be indistinguishable from a t-tuple of random independent values. Moreover, the subsequent runnup rounds during which the keystream generator is run without outputting keystream bits provide an extra security margin, since only high degree functions of x^1 to x^t are available to an adversary instead of quadratic functions. If instead of the proposed key and IV setup the key and IV values the IV had been loaded into the initial state and an insufficient number of quadratic mappings had been applied to the initial state before activating the keystream generation, then chosen-IV attacks exploiting the higher degree differential properties of low degree functions could have been mounted.

Dual Ciphers: because of the structure of the QUAD equations, it is easy to find dual ciphers of QUAD, i.e. simple (e.g. linear) transformations f and g of the key K and the keystream as to ensure that for each triplet of quadratic systems (S, S_0, S_1) there exist quadratic systems (S', S'_0, S'_1) such that for any key K and any IV value IV, the keystream associated with $(f(K), IV, S', S'_0, S'_1)$ is the image by g of the keystream associated with $(f(K), IV, S, S_0, S_1)$. We do not expect this property to represent a security threat for QUAD.

6 Performance

In this Section we give performance results for our recommended version of QUAD, which has 160 bits of internal state, an expansion factor of 2 and a 80-bit key and IV length. On a Pentium IV clocked at 2.5GHz with 512 kByte of cache and using the Intel compiler, our recommended version of QUAD reaches a speed of 4347 cycles/byte (4.6 Mbit/s). On a Pentium 4 with 1MByte of cache, the same version reaches a speed of 2915 cycles/byte (5.7 Mbit/s). This cache effect is due to the fact that the quadratic system used contains more than 4 millions of binary coefficients, which requires around 1MByte to store. A version of QUAD running on an Opteron clocked at 2.1 GHz with a 64-bit architecture reaches the speed of 2176 cycles/byte (quite close from 1MByte/s). An optimised version of Blum Blum Shub's generator with an internal state of 1024 bits, which is far from the number of bits of the internal state required for proven security, reaches 30374 cycles/byte. In his paper[16], Gennaro claimed his discrete logarithm based generator to be twice faster for these parameters. We can therefore assume that this generator runs at about 15000 cycles/byte. Though QUAD is significantly slower than AES, which runs at 25 cycles/byte, it is much more efficient than other provably secure pseudo random generator. Moreover, implementations of QUAD with quadratic system over larger fields (e.g. $GF(16)$ or $GF(256)$) are much faster and even reach 106 cycles/byte.

7 Conclusion

In this paper we introduced QUAD, a novel synchronous stream cipher based on MQ with a security proof in the concrete security model. Eventhough this construction relies on a mathematical problem and has a proof of security, its internal state is of small size n and it extracts a small multiple of n bits at each round. A software implementation of our recommended version of QUAD reaches a speed of 4.6 Mb/s on a standard PC. This makes QUAD of great interest for applications where security is the main concern. We do not preclude that it might be possible to derive tighter bounds in some parts of the proof, which would allow us to further reduce the internal state size and increase the number of extracted bits.

We would like to thank Matt Robshaw and Olivier Billet for helpful comments.

References

1. Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université Paris VI, 2004.
2. Mihir Bellare. The Goldreich-Levin Theorem. `http://www-cse.ucsd.edu/users/mihir/courses.html`, 1999.
3. Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.
4. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
5. Don Coppersmith, Shai Halevi, and Charanjit S. Jutla. Cryptanalysis of stream ciphers with linear masking. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.
6. Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography*, pages 211–227, 2002.
7. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2000.
8. Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
9. Nicolas Courtois and Jacques Patarin. About the XL Algorithm over $GF(2)$. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer-Verlag, 2003.
10. Claus Diem. The XL-Algorithm and a Conjecture from Commutative Algebra. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 323–337. Springer-Verlag, 2004.
11. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at http://www.ecrypt.eu.org/stream/, Accessed September 29, 2005, 2005.
12. Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, Makoto Sugita, and Gwénolé Ars. Comparison Between XL and Grbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer-Verlag, 2004.
13. Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *EUROCRYPT*, pages 245–255, 1996.

14. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of solving algebraic equations. *Inf. Process. Lett.*, 10(4/5):178–179, 1980.
15. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter 7.2 Algebraic Equations over $GF(2)$. W H Freeman & Co, 1979.
16. Rosario Gennaro. An improved pseudo-random generator based on discrete log. In *CRYPTO*, pages 469–481, 2000.
17. Oded Goldreich. Three xor-lemmas an exposition. Technical report, Weizmann Institute of Science, Revohot, Israel, 1995.
18. Oded Goldreich. *Fondationsof Cryptography*, volume 1. Cambridge University Press, 2001.
19. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
20. Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. Available at `http://www-cse.ucsd.edu/users/mihir/courses.html`, 2001.
21. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
22. Russel Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In D.S.Johnson, editor, *21th ACM Symposium on Theory of Computing – STOC '89*, pages 12–24. ACM Press, 1989.
23. Russel Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.
24. Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.
25. Leonid A. Levin and Oded Goldreich. A hard-core predicate for all one-way functions. In D. S. Johnson, editor, *21th ACM Symposium on Theory of Computing – STOC '89*, pages 25–32. ACM Press, 1989.
26. Rudolf Lidl and Haradl Niederreiter. *Finite Fields.* Cambride University Press, 1997.
27. National Institute of Standards and Technology. FIPS-197: Advanced Encryption Standard, November 2001. Available at http://csrc.nist.gov/publications/fips/.
28. Jacques Patarin and Louis Goubin. Asymmetric cryptography with s-boxes. In *ICICS*, pages 369–380, 1997.
29. Jacques Patarin and Louis Goubin. Asymmetric cryptography with s-boxes. In *ICICS*, pages 369–380, 1997.
30. Johan Håstad and Mats Näslund. Bmgl: Synchronous key-stream henerator with provable security. submitted to Nessie Project, 2000.
31. Andrew Yao. Theory and applications of trapdoor function. In *Foundations of Cryptography FOCS 1982*, 1982.

Appendix: Proof of lemma 1

We denote by L_i, $1 \leq i \leq n$ the n-bit to 1-bit linear forms defined by $L_i(x)=x_i$, where x is represented by the binary string $x_1 x_2 \cdots x_n$. The idea of the proof is to call algorithm B sufficiently many times to recover all the $x_i = L_i(x)$ one by one. To do so, we introduce a parameter t, whose order of magnitude is $\log n$ which will be specified later. We use t randomly chosen n-bit to 1-bit linear forms $R_1, \ldots, R_t$ to randomize our requests to algorithm B. For each $L_i(x)$ we want to retrieve, we call algorithm B 2^t times, using the 2^t linear combinations $\bigoplus_j \alpha_j R_j$ of the R_k forms in order to randomize L_i. Suppose we know the t values for $R_j(x)$, then for any α we can also compute the value of $\bigoplus_j \alpha_j R_j(x)$ and add this value to $B(\bigoplus_j \alpha_j R_j \oplus L_i, f(x))$. We denote

$$C(i,\alpha) = B(\bigoplus_j \alpha_j R_j \oplus L_i, f(x)) \oplus \bigoplus_j \alpha_j R_j(x)$$

If we make a correct assumption on the t values $R_1(x)$ to $R_t(x)$ and if B returned the right value of $(\bigoplus_j \alpha_j R_j \oplus L_j)(x)$, then we have

$$\begin{aligned} C(i,\alpha) &= (\bigoplus_j \alpha_j R_j \oplus L_i)(x) \oplus \bigoplus_j \alpha_j R_j(x) \\ &= L_i(x) \oplus \bigoplus_j \alpha_j R_j(x) \oplus \bigoplus_j \alpha_j R_j(x) = L_i(x). \end{aligned}$$

For all the possible α values, we collect the vote $C(i,\alpha)$ for the value of $L_i(x)$. Since algorithm B is supposed to answer correctly most of the time, taking the majority of the votes $C(i,\alpha)$ will provide us with the value of $L_i(x)$ with a high probability if we assume that 2^t requests are enough. The counterpart of this technique is that we have to guess the real values of $R_j(x)$ for all j but since t is of logarithmic size this is achievable.

We now give a more formal proof with a small difference: we use fast Walsh transform computations to simultaneously compute the 2^t results of the votes on the $C(i,\alpha)$ values for all the 2^t possible t-tuples of assumptions $R_j(x)$, $1 \leq j \leq t$, instead of computing them independently.

Before we give the proof, we need to recall some results on the Walsh transform. Given a real function of t binary variables $g(x_1,\ldots,x_t)$, the Walsh transform of g is the real function of t binary variables $G = W(g)$ defined by

$$G(u_1,\ldots,u_t) = \sum_{x_1,\ldots,x_t \in \{0,1\}^t} f(x_1,\ldots,x_t)(-1)^{u_1 x_1 + \ldots + u_t x_t}$$

It is known that the time needed to compute the Walsh transform of a function of t binary variables is $t \cdot 2^t$.

Proof. The algorithm C works as follows : first it randomly selects t elements $R_1,\ldots,R_t$ of the n-dimensional vector space over $GF(2)$ of the n-bit to 1-bit linear forms.

Then for each $i = 1,\ldots,n$ it executes the following process: for all the 2^t possible $\alpha = (\alpha_1,\ldots\alpha_t)$ t-tuples $\in \{0,1\}^t$ store $(-1)^{B(\bigoplus_j \alpha_j r_j \oplus L_i, f(x))}$ in a table of size 2^t, say $(c_0,\ldots c_{2^t-1})$ (thus the coefficient associated with α is $c_{\sum_{j=0}^{t-1} \alpha_j \cdot 2^{j-1}}$). Then it applies the Walsh transform to this table (which represents a function of α. This gives 2^t numbers $(\beta_0^i,\ldots,\beta_{2^t-1}^i)$ such that

$$\begin{aligned} \beta_k^i &= \sum_\alpha (-1)^{B(\bigoplus_j \alpha_j R_j \oplus L_i, f(x))}(-1)^{<k,\alpha>} \\ &= |\{\alpha | C(i,\alpha) = 0\}| - |\{\alpha | C(i,\alpha) = 1\}| \end{aligned}$$

β_k^i is the difference of the number of 0 votes and 1 for $L_i(x)$ corresponding to the assumption that $R_j(x) = k_j$ for all j comprised between 1 and t. Consequently if β_k^i is positive, then C sets bit i of the n-bit candidate value C_k associated with the assumption k to $C_k^i = 0$, otherwise this bit is set to $C_k^i = 1$.

After this process has been completed for all the n values of i, one is left with a list of 2^t n-bit candidate values for x corresponding to each of the 2^t assumptions for $R_1(x)$ to $R_t(x)$. For each candidate value C_k, algorithm C then computes $f(C_k)$ and compares it to $f(x)$. If a match occurs, C keeps C_k in the list of at most 2^t candidate values for x it outputs, otherwise C_k is discarded from the list.

The total running time of algorithm C is $n2^t(T + t + 2) + 2^t T_f$ where T_f is the time needed to compute $f(y)$ for an n-bit value y.

Let us now upper bound the probability that algorithm C fails to select x in the list of pre-images of $f(x)$ it produces. Over the 2^t assumptions for $R_1(x)$ to $R_t(x)$, only the correct one is to be considered. The failure probability of C is upper bounded by the sum of the n probabilities p_i that the vote for $L_i(x)$ is incorrect and we have:

$$p_i = Pr\left\{|\{\alpha|C(i,\alpha) = L_i(x)\}| < \frac{2^t}{2}\right\}$$

$|\{\alpha|C(i,\alpha) = L_i(x)\}|$ is the sum of the 2^t pairwise independent 0-1 variables $C(i,\alpha) \oplus L_i(x) \oplus 1$ of average value $\mu_\alpha \geq \frac{1}{2} + \frac{\epsilon}{2}$ and variance $v_\alpha = \frac{1}{4} - \frac{\epsilon^2}{4}$. Thus p_i has average value $\mu = 2^t\left(\frac{1}{2} + \frac{\epsilon}{2}\right)$ and variance $\sigma^2 = 2^t\left(\frac{1}{4} - \frac{\epsilon^2}{4}\right)$. By applying Chebyshev's inequality, we have

$$\begin{aligned} p_i &= Pr\left\{\sum_\alpha C(i,\alpha) \oplus L_i(x) \oplus 1 < \frac{2^t}{2}\right\} \\ &= Pr\left\{\sum_\alpha C(i,\alpha) \oplus L_i(x) \oplus 1 - \mu < -\frac{2^t\epsilon}{2}\right\} \\ &\leq Pr\left\{\left|\sum_\alpha C(i,\alpha) \oplus L_i(x) \oplus 1 - \mu\right| > \frac{2^t\epsilon}{2}\right\} \leq \frac{\sigma^2}{(2^t\frac{\epsilon}{2})^2} \leq \frac{1}{2^t\epsilon^2} \end{aligned}$$

Thus the failure probability of C is upper bounded by $\frac{n}{2^t\epsilon^2}$. If we want to have a probability of success for algorithm C higher than $\frac{1}{2}$, then we have to choose t such that $2^t = \frac{n}{\epsilon^2}$. Finally the total complexity of algorithm C is given by

$$\frac{2n^2}{\epsilon^2}\left(T + \log(\frac{2n}{\epsilon^2}) + 2\right) + \frac{2n}{\epsilon^2}T_f$$