On the Security of IV Dependent Stream Ciphers

Côme Berbain and Henri Gilbert

France Télécom R&D
38–40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 — France
{firstname.lastname}@orange-ftgroup.com

Abstract. Almost all the existing stream ciphers are using two inputs: a secret key and an initial value (IV). However recent attacks indicate that designing a secure IV-dependent stream cipher and especially the key and IV setup component of such a cipher remains a difficult task. In this paper we first formally establish the security of a well known generic construction for deriving an IV-dependent stream cipher, namely the composition of a key and IV setup pseudo-random function (PRF) with a keystream generation pseudo-random number generator (PRNG). We then present a tree-based construction allowing to derive a IV-dependent stream cipher from a PRNG for a moderate cost that can be viewed as a subcase of the former generic construction. Finally we show that the recently proposed stream cipher QUAD [3] uses this tree-based construction and that consequently the security proof for QUAD's keystream generation part given in [3] can be extended to incorporate the key and IV setup.
Keywords: stream cipher, PRNG, IV setup, provable security

1 Introduction

Stream ciphers and block ciphers are the two most popular families of symmetric encryption algorithms. Unlike block ciphers, stream ciphers do not produce a key-dependent permutation over a large blocks space, but a key-dependent sequence of numbers over a small alphabet, typically the binary alphabet $\{0, 1\}$. To encrypt a plaintext sequence, each plaintext symbol is combined with the corresponding symbol of the keystream sequence using a group operation, usually the exclusive or operation over $\{0, 1\}$.

Nearly all stream ciphers specified recently use two inputs to generate a keystream sequence: a secret key and an additional parameter named the initial value (IV), that is generally not secret. The purpose of the initial value is to allow

[0] The work described in this paper has been supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

to derive several "independent" keystream sequences from one single key, and thus to provide a convenient method for encrypting several plaintext sequences under the same secret key, by "resynchronizing" the stream cipher each time with a new IV value. This represents an obvious practical advantage over formerly proposed stream ciphers which single input was the secret key. But on the other hand, the use of an IV input has considerable impacts on the cryptanalysis and on the formalization of the security requirements on stream ciphers.

As for cryptanalytic implications, the quite numerous attacks of IV-dependent stream ciphers published in the past years clearly indicate that IVs result in additional attack opportunities, and that the key and IV setup procedure still represents one of the less well understood aspects of stream ciphers design. As a matter of fact an adversary can compare the keystream sequences associated with several known, related or chosen IV values, and potentially derive information upon the corresponding internal state values that could not be derived from one single keystream sequence. This is illustrated by Fluhrer, Mantin, and Shamir's attack on the key and IV loading method of the RC4-based cipher used in certain WiFi systems [10], by Ekdahl and Johansson's cryptanalysis of the GSM cipher A5/1 [9], by Joux and Muller's differential known or chosen IV attacks on various ciphers [16, 17], by Daemen, Govaerts, and Vandewalle's and by Armknecht, Lano, and Preneel's resynchronization attacks [8, 1] or more recently by attacks against some of the eSTREAM candidates.

As for the implications of IVs on the formalization of security requirements on stream ciphers, they can be outlined as follows:

In the case of a stream cipher without IV, the requirements are conveniently captured by the theory of pseudo-random numbers generators which has been stemming from the seminal work of Shamir [18], Yao [19], Blum and Micali [6] in the early 80's. A stream cipher is considered secure if the associated key to keystream function is a pseudo-random number generator (PRNG), i.e. an input-expanding function allowing to expand a short seed (the key) into a strictly longer output (the keystream) in such a way that if the secret input seed is uniformly distributed, then the probability distribution of the corresponding output is computationally indistinguishable with non negligible probability from the uniform distribution.

In the case of an IV-dependent stream cipher, no as unanimously accepted formalization of the security requirements has emerged so far. However, most cryptologists would probably agree that a sufficient security condition is that the associated function generator which maps the secret key onto the IV to keystream function be a pseudo-random function generator (PRF), i.e. a random function generator indistinguishable with non negligible probability from a perfect random function generator. To quote an example, this is the condition Halevi, Coppersmith, and Jutla are using to express the security requirements on the IV-dependent stream cipher Scream [15]. We will briefly discuss the validity of this PRF-based formalization in Section 3 hereafter, and conclude that it indeed captures the most natural generalization to IV-dependent stream ciphers of the well accepted (PRNG based) formalization of IV less stream ciphers.

One might argue that since constructing a secure PRF can be expected to be more demanding than constructing a secure PRNG and nearly as difficult as constructing a block cipher, introducing IVs in stream ciphers looses all performance advantages of stream ciphers over block ciphers and requires the same kind of techniques than designing a block cipher. This is however not necessarily the case, as will be shown in the sequel.

The purpose of this paper is twofold. Firstly, to clarify the security requirements upon an IV-dependent stream cipher (Section 3) and to identify sufficient conditions on its key and IV setup and key generation parts in order for the whole stream cipher to be secure (Section 4). Secondly, to propose a practical construction allowing to meet these conditions (Section 5), and therefore to derive an IV-dependent stream cipher with a provable security argument. Finally we show that as an application of this construction the security arguments of QUAD can be extended in order to include the key and IV setup (Section 6). An overview of our main results is given in Section 2.

2 Outline of our results

For all the stream ciphers we consider in this paper, the keystream derivation is split, as in nearly all existing IV-dependent stream ciphers, into the two following separate phases, according to the generic construction illustrated in Figure 1:

- (1) **Key and IV setup:** an m-bit initial state value is derived from the key and IV value.
- (2) **Keystream generation:** the keystream is derived from the m-bit initial state obtained in the key and IV-setup phase. For that purpose, the m-bit initial state is taken as the seed input of a number generator [1].

We formally establish, in Section 4, the validity of the following "folklore" belief implicitly invoked in the security argumentation of several existing IV-dependent stream ciphers [5, 2]: if the family $\{F_K\}$ of IV to initial state functions parametrized by the key K is a PRF and if the number generator g is a PRNG, then the family $\{G_K = g \circ F_K\}$ of IV to keystream sequence functions is a PRF. This provides useful sufficient conditions in order for the IV-dependent stream cipher resulting from the generic construction of Figure 1 to be secure. Our security proof relies upon a simple "composition theorem". A specific construction directly suggested by the former security results might consist of using a trusted block cipher for the IV-setup, as done for instance in the stream cipher candidates LEX [5] and SOSEMANUK [2], both selected as focus ciphers for

[1] Though our constructions are potentially applicable to any number generator with a sufficiently long input size, they are mainly intended for number generators based upon the iterated invocation of a finite state machine (FSM). The keystream sequence generation procedure of nearly all existing stream ciphers has this specific structure, i.e. uses the state transition function of a FSM to update an m-bit internal state and derive at each iteration a t-bit keystream portion by means of a fixed output function.

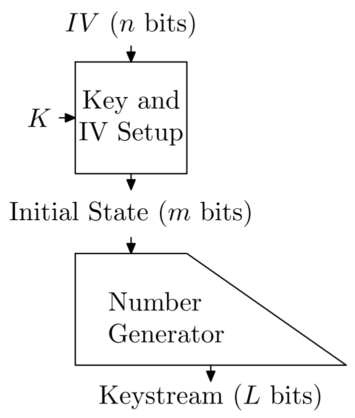

Fig. 1. IV-dependent stream cipher: (generic construction)

third evaluation phase of the stream cipher initiative eSTREAM of the European network ECRYPT. One may however argue that this construction results in a lack of design unity when (unlike in LEX) the keystream generation does not reuse the trusted block cipher used for the key and IV setup.

We propose, in Section 5 hereafter, another specific construction also supported by former security results, which has the additional advantage that it better preserves the design unity of stream ciphers. This construction consists of applying the so called tree-based construction proposed by Goldreich, Goldwasser, and Micali in [13] for deriving the PRF needed for the key and IV setup from any n-bit to $2n$-bit PRNG. This PRNG can be essentially the same as the one used in the keystream generation phase. The later option allows to even better preserve the unity of the design, and to achieve substantial savings in the hardware and software implementation complexity of the stream cipher, since the key and IV setup and the keystream generation are then using the same computational ingredients.

Last of all, we focus in Section 6 on a particular stream cipher where the tree-based construction of Section 5 is applied in the key and IV setup, namely the recently proposed stream cipher QUAD [3]. We show that the partial proof of security of [3] (which gives some evidence that the keystream generation part of QUAD is secure) can be extended to incorporate the key and IV setup. This allows to reduce the security of the whole stream cipher to the difficulty of solving a random multivariate quadratic system.

3 Security Model

3.1 Basic Security Notions

We first recall definitions of advantages for distinguishing a number generator from a perfect random generator and a function generator from a perfect random function generator, and the notions of Pseudo-Random Number Generator

(PRNG) and Pseudo-Random Function (PRF). All the security definitions used throughout this paper relate to the concrete (non asymptotic) security model. In the sequel, when we state that a value u is randomly chosen in a set U, we implicitly mean that u is drawn according to the uniform law over U.

Single-query distinguisher for a number generator: let us consider a number generator $g : \{0,1\}^n \longrightarrow \{0,1\}^L$ with input and output lengths $L > n$, used to expand an n-bit secret random seed into an L-bit sequence. A distinguisher in time t for g is a probabilistic testing algorithm A which when input with an L-bit string outputs either 0 or 1 with time complexity at most t. We define the advantage of A for distinguishing g from a perfect random generator as

$$\mathbf{Adv}_g^{prng}(A) = \left|\Pr_{x\in\{0,1\}^n}(A(g(x)) = 1) - \Pr_{y\in\{0,1\}^L}(A(y) = 1)\right|,$$

where the probabilities are not only taken over the value of an unknown randomly chosen $x \in \{0,1\}^n$ (resp. of a randomly chosen $y \in \{0,1\}^L$), as explicitly stated in the above formula, but also over the random choices of the probabilistic algorithm A.

We define the advantage for distinguishing the function g in time t as

$$\mathbf{Adv}_g^{prng}(t) = \max_A\{\mathbf{Adv}_g^{prng}(A)\},$$

where the maximum is taken over all testing algorithms of time complexity at most t.

A function g is said to be a PRNG if $\mathbf{Adv}_g^{prng}(t)$ is negligible (for example less than 2^{-40}) for values of t strictly lower than a fixed threshold (for example 2^{80} or 2^{128}). The definition of a PRNG is therefore dependent upon thresholds reflecting the current perception of an acceptably secure number generator.

Multiple-query distinguisher for a number generator: let us still consider a function g from n bits to L bits. A q-query distinguisher in time t for g is a probabilistic testing algorithm A which when input with a q-tuple of L-bit words outputs either 0 or 1 with time complexity at most t. We define the advantage of A for distinguishing g from a perfect random generator as

$$\mathbf{Adv}_g^{prng}(A) = \left|\Pr(A(g(x_1), \ldots, g(x_q)) = 1) - \Pr(A(y_1, \ldots, y_q) = 1)\right|,$$

where the probabilities are taken over the q-tuples of n-bit values x_i (resp. of L-bit values y_i) and on the random choices of the probabilistic algorithm A. We also define the advantage for distinguishing the function g in time t with q queries as

$$\mathbf{Adv}_g^{prng}(t, q) = \max_A\{\mathbf{Adv}_g^{prng}(A)\},$$

where the maximum is taken over all testing algorithms A having time-complexity at most t and using q inputs.

Distinguisher for a function generator: let us now consider a function generator, i.e. a family $F = \{f_K\}$ of $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions indexed by a key K randomly chosen from $\{0,1\}^k$. A distinguisher in time t with q queries for F is a probabilistic testing algorithm A^f able to query an n-bit to m-bit oracle function f up to q times. Such an algorithm allows to distinguish a randomly chosen function f_K of F from a perfect random function f^* randomly chosen in the set $F^*_{n,m}$ of all $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions with a distinguishing advantage

$$\mathbf{Adv}_F^{prf}(A) = \left|\Pr(A^{f_K} = 1) - \Pr(A^{f^*} = 1)\right|,$$

where the probabilities are taken over $K \in \{0,1\}^k$ (resp $f^* \in F^*_{n,m}$) and over the random choices of A. We define the advantage for distinguishing the family F in time t with q queries as

$$\mathbf{Adv}_F^{prf}(t,q) = \max_A\{\mathbf{Adv}_F^{prf}(A)\},$$

where the maximum is taken over all testing algorithms A working in time at most t and capable to query an n-bit to m-bit oracle function up to q times.

A family of functions $F = \{f_K\}$ is said to be a PRF if $\mathbf{Adv}_F^{prf}(t,q)$ is negligible for values of t and q strictly lower than the respective threshold (for example 2^{80} or 2^{128} for t and 2^{40} for q).

3.2 Security Requirements for an IV-dependent Stream Ciphers

Let us consider an IV-dependent stream cipher, i.e. a family $G = \{g_K\}_{K\in\{0,1\}^k}$ of IV to keystream functions $g_K : \{0,1\}^n \longmapsto \{0,1\}^L$, where k is the size of the key, n is the size of the IVs and L is the maximum number of keystream bits that can be produced for a given IV.

Such an IV-dependent stream cipher can be viewed as a special number generator, allowing to expand a k-bit secret seed onto an exponentially long sequence of 2^n L-bit keystream words $\{Z_{IV} = g_K(IV)\}_{IV\in\{0,1\}^n}$, with the additional property that while this sequence is too long to be entirely accessed in a sequential manner, it can be directly accessed, i.e. that for any value of $IV \in \{0,1\}^n$ the computational cost for accessing the L-bit subsequence Z_{IV} is constant.

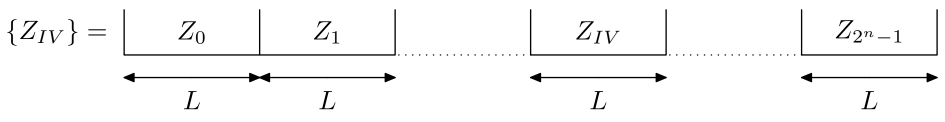

Fig. 2. Exponentially long sequence with direct access associated to an IV-dependent stream cipher g

This observation can be used in order to try to generalize the well accepted formalization of the security requirements on an IV-less stream cipher by means of a PRNG in a natural manner. An IV-less stream cipher is considered secure if and only if no testing algorithm, when given access either to a Λ-bit output of the generator corresponding to a random secret input or to a random Λ-bit sequence, can distinguish both situations in time less than a sufficiently large threshold (say 2^{80}) with a non-negligible advantage. It can be reasonably argued that in the case of an IV-dependent stream cipher, the most natural generalization of the above security definition is to require that no testing algorithm, when given a sufficiently large number q (say for instance $\min(2^{80}, 2^n)$) of direct accesses to L-bit subsequences of a sequence $\{Z_{IV}\}$ associated with a random unknown key K or to a uniformly drawn sequence of 2^n L-bit subsequences, can distinguish both situations in time less than a sufficiently large threshold (say 2^{80}) with a non-negligible advantage.

But it is easy to see that both the sequence $\{Z_{IV}\}$ and the uniformly drawn sequence of $2^n \cdot L$ bits can be viewed as n-bit to L-bit functions, and that direct accesses to these sequences can be viewed as oracle queries to these functions. Therefore the requirements formulated above are strictly equivalent to saying that the function family $G = \{g_K\}_{K \in \{0,1\}^k}$ is a PRF.

In other words, we have given some evidence that an IV-dependent stream cipher $G = \{g_K\}$ for $K \in \{0,1\}^k$ with $g_K : \{0,1\}^n \longrightarrow \{0,1\}^L$ can be viewed as a family of functions, and can be considered secure if and only if it is a PRF , i.e. sufficiently indistinguishable from a perfect random function.

Related key attacks, which relevance, when it comes to security requirements on symmetric ciphers, is a controversial issue [4], are not covered by our security model.

4 Security of the Generic Construction

4.1 A Simple Composition Theorem

In this Section, we define the composition G of a family of function F and a function g, relate the indistinguishability of G to the one of F and g, and show that this composition theorem results in a secure construction allowing to derive a secure IV-dependent stream cipher from a PRF and a PRNG.

Definition 1. *The composition $G = g \circ F$ of an n-bit to m-bit family of functions $F = \{f_K\}$ and of an m-bit to L-bit function g is the n-bit to L-bit family of functions*

$$G = \{g \circ f_K\}.$$

Theorem 1. *Let us consider a PRF $F = \{f_K\}$ where $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ and a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ that produces L bits in time T_g^L. The advantage in time t with q queries of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows*

$$\mathbf{Adv}_G^{prf}(t, q) \leq \mathbf{Adv}_F^{prf}(t + qT_g^L) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

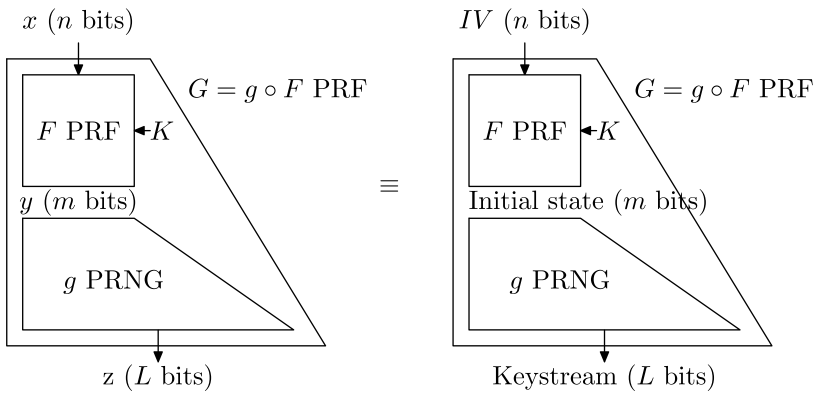

Fig. 3. Composing a PRF and a PRNG gives a PRF

In order to prove Theorem 1 we first establish a useful lemma which relates the single-query and multiple-queries advantages of any PRNG.

Lemma 1. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which can be computed in time T_g^L. The q-query advantage for distinguishing g in time t is related to the single-query advantage for distinguishing g by the inequality*

$$\mathbf{Adv}_g^{prng}(t,q) \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

A proof of the above lemma is given in Appendix 1; this proof is similar to the one of a proposition relating the single-sample indistinguishability and the multiple-sample indistinguishability of polynomial-time constructible ensembles established by Goldreich and Krawczyk [14, 12]. Using this lemma, we can prove Theorem 1. Our proof is given in Appendix 2. Theorem 1 is illustrated on the left part of Figure 3.

Application to the security of IV-dependent stream ciphers. A direct application of Theorem 1 is depicted on the right part of Figure 3. As said in Section 3, an IV-dependent stream cipher can be considered secure if and only if the IV to keystream function g_K parametrized by the key is a PRF. Theorem 1 implies that this is indeed the case, i.e. that the stream cipher is secure if:
1) the n-bit to m-bit IV to initial state function parametrized by the key representing the IV setup of a stream cipher is a PRF;
2) the m-bit to L-bit initial state to keystream function is a PRNG;
3) the upper bounds on the advantage for distinguishing $\{g_K\}$ given by Theorem 1 guarantee a sufficient resistance against attacks.

From now on, we consider the following stream cipher design problem. Let us assume that a trusted number generator g allowing to expand an m-bit initial state into an L-bit sequence is available. We are now faced with the issue of constructing a key and IV setup PRF in order to compose this PRF with g to get a secure IV-dependent stream cipher.

A straightforward construction for such a PRF, directly suggested by the former security results, would consist of using a trusted block cipher with a sufficient block-length to fit the initial state length m of g. As a matter of fact it is usual to conjecture that the family of key dependent encryption permutations associated with a secure block cipher represents both a pseudo-random family of permutations (PRP) and a PRF. The value of m must be typically at least 160 bits in the frequent case where g has a finite state automaton structure, in order to avoid generic time-memory trade-offs. This suggests that one could for instance use the variant of Rijndael with a 256-bit block size, or a truncated instance of this cipher if m is smaller than 256, or an appropriate "one to many blocks" mode of operation of any block cipher [11] for initial state sizes larger than 256 bits. This would allow to accommodate keys and IVs of size up to one block. Such an approach can certainly be considered more conservative than the key and IV setup procedure of many existing stream ciphers. On the other hand, one may argue that it results in a strong performance penalty for the encryption of short messages and an increased implementation complexity, and in a lack of design unity if the trusted number generator g does not reuse the same ingredients as the trusted block cipher.

5 A Tree Based Stream Cipher Construction

Is this Section we present a key and IV setup procedure derived from the Tree Based Construction introduced by Goldreich, Goldwasser, and Micali in [13]. This construction allows to derive a PRF from a PRNG and to relate their securities. Though initially introduced for theoretical purposes (namely show in the asymptotic model that the existence of a PRNG implies the existence of a PRF) it is also of practical interest since it allows, as shown here, to build a stream cipher from two number generators: the Tree Based Construction is used to transform the first number generator into an efficiently computable key and IV setup; the second number generator is initialized with the value given by the key and IV setup, and generates the keystream. These two number generators can advantageously be the same. Thus it becomes possible to build an IV-dependent stream cipher from one single number generator.

5.1 The Tree Based Construction

The Tree Based Construction allows to derive a PRF F^g from a PRNG g. Let us consider a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ where $L \geq 2m$ and let us denote the L bit image of $y \in \{0,1\}^m$ by $g(y) = z_0 z_1 \dots z_{L-1}$. We derive from g two functions $g_0 : y \in \{0,1\}^m \longmapsto z_0, \dots, z_{m-1}$ and $g_1 : y \in \{0,1\}^m \longmapsto z_m, \dots, z_{2m-1}$ from m bits to m bits which on input $y \in \{0,1\}^m$ respectively produce the first and the second m-bit string generated by g when input with y.

The PRF F^g is the family of functions $\{f_y\}_{y \in \{0,1\}^m}$ where

$$\begin{aligned} f_y : \{0,1\}^n &\longrightarrow \{0,1\}^m \\ (x_1, x_2, \dots, x_n) &\longmapsto f_y(x_1, x_2, \dots, x_n) = g_{x_n} \circ g_{x_{n-1}} \dots \circ g_{x_1}(y) \end{aligned}$$

This construction is illustrated on Figure 4: the input bits determine a path trough the binary tree leading to the output of the function.

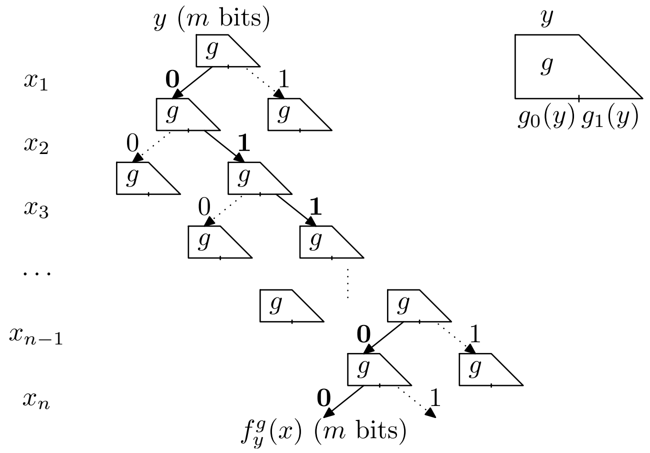

Fig. 4. Tree Based Construction

Theorem 2. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which generates $L \geq 2m$ outputs bits and produces its $2m$ first output bits in time T_g^{2m} and let $F^g = \{f_y\}_{y \in \{0,1\}^m}$ be the family of n-bit to m-bit functions derived from g by the Tree Based Construction. The (t, q) advantage of PRF F^g is related to the single-query advantage of PRNG g by the following inequality:*

$$\mathbf{Adv}_{F^g}^{prf}(t, q) \leq nq\mathbf{Adv}_g^{prng}(t + q(n + 1)T_g^{2m}).$$

A proof of Theorem 2 is given in Appendix 3. This proof is essentially the same as the security proof of the Tree Based Construction due to Goldreich, Goldwasser, and Micali[13], a detailed version of which is given by Goldreich in [12], up to the fact that we consider the concrete security model instead of the asymptotic (polynomial time indistinguishability) security model.

5.2 Resulting Stream Cipher Construction

To build an IV-dependent stream cipher from a m-bit to L-bit PRNG g representing an IV-less stream cipher, we apply the Tree Based Construction to a m to L'-bit PRNG g' ($L' \geq 2m$) typically equal to g itself, in order to derive the key and IV setup function, which produces the initial state of g, following the construction of Figure 4. For a key K and an IV IV, the value $f_K^{g'}(IV)$ is the initial state of g. Thanks to Theorem 1 and since the Tree Based Construction provides a PRF, the resulting stream cipher is also a PRF. The security of the final stream cipher only depends on the security of the PRNG.

In case K is smaller than m bits, we need to extend K to a value randomly chosen in $\{0,1\}^m$. In order to achieve this we can use an additional PRNG $h : \{0,1\}^k \longrightarrow \{0,1\}^m$. The proof for the Tree Based Construction can easily be extended and we have:

$$\mathbf{Adv}_{F^g}^{prf}(t,q) \leq nq\mathbf{Adv}_{g'}^{prng}(t + q(n+1)T_{g'}^{2m}) + q\mathbf{Adv}_{h}^{prng}(t + q(n+1)T_h^m).$$

Theorem 3. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which generates L outputs bits in time T_g^L and $g' : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ be a PRNG. We can define a stream cipher $G = \{G_K\} = g \circ F^{g'}$, where $F^{g'}$ is derived from g' using the Tree Based Construction*

$$G_K(IV) = g \circ f_K^{g'}(IV)$$

Moreover G is a PRF and we have:

$$\mathbf{Adv}_{G}^{prf}(t,q) \leq nq\mathbf{Adv}_{g'}^{prng}(t + qT_g^L + q(n+1)T_{g'}^{2m}) + q\mathbf{Adv}_{g}^{prng}(t + qT_g^L)$$

If g and g' are equal, then we have

$$\mathbf{Adv}_{G}^{prf}(t,q) \leq q(n+1)\mathbf{Adv}_{g}^{prng}(t + q(n+2)T_g^L)$$

Proof. To prove this result we only have to use Theorem 1 and Theorem 2.

The above key and IV setup construction is of practical interest: suppose we have a trusted PRNG, for example the Shrinking Generator [7], we can build an IV-dependent stream cipher based on this PRNG without introducing any additional feature for a moderate computational cost.

5.3 Efficiency Considerations

Let us assume for instance that we want to build from a PRNG g of initial state length 160 bits a stream cipher with a 160-bit key, a 80-bit IV, and a target security of 2^{80}, using the previously described construction. Then the time required to compute the key and IV setup with the above construction is the time required by g to produce 3200 bytes. Considering a very fast PRNG, running at 5 cycles per byte, the key and IV setup requires about 16000 cycles on a standard PC. This is slower than using a block cipher like AES, which would require about 1000 cycles. Therefore for software applications where resynchronization has to be done frequently, this construction is not at all efficient and using a block cipher for the key and IV setup should be considered. However for applications where the keystream generated for a single IV is very long compared to these 3200 bytes our construction can be competitive.

Now in the case of hardware applications, the above construction can be of real interest, in order to minimize the hardware complexity since it uses a single PRNG for the key and IV setup and the keystream generation. Then only a few additional gates are required to implement the key and IV setup. If a block cipher were used instead, then the total number of gates required to implement the stream cipher would be much higher than the number of gates required for a PRNG.

6 Application to the QUAD Stream Cipher

The stream cipher QUAD is a practical stream cipher with some provable security which was introduced [3] by Berbain, Gilbert, and Patarin at Eurocrypt 2006. The provable security argument relates, in the $GF(2)$ case, the indistinguishability of the keystream generated by QUAD to the conjectured hardness of solving random quadratic systems. QUAD iterates a one way function, namely a quadratic system, upon an internal state and extracts a certain number of bits of this step at each iteration.

The keystream generation makes use of two systems $S_{\text{in}} = (Q_1, \ldots, Q_m)$ and $S_{\text{out}} = (Q_{m+1}, \ldots, Q_{km})$ of multivariate quadratic equations both sharing the same m unknowns over $\text{GF}(q)$, typically $GF(2)$ as described on Fig. 5. The first system S_{in} is used to update the internal state and thus contains m equations, whereas the second system S_{out} produces the keystream and contains $(k-1)m$ equations. As explained in [3], the quadratic systems S_{in} and S_{out}, though randomly generated, are both publicly known.

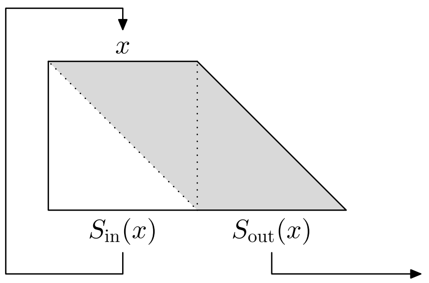

Fig. 5. Stream cipher QUAD

Given an internal state $x = (x_1, \ldots, x_m)$, the keystream generation amounts to iterating the following steps:

- compute $\big(S_{\text{in}}(x), S_{\text{out}}(x)\big) = \big(Q_1(x), \ldots, Q_{km}(x)\big)$, from the internal state x;
- output the sequence $S_{\text{out}}(x) = \big(Q_{m+1}(x), \ldots, Q_{km}(x)\big)$ of $(k-1)m$ keystream elements of $\text{GF}(q)$;
- update the internal state x with the sequence $S_{\text{in}}(x) = \big(Q_1(x), \ldots, Q_m(x)\big)$.

Before generating any keystream the internal state x needs to be initialized, 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 will assume in the sequel that $|K|$ is equal to m. The initialization is done as follows: two publicly known chosen multivariate quadratic systems S_0 and S_1 of m equations over m unknowns are used. The initial state is filled with the key. 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 $GF(q)^m$ value $S_0(x)$; if $IV_i = 1$, x is replaced by the $GF(q)^m$ value $S_1(x)$. Finally the cipher is clocked m additional times as described before, but without outputting the keystream.

We now extend the partial proof over GF(2) given in [3] to incorporate the key and IV Setup. We denote by $S = (S_{\text{in}}||S_{\text{out}})$ the randomly chosen system of km equations on m variables and $S' = (S_0||S_1)$ the randomly chosen system of $2m$ equations in m variables. We also denote by $g^S : \{0,1\}^m \longrightarrow \{0,1\}^L$ and $g^{S'} : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ the corresponding PRNGs.

The key and IV setup proposed in [3] can be divided into two parts: in the first part the Tree Based Construction for $g^{S'}$ is applied and the value $y = f_K^{g^{S'}}(IV)$ is computed. Then in the second part, y is used to initialize g^S which is clocked m times but the corresponding $(k-1)m^2$ bits of keystream are not used. The value of the internal state after these m clocks is used to produce L bits of keystream. The security of the first part is related to the security of $g^{S'}$ thanks to Theorem 2.

$$\mathbf{Adv}^{prf}_{F^{g^{S'}}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t + q(n+2)T^{2m}_{g^{S'}})$$

The security of the PRNG g_{real} which starts by running m clocks like g^S without producing any keystream to reflect the runup of QUAD and then produces L bits of keystream as g^S does, is related to the security of generator $\tilde{g}^S : \{0,1\}^m \longrightarrow \{0,1\}^{L+(k-1)m^2}$ which iterates S to produce $L + (k-1)m^2$ bits, since g_{real} produces the same keystream as $\tilde{g}^S$ up to the fact that the first $(k-1)m^2$ bits of $g^{S'}$ are discarded. Consequently a distinguisher on g_{real} is also a distinguisher for $\tilde{g}^S$. Thus the advantage of g_{real} is upper-bounded by the advantage of $\tilde{g}^S$.

$$\mathbf{Adv}^{prng}_{g_{real}}(t) \leq \mathbf{Adv}^{prng}_{\tilde{g}^S}(t + T^{L+(k-1)m^2}_{\tilde{g}^S})$$

Finally the security of the stream cipher is related to the securities of $\tilde{g}^S$ and of $g^{S'}$ by the composition theorem of Section 4.

$$\mathbf{Adv}^{prf}_{\text{QUAD}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t+q(n+3)T^{L}_{g^{S'}})+q\mathbf{Adv}^{prng}_{\tilde{g}^S}(t+qT^{L}_{\tilde{g}^S}+T^{L+(k-1)m^2}_{\tilde{g}^S})$$

Those two generators are based on the iteration of a randomly chosen quadratic system of km equations in m variables (with $k = 2$ for S'). We can use the main result of [3], which relates the security of this kind of PRNG to the difficulty of inverting a randomly chosen multivariate quadratic system to say that the security of QUAD as a stream cipher is related to the difficulty of the MQ Problem, i.e. an adversary able to distinguish QUAD from a PRF in time t with q queries is then able to construct a MQ solver:

$$\begin{aligned}\mathbf{Adv}^{prf}_{\text{QUAD}}(t,q) &\leq 3q\mathbf{Adv}^{prng}_{g^S}(t + qT^{L+(k-1)m^2}_{\tilde{g}^S}) \\ &\leq \mathbf{Adv}^{MQinversion}(t'[t + qT^{L+(k-1)m^2}_{\tilde{g}^S}]),\end{aligned}$$

where $t'[t + qT^{L+(k-1)m^2}_{\tilde{g}^S}]$ is the running time of the MQ inverter algorithm given by the main reduction theorem of [3], which is recalled in Appendix.

7 Conclusion

In this paper we investigated security issues arising for IV-dependent stream ciphers. We confirmed the "folklore" belief that the composition of a key and IV setup PRF and a key generation PRNG provides a secure stream cipher, which furnishes a proof that initializing a PRNG with a block cipher is secure provided that the block cipher's block length is sufficiently large. Moreover we described a practical construction that allows to derive an IV-dependent stream cipher from a PRNG (or equivalently an IV-less stream cipher) for a moderate additional cost. This construction is quite simple and does not require additional components. Finally we showed an application of this provably secure construction to the stream cipher QUAD, by incorporating the key and IV setup in the security proof given by the authors of QUAD. The resulting extended proof relates the security of the whole stream cipher (not only the keystream generation part) to the conjectured intractability of the MQ problem.

References

1. Frederik Armknecht, Joseph Lano, and Bart Preneel. Extending the Resynchronization Attack. In Helena Handschuh and Anwar Hasan, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, page 19. Springer-Verlag, 2004.
2. Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert. Sosemanuk, a Fast Software-Oriented Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. http://www.ecrypt.eu.org/stream.
3. Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: a Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes in Computer Science. Springer-Verlag, 2006.
4. Daniel J. Bernstein. Related-key attacks: Who cares? eSTREAM, ECRYPT Stream Cipher Project, 2006. http://www.ecrypt.eu.org/stream/phorum.
5. Alex Biryukov. A new 128 bit key Stream Cipher : LEX. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. http://www.ecrypt.eu.org/stream.
6. Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
7. Don Coppersmith, Hugo Krawczyk, and Yishay Mansour. The Shrinking Generator. In Douglas Robert Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 22–39. Springer-Verlag, 1993.
8. Joan Daemen, Ren Govaerts, and Joos Vandewalle. Resynchronization Weaknesses in Synchronous Stream Ciphers. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 159–167. Springer-Verlag, 1993.
9. Patrik Ekdahl and Thomas Johansson. Another Attack on A5/1. *IEEE Transactions on Information Theory*, 49(1):284–289, 2003.
10. Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *Selected Areas in Cryptography*, pages 1–24, 2001.

11. Henri Gilbert. The Security of "One-Block-to-Many" Modes of Operation. In Thomas Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 376–395. Springer-Verlag, 2003.
12. Oded Goldreich. *The Foundations of Cryptography - Volume 1.* Cambridge University Press, 2001.
13. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, 1986.
14. Oded Goldreich and Hugo Krawczyk. Sparse Pseudorandom Distributions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 113–127. Springer-Verlag, 1989.
15. Shai Halevi, Don Coppersmith, and Charanjit S. Jutla. Scream: A Software-Efficient Stream Cipher. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encrytion – FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, 2002.
16. Antoine Joux and Frédéric Muller. A Chosen IV Attack Against Turing. In Mitsuru Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography – SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 194–207. Springer-Verlag, 2003.
17. Frédéric Muller. Differential Attacks against the Helix Stream Cipher. In Willi Meier and Roy Bimal, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, page 0. Springer-Verlag, 2004.
18. Adi Shamir. On the Generation of Cryptographically Strong Pseudo-Random Sequences. In *ICALP*, pages 544–550, 1981.
19. Andrew Yao. Theory and Applications of Trapdoor Function. In *Foundations of Cryptography FOCS 1982*, 1982.

Appendix

Proof of Lemma 1

Lemma 1. Let us consider a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ which can be computed in time T_g^L. Then we have

$$\mathbf{Adv}_g^{prng}(t,q) \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Proof. Suppose there is an algorithm A that distinguishes q L-bit keystream sequence produced by g from q unknown randomly chosen initial internal state $x_i \in \{0,1\}^m$ from q random L-bit sequences in time t with advantage ϵ. Then we are going to build an algorithm B that distinguishes $g(x)$ corresponding to an unknown random input x, from a random value of size L in time $t' = t + qT_g^L$ with advantage $\frac{\epsilon}{q}$.

We introduce the hybrid probability distributions D^i over $\{0,1\}^L$ for any $0 \leq i \leq q$ respectively associated with the random variables

$$Z^i = (g(x_1), g(x_2), \ldots, g(x_i), r_{i+1}, \ldots, r_q)$$

where the r_j and x_i are random independent uniformly distributed values of $\{0,1\}^L$ and $\{0,1\}^m$ respectively. Consequently D^q is the distribution of the q

L-bit keystream produced by g and D^0 is the distribution of q L-bit uniformly distributed values of $\{0,1\}^L$.

We denote by p^i the probability that A accepts a random qL-bit sequence distributed according to D^i.

We have supposed that algorithm A distinguishes between D^0 and D^q with advantage ϵ, in other words that $|p^0 - p^q| \geq \epsilon$.

Algorithm B works as follows : on input $y \in \{0,1\}^L$ it selects randomly an i such that $1 \leq i \leq q$ and constructs the vector

$$Z(y) = (g(x_1), g(x_2), \dots, g(x_{i-1}), y, r_{i+1}, r_{i+2}, \dots, r_q)$$

with x_i and r_i randomly chosen values. If y is distributed accordingly to the output distribution of g, i.e. $y = g(x)$ for a uniformly distributed value of x, then

$$Z(y) = (g(x_1), g(x_2), \dots, g(x_{i-1}), g(x), r_{i+1}, r_{i+2}, \dots, r_q)$$

is distributed according to D^i. Now if y is distributed according to the uniform distribution, then

$$Z(y) = (g(x_1), g(x_2), \dots, g(x_{i-1}), y, r_{i+1}, r_{i+2}, \dots, r_q).$$

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

$$\begin{aligned} &\left|\Pr_x(B(g(x)) = 1) - \Pr_y(B(y) = 1)\right| \\ &= \Big|\frac{1}{q}\sum_{i=0}^{q-1} p^i - \frac{1}{q}\sum_{i=1}^{q} p^i\Big| = \frac{1}{q}|p^0 - p^q| \geq \frac{\epsilon}{q}. \end{aligned}$$

Thus B distinguishes the output distribution of g from the uniform distribution with probability at least $\frac{\epsilon}{q}$ in time $t + qT_g^L$. Consequently we have:

$$\begin{aligned} \mathbf{Adv}_g^{prng}(A) &= q\mathbf{Adv}_g^{prng}(B) \\ &\leq q\mathbf{Adv}_g^{prng}(t + qT_g^L) \end{aligned}$$

which gives us the final result:

$$\mathbf{Adv}_g^{prng}(t,q) \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Proof of Theorem 1

Theorem 1. Let us consider $F = \{f_K\}$ where $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ a PRF and $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ a PRNG that produces L bits in time T_g^L. The advantage in time t with q queries of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows

$$\mathbf{Adv}_G^{prf}(t,q) \leq \mathbf{Adv}_F^{prf}(t + qT_g^L) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Proof. We want to upper bound the advantage of an algorithm A that in time t with q requests distinguishes a random instance $g_K = g \circ f_K$ of $G = g \circ F$ from a perfect random function $g^* \in F^*_{n,L}$. We can write the advantage of A as

$$\mathbf{Adv}_G^{prf}(A) = \left|\Pr(A^{g_K} = 1) - \Pr(A^{g^*} = 1))\right|.$$

A is making at most q distinct queries to an oracle function instantiated by g_K, resp. g^*. In order to upper bound $\mathbf{Adv}_G^{prf}(A)$, we consider the intermediate situation where the oracle function is neither g_k nor g^*, but a random instance $g \circ f^*$ of the composition of a perfect random function $f^* \in F^*_{n,m}$ and g. Due to the triangular inequality, we have

$$\begin{aligned}\mathbf{Adv}_G^{prf}(A) \leq & \left|\Pr(A^{g_K} = 1) - \Pr(A^{g \circ f^*} = 1)\right| \\ & + \left|\Pr(A^{g \circ f^*} = 1) - \Pr(A^{g^*} = 1)\right|.\end{aligned}$$

We denote the first and the second absolute values of the right expression by δ_1 and δ_2.

Let us first upper bound δ_2. It is easy to see that instantiating the oracle function of A with $g \circ f^*$ (resp. g^*) amounts to answering the up to q distinct oracle queries of A with a q-tuple $(g(y_1), \ldots, g(y_q))$ of L-bit values, where the q-tuple $(y_1, \ldots, y_q)$ is a randomly drawn from $\{0,1\}^{mq}$, resp. with a q-tuple $(z_1, \ldots, z_q)$ of answers randomly drawn from $\{0,1\}^{Lq}$. In both case, the q-tuple of oracle answers is independent of the up to q distinct values of the oracle queries. Using this fact we can derive an algorithm B that distinguishes q values $(g(x_1), \ldots, g(x_q))$ from q random values of $\{0,1\}^L$. Algorithm B works as follows: on input $(y_1, \ldots, y_q)$ it runs algorithm A. Consequently it has to answer A's n-bit oracle queries x_i with L-bit responses. On each distinct query x_i, B simply answers y_i. When A halts, B halts also and returns the output of A. We can easily see that $\Pr(B(g(x_1), \ldots g(x_q)) = 1) = \Pr(A^{g \circ f^*} = 1)$ and that $\Pr(B(y_1, \ldots y_q) = 1) = \Pr(A^{g^*} = 1)$. Consequently we have

$$\delta_2 = \mathbf{Adv}_g^{prng}(B) \leq \mathbf{Adv}_g^{prng}(t, q).$$

Lemma 1 now provides:

$$\delta_2 \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

In order to upper-bound $\mathbf{Adv}_G^{prf}(A)$ we still have to upper bound δ_1. This can be done by deriving from algorithm A an algorithm C that is able by invoking A one single time to distinguish a random instance of the n-bit to m-bit PRF F from a perfect random function $f^* \in F^*_{n,m}$ with an advantage also equal to δ_1.

Algorithm C has access to an n-bit to m-bit oracle function f. C works as follows: first it invokes algorithm A. Consequently it has to answer A's n-bit oracle queries x_i with L-bit responses. For such a query, C queries its own oracle function f with the same query value x_i, gets an m-bit answer $y_i = f(x_i)$, computes the L-bit value $g(y_i)$, and answers this value to algorithm A. When A halts, C halts as well and outputs the same output as A. Therefore we have $\Pr(C^{f_K} = 1) = \Pr(A^{g \circ f_K} = 1) = \Pr(A^{g_K} = 1)$ and $\Pr(C^{f^*} = 1) = \Pr(A^{g \circ f^*} = 1)$. This implies that $\left|\Pr(C^{f_K} = 1) - \Pr(C^{f^*} = 1)\right|$ is equal to $\left|\Pr(A^{g_K} = 1) - \Pr(A^{g \circ f^*} = 1)\right|$, i.e.

$$\mathbf{Adv}_F^{prf}(C) = \delta_1.$$

Furthermore the time required for algorithm C is equal to the time required for algorithm A plus q times the time of computing g. Therefore $\mathbf{Adv}_F^{prf}(C)$ is upper-bounded by $\mathbf{Adv}_F^{prf}(t + qT_g^L, q)$, i.e. $\delta_1 \leq \mathbf{Adv}_F^{prf}(t + qT_g^L, q)$. Finally we have for any A

$$\mathbf{Adv}_G^{prf}(A) \leq \delta_1 + \delta_2 \leq \mathbf{Adv}_F^{prf}(t + qT_g^L, q) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Consequently

$$\mathbf{Adv}_G^{prf}(t, q) \leq \mathbf{Adv}_F^{prf}(t + qT_g^L, q) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).\square$$

Proof of Theorem 2

Theorem 2. Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which generates $L \geq 2m$ outputs bits and produces its $2m$ first output bits in time T_g^{2m} and let $F^g = \{f_y\}_{y \in \{0,1\}^m}$ be the family of n-bit to m-bit functions derived from g by the Tree Based Construction. The (t, q) advantage of PRF F^g is related to the single-query advantage of PRNG g by the following inequality:

$$\mathbf{Adv}_{F^g}^{prf}(t, q) \leq nq\mathbf{Adv}_g^{prng}(t + q(n+1)T_g^{2m}).$$

Proof. First we define, for $0 \leq i \leq n$, a family F_i^g of $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions; each F_i^g can be viewed as an intermediate PRF between F^g and the set $F_{n,m}^*$ of perfect random n-bits to m-bit functions.

- $F_0^g = \{f_{y_0}^g\}_{y_0 \in \{0,1\}^n}$ where $f_{y_0}^g : (x_1, \ldots, x_n) \longmapsto g_{x_n} \circ \ldots \circ g_{x_1}(y_0)$.
- $F_1^g = \{f_{y_0,y_1}^g\}_{(y_0,y_1) \in \{0,1\}^{2n}}$
  where $f_{y_0,y_1}^g : (x_1, \ldots, x_n) \longmapsto g_{x_n} \circ \ldots \circ g_{x_2}(y_{x_1})$.
- $F_i^g = \{f_{y_0,y_1,\ldots,y_{2^i-1}}^g\}_{y_0,y_1,\ldots,y_{2^i-1} \in \{0,1\}^{2^i n}}$
  where $f_{y_0,\ldots,y_{2^i-1}}^g : (x_1, \ldots, x_n) \longmapsto g_{x_n} \circ \ldots \circ g_{x_{i+1}}(y_{x_1 \ldots x_i})$
  (in the former expression $y_{x_1,\ldots x_i}$ represents $y_{\sum_{t=1}^{i} x_i 2^{i-1}}$)
- $F_n^g = \{f_{y_0,y_1,\ldots,y_{2^n-1}}^g\}_{y_0,y_1,\ldots,y_{2^n-1} \in \{0,1\}^{2^n}}$
  where $f_{y_0,\ldots,y_{2^n-1}}^g : (x_1, \ldots, x_n) \longmapsto (y_{x_1 \ldots x_n})$.

It is easy to see that F_0^g is equal to F^g, and that F_n^g is the set $F_{n,m}^*$ of all n-bit to m-bit functions.

Let us consider any (t, q) distinguishing algorithm A for F^g, i.e. a testing algorithm capable to query an n-bit to m-bit oracle function up to q times, and let us denote its distinguishing probability by

$$\epsilon = \left|\Pr_{f \in F^g}(A^f = 1) - \Pr_{f \in F^*_{n,m}}(A^f = 1)\right|.$$

We denote $\Pr_{f \in F^g_i}(A^f = 1)$ by p_i. Thus we have

$$\epsilon = |p_0 - p_n|.$$

We now construct a q-query distinguisher B for g, which when input with a q-tuple $(z_1, \ldots, z_q)$ of 2m-bit words is using one invocation of algorithm A to output either 0 or 1. In order to processes an input q-tuple $(z_1, \ldots, z_q)$, B first randomly draws an integer i comprised between 0 and $n-1$, and then inputs (z_1, z_q) to a testing algorithm B_i, and outputs B_i's binary output. Each testing algorithm B_i is defined as follows: B_i invokes algorithm A, and computes the answers to the up to q distinct n-bit oracle queries of A. For that purpose, B_i uses its random generation capability to simulate an auxiliary random function $\alpha : \{0,1\}^i \longrightarrow \{1, q\}$ that is initially undetermined. At each novel n bit oracle query $x^j = (x^j_1, \ldots, x^j_n)$ of A, algorithm B_i uses:

– the bits x^j_1 to x^j_i to determine a $2m$-bit value z_k as follows: B_i first checks if α is defined on point $(x^j_1, \ldots, x^j_i)$. If not, it selects randomly a value in $\{1, q\}$, affects it to $\alpha(x^j_1, \ldots x^j_i)$ and stores the new point of α. Otherwise B_i simply read the previously stored value. In both case, we denote by k the obtained value of $\alpha(x^j_1, \ldots x^j_i)$; k is used to select the k-th input z_k from B_i's input $(z_1, \ldots, z_q)$;
– the bit x^j_{i+1} to select an m-bit word y equal to the substring of the m left bits of z_k if $x^j_{i+1} = 0$, and of the m right bits of z_k if $x^j_{i+1} = 1$;
– the bits x^j_{i+2} to x^j_n to compute A^s L-bit oracle response $g_{x^j_n} \circ \ldots \circ g_{x^j_{i+2}}(y)$.

Finally when A halts, B_i halts also and returns A's binary output.
It is not too difficult to see that:

– if B_i's input is $(g(a_1), \ldots, g(a_q))$, where $(a_1, \ldots, a_q)$ is a randomly drawn q-tuple of m-bit, then A's oracle queries and response pairs have exactly the same probability distribution as if A were run with an n-bit to m-bit oracle function f randomly drawn from the family F^g_i:

$$\Pr(B_i((g(a_1), \ldots, g(a_q)) = 1) = \Pr_{f \in F^g_i}(A^f = 1) = p_i.$$

– if B_i's input is is a randomly drawn q-tuple $(z_1, \ldots, z_q)$ of $2m$-bit values, then A's oracle queries and response pairs have exactly the same probability distribution as if A was run with an n-bit to m-bit oracle function f randomly drawn from the family F^g_{i+1}:

$$\Pr(B_i(z_1, \ldots, z_q) = 1) = \Pr_{f \in F^g_{i+1}} = (A^f = 1) = p_{i+1}.$$

The above equalities imply:

$$\begin{aligned}&\left|\Pr(B((g(a_1),\ldots,g(a_q))=1)-\Pr(B(z_1,\ldots,z_q)=1)\right|\\&=\Big|\frac{1}{n}\sum_{i=0}^{n-1}p_i-\frac{1}{n}\sum_{i=1}^{n}p_i\Big|=\frac{1}{n}|p_0-p_n|=\frac{\epsilon}{n}.\end{aligned}$$

In other words:

$$\mathbf{Adv}_g^{prng}(B)=\frac{1}{n}\mathbf{Adv}_{F^g}^{prf}(A).$$

However, algorithm B requires at most $t+qnT_g^{2m}$, where t is the time required by A and $T_{g'}^{2m}$ is the time required by g' to produce $2m$ bits. Therefore for any A we have

$$\mathbf{Adv}_{F^g}^{prf}(A)\le n\mathbf{Adv}_g^{prng}(t+qnT_g^{2m},q).$$

Finally, since $\mathbf{Adv}_g^{prng}(t+qnT_g^{2m},q)\le q\mathbf{Adv}_g^{prng}(t+q(n+1)T_g^{2m})$ due to Lemma 1, we obtain

$$\mathbf{Adv}_{F^g}^{prf}(t,q)\le qn\mathbf{Adv}_g^{prng}(t+q(n+1)T_g^{2m}).\square$$

Main Reduction Theorem of [3]

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'.*

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