Two Dragons
A Family of Fast Word-based Stream Ciphers

Matt Henricksen
*Institute of Infocomm Research, A*STAR, Singapore, Singapore*

Keywords: Dragon, Stream Ciphers, AES-NI, Cryptology.

Abstract: The EU eSTREAM competition selected two portfolios of stream ciphers, from among thirty-four candidates, with members that were either fast in software or compact in hardware. Dragon was among the eight finalists in the software category. While meeting the performance requirement of being faster than the Advanced Encryption Standard (AES) on many platforms, it was less efficient than the four ciphers selected for the portfolio. Cryptanalysis revealed some less-than-ideal properties. In this paper, we provide some new insights into Dragon, and propose two modifications: Black Dragon, which is tailored for efficient implementation in modern SIMD architectures; and Yellow Dragon, which utilizes recent developments in Chinese block ciphers. We show the improved security and performance of these two variants.

1 INTRODUCTION

Symmetric ciphers are primitives that form the backbone of information security by providing an efficient way to encrypt and authenticate data. They comprise stream ciphers, which process a secret state, and stateless block ciphers. Stream ciphers can be faster and more compact than block ciphers.

The most famous symmetric cipher is the Advanced Encryption Standard (AES) (Daemen and Rijmen, 2002), ratified by the US' National Institute of Standards in 2001 after a competition that saw cryptanalysts evaluating fifteen block ciphers. The winning candidate, Rijndael, has high efficiency, provable security against common attacks, and an elegant design.

Following the model of the AES competition, the EU ECRYPT node of excellence held several competitions for new stream ciphers. The first competition failed to find candidates with sufficient security. The second, the ECRYPT eSTREAM project, had more success in identifying stream ciphers at least as secure as the AES when used with a 128-bit key, but faster in software; or smaller than the AES in hardware, while achieving at least 80 bits of security. The competition yielded two portfolios: one for hardware, with three members, and one for software, with four members.

There were many finalists that were not selected for various reasons. The final report on the eSTREAM finalists said:

> Dragon cipher appears to be of solid construction. The downside to the current design of Dragon, is that while its performance is competitive with the AES, it does not compare too well to the other submissions in the final phase. It would certainly be interesting if a future version of the cipher were able to maintain some of the successful design ideas while delivering even better performance" (eSTREAM, 2008)

Delivering better performance for equivalent security is the motivation for this paper. It is made timely by the delivery of a ubiquitous and very fast implementation of a 32×32 non-linear mapping that can be used to replace the weakest component of Dragon[1].

In Section 2, we discuss the original design of Dragon, as presented to eSTREAM, and in Section 3, the problems with the cipher. In Section 4, we present two new Dragons: Black Dragon, which makes use of the new AES-NI instruction set, and Yellow Dragon, which utilizes the structure of the SMS-4. In Section 5, we indicate our reasons for these designs. In Section 6, we give an analysis of their security, and in Section 7, performance benchmarks and implementation notes. We conclude in Section 8.

[1] On a light note, the timing is also impeccable as 2012 is the year of the Dragon

Henricksen M..
Two Dragons - A Family of Fast Word-based Stream Ciphers.
DOI: 10.5220/0004014000350044
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 35-44
ISBN: 978-989-8565-24-2

2 THE DRAGON CIPHER

Dragon (Chen et al., 2004) is a stream cipher with a conservative design. It is based on a 1,024-bit NLFSR, with an additional 64-bit register intended to give assurances about the cipher period. A 192×192-bit function F serves both as the update function on the state, and the output filter. The state is initialized by a 128- or 256-bit key, using slightly different methods, both of which use F.

2.1 The F function

The F function is shown in Figure 1. It has three layers. The first and third layers perform word-based diffusion, using binary addition ($\oplus$) and addition modulo 2^{32} ($\boxplus$) while the second introduces strong nonlinearity through the G and H functions.

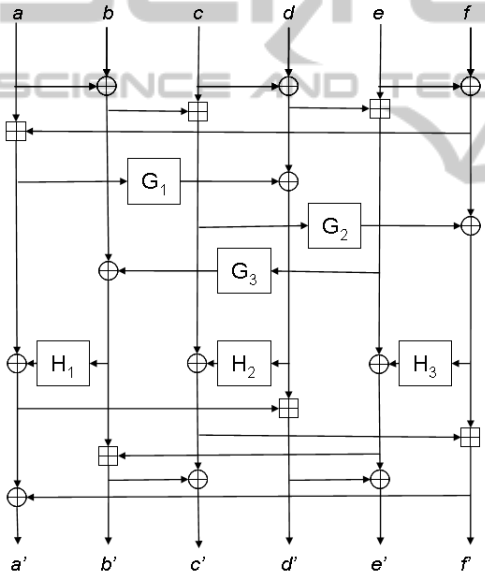

Figure 1: Schematic of Dragon's F Function.

The G and H functions are 32×32-bit mappings based on a set of heuristically designed 8×32-bit s-boxes S_1 and S_2. The construction of the functions is simple. For $x = x_0|x_1|x_2|x_3$, $G/H(x) = S_a(x_0) \oplus S_b(x_1) \oplus S_c(x_2) \oplus S_d(x_3)$, where each S represents one of two s-boxes S_1 or S_2. In each of the six functions G/H, either of the s-boxes is used three times, and the other once. This asymmetry prevents cancellation in a byte-symmetric input word.

2.2 Key Initialization Algorithm

Due to a historical quirk, Dragon has different key initialization algorithms for its two key sizes of 128- and 256-bits. During key initialization, the 1,024 bit NLFSR state W is divided into eight 128-bit words. For the 256-bit master key, the state is filled by concatenating K and IV with their bitwise sum and complement such that $W = K \parallel K \oplus IV \parallel \overline{K \oplus IV} \parallel IV$, with the over-score representing complementation. For the 128-bit master key, the state is filled by $(K||K' \oplus IV'||IV||K \oplus IV'||K'||K \oplus IV||IV'||K' \oplus IV)$ where x' denotes swapping of the upper and lower halves of the 128-bit quantity x. The 64-bit register is filled with a constant representing the ASCII representation of 'Dragon'.

The cipher is clocked sixteen times. During each clock, the the six input words to the F function are chosen as 128-bit quantity $(W_0 \oplus W_6 \oplus W_7)$ and 64-bit quantity M. Every word in the NLFSR is shifted one place to the right, the topmost word being discarded. The void at the bottom-most position in the NLFSR is filled by four of the six 32-bit output words of F. The remaining two words overwrites the contents of M. The phase completes when every stage of the NLFSR has been overwritten twice.

2.3 Keystream Generation Algorithm

Each clock of the cipher during keystream generation mode produces a 64-bit keystream word, which is chosen as the concatentation of the F output $a'||e'$. The output $b'||c'$ is used as feedback to the NLFSR. Outputs d and f are discarded (so in the keystream generation mode, the function F can be modelled as a $192 \rightarrow 128$-bit function). During each clock, the memory M is incremented by 1.

Fortunately, the designers of Dragon stipulated that after 2^{64} bits of keystream have been generated under a single key-IV pair, the key initialization algorithm must be executed with a new key-IV pair.

3 WHAT'S WRONG WITH DRAGON?

The straight answer to the question 'What's wrong with Dragon?' is 'nothing really', except that it doesn't do anything better than its competitors. It was scrutinized for a long time by cryptanalysts during the eSTREAM competition, with only a theoretical distinguishing attack, which could not occur under the correct usage model, being noted.

But in order that anyone other than its designers use it, however, improvements need to be made in two areas - performance, and security.

Table 1: Number of s-box lookups per 32-bit word of keystream.

Cipher	S-box size	Number lookups
Dragon	8×32	12
SNOW 2.0	8×8	4
HC-128	9×32	8
Rabbit	None	0
Sosemanuk	Bitsliced	0
Salsa-20	None	0

3.1 Performance

In the context of the eSTREAM competition, Dragon has two problems with performance. The first is that it is a cipher designed for a 256-bit key, but retrofitted with a key initialization algorithm for a 128-bit key. If it competes with a cipher designed for a 128-bit key (eg. eSTREAM finalists HC (Wu, 2008) or SOSEMANUK (Berbain et al., 2008) [2]), it is unsurprising it will be slower.

The second problem is the design of its G and H functions, which provide most of the strength in the cipher while also being quite slow.

3.1.1 S-boxes

On most architectures, look-up tables are not atomic operations like additions, multiplications or exclusive-ors. They are usually composed from three or four instructions, including retrieving data from memory, which incurs a penalty if the data is not already in the cache.

Table 1 compares the number of s-box lookups, among Dragon and the eSTREAM software portfolio members, per 32-bit word of keystream. Dragon uses the largest number of s-boxes. Most of the portfolio members do not use s-boxes, or provide ways in which to implement them using logical operations.

It is not only the number of s-boxes that is important, but also whether they can be parallelized, to alleviate some of the stress on the execution ports responsible for retrieving data from memory. In Dragon, all of the s-box lookups are localized to the middle 'S-box' layer of F. Contemporary super-scalar architectures can execute multiple additions or exclusive-ors in a single cycle, but only one s-box at a time. The middle layer of F is the bottleneck.

One way to fix this problem is to reduce the number of s-boxes, but a better solution is to properly implement a sufficient number of s-boxes to provide a good amount of security.

3.1.2 State Size

Dragon was originally designed for a 256-bit key. The state size of 1,088 reflects this, being just a bit larger than double the combined length of key and IV, to protect against time-memory-data tradeoff attacks.

The 128-bit key initialization scheme was retroactively fitted to the cipher in order to meet eSTREAM requirements. The state size should have been reduced accordingly, but was not. So the state is very large, and uses many gates in hardware unnecessarily.

The key agility of Dragon for a 128-bit key also suffers, since the design principle states that every stage of the NLFSR must be modified twice during key initialization. Since there are twice as many stages as necessary, the key agility of Dragon with a 128-bit key is unnecessarily halved.

3.2 Security

3.2.1 S-boxes and Linear Cryptanalysis

The s-boxes for Dragon were designed by experts in with expertise in boolean functions and genetic algorithms. They used genetic algorithms to optimize the s-boxes, with respect to non-linearity (116), algebraic degree (6 or 7), and a range of other properties. One thing they did not optimize for was resistance to linear cryptanalysis.

Cho (Cho, 2008) points out that for input mask 0 and output mask 0x61300000, the number of masked inputs to s-box S_1 that have the same parity as the masked outputs is only 92. In other words, the bias of the s-box is $2^{-1.83}$. The same bias occurs for S_2 with output mask 0x60020300. Compare this to the maximum bias of the AES 8×8 s-box, which is 2^{-3}. The resistance of the Dragon s-boxes to linear cryptanalysis, despite the sophisticated design techniques, is terrible.

Englund and Maximov (Englund and Maximov, 2005) note that G and H, although being $\mathbf{Z}_{2^{32}} \rightarrow \mathbf{Z}_{2^{32}}$ functions, cannot be bijective because of the way that they are constructed using $\mathbf{Z}_{2^8} \rightarrow \mathbf{Z}_{2^{32}}$ sub-functions (the S_1 and S_2 s-boxes). Our own experiments show that only 37% of the possible outputs are reached through each G and H function.

The best attack on Dragon (Cho, 2008) utilizes both of these points, along with newly developed analysis of adjacent modular additions. Given 2^{133} words of keystream generated under a single key-IV pair, the attacker can use the bias of about 2^{-36} in F

[2] SOSEMANUK accepts a 128-bit key or a 256-bit key but is vulnerable to guess and determine attacks with complexity O(2^{176}) (Feng et al., 2010), indicating it is only suitable for use with 128-bit keys

to distinguish the keystream from random. It is a theoretical attack. If the attacker adheres to the usage model of Dragon, he can never obtain this amount of keystream. If he abuses the usage model, he is able to distinguish Dragon from random, but not to recover the key or predict keystream. Such an attack is of limited usefulness, other than for highlighting the weak points of Dragon, ie. the s-boxes.

The technique uses two types of approximations, including applying 'bypassing' by approximating s-boxes that are not influenced by other s-boxes, and 'cutting' by setting the input masks of s-boxes influenced by other s-boxes to zero. This means that the contribution of the other s-boxes to the 'cut' s-boxes can be ignored. If the s-boxes are bijective, cutting does not work, since setting the input mask to zero gives a bias of zero on any output mask

3.3 Elegance

3.3.1 Key Initialization Algorithm

The initial population of state using combinations of key and initialization vector differs quite arbitrarily according to the size of the master key. The designers of Dragon (Chen et al., 2004) state that this is so to avoid 256-bit key/IV pairs mapping to 128-bit key/IV pairs, leading to a reduction of the key space. The solution is inelegant. It is easy to achieve the same ends more naturally by incorporating the length of the key into the initial state. Having a unified key initialization algorithm also allows simpler code, reduced space in hardware, and scalability to other key sizes.

3.3.2 The Memory

Unlike ciphers based on LFSRs with primitive feedback polynomials, *n*-bit NFLSRs do not give guarantees on lower bounds of period, but instead provide expected periods of $2^{n/2}$.

The designers of Dragon indicate that the 64-bit counter M used during keystream generation gives a lower bound to Dragon's period of 2^{64}. This is a heuristic argument that holds limited usefulness, since it doesn't combine the properties of the counter successfully with the properties of the remaining parts of the cipher.

As Englund and Maximov indicate (Englund and Maximov, 2005), the left half of the counter, which changes very slowly with time, shifts the distribution of samples made during linear cryptanalysis attacks, but does not change the bias. It is better that M retains its use as an unpredictable counter during keystream generation. Even without the counter, it is unlikely that a 1024-bit NLFSR with a bijective feedback function has short cycles.

4 TWO DRAGONS

The term Dragon-2 encompasses the family of stream ciphers that we are about to describe here. We present six variants that vary in the size of the key and nonlinear function.

The Dragon-2 state comprises an NLFSR W and a 128-bit memory M. A series of algorithms operates on the state. The key initialization algorithm uses a key and initialization vector (IV) to populate and mix the state. The key can be 80-, 128- or 256-bits. The IV must be the same length as the key. We refer to versions of Dragon-2 with 80-bit, 128-bit and 256-bit keys respectively as Dragon-80, Dragon-128, and Dragon-256.

The size of the NLFSR is proportional to the size of the key. Respectively, for 80-bit, 128-bit and 256-bit keys, the size s of the NLFSR is three quadwords (384 bits), four quadwords (512 bits) and eight quadwords(1024 bits).

The keystream generation algorithms produce a 128-bit blocks of keystream. These algorithms include a feedback function, which modifies part of the state, and an output filter which processes part of the state in a non-invertible way to produce the block of keystream. The feedback function and output filter are more decoupled than in Dragon-1.

Both the key initialization and keystream generation algorithms make use of a 256×256-bit function F, which is described in Section 4.3, along with its sub-functions F_1 and F_2.

4.1 Key Initialization Algorithm

Dragon-2 has a two-phase initialization process. The first phase adds the key to the state, which it mixes. The second phase adds the IV to the state, and again mixes. Separating the addition of key and IV to the state means that IV-only rekeying can be made much more efficient.

To permit a single procedure irrespective of key size, extended keys and IVs are created. Extended key EK is generated from the supplied key K as as $EK = K||K \lll 32||K \lll 64||K \lll 96$ (where $\lll$ is bitwise rotation) then truncated to s quadwords. Likewise, the extended IV is generated as $EIV = IV||(IV \lll 32) \boxplus 1||(IV \lll 64) - 1||(IV \lll 96) \boxplus 2^{127} - 1$, then truncated to s quadwords.

The key initialization algorithm, using key K of length len, is shown in Table 2.

Table 2: Dragon-2 two-phase key initialization algorithm.

Input: EK, EIV		
Phase 1: Key Injection		
1.	$W_0 = EK_0 \oplus len$	$W_i = EK_i, 1 \le i \le s-1$
2.	$M = 0$	
Perform steps 3-5 (s) **times**		
3.	$O_F = F_1(W)$;	$O_Z = F_2(W,M)$
4.	$W_0 = O_F$;	$W_i = W_{i-1}, 1 \le i \le s-1$
5.	$M = O_Z$	
Phase 2: IV Injection		
6.	$W_i = W_i \oplus EIV_i$	$0 \le i \le s-1$
Perform steps 3-5 (s) **times**		
Output: W, M		

Table 3: Dragon-2 keystream generation algorithm.

Input: W, M		
1.	$O_F = F_1(W)$	$z = F_2(W,M)$
2.	$M = M \boxplus W_{s-1}$	
3.	$W_0 = O_F$;	$W_i = W_{i-1}, 1 \le i \le s-1$
Output: W, M, z		

4.2 Keystream Generation Algorithm

After the key initialization mode has finished, the keystream generation algorithm can be invoked. Each time it is invoked, it produces a 128-bit keystream block z. The procedure for generating the keystream is shown in Table 3.

Since the attacker uses keystream z as a window into the state, the client is free to mask out part of z to improve security at the expense of throughput.

Rekeying, using at least a fresh IV, must be carried out after at most $min(2^{\frac{len}{2}}, 2^{64})$ bits of input have been generated for any key-IV pair.

4.3 The F function

The function F comprises two sub-functions, the $128 \times 128-$bit feedback function F_1, and the $256 \times$ 128-bit output filter F_2. In these function, as per Dragon-1, diffusion is provided by a network of modular and binary additions, denoted by $\boxplus$ (mod 2^{32}) and $\oplus$ respectively; and confusion is provided by G. In later sections, we specify two Dragons, Black and Yellow, using two well-known functions for G.

The feedback function F_1 is shown in Table 4. This function is independent of the feedback function F_2. For Dragon-80, the input I_F is formulated as $W_0 \oplus W_2$; for the other variants, it is formulated as $W_0 \oplus W_{(s/2-1)} \oplus W_{(s-1)}$. The G function takes the all-zero string as its second parameter.

The output filter F_2 is shown in Table 5. This function uses the output of the feedback function, O_F, as input parameter K. Other inputs include memory M, and I_Z, which is formulated as $W_{\lfloor s/2 \rfloor}$.

The combination of sub-functions F_1 and F_2 into F is shown in Figure 2.

Table 4: The feedback function F_1.

Input: $I_F = \{a,b,c,d\}$		
S-box Layer:		
1.	$(g_0,g_1,g_2,g_3) = G(a\|\|b\|\|c\|\|d,0)$;	
Mixing Layer:		
2.	$b = b \oplus a$;	$d = d \oplus c$;
3.	$c = c \boxplus b$;	$a = a \boxplus d$;
4.	$a = a \oplus g_0$;	$b = b \oplus g_1$;
	$c = c \oplus g_2$;	$d = d \oplus g_3$;
Output $O_F = \{a',b',c',d'\}$		

Table 5: The output filter F_2.

Input: $I_Z = \{e,f,g,h\}, M = \{M_0\|\|M_1\|\|M_2\|\|M_3\}, K$		
Mixing Layer:		
1.	$e = e \oplus M_0$;	$f = f \oplus M_1$;
	$g = g \oplus M_2$;	$h = h \oplus M_3$;
2.	$f = f \oplus e$;	$h = h \oplus g$;
3.	$g = g \boxplus f$;	$e = e \boxplus h$;
4.	$f = f \oplus e$;	$h = h \oplus g$;
5.	$g = g \boxplus f$;	$e = e \boxplus h$;
S-box Layer:		
6.	$(e',f',g',h') = (e,f,g,h) \oplus G(a\|\|b\|\|c\|\|d,K)$;	
Output: $O_Z = \{e',f',g',h'\}$		

4.4 黑龙 Black Dragon

Black Dragon is defined as a version of Dragon-2 that uses the AES round function for G.

This round function consists of four operations: *ByteSub*, which applies sixteen 8×8 s-boxes to the 128-bit input, followed by *ShiftRows* and *MixColumn* which diffuse bytes within four groups, followed by *AddKey*, which exclusive-ors the result with another 128-bit quantity (the key) to produce the 128-bit output.

Please refer to (Daemen and Rijmen, 2002) for details on how to implement G for Black Dragon.

4.5 黄龙 Yellow Dragon

Yellow Dragon is defined as a version of Dragon-2 that uses the SMS-4 round function within G.

SMS-4 is the block cipher used in the Chinese National Standard for Wireless Local Area Networks (WLANs) Wired Authentication and Privacy Infras-

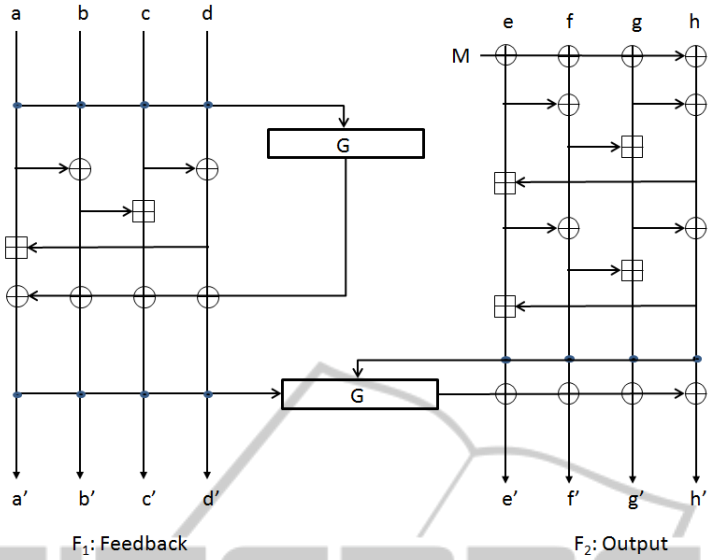

Figure 2: Schematic of Dragon-2's F Function.

tructure (WAPI) as an alternative to the American ratified AES.

In SMS-4, the round function comprises four layers. The first compresses the 96-bit source into a 32-bit word using exclusive-or. In the second, a 32-bit round key is added. In the third layer, a row of parallel 8×8 s-boxes, different to the AES s-box, but designed using a similar methodology, adds non-linearity. The fourth is the linear 32×32-bit sub-function $L(x) = x \oplus (x \lll 2) \oplus (x \lll 10) \oplus (x \lll 18) \oplus (x \lll 24)$.

In Yellow Dragon, the round function is iterated four times within the G function, using the same unbalanced Feistel sub-structure, with 96-bit source and 32-bit target, as does SMS-4. The 128-bit key is divided naturally between the four invocations of the round function.

Please refer to the SMS-4 specification document (People's Republic of China Office of State Commercial Cryptography Administration, 2006) for details on how to implement G for Yellow Dragon.

5 DESIGN PRINCIPLES

The design of Dragon-2 is strongly influenced by what we perceive to be weak about Dragon-1, and also for easy of implementation in software. The guiding principle is that it can implemented on modern SIMD architectures, such as the Intel x86. The i7 we used for development has sixteen 128-bit XMM registers, and hundreds of operations that apply to them, including the AES-NI, which implement an AES round instruction with latency of eight cycles, and throughput of four cycles (Fog, 2011). This provides a safe and efficient way to use good s-boxes, circumventing most of the problems with Dragon-1.

5.1 State

5.1.1 NLFSR

We don't like that the state size of the NLFSR in Dragon-1 remains the same for different sized keys, but the algorithm changes. In hardware, a 1024-bit state is very large, and is only justified by a large sized key. There is no security reason for having a 1024-bit state for a 128-bit key. Time-memory-data tradeoff attacks suggest that 512-bits is the sweet spot. Consequently, in Dragon-2, the size of the state varies appropriately with the size of the key.

The size of the NLFSR is the lowest multiple of 128 bits that is equal to or greater than four times the size of the key. This permits the cipher to resist time-data-memory trade-off attacks, and to be implemented easily in 128-bit registers.

All of the operations that apply to the NLFSR, with the exception of the G function in Yellow Dragon, are designed for implementation on the Intel x86-64 SIMD architectures. However, there are no operations that cannot be implemented efficiently using general purpose 32-bit registers.

When the NLFSR is divided into quadwords, the choice of words to use as input into the F function becomes quite constrained. This is different to Dragon-1, in which the choice of 32-bit words is according to a full positive difference set, to maximize resistance against guess and determine attacks. But in the latter

approach, many different words need to be marshalled and prepared for input into F. In the Dragon-2 approach, no such manipulation is required. The words are present in XMM registers, or in hardware, to be presented to the F function at no additional cost.

5.1.2 Memory

The behaviour of the memory M is different in key generation than it was in Dragon-1. In Dragon-1, the memory behaved as a counter. As illustrated in Section 3, it is better for it to behave pseudo-randomly.

One reason is that in Dragon-2, the attacker needs to guess 128 bits (the contribution of the feedback function to the output filter) in order to derive 256 bits of state. Of this, 128 bits is combined NLFSR-memory material. If the memory acts as a counter, then the attacker can repeat the guess at a known time, in order to derive a further 256 bits of material, then untangle the 512 bits of NLFSR material from the 256 bits of counter material. Although he has guessed equivalent amounts of material as the master key, one of our principles is to limit the amount of material gained in this way. Implementing M as a unpredictable memory rather than a counter means that, using this method, the attacker has to guess x bits of material in order to gain x bits of state, where x is a multiple of 128 bits.

5.2 The Key Initialization Algorithm

In Dragon-2, the key initialization algorithm is unified across key sizes, rather than arbitrarily different, as in Dragon-1. We use byte shuffling rather than complementation to pad the key, since this is easier to implement in hardware. So long as the key and IV are loaded across the state in subtly different ways, the choice of operations is not very important. The aim is to spread every bit of the key into every bit of the state as quickly as possible.

We also changed the algorithm to be two phase to enable quicker IV-refreshing. Quick diffusion in key initialization is enabled by choosing multiple NLFSR words, which are combined using exclusive-or, as input into the feedback function.

5.3 The Keystream Generation Algorithm

In Dragon-1, the key initialization algorithm and keystream generation algorithm both use the F function, but otherwise they are both quite different. This wastes space in hardware, and makes the cipher more difficult to program. The division of the NLFSR into

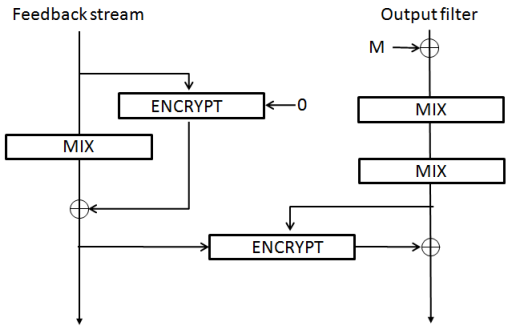

Figure 3: Abstract view of Dragon-2's F function.

different quantities - either 32-bit or 128-bit, depending on mode, is also awkward.

In Dragon-2, the key initialization algorithm and keystream generation algorithm are almost but not exactly the same, in order to save hardware space, but avoid slide-like attacks.

5.4 The F Function

In Dragon-1, the F function separates parts of the output of a monolithic 192×192 function for feedback and keystream. There is no separate output filter. A 192-bit block is unnatural to deal with. With advent of SSE, it is easier to deal with 2^x-bit blocks.

In Dragon-2, we made the decision to decouple the feedback and output filter modules. An abstract of the design is shown in abstract in Figure 3, The feedback function is bijective. The output function is deliberately non-invertible, due to the addition of whitening material from the feedback function.

While the output filter is post-whitened using material from the feedback function, the output filter has no direct influence on the feedback function. This is to reduce the ability of the attacker to guess parts of the internal state.

During one clock of the F, thirty-two 8×8 s-boxes are invoked, compared to twenty-four invocations of 8×32 s-boxes in the original Dragon. In both Black Dragon and Yellow Dragon, the s-boxes are bijective, and of much better quality.

5.4.1 The F_1 Function

The F_1 function does not provide full diffusion. By the time its material is exposed to an attacker, it has been further processed by F_2, which does provide full diffusion. This enables the function to remain efficient.

The feedback function must be a bijection in order to avoid loss of entropy. Provided that G is a bijection, then the feedback function will also be a bijection. For this reason, G is keyed with a null key.

There is no difference in choosing the contribution to F_2 from the start or end of the function, since the material at the end of one function is the same as at the start of another.

5.4.2 The F_2 Function

By including two mix modules, and one G module, the F_2 function provides full diffusion.

It is not possible to use a fixed key with G if it is the last operation in the output filter, since the entire sub-function could be inverted. By using pseudo-random material from the feedback function as the key, the attacker has to guess a large amount of material from the state to invert the function.

For Black Dragon, which uses the AES round, this material is exclusive-ored as the last operation. The attacker is forced to guess the 128 bits of state used as input into the feedback filter in order to utilize the 128 bits of output to unravel to the 128 bits of state and 128 bits of memory used as input to the filter.

6 ANALYSIS

6.1 Time-Memory-Data Tradeoff Attacks

Time-Memory-Data Tradeoff (TMDT) attacks use a pre-computation to compute tables that, in an on-line phase, reduce the time needed to identify the key or contents of the cipher state, by using keystream sequences as an index. Generally, a tradeoff can be made by varying the parameters in the equation $N^2 = TM^2D^2$ where $T \geq D^2$ (Biryukov and Shamir, 2000), N is the size of the combined key-IV or the size of the state, T is the amount of time used to identify the key/state, M is the amount of memory required, and D is the number of data points. Clearly, increasing the size of N also increases the value of at least one of the parameters.

The rules-of-thumb for defending against TMD attacks is to use an IV equal in length to the key, and a state at least twice the size of the combined key-IV. As the NLFSR size of Dragon-2 was chosen specifically in relation to the size of the key, with the 128-bit memory providing an additional margin, the cipher is not vulnerable to this kind of attack.

6.2 Related-key Attacks

Dragon-2 uses the optimal branch number of the diffusion components in G to resist related-key and related-IV attacks. When analysing the effect of related-keys and IVs, we can ignore the output filter since it does not effect the NLFSR. When the NLFSR is thoroughly mixed, the attacker loses control over manipulating the keystream.

The simplest differential through F is $(0,0,0,0,\Delta,0,0,0)$, where $\Delta = 0x8000$. For a difference in which only the most significant bit is set, additions behave like exclusive-ors, and it is easy to cancel like differences. The output difference for this differential is $(0,0,0,0,0,\Delta^1,\Delta^2,\Delta^3)$, with two s-boxes activated (ie. a maximum probability of 2^{-12}). This differential would permit a zero difference to be maintained in the NLFSR throughout key initialization. However, this high probability differential cannot be used, because of the way in which the initialization vector is set into state, and the high diffusion of the NLFSR.

Assume that $\Delta K = 0$ and $\Delta IV = \Delta << 96$. Also assume the simplification that no constants are added to the IV during step 6 of Table 2. The state is set as $(\Delta IV, \Delta IV, \Delta IV, \Delta IV)$. The input I_F to sub-function F_1 is $(\Delta,0,0,0)$. F_1 provides incomplete diffusion, but its output at time t forms part of its input at time $t+1$. So the difference introduced has been modified by two invocations of the G function by time $t+2$. Two adjacent rounds of G propagate a single byte difference into all positions into the quadword. Subsequently all sixteen s-boxes are likely to be activated, with a maximum probability $2^{-21\times 6}$. Four adjacent rounds activate at least twenty-five s-boxes, which is sufficient to defeat differential cryptanalysis for Dragon-80 and Dragon-128. As Dragon-256 invokes sixteen rounds during key mixing state, we would expect activation of at least seventy-five boxes to influence all positions in the NLFSR, meaning any differential would have a maximum probability of $2^{-6\times 75}$, well below a successful probability of 2^{-256}.

The addition of constants during the IV-keying phase, to disrupt the attacker's ability to cancel parts of the IV difference within the diffusion network, and the weak non-linearity of the modular additions in the F function are expected to further weaken the probability of any differential.

6.3 Linear Cryptanalysis

Linear cryptanalysis has been the most effective attack on the Dragon structure, due in part to poor s-boxes with a maximum bias of $2^{-1.83}$. In order to attack Dragon with linear cryptanalysis using the same techniques of Cho (Cho, 2008) under the correct usage model, the F function must have a bias of greater than 2^{-16}. As each s-box has a maximum bias of

2^{-3}, no more than five s-boxes of the thirty-two s-boxes can be activated in any successful approximation. The branch number of the linear components in the G functions is five, meaning that the sum of active input and output bytes for each G is at least five. The mandatory incorporation of two G functions in the F function means that a maximum bias of 2^{-16} is not achievable.

6.4 Guess and Determine Attacks

Analysing Dragon-2 against guess-and-determine attacks is uncomplicated because of its large word size, and the diffusion network that means guessing bytes or 32-bit words quickly involves guessing as much material as in a quadword. For Dragon-80 or Dragon-128, guessing quadwords is unproductive.

For Dragon-256, the best case for the attacker is to guess the combination of input and memory to the output filter at time t. As he does not know the value of the memory, this does not immediately give insight into the NLFSR state. But in conjunction with the keystream, it allows him to calculate the input and output to the feedback function, permitting him to know the first word of the NLFSR at time $t+1$. As both the feedback function and output filter use multiple words to construct inputs, he cannot use this knowledge unless he guesses another word, by which time his cumulative guessing is equal to the effort of guessing the master key.

6.5 Algebraic Attacks

Both Black Dragon and Yellow Dragon use s-boxes constructed algebraically, so they might seem to be good candidates for algebraic analysis, by constructing a system of equations that can be solved to recover the internal state.

However, the G functions are embedded in a network containing many 32-bit modular additions. As shown in analysis of SNOW 2.0 (Billet and Gilbert, 2005), which like Black Dragon also uses the AES round function, the prolific use of additions is an effective deterrent to algebraic attacks, since they cause the degree of the collected equations to increase very quickly. Coupled with the need to refresh the key-IV pair every 2^{64} bits, and the poor success rate of algebraic analysis to word-based ciphers, it seems very unlikely that Dragon-2 will be vulnerable.

6.6 Cache-timing Attacks

When implemented using AES-NI, Black Dragon does not perform any memory lookups in conjunction with s-boxes so cache-timing attacks cannot apply. Yellow Dragon is intended to be used with similar hardware support for the SMS-4 round function, so similar reasoning applies.

Even without hardware support, the F network provides some mitigation against cache-timing attacks by ensuring that the output of the G function, which is the only place that might contain s-boxes, never overwrites any values, but is combined to them using exclusive-or. If an attacker learns half the bits in n-bit value X, he will need to guess the equivalent amount of bits in a n-bit value Y to learn the corresponding portion of $Z = X \oplus Y$. He has obtained no advantage over directly guessing the bits in Z.

Table 6: Speed of stream ciphers on the Intel i7 (cycles/byte).

Cipher	Long message	4,096 bit message
Black Dragon	3.6	3.8
Dragon	8.8	9.2
HC-128	2.2	7.2
Rabbit	4.8	5.1
Salsa-20	1.8	1.8
SNOW 2.0	3.7	3.8
Sosemanuk	2.5	3.2

7 IMPLEMENTATION

The current speeds of Dragon-2, eSTREAM software portofolio members and benchmark cipher SNOW 2.0 are shown in Table 6 for the Intel i7. Figures for the ciphers, excluding Dragon and Black Dragon, come from the Vampire benchmarking site(VAMPIRE - Virtual Applications and Implementations Research Lab, 2012). We note that Black Dragon is faster than the benchmark cipher SNOW 2.0, and significantly faster than Rabbit and Dragon. It is also worth noting that much effort has gone into optimizing the relatively mature eSTREAM ciphers, compared to Dragon-2, and we would expect the most improvement in future optimization to occur in our cipher.

It is very likely that the core function of SMS-4 will be implemented in hardware and as native instructions on Chinese-built processors in the future. This will provide similar advantages to the implementation of Yellow Dragon, as for Black Dragon.

It is interesting to note that of the ciphers listed here, the AES-NI only provides benefits to SNOW 2.0 and Black Dragon. Where AES-NI is not available, Black Dragon can be implemented in software using the efficient tabular approach of the AES.

8 DISCUSSION

In this paper, we have documented improvements to the Dragon cipher, and presented them as Dragon-2. Users of Dragon-2 have a choice to use the AES round function within the cipher, gaining the benefit of the recent AES-NI set, which provides blisteringly fast encryption support. This version of the cipher is called Black Dragon. Alternatively, the user can specify four iterations of the SMS-4 round function, in which case the cipher is called Yellow Dragon.

Dragon-2 represents a relatively conservative design that benefits from good hardware support. In the future, once more widespread analysis has been conducted, it might be considered a viable alternative to the eSTREAM software portfolio members.

For example, HC-128 is very fast for long messages, but its much larger state requires a long time for rekeying, and so it is not as agile as Dragon-2. Depending on the application, Dragon-2 might be preferable.

Salsa-20 relies on the iterated weak non-linearity of addition, compared to the proven high non-linearity of the AES and SMS-4 s-boxes. Relying on the properties of a single operation does not provide robustness. Advances in differential cryptanalysis are likely to weaken Salsa-20. Dragon-2 is a more conservative and equally efficient choice.

Sosemanuk has been shown to be unable to provide 256 bits of security; it is not termed broken only because its designers specified 128 bits of security even for the larger key. Rabbit appears to be strong, but only accepts a 128-bit key (Robshaw and Billet, 2008). In either case, if Dragon-2 withstands cryptanalysis, it is a stronger choice.

There is still much work to do on Dragon-2. We continue to cryptanalyse it, but it must be scrutinized by impartial cryptographers. We have met the remark made in the eSTREAM report, as presented in the first section of this paper. Due to space limitations in this paper, test vectors are available upon request.

REFERENCES

Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., and Sibert, H. (2008). SOSEMANUK, a Fast Software-Oriented Stream Cipher. In (Robshaw and Billet, 2008), pages 98–118.

Billet, O. and Gilbert, H. (2005). Resistance of SNOW 2.0 against algebraic attacks. In Menezes, A. J., editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 19–28. Springer.

Biryukov, A. and Shamir, A. (2000). Cryptanalytic time/memory/data tradeoffs for stream ciphers. In Okamoto, T., editor, *Advances in Cryptology - Proceedings of Asiacrypt 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer.

Chen, K., Henricksen, M., Millan, W., Fuller, J., Simpson, L. R., Dawson, E., Lee, H., and Moon, S. (2004). Dragon: A fast word based stream cipher. In Park, C. and Chee, S., editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 33–50. Springer.

Cho, J. Y. (2008). An improved estimate of the correlation of distinguisher for Dragon. In *SASC2008*, pages 11–20, Lausanne, Switzerland. Special Workshop hosted by the ECRYPT Network of Excellence. Proceedings available at http://www.ecrypt.eu.org/stvl/sasc2008/.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer.

Englund, H. and Maximov, A. (2005). Attack the dragon. In Maitra, S., Madhavan, C. E. V., and Venkatesan, R., editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 130–142. Springer.

eSTREAM (2008). Third phase report. At http://www.ecrypt.eu.org/stream/index.html.

Feng, X., Liu, J., Zhou, Z., Wu, C., and Feng, D. (2010). A Byte-Based Guess and Determine Attack on SOSEMANUK. In *ASIACRYPT'10*, pages 146–157.

Fog, A. (2011). Instruction tables. Lists of instruction latencies, throughputs and microoperation breakdowns for Intel, AMD and VIA CPUs. At www.agner.org/assem/.

People's Republic of China Office of State Commercial Cryptography Administration (2006). The SMS4 Block Cipher. Archive available at http://www.oscca.gov.cn/UpFile/20062101642319799 0.pdf (in Chinese).

Robshaw, M. and Billet, O., editors (2008). *New Stream Cipher Designs: The eSTREAM Finalists*. Number 4986 in Lecture Notes in Computer Science. Springer.

VAMPIRE - Virtual Applications and Implementations Research Lab (2012). eBACS: ECRYPT Benchmarking of Cryptographic Systems. http://bench.cr.yp.to/results-stream.html.

Wu, H. (2008). The stream cipher HC-128. In (Robshaw and Billet, 2008), pages 39–47.