FALCON: A Fast 256-Bit Block Cipher

Christian Daley

January 17, 2016

Abstract

A new block cipher is presented that is faster in software than AES on 64-bit machines. It operates on a 256-bit block and accepts a variable length key up to 256 bits. FALCON uses up to 20 rounds of encryption, with the recommended number of rounds being 16. The round function of FALCON utilizes an 8x8 MDS matrix and an S-box identical to Rijndael's S-box, along with modular additions, bit rotations, and bitwise XORs to achieve nonlinearity and diffusion. Optimized 16-round FALCON encrypts at 10.5 cycles/byte on a 2.6 GHz Intel Core i7 processor, faster than optimized AES-128 which encrypts at 12.5 cycles/byte on the same machine.

Keywords: Cipher, Software, Encryption, FALCON

Contents

1 Introduction

1.1 A Note About FALCON

FALCON was created for a student project. It has not been subjected to rigorous cryptanalysis and therefore should not be used to protect sensitive data. Software implementations of FALCON have been provided in order to demonstrate its speed and capabilities, but not to encourage the use of FALCON as a serious encryption tool.

1.2 Design Rationale

FALCON was designed specifically to be efficient in software on 64-bit machines. The round function of FALCON operates on 64-bit words and uses modular additions, bit rotations, and bitwise XORs. However, FALCON also requires lookup tables. This means that it is not as fast as some ciphers that use only additions, rotations, and exclusive-ors (ARX structure) such as Salsa20 and Threefish.

2 Specifications

2.1 Notations

Table 1: **List of Notations**

Notation	Description
$\oplus$	bitwise XOR (exclusive-or)
$+$	addition mod 2^{64}
$-$	subtraction mod 2^{64}
$\lll$	circular left rotation
$\ggg$	circular right rotation

FALCON operates on 64-bit words using little-endian convention. The 32-byte plaintext (p_0, ..., p_{31}) is considered to be four unsigned 64-bit words (P_0, P_1, P_2, P_3) such that

$$P_i = \sum_{j=0}^{7} p_{(8i + j)} \cdot 2^{8j}$$

Likewise, the 32-byte cipherext and each 32-byte round key R_i are represented as (C_0, C_1, C_2, C_3) and (K_{4i}, K_{4i+1}, K_{4i+2}, K_{4i+3}), respectively.

2.2 Overview of FALCON

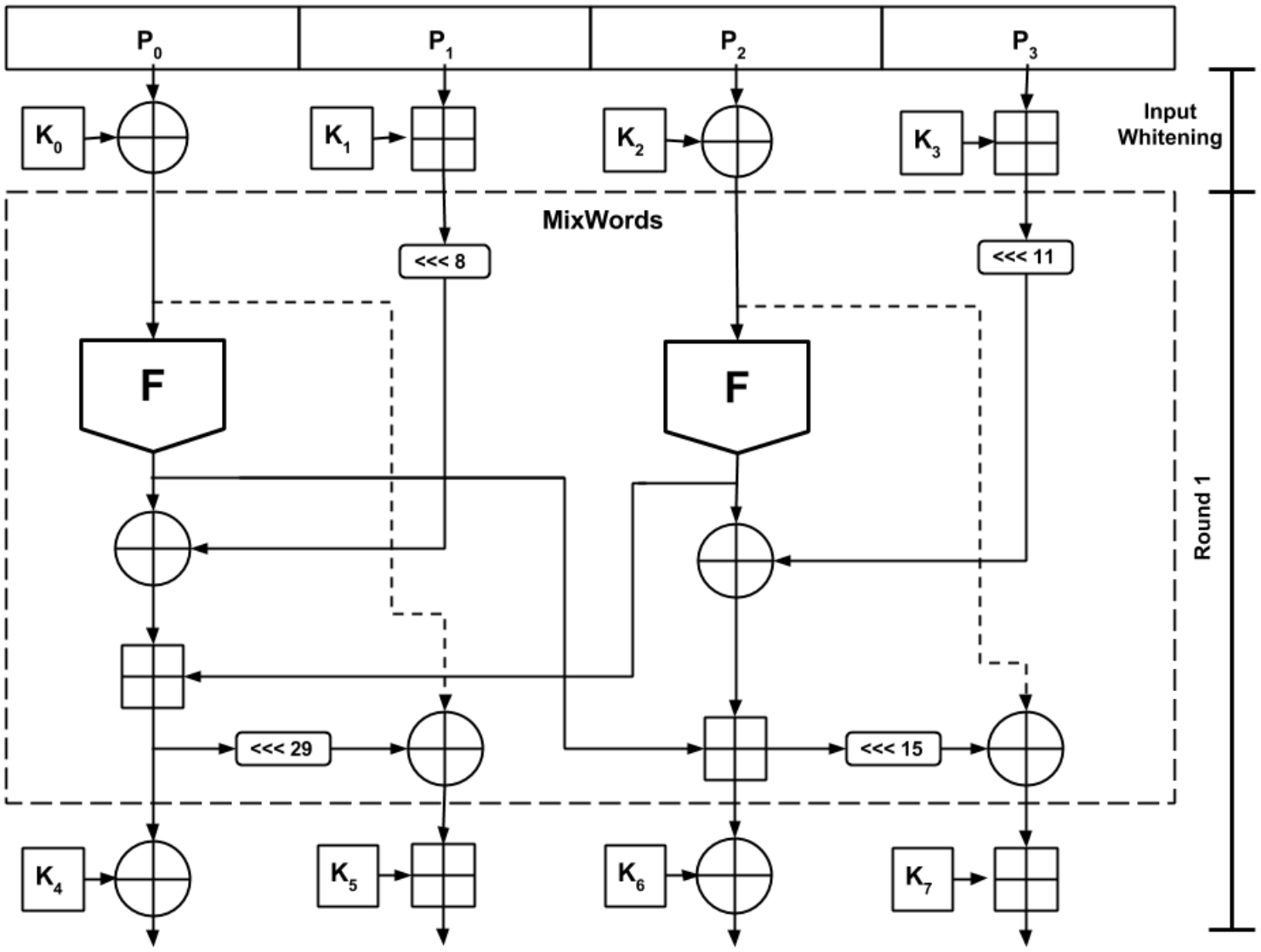
A round of encryption involves passing the data through the MixWords function and then applying a round key. To apply a round key to the data, the first and third words of the round key are XORed with the first and third words of the state, and the second and fourth words of the round key are added to the second and fourth words of the state mod 2^{64}.

Algorithm 1 Encryption process of FALCON

```
function ENCRYPT
    ApplyRoundKey
    for i := 1 to Nr do
        MixWords
        ApplyRoundKey
    end for
end function
```

The MixWords function is the core of FALCON. It uses modular additions, bit rotations, and bitwise XORs. It also utilizes a function called "F" that provides high diffusion and nonlinearity. F involves an S-box and an 8x8 maximum distance separable (MDS) matrix. Half of the input bits are capable of affecting all of the output bits to MixWords, while the other half of the input bits do not greatly affect the output of MixWords. Full diffusion occurs with two rounds of encryption.

Diagram 1: The encryption process of FALCON



2.3 The S-box

The S-box of FALCON is identical to the S-box of Rijndael. This S-box has been studied and is known to have good nonlinear properties. It seemed reasonable to use a tried and tested S-box rather than creating a new one.

2.4 MixWords

MixWords is used as the round function in order to achieve diffusion, and is also used to generate the key expansion. It is a nonlinear, bijective function. It takes four 64-bit words (W_0, W_1, W_2, W_3) as an input and returns four 64-bit words (W'_0, W'_1, W'_2, W'_3) as an output.

Algorithm 2 The function MixWords

function MixWords(W_0, W_1, W_2, W_3)
$Z_0 \coloneqq \mathrm{F}(W_0)$
$Z_1 \coloneqq \mathrm{F}(W_2)$
$W'_0 \coloneqq ((W_1 \lll 8) \oplus Z_0) + Z_1$
$W'_1 \coloneqq (W'_0 \lll 29) \oplus W_0$
$W'_2 \coloneqq ((W_3 \lll 11) \oplus Z_1) + Z_0$
$W'_3 \coloneqq (W'_2 \lll 15) \oplus W_2$
return (W'_0, W'_1, W'_2, W'_3)
end function

A small change to W_0 or W_2 will result in a full avalanche effect, potentially affecting all bits of the output (due to the fact that F provides high diffusion). Changes to W_1 and W_3 will not result in an avalanche effect because W_1 and W_3 are not passed through F and do not affect the whole output. Full diffusion occurs after two uses of MixWords.

2.4.1 The Function F

F is a nonlinear, bijective function that has good diffusion properties. It takes one 64-bit word as an input and returns one 64-bit word as an output. F works by separating the 64-bit input word into eight bytes. Each byte is replaced by the corresponding S-box entry. Then, the bytes are viewed as a vector with eight elements in GF(2^8) and multiplied by an 8x8 MDS matrix [2]. This matrix was first studied in the paper "Improving Diffusion Power of

AES Rijndael with 8x8 MDS Matrix", and it has now found use in FALCON.

$$\begin{pmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{pmatrix}$$

Multiplication is done in Rijndael's finite field [1]. After the matrix multiplication the eight bytes are concatenated into one 64-bit word and returned as the output of F.

The process of F: Given a 64-bit input X, represent X as an array of eight bytes in little-endian format:

$$X = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = \sum_{i=0}^{7} x_i \cdot 2^{8\mathrm{i}}$$

A new 64-bit word Y is now defined.

$$Y = (y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) \coloneqq (S(x_0), S(x_1), S(x_2), S(x_3), S(x_4), S(x_5), S(x_6), S(x_7))$$

where $\mathrm{S}(x_i)$ denotes passing byte x_i through the S-box. Y is then multiplied by the MDS matrix under the finite field $\mathrm{GF}(2^8)$, resulting in Z.

$$Z = (z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7) \coloneqq \begin{pmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix}$$

Z is returned as the output of F.

F can be efficiently implemented in software by combining the S-box and matrix multiplication into a series of table lookups in the same way as optimized versions of Rijndael [1]. The result of F can be computed with eight table lookups and seven bitwise XORs, requiring 16 KiB of precomputed data.

2.5 Key Schedule

FALCON has an interesting key schedule. It accepts any key of length N bits such that $0 \leq N \leq 256$. The key schedule ensures that every possible master key necessarily results in a unique key expansion. For example, a maser key of 256 zero bits will result in a different key expansion than a master key of 255 zero bits. Because every possible key of length less than or equal to 256 results in a unique key expansion, the total size of FALCON's keyspace is

$$\sum_{i=0}^{256} 2^i = 2^{257} - 1$$

This claim is formally proven in a later section.

FALCON can even accept a key of length zero (no key), although this would offer no security. The minimum recommended key length is 112 bits, the same key length as two key 3DES.

Note: For the purpose of this paper, the notation FALCON-N will be used to denote FALCON operating with an N bit key.

The key schedule works by initializing a 512-bit (64 byte) state S based on the master key, M. MixWords is then repeatedly used to derive round keys from S, and S is updated after each call to MixWords. Given a master key M of length N, the 512-bit state S is constructed as follows:

1. The first N bits of S are the N bits of M.
2. Append $256 - N$ "zero" bits to S. That is, pad S with "zero" bits until there are 256 total bits.
3. Append N "one" bits to S.
4. Append $256 - N$ "zero" bits to S, resulting in 512 total bits.

S is viewed as a series of eight 64-bit words (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7) and the round keys are derived according to the following algorithm:

Algorithm 3 The key schedule of FALCON

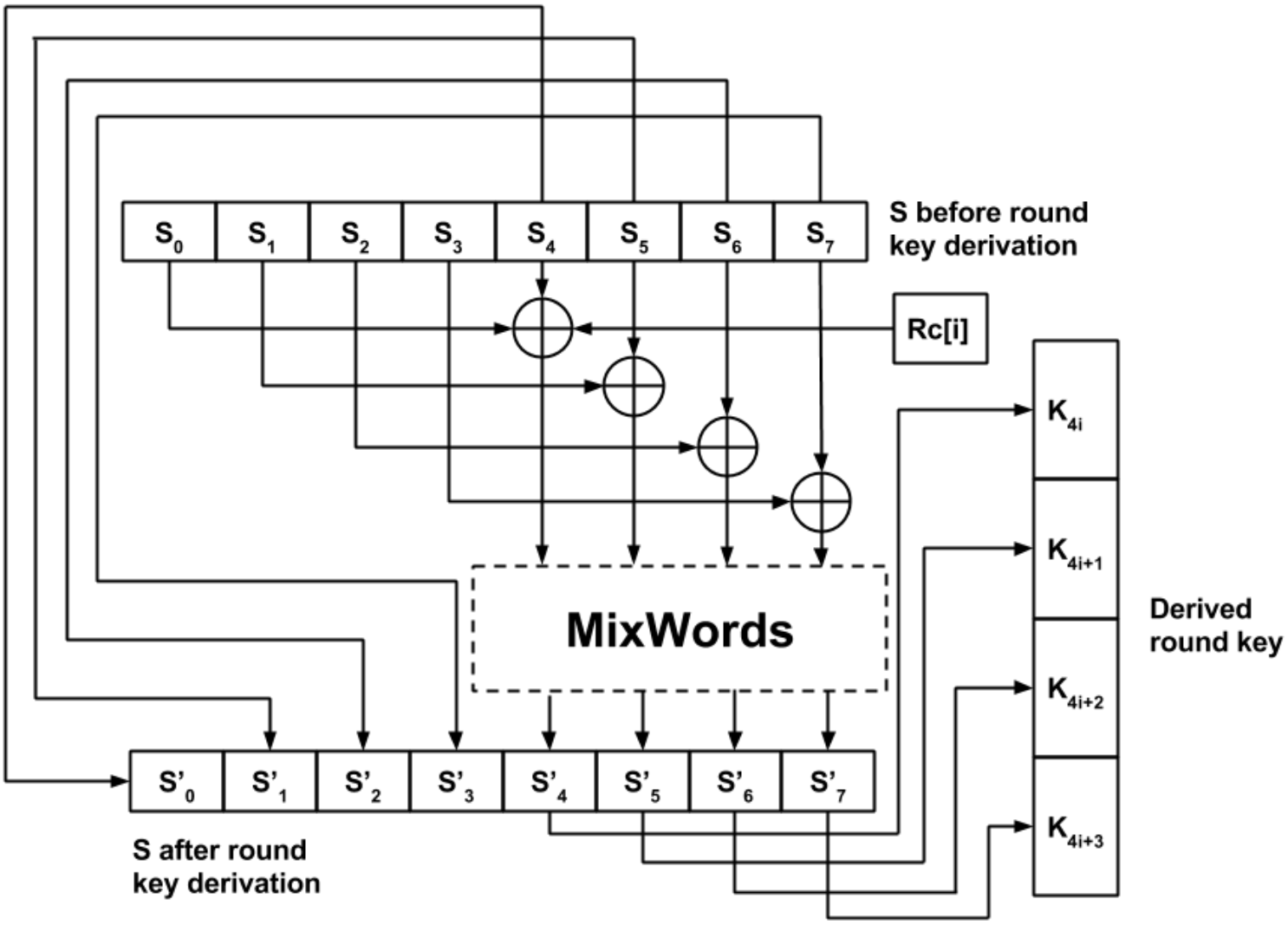
for i $\coloneqq$ 0 to Nr **do**
  $(\mathrm{K_{4i}}, \mathrm{K_{4i+1}}, \mathrm{K_{4i+2}}, \mathrm{K_{4i+3}}) \coloneqq \mathrm{MixWords}(S_0 \oplus S_4 \oplus Rc[i], S_1 \oplus S_5, S_2 \oplus S_6, S_3 \oplus S_7)$
  $S_0 \coloneqq S_4$
  $S_1 \coloneqq S_5$
  $S_2 \coloneqq S_6$
  $S_3 \coloneqq S_7$
  $S_4 \coloneqq \mathrm{K_{4i}}$
  $S_5 \coloneqq \mathrm{K_{4i\ +1}}$
  $S_6 \coloneqq \mathrm{K_{4i+2}}$
  $S_7 \coloneqq \mathrm{K_{4i+3}}$
end for

Where Nr is the number of rounds. Thus, the key schedule provides a total of Nr+1 round keys. It should also be noted that the master key itself is discarded after the key expansion is generated and is never directly used for encryption. This is somewhat unusual, as most ciphers use the master key as their first round key.

Rc[i] is a 64-bit round constant that differs for each round of key derivation. The round constants are defined by the hexadecimal expansion of π.

$$Rc = \begin{pmatrix} \text{0x243f6a8885a308d3}, & \text{0x13198a2e03707344}, & \text{0xa4093822299f31d0}, \\ \text{0x082efa98ec4e6c89}, & \text{0x452821e638d01377}, & \text{0xbe5466cf34e90c6c}, \\ \text{0xc0ac29b7c97c50dd}, & \text{0x3f84d5b5b5470917}, & \text{0x9216d5d98979fb1b}, \\ \text{0xd1310ba698dfb5ac}, & \text{0x2ffd72dbd01adfb7}, & \text{0xb8e1afed6a267e96}, \\ \text{0xba7c9045f12c7f99}, & \text{0x24a19947b3916cf7}, & \text{0x0801f2e2858efc16}, \\ \text{0x636920d871574e69}, & \text{0xa458fea3f4933d7e}, & \text{0x0d95748f728eb658}, \\ \text{0x718bcd5882154aee}, & \text{0x7b54a41dc25a59b5}, & \text{0x9c30d5392af26013} \end{pmatrix}$$

Diagram 2: One round of FALCON's key schedule



3 Inverting FALCON

Inverting FALCON involves reversing the order of the round keys that are applied to the data (using subtraction rather than addition to apply the keys), and inverting the MixWords function.

3.1 Inverting MixWords

As described earlier, MixWords is a function that takes four 64-bit words (W_0, W_1, W_2, W_3) as an input and returns four 64-bit words (W'_0, W'_1, W'_2, W'_3) as an output. The algorithm for inverting MixWords is as follows:

Algorithm 4 The inverse of MixWords

function InverseMixWords(W'_0, W'_1, W'_2, W'_3)
$W_0 \coloneqq (W'_0 \lll 29) \oplus W'_1$
$W_2 \coloneqq (W'_2 \lll 15) \oplus W'_3$
$Z_0 \coloneqq \mathrm{F}(W_0)$
$Z_1 \coloneqq \mathrm{F}(W_2)$
$W_1 \coloneqq ((W'_0 - Z_1) \oplus Z_0) \ggg 8$
$W_3 \coloneqq ((W'_2 - Z_0) \oplus Z_1) \ggg 11$
return (W_0, W_1, W_2, W_3)
end function

4 Security of FALCON

No rigid security claims are made about FALCON, as it has not been properly cryptanalyzed. However, some speculations and tentative assumptions are made.

4.1 Comparision to Rijndael

Rijndael (AES) has been extensively studied/cryptanalyzed and is known to be resistant to attacks such as linear and differential cryptanalysis. The S-box of Rijndael has optimal worst-case nonlinearity for an 8-bit by 8-bit S-box. For this reason, FALCON uses the same S-box in order to rely on a time-tested feature and achieve good nonlinearity. The use of alternating modular additions, bit rotations, and bitwise XORs in the MixWords function adds further nonlinearity to FALCON. Also, FALCON uses a larger MDS matrix with a higher branch number than the MDS matrix of Rijndael (nine vs five)

[1, 2]. FALCON achieves full diffusion with two rounds of encryption, just as Rijndael also requires two rounds to achieve full diffusion. In addition, the recommended number of rounds for FALCON is 16, higher than the number of rounds used by Rijndael.

With these considerations in mind, it may be reasonable to conclude that FALCON has at least *some* degree of resistance to linear and differential cryptanalysis, although this is not guaranteed.

4.2 Analysis of the Key Schedule

It is possible for two different master keys to result in identical key expansions for a particular round. This is an obvious fact due to the pigeon-hole principle: there are $2^{257}-1$ possible input keys to FALCON, but only 2^{256} possible round keys for any specific round. However, the key schedule was designed in such a way that two different states may only produce identical keys for one round. After one round they will necessarily began producing different keys.

Proof. MixWords is used to derive round keys, and it is a bijective function. This means that when operating on different inputs MixWords will always produce different outputs. Therefore, the only instances where two different states can produce identical round keys are instances where the inputs to MixWords are identical. Consider two possible states of the key schedule, S and S'. In order for them to result in identical inputs to MixWords the following conditions must hold:

$$S_0 \oplus S_4 = S'_0 \oplus S'_4$$

$$S_1 \oplus S_5 = S'_1 \oplus S'_5$$

$$S_2 \oplus S_6 = S'_2 \oplus S'_6$$

$$S_3 \oplus S_7 = S'_3 \oplus S'_7$$

This results in another condition:

$$S_i \neq S'_i \iff S_{\text{i+4 mod 8}} \neq S'_{\text{i+4 mod 8}}$$

Another way of stating this condition is: if S and S' differ in one (or more) of their first four words then they must also differ in one (or more) of their last four words (and vice versa) in order to produce identical keys.

Let S and S' be two different states, $S \not\equiv S'$, that produce the same round key, K = (K_0, K_1, K_2, K_3). As mentioned above, this implies that S and S'

have differences in their first four words and their last four words. After the round key has been derived the states are changed to the following values:

$$S \coloneqq (S_4, S_5, S_6, S_7, K_0, K_1, K_2, K_3)$$

$$S' \coloneqq (S'_4, S'_5, S'_6, S'_7, K_0, K_1, K_2, K_3)$$

Now S and S' have differences in their first four words, but their last four words are identical. The required conditions for producing identical round keys are no longer met. These two states will necessarily produce different round keys for at least the next two rounds.

This also has the effect of proving the claim that the size of FALCON's keyspace is

$$\sum_{i=0}^{256} 2^i = 2^{257} - 1$$

It has just been shown that two different states may produce at most one identical round key before their outputs diverge. Therefore, two different states may never produce completely identical key expansions. Every possible master key corresponds to a unique initial state of the key schedule, so every possible master key will result in a unique key expansion. There are $2^{257} - 1$ unique master keys of length less than or equal to 256-bits, therefore there are $2^{257} - 1$ unique key expansions. □

4.3 Side Channel Attacks

Because FALCON is reliant upon lookups tables in the same manner as AES, it is vulnerable to side channel attacks such as cache timing.

5 Performance

FALCON is fast in software on 64-bit machines. Optimized 16-round FALCON is faster than AES-128. 18-round FALCON is also faster than AES-128. 20-round FALCON is slower than AES-128 but faster than AES-192 and AES-256. The speed of FALCON does not depend on the size of the key used, and decryption is equal in speed to encryption. All tests were performed on a 2.6 GHz Intel Core i7 Ivy Bridge processor.

5.1 Encryption Speeds

The following table compares the speed of optimized FALCON to AES-128, AES-192, and AES-256. The algorithms were tested by repeatedly encrypting

one block until the designated quantity of data had been encrypted. The test was performed 5 times, and the values in the table represent averages. The specific implementation of AES used for this test is the optimized ANSI C source code for Rijndael written by Vincent Rijmen, Antoon Bosselaers, and Paulo Barreto.

Table 2: **Encryption of 4 GB of data: slowest to fastest**

Cipher	**Rounds**	**Block Size**	**Time (s)**	**MB/s**	**Cycles/Block**	**Cycles/Byte**
AES-256	14	128-bit	25.4	157.7	263.7	16.5
AES-192	12	128-bit	22.2	180.3	230.7	14.4
FALCON	**20**	**256-bit**	**20.0**	**200.3**	**415.4**	**13.0**
AES-128	10	128-bit	19.2	208.8	199.2	12.5
FALCON	**18**	**256-bit**	**18.1**	**221.2**	**376.1**	**11.8**
FALCON	**16**	**256-bit**	**16.1**	**248.4**	**335.0**	**10.5**
FALCON	**14**	**256-bit**	**14.1**	**283.1**	**293.9**	**9.2**
FALCON	**12**	**256-bit**	**12.1**	**329.7**	**252.3**	**7.9**
FALCON	**10**	**256-bit**	**10.1**	**393.8**	**211.3**	**6.6**

When FALCON operates with the same number of rounds as AES, it encrypts about 1.8 times faster. This allows a comparatively high number of rounds (16) to be recommended while still achieving good encryption speeds. Even when 20 rounds are used FALCON performs well, and it would not be difficult to adjust FALCON to use more than 20 rounds if needed.

5.2 Key Setup Speeds

FALCON's key schedule is slower than the key schedule of AES, although it is still quite fast. A key expansion can be generated in about the time it takes perform 1.5 encryptions. The following table compares the key setup speeds of FALCON and AES averaged over one hundred million trials. FALCON was given a master key of 256 bits for the purpose of this benchmark. A master key with a length that is not a multiple of 8 takes slightly longer to setup.

Table 3: **Key setup speed: slowest to fastest**

Cipher	Rounds	Time (ns)	Cycles	Expansion Size (bytes)	Cycles/Byte
FALCON	**20**	**218.9**	**569.3**	**672**	**0.8**
FALCON	**18**	**199.1**	**517.8**	**608**	**0.9**
FALCON	**16**	**179.4**	**466.6**	**544**	**0.9**
FALCON	**14**	**159.5**	**414.7**	**480**	**0.9**
FALCON	**12**	**140.0**	**364.1**	**416**	**0.9**
FALCON	**10**	**119.7**	**311.3**	**352**	**0.9**
AES-256	14	65.0	168.9	240	0.7
AES-128	10	49.3	128.4	176	0.7
AES-192	12	46.0	119.7	208	0.6

FALCON's key schedule is slower than AES's key schedule for several reasons. FALCON requires significantly more key material than AES due to the fact that its block size is twice as large as AES. In addition, the use of MixWords makes FALCON's key schedule much more complex than the key schedule of AES.

6 Reference Code

Both reference and optimized implementations of FALCON are provided along with the code for the performance benchmarks.

https://github.com/christiandaley/FALCON

7 Test Vectors for 16 Rounds

Any extra bits in the key (past the specified length) should be ignored by the key schedule.

7.1 FALCON-112

Key: 4478247e37860affc3167c5302f7

Plaintext: 0123456789abcdeffedcba98765432100123456789abcdeffedcba9876543210

Ciphertext: 001b4cb2e84e3cf96e5430437143aa4959872dc74425cc156280eeaca6d6d904

7.2 FALCON-128

Key: 01f1404cc287212544226a80b67574d0

Plaintext: bef6561bfefda682495df67aab2d705ad45d83a77f8839cda855572f9445d63c

Ciphertext: b165e416fc6e566717aa832f1e30520ad25dae4b48a8892b01506f3154d04dbd

7.3 FALCON-127

Key: 01f1404cc287212544226a80b67574d0 (final bit ignored)

Plaintext: bef6561bfefda682495df67aab2d705ad45d83a77f8839cd a855572f9445d63c

Ciphertext: 53bb30c74d4703d5b10fbd99c9c9f3a5da356fa8d54d75337747825a4741ba8e

7.4 FALCON-168

Key: 17470bdc7626afefc9942854787c6c3db74f8dcbb7

Plaintext: 270ecddb10737f53c6547d9df39fbf22248f4c8dde6f920dd9973b6aef56ab60

Ciphertext: cc74544095d228c473f716ef42d120fe5aa23923fee35704043e85e2ebe9ff57

7.5 FALCON-192

Key: 000102030405060708090a0b0c0d0e0f1011121314151617

Plaintext: dfe3866f212746e1b15d39db5b17f4ff9ba63b261913297f3c4906cb5db0478b

Ciphertext: 44be694526f98227ba865c5b99307651397e389c436a642ec0649839fb44c47a

7.6 FALCON-224

Key: 7ef8d41c3c21adf9c66ff2facce0287c808698d6795e2e57caa13a6b

Plaintext: 05d3f57aec2dc358feef14e004e6e2f38cfffa96cd2c9e7fd07afc5c859ae515

Ciphertext: 3c2f7da9a9ae6f80a831acba57475406d639f903c900e23d1a2b4cdfd28260e8

7.7 FALCON-256

Key: 0000000000000000000000000000000000000000000000000000000000000000

Plaintext: 00000000000000000000000000000000000000000000000000000000000000000000000000000000

Ciphertext: 62ecab567062e397eb78fdee2d0959a2c440c324f20b09c03206267b09f9bc2d

7.8 FALCON-255

Key: 0000000000000000000000000000000000000000000000000000000000000000
(final bit ignored)

Plaintext: 00000000000000000000000000000000000000000000000000000000000000000000000000000000

Ciphertext: 7045a5717b38c9973ee42996dee52533c79ce8230e16c396151c77d24bf8bc11

References

[1] Joan Daemen and Vincent Rijmen. *AES Proposal: Rijndael*. URL: `http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf`.

[2] R.Elumalai and Dr.A.R.Reddy. *Improving Diffusion Power of AES Rijndael with 8x8 MDS Matrix*. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.5460&rep=rep1&type=pdf`.