**Public Comments Regarding**
**The Advanced Encryption Standard (AES)**
**Development Effort**

**Round 1 Comments**
[in response to a notice in the September 14, 1998 Federal Register
(Volume 63, Number 177; pages 49091-49093)]

---

From: "**Yasuyoshi Kaneko**" <kaneko@yokohama.tao.or.jp>
To: <aesfirstround@nist.gov>
Subject: AES Candidate Comments
Date: Tue, 25 Aug 1998 11:43:28 +0900
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.2106.4
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.2106.4

Messrs. Information Technology Laboratory

I am Y.Kaneko belonging **Telecommunications Advancement Organization of Japan(TAO)**.
I have found the specifications of all AES candidate Algolizums on web and begun to read.
In my opinion, these 15 ciphers are classified to 4 groups.
Namely 'DES-type' , 'IDEA-type' , 'SAFER-type' and 'other-type' are found out from these specifications.
'DES-type' ciphers are 'DEAL', 'DFC', 'E2' and 'LOKI97' , these ciphers have feistel structures with a F-function.
'IDEA-type' ciphers are 'CAST-256', 'MARS', 'RC6' and 'TWOFISH', these ciphers have paralell 4 inputs and 4 outputs with some (F-) functions.
'SAFER-type' ciphers are 'CRYPTON', 'MAGENTA', 'RIJENDA', 'SAFER+' and 'SERPENT', these ciphers have a kind of Pseude-Hadamard Transform structure with invertible functions per same bits-length input/output.
'other-type' ciphers are 'HPC' and 'FLOG' these ciphers have particular structures with original algolithms.
The differences of the first three types are summarized as followings:
1) Differential and Linear cryptanalysis
From the viewpoint of the differential and linear cryptanalysis, supposing that r-round cipher (which round numbers would be logically and experimentaly obtained) gives an enough strong ciphers for each types then
'DES-type' ciphers need r+3 or r+2 round to keep security, because (r-2)-round or (r-3)-round differential or linear attacks are possible for the feistel structure,
'IDEA-type' ciphers and 'SAFER-type' ciphers need r+1 round to keep security in general meaning.
2) Diffusion
From the viewpoint of the effectiveness of diffusion, comparing with 1-round, 'DES-type' ciphers don't have an effective structure because one input bit may influence only one out bit,

'IDEA-type' ciphers depend on the design of the structure, that is to say some cipher has an influence of one input bit to two out bits, some other has an influence of one input bit to three out bits and some other has an influence of one input bit to four out bits,
'SAFER-type' ciphers are usually designed to consider the diffusion effect as possible as to get a large influenced out bit.
3) Functions
'DES-type' ciphers permit any kind of functions which are necessarily bijective,
'IDEA-type' ciphers also permit any kind of functions but usually have some invertible functions to mix input extended keys.
'SAFER-type' ciphers only permit bijective functions.

These above information is merely based on my personal opinion, however
I would like to propose my view point to NIST's AES project to hope that these information would be help to select new and good ciphers for the half of 21-century.

Best regards

Yasuyoshi Kaneko
Telecommunications Advanced Organization of Japan(TAO),
1-1-32 Shin'urashima, Kanagawa-ku, Yokohama 221-0031 Japan
tel:+81-45-450-1245 fax::+81-45-450-1246
e-mail: kaneko@yokohama.tao.go.jp

From: "**Yasuyoshi Kaneko**" <kaneko@yokohama.tao.or.jp>
To: <aesfirstround@nist.gov>
Subject: AES Candidate Comments(2)
Date: Thu, 27 Aug 1998 19:05:20 +0900
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 4.72.2106.4
X-MimeOLE: Produced By Microsoft MimeOLE V4.72.2106.4

Messrs. Information Technology Laboratory

I am Y.Kaneko belonging **Telecommunications Advancement Organization of Japan(TAO)**.
This is the second time to send my opinion.
In the first my opinion, I proposed a classification of 4-groups of AES candidate ciphers. This classification is based on the structure of each cipher. In this time I try to classify the all ciphers based on the S-boxes of each ciphers.
I feel that 15 candidate ciphers are a few many and selecting good and attractive ciphers is like to choose a pleased driving car. In the case of cars, the style of the body and the ability of engines are very important besides with its price and popularity. In the case of selecting ciphers, comparing with cars, the style of cars corresponds to the structure of ciphers as I mentioned in the first my opinion and engines correspond to the S-boxes.
In this time these 15 ciphers are also classified to 4 groups as following.
The 1-st group is based on some kind of 'secure design principles', the 2-nd group uses some 'known cryptical functions', the 3-rd group uses some(like plus, shiftrotation) 'arithmetic operations', and the 4-th group uses some 'new cryptical functions'.
'Secure design principles' group consists of CAST256, CRYPTON, E2, LOKI97, RIJNDAEL and TWOFISH.
This means that CAST256 uses bent-function-base large S-boxes which satisfy a few many cryptical criterion, CRYPTON and TWOFISH uses some kind of secure feistel functions like one which is seen in the case of CS-cipher, E2 and RIJNDAEL uses a provable secure and high degree function x^* over a finite field, and LOKI97 uses a provable secure functions x^3 over a finite field.
'Known cryptical functions' group consists of DEAL, MARS, SAFER+ and SERPENT.
DEAL uses DES as a F-function, MARS uses a hush function SHA, SAFER+ uses SAFER's exponential and logarithm functions, and SERPENT uses 32 DES's S-boxes.
'Arithmetic operations' group consists of HPC, MARS and RC6.
Especially about RC6, it uses a function y=x(2x+1) mod 2^w.
'New cryptical functions' group consists of DFC, FROG and MAGENTA.
DFC uses a function which is constructed with a pairwise decorrelation module, FROG uses a particular kind of permutation, and MAGENTA uses a function y=α^x where α is a primitive element of a finite field.
To sumarize all my classification, the following (Structure\S-box)- table is obtained.

| S-box Structure | Secure design principles | Known cryptical functions | Arithmetic Operations | New cryptical functions |
|---|---|---|---|---|
| DES-type | E2, LOKI97 | DEAL | - | DFC |
| IDEA-type | CAST256, Twofish | MARS | MARS, RC6 | - |
| SAFER-type | CRYPTON, Rijndael | SAFER+, Serpent | - | MAGENTA |
| OTHER-type | - | - | HPC | FROG |

This my classification may one of classifications, however as reading from above table, if using with necessary check lists about all ciphers, an right classification will give a clear and objective viewpoint which help to discuss about the property of all ciphers.

Best regards

Yasuyoshi Kaneko
Telecommunications Advanced Organization of Japan(TAO),
1-1-32 Shin'urashima, Kanagawa-ku, Yokohama 221-0031 Japan
tel:+81-45-450-1245 fax::+81-45-450-1246
e-mail: kaneko@yokohama.tao.go.jp

X-ROUTED: Thu, 10 Sep 1998 07:04:20 -0600
X-TCP-IDENTITY: Billp
Date: Thu, 10 Sep 1998 06:44:43 -0600
From: **bill payne** <billp@nmol.com>
X-Mailer: Mozilla 3.03 (Win95; I)
To: aesfirstround@nist.gov, lb@qainfo.se
CC: ted lewis <lewis@nps.navy.mil>,
masanori fushimi <fushimi@misojiro.t.u-tokyo.ac.jp>, jy@jya.com,
j orlin grabbe <kalliste@aci.net>, vicepresident@whitehouse.gov,
cypherpunks@toad.com, ukcrypto@maillist.ox.ac.uk,
biham@cs.technion.ac.il, even@cs.technion.ac.il, wpi@wpiran.org,
abd@CDT.ORG, merata@pearl.sums.ac.ir, lawya@lucs-01.novell.leeds.ac.uk,
sjmz@hplb.hpl.hp.com, senatorlott@lott.senate.gov,
senator_leahy@leahy.senate.gov, conrad_burns@burns.senate.gov,
larry_craig@craig.senate.gov, senator@wyden.senate.gov,
dpcintrn@osd.pentagon.mil, abumujahid@taliban.com
Subject: Official comment

http://csrc.nist.gov/encryption/aes/aes_home.htm#comments

OFFICIAL Comments - Anyone may provide NIST with OFFICIAL public comments on the AES candidate algorithms. NOTE THAT ALL COMMENTS RECEIVED SHALL BE MADE PART OF THE PUBLIC RECORD. A comment may be submitted by sending an e-mail message to AESFirstRound@nist.gov.

OFFICIAL Comment

http://www.aci.net/kalliste/bw1.htm

to appear at http://zolatimes.com/
Content-Type: text/html; charset=iso-8859-1; name="bw1.htm"
Content-Disposition: inline; filename="bw1.htm"
Content-Base: "http://www.aci.net/kalliste/bw1.htm"

X-MIME-Autoconverted: from 8bit to quoted-printable by email.nist.gov id IAA21110

# Black and White Test of Cryptographic Algorithms

***by William H. Payne***

---
**The purpose of this article is to explain the underlying principles of cryptography by examples, and to show some criteria that should be met by cryptographic algorithms before they are seriously considered for adoption.**
---

Cryptography is the art or science of scrambling plaintext into ciphertext with a key so that it cannot be read by anyone who does not possess the key.

Digital information is stored in the form of 1s and 0s, called BINARY.

# Binary Numbers

Let’s count from DECIMAL 0 to 15 in BINARY by adding 1 to the previous number.

| **decimal:** | **binary:** |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | |
| 14 | |
| 15 | |

1101
1110
1111

Notice that first we start with a single number position, which can be 0 or 1 in BINARY. This number position is a *bit*. Call this first bit b0. Notice that b0 is either 0 or 1. That is, b0 = 0 or b0 = 1.

To get to DECIMAL 2, we have to introduce a second BINARY bit--call it b1. We have b1b0 = 10. Next, for DECIMAL 3, we have BINARY b1b0 = 11.

**binary:**
0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1110
1111

**numbered bits:**
b0
b0
b1b0
b1b0
b2b1b0
b2b1b0
b2b1b0
b2b1b0
b3b2b1b0
b3b2b1b0
b3b2b1b0
b3b2b1b0
b3b2b1b0
b3b2b1b0
b3b2b1b0
b3b2b1b0

Notice that the bit subscript represents a power of 2. That is, b0 really means

b**0***2^**0**,

where * is multiplication, and ^ exponentiation (for example, 2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8). The subscript on b is the same as the power on 2.

If we had b26, we would know its meaning was b**26***2^**26**. If b26 = 0, then this value is 0. If b26 = 1, then this value is 2^26.

Now look at "1111" (which in BINARY is equal to DECIMAL 15). In this case, b3b2b1b0 = 1111. The right-most BIT (b0) is the *least*-significant bit, because it corresponds to the lowest power of 2.

# Converting Binary Numbers to Decimal Numbers

To convert a BINARY number ...b3b2b1b0 to a DECIMAL number Y, we simply write

Y = b0 + b1 * 2 + b2 * 2^2 + b3 * 2^3 + ...

The bits b0, b1, b2, b3 are limited to the values 0 and 1 ONLY.

Performing the exponentiation of powers of 2 and reversing the bits gives

Y = . . . + b3 * 8 + b2 * 4 + b1 * 2 + b0 .

Most of us were brought-up to understand that the most significant digits are to the LEFT of the previous digit.

For sake of economy of writing and easy conversion, binary numbers are frequently represented in base 16, or HEXADECIMAL, abbreviated HEX.

# Hexadecimal Numbers

| Binary weights 8 4 2 1 | HEXDECIMAL |
|---|---|
| 0 0 0 | 0 |
| 1 1 1 | 1 |
| 1 0 2 | 2 |
| 1 1 3 | 3 |
| 1 0 0 4 | 4 |
| 1 0 1 5 | 5 |
| 1 1 0 6 | 6 |
| 1 1 1 7 | 7 |
| 1 0 0 0 8 | 8 |
| 1 0 0 1 9 | 9 |
| 1 0 1 0 A | 10 |

1 0 1 1 B 11
1 1 0 0 C 12
1 1 0 1 D 13
1 1 1 0 E 14
1 1 1 1 F 15

Conversion from binary to hexadecimal or hexadecimal to binary is easy if you remember

1010 is A
1100 is C
1110 is E

B is one greater than A, 1011. D is one greater than C, 1101. And F is one greater than E, 1111.

# Computer Memory

Computer memory is frequently organized as BYTEs which are eight bits.

Since one hexadecimal digit represents 4 bits, it takes two hexadecimal digits to represent one byte.

There are 2^8 = 256 different binary values that can be represented in a byte. These 256 values (written in HEX for brevity) are:

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
99 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
FF F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

# Representing Language on a Computer

The problem of how to represent characters in a computer has been solved several ways. One way is the American Standard Code of Information Interchange (ASCII).

ASCII represents characters as 7 bits.

Here is table modified from a web site ( http://members.tripod.com/~plangford/index.html).

| Hex | Char | Description | Hex | Char | Hex | Char | Hex | Char |
|---|---|---|---|---|---|---|---|---|
| 00 | | NUL (null) | 20 | | 40 | @ | 60 | ` |
| 01 | | SOH (start of heading) | 21 | ! | 41 | A | 61 | a |
| 02 | | STX (start of text) | 22 | " | 42 | B | 62 | b |
| 03 | | ETX (end of text) | 23 | # | 43 | C | 63 | c |
| 04 | | EOT (end of transmission) | 24 | $ | 44 | D | 64 | d |
| 05 | | ENQ (enquiry) | 25 | % | 45 | E | 65 | e |
| 06 | | ACK (acknowledge) | 26 | & | 46 | F | 66 | f |
| 07 | | BEL (bell) | 27 | ' | 47 | G | 67 | g |
| 08 | | BS (backspace) | 28 | ( | 48 | H | 68 | h |
| 09 | | TAB (horizontal tab) | 29 | ) | 49 | I | 69 | i |
| 0A | | LF (line feed, new line) | 2A | * | 4A | J | 6A | j |
| 0B | | VT (vertical tab) | 2B | + | 4B | K | 6B | k |
| 0C | | FF (form feed, new page) | 2C | , | 4C | L | 6C | l |
| 0D | | CR (carriage return) | 2D | - | 4D | M | 6D | m |
| 0E | | SO (shift out) | 2E | . | 4E | N | 6E | n |
| 0F | | SI (shift in) | 2F | / | 4F | O | 6F | o |
| 10 | | DLE (data link escape) | 30 | 0 | 50 | P | 70 | p |
| 11 | | DC1 (device control 1) | 31 | 1 | 51 | Q | 71 | q |
| 12 | | DC2 (device control 2) | 32 | 2 | 52 | R | 72 | r |
| 13 | | DC3 (device control 3) | 33 | 3 | 53 | S | 73 | s |
| 14 | | DC4 (device control 4) | 34 | 4 | 54 | T | 74 | t |
| 15 | | NAK (negative acknowledge) | 35 | 5 | 55 | U | 75 | u |
| 16 | | SYN (synchronous idle) | 36 | 6 | 56 | V | 76 | v |
| 17 | | ETB (end of trans. block) | 37 | 7 | 57 | W | 77 | w |
| 18 | | CAN (cancel) | 38 | 8 | 58 | X | 78 | x |
| 19 | | EM (end of medium) | 39 | 9 | 59 | Y | 79 | y |
| 1A | | SUB (substitute) | 3A | : | 5A | Z | 7A | z |
| 1B | | ESC (escape) | 3B | ; | 5B | [ | 7B | { |
| 1C | | FS (file separator) | 3C | | 5E | ^ | 7E | ~ |
| 1F | | US (unit separator) | 3F | ? | 5F | _ | 7F | DEL |

Now let us take two different one-word messages we might wish to cipher: "black" and "white". We can use the preceding table to find the ASCII codes for the characters in "black" and "white".

message

character
ASCII (Hex)
Binary

Message 1

black
62 6C 61 63 6B
0110 0010 0110 1100 0110 0001 0110 0011 0110 1011

Message 2

white
77 68 69 74 65
0111 0111 0110 1000 0110 1001 0111 0100 0110 0101

But before doing this, we must understand the general "cipher problem."

# The Cipher Problem

We have three elements in the encryption process:

1. The plaintext message
2. The key
3. The ciphertext

Let's start REAL SIMPLE. Let's consider a situation where the plaintext message, the key, and the ciphertext *are all the same length*. To make it even simpler, let's make ***each one only one bit long***. So the key can be one of two possibilities (0 or 1), and so can the plaintext and the ciphertext. So, in all, there are 2*2*2 = 8 total possible encipherment circumstances.

Let's enumerate ALL 8 POSSIBILITIES.

**Possibilities Table**

| Possibility | Key | Plaintext | Ciphertext |
|---|---|---|---|
| **1** | 0 | 0 | 0 |
| **2** | 0 | 0 | 1 |
| **3** | 0 | 1 | 0 |
| **4** | 0 | 1 | 1 |
| **5** | 1 | 0 | 0 |
| **6** | 1 | 0 | 1 |
| **7** | 1 | 1 | 0 |
| **8** | 1 | 1 | 1 |

That's it! There are no more possibilities than these 8. What does this mean for the encryption process--the "algorithm"?

An ALGORITHM is a deterministic processes that accepts inputs and transforms them into outputs.

"Deterministic" is important in that the same inputs ALWAYS produce the same output.

Consider ANY algorithm which takes as its inputs the key values of 0 or 1 and the plaintext message values of 0 or 1.

ANY algorithm can only produce one of the ciphertext outputs seen above. Image the following hypothetical but REAL SIMPLE algorithm:

> **Hypothetical Algorithm**: Find the value of the Key in column 2 of the Possibilities Table and the Plaintext message in column 3; then chose as the Ciphertext output whatever number is found in column 4 of that row.

But there, of course, is a catch with a valid CRYPTOGRAPHIC ALGORITHM.

Given the Key and the Ciphertext, one must be able to get back the Plaintext!

A cryptographic algorithmic should have an INVERSE.

So a cryptographic algorithm could not produce ALL of the eight combinations seen above for the reason that it is impossible to invert some of the possibilities. For example, some of the mappings are incompatible from an inverse standpoint, because same values of the key and ciphertext can lead to two different values of the plaintext. Notice how the following pairs of possibilities conflict: 1 and 3; 2 and 4; 5 and 7; 6 and 8.

PossibilityKeyPlaintextCiphertext
**1**000
**3**010
**2**001
**4**011
**5**100
**7**110
**6**101
**8**111

But there two different cryptographic algorithms that could be made from the Possibilities Table, both of which have inverses:

**Cryptographic Algorithm 1**

PossibilityKeyPlaintextCiphertext
**1**000
**4**011
**6**101
**7**110

**Cryptographic Algorithm 2**

PossibilityKeyPlaintextCiphertext
**2**001
**3**010
**5**100
**8**111

Of course, the output of Algorithm 2 is merely the same as the output of Algorithm 1, with 0s and 1s switched. (This is called a logical NOT operation.)

# Logic and Its Electronic Representation

Logic, sometimes called Boolean logic when it is dealing with 0s and 1s, has several elementary rules.

In computers, TRUE is usually represented by a 1. FALSE is represented by a 0.

Electrically a 1 is usually, but not always, represented by a HIGH VOLTAGE. A zero by a LOW VOLTAGE.

The three basic operations in logic are NOT, AND, and OR:

Logical Operation
Input(s)Output
NOT0 1
1 0

AND000
010
100
111

OR 000
011
101
111

A derivative operation called an EXCLUSIVE-OR, abbreviated **XOR**, is defined as follows:

Logical Operation
Input(s)Output
XOR000
011
101
110

**In XOR, if the two input bits have the the same value, they sum to 0. If they have different values, they sum to 1.**

Now look back at Cryptographic Algorithm 1. It is, in fact, the exclusive-or (XOR) of the key and plaintext.

**Algorithm 1**: Ciphertext Output = Key XOR Plaintext.

Cryptographic Algorithm 2, meanwhile, is just the NOT of Algorithm 1.

**Algorithm 2**: Ciphertext Output = NOT (Key XOR Plaintext).

The REALLY IMPORTANT property of the **XOR** is THAT IT HAS AN INVERSE.

By contrast, logical AND does not have an inverse for the reason that if the Key and (Key AND Plaintext) are both 0, then the Plaintext itself is ambiguously either 0 or 1.

Logical OperationKey InputPlaintext InputKey AND Plaintext
AND000oops
010oops
100
111

Likewise, logical OR does not have an inverse for the reason that if the Key and (Key OR Plaintext) are both 1, then the Plaintext itself is ambiguously either 0 or 1.

Logical OperationKey InputPlaintext InputKey OR Plaintext
OR 000
011
101oops
111oops

So logical AND and OR don't work well for a crypto algorithm, but the XOR does because it has an inverse.

# How to Create Two Keys for Deniability

The XOR works even better from a legal standpoint. Imagine the following conversation:

**Ciphercop**: We have the ciphertext 0 and we CAUGHT you with the key with a bit value of 1, so you sent a plaintext 1.
**Citizen**: No I did not! You PLANTED the key with bit value 1. The real key bit is a 0, and I sent 0 as the plaintext.

Let's generate a key for the REAL WORLD crypto messages "black" and "white",

message

character
ASCII (Hex)
Binary

Message 1

black

62 6C 61 63 6B
0110 0010 0110 1100 0110 0001 0110 0011 0110 1011

Message 2

white
77 68 69 74 65
0111 0111 0110 1000 0110 1001 0111 0100 0110 0101

and see if we can produce a REAL EXAMPLE of a SECOND KEY.

Here's a key, which we will call key 1:

**key 1:** 1010 0101  1100 0011  1110 0111  1111 0000  0110 1001

Key 1 doesn't look too random. Each group of four bits is followed by its logical NOT (e.g. NOT(1010) = 0101, etc.). Which leads to another lesson.

> **To claim that a sequence of 0s and 1s is *random* requires statistical testing. Otherwise, the state of the sequence is UNKNOWN.**

**Here's another key, which we will call key 2:**

key 2: **1011 0000  1100 0100  1110 1111  1110 0111  0110 0111**

**These two keys produce the** same ciphertext **for the two different messages "black" and "white".**

black0110 0010 0110 1100 0110 0001 0110 0011 0110 1011
key 11010 0101 1100 0011 1110 0111 1111 0000 0110 1001
ciphertext (XOR)1100 0111 1010 1100 1000 0110 1001 0011 0000 0010

white0111 0111 0110 1000 0110 1001 0111 0100 0110 0101
key 21011 0000 1100 0100 1110 1111 1110 0111 0110 0111
ciphertext (XOR)1100 0111 1010 1100 1000 0110 1001 0011 0000 0010

So when the ciphercops FALSELY accuse you of encrypting "black", you SCREAM "Bull pucky!", and produce key 2 to show that you, IN FACT, encrypted "white". Then sue the government--*pro se*, of course. (See http://jya.com/whpfiles.htm.)

The recipe for producing the second key in this example is simple. Take two plaintext messages of the same length. Encrypt one of them with an arbitrary key that yields a ciphertext of the same length. XOR the ciphertext with the second plaintext message. The result is the second key. Store this one for plausible deniability.

So from the standpoint of plausible deniability it is BEST to have TWO KEYS for any given encryption:

1. The REAL KEY
2. The key you can CLAIM was the REAL KEY. Just in case you get caught.

**"Have we gone beyond the bounds of reasonable dishonesty?"**
**--CIA memo**

(The CIA quote is from *Weird History 101*, by John Richard Stephens, page 55.)

None of us want to get caught going beyond the bounds of reasonable dishonesty.

Thus far two criteria of a worth candidate for cryptographic algorithm have been established.

# Criterion 1: The ciphertext is invertible with the help of a key back into the plaintext.

# Criterion 2: There is ALWAYS a second key. Just in case you get caught.

## Plaintext and Ciphertext Sizes

The plaintext and ciphertext should be the same size.

First, note that if the plaintext is *longer* than the ciphertext, then the ciphertext is not invertible. For example, let's suppose that the plaintext is two bits long and the ciphertext is one bit long.

PlaintextCiphertext
0 00 or 1
0 11 or 0
1 0oops
1 1oops

After the first two ciphertext bits have been assigned to plaintext pairs, the next two plaintext pairs (10,11) must conflict with this assignment. The ciphertext thus correspondents to more than one plaintext possibility.

We run into problems for the reason that we cannot establish a one-to-one correspondence between the plaintext and cipher text and, therefore, can't possibly have an inverse.

Second, if the plaintext is *shorter* than the ciphertext, then the ciphertext can't be trusted. It may include too much information. For example, let's suppose that the plaintext is one bit long, the key is one bit long, and the cipher text is two bits long.

**Iran Cryptographic Algorithm**

KeyPlaintextCiphertext
0 0 00
0 1 10
1 0 11
1 1 01

Not only is the above algorithm invertible, but now the crypto key has been sent along with the ciphertext in the second bit position!

That is, the first bit in the ciphertext is is the value of (key XOR plaintext). The second bit is the key itself. So if you XOR the two ciphertext bits with each other, you recover the plaintext bit.

You might ask who would be audacious enough to pull such stunt. The Great Satan, of course.

For the story of how the National Security Agency (NSA) bugged the encryption equipment that was sold by a Swiss company to 140 nations around the world, see the following links:

http://www.aci.net/kalliste/speccoll.htm
http://caq.com/cryptogate
http://www.qainfo.se/~lb/crypto_ag.htm
http://jya.com/whpfiles.htm

And the Great Satan got caught. No plan B. Or in crypto parlance, no second key.

So we have a third criterion for a cryptographic algorithm we might wish to adopt.

# Criterion 3: The length of the plaintext equals the length of the ciphertext.

In simple terms, if more bits come out of a crypto algorithm than go in, WATCH OUT!

# Otis Mukinfuss and the Advanced Encryption Standard

Bruce Hawkinson (BHAWKIN@sandia.gov) WAS Sandia National Laboratories Lab News editor some years ago.

In one editorial, Hawkinson wrote that while we was traveling for Sandia, he spent his motel time looking up strange names in the phone book. One name I recall mentioned was Steely Gray who was a government office furniture salesman.

Hawkinson concluded his article by writing his all-time favorite name was Otis Mukinfuss.

Hawkinson was no longer editor of Sandia's Lab News shortly thereafter.

J. Orlin Grabbe has done an excellent job writing about cryptographic algorithms in Cryptography and Number Theory for Digital Cash.

One inescapable conclusion from Grabbe's internet article is that from a layman's standpoint public key cryptography is an incomprehensible mess. A Muckinfuss.

The National Institute of Standards and Technology (NIST) is holding a CONTEST to select an Advanced Encryption Standard to replace the current Data Encryption Standard (DES).

Click through the candidates to view some additional examples of Mukinfusses.

So another criterion has been established for a cryptographic algorithm to be considered for adoption.

## Criterion 4: The crypto algorithm must be simple and CONVINCE EVERYONE that it is ONLY PERFORMING ITS SIMPLE INTENDED FUNCTION.

While we are at the NIST web site, the NIST Advanced Encryption Standard contest reminds me of a the plot of a recent movie, *The Game*, starring Michael Douglas and Sean Penn:

> The film is a thriller directed by David Fincher (Se7en). "The Game" is what begins when a high-powered businessman named Nicholas Van Orton (Douglas) receives the birthday gift of a lifetime from his brother he alienated years ago (Penn). What Nicholas gets is entry into a mysterious new form of entertainment provided by C.R.S. (Consumer Recreational Services) simply called "The Game." It proves to be an all-consuming contest with only one rule: there are no rules. By the time Van Orton realizes he is in, it is too late to get out. ... (See http://www.movietunes.com/soundtracks/1997/game/.)

NIST does not appear to publish any criteria for winning the AES contest! Look at http://www.nist.gov/public_affairs/confpage/980820.htm and decide for yourself.

## Perfect Cryptography

Here we have described a process of encrypting a plaintext by XORing it with a key of the same length. This encryption technique is called a "one-time pad", or Vernam cipher. Just as long as each key is only used once, the encryption technique is perfectly secure.

The one-time pad described here satisfies all criteria mentioned so far:

1. The ciphertext is invertible with the help of a key back into the plaintext.
2. There is ALWAYS a second key. Just in case you get caught.
3. The length of the plaintext equals the length of the ciphertext.
4. The crypto algorithm must be simple and CONVINCE EVERYONE that it is ONLY PERFORMING ITS SIMPLE INTENDED FUNCTION.

I add

## Criterion 5: The length of the key must equal the length of the plaintext.

Extensive mathematics or complication fails Criterion 4.

Public key cryptograpy that uses the RSA algorithm MAY fail Criterion 1 if the message divides the product of the two prime numbers, p and q, used in the modulus.

Most crypto algorithms are designed so that the key cannot be recovered from a plaintext-ciphertext pair. Therefore, they fail Criterion 2.

Criterion 3 is much more difficult to ensure against.

## Black and White Hats

American western movie producers used to aid their audiences in identification of the heroes and villains. The heroes wore white hats. The villains, black hats.

US government agencies adopted the term ‘black hatter’ to describe an employee whose job it is to break into THINGS. Or screw them up: http://www.jya.com/whp1.htm

A ‘white hatter’ is one who analyzes THINGS to make sure they cannot be broken into. And they can’t be screwed up.

But the empirical fact is that the ‘black hatters’ can figure-out methods to transmit the key on a covert channel, tell the ‘white hatters’ they did this. And the ‘white hatters’ can’t find out how they did it.

# Algorithmic Processes

Suppose the key is five bits:

1 0 1 0 1

Suppose the plaintext is six bits:

1 1 1 1 1 1

And the ciphertext is also six bits:

1 0 1 1 0 1

Ask the cryptographer give you a key which changes ONLY the sixth bit of the ciphertext, as in the following:

1 0 1 1 0 0

You like the other 5 bits just fine.

If the cryptographer can't, then you might look for another algorithm to adopt.

# Conclusion

We have five criteria to judge the outcome of the NIST Advanced Encryption Standard contest.

If none of the algorithms pass the five tests, we will not be discouraged.

We know that Gilbert S. Vernam and Joseph O. Mauborgne solved the crytptography problem in 1918, when they created the one-time pad. (See "What is a One-Time Pad?".)

William H. Payne
P.O. Box 14838
Albuquerque, New Mexico 87191
505-292-7037
Embedded Controller Forth.
Generalized Feedback Shift Registers

Here are some links to some of my students:

Ted Lewis (Friction-Free Economy)
John Sobolewski (HPCERC)

Date: Wed, 30 Sep 1998 19:30:35 -1000
From: "**A. C. Folmsbee**" <poiubnm@yahoo.com>
Organization: Me
X-Mailer: Mozilla 2.01 (Win95; I; 16bit)
To: aesfirstround@nist.gov
Subject: A new name for AES
X-URL: http://csrc.nist.gov/encryption/aes/aes_home.htm#comments

Here is a possible new name for AES :

Millennial Encryption Standard Algorithm

MESA

One level above DES for cryptographic purposes.

A. C. Folmsbee, 1998/9/30

From: John.Lamertina@smed.com
X-Lotus-FromDomain: SMS
To: AESFirstRound@nist.gov
Cc: BBraithw@osaspe.dhhs.gov
Date: Mon, 26 Oct 1998 13:41:52 -0500

Dear Reviewer,

You may be aware that the Health Insurance Portability and Accountability Act (HIPAA) was enacted in 1996 and requires all health care providers to secure and keep confidential all electronic patient records. The Secretary of the Department of Health and Human Services (DHHS) submitted supporting security requirements to the Federal Register on August 12, 1998.

The DHHS security and electronic signature proposals for HIPAA require that any patient data transmitted on an open network (e.g. the internet or standard dial-up lines) be encrypted. These requirements may go into affect as early as October 2000.

Has there been any coordination of the AES with DHHS or HIPAA? Do you know if the DHHS is planning to mandate a particular encryption methodology? Do you have suggestions for how affected organizations might best improve their software to be compliant with the DHHS and HIPAA encryption requirements?

Does the AES (or the DES which it replaces) require some minimum hardware capacity? Is that capacity measured in word lengths (e.g. 16-bit, 32-bit or 64-bit)? Minimum bit lengths could limit which operating system is used (DOS, Windows 3.1, Windows 95, Windows 98 or NT).

Thank you.

**John Lamertina**, Manager of Interface Development
**Delta Health Systems, An SMS Company**
Altoona, PA 16603

To: AESFirstRound@nist.gov
From: maro@isl.ntt.co.jp
Subject: comments to NIST preliminary efficiency test
Date: Sun, 17 Jan 1999 16:33:52 JST
Sender: maro@sucaba.isl.ntt.co.jp

Dear NIST,

We have comments on NIST preliminary analysis put on
http://csrc.nist.gov/encryption/aes/round1/nistefficiency1.pdf.

[1]Clarify how to measure the speed. Open the program for speed measuring if possible.

[2]Some candidates such as CRYPTON do not meet NIST API requirements. The measurement of such candidates should be normalized and how to normalize should be opened. If this is difficult, the fact that they don't follow NIST API should be stated clearly.

[3]If the source codes which contain conditional branches of C preprocessor do not follow ANSI C, the fact should be noted. Moreover, performance comparison should be done among only the candidates whose codes follow ANSI C.

Regards,

**K. Aoki of NTT Laboratories**

Date: Sun, 31 Jan 1999 19:20:33 +0100 (MET)
From: **Robert Harley** <Robert.Harley@inria.fr>
To: AESFirstRound@nist.gov
Subject: AES Official comment

I would like to submit the following as an official comment on the first round of the AES process.

The first section is a message I sent to a Usenet disucssion and I am including it verbatim as it articulates my point of view as a mathematician, quite well.

For a Usenet reader who sseems to have kipped straight to the end of that message, I clarified what "proof of security" meant in part of a second message which I have also included below.

Thank you,

Robert Harley

Robert.Harley@inria.fr

---

From: Robert Harley <harley@corton.inria.fr>
Subject: Re: Who will win in AES contest ??
Newsgroups: sci.crypt
Date: 26 Jan 1999 17:25:07 +0100
Organization: I.N.R.I.A Rocquencourt

Serge Vaudenay wrote:
>People says DFC has not enough rounds, which is why it is faster. Then
>increase it from 8 to 12 rounds, and it will be as fast as the popular
>ones!

jerome@mycpu.org () replied:
>i missed something... a dfc with 12 rounds is not faster on 64b cpu and
>looses its speed advantage.

I think Serge's remark was somewhat tongue-in-cheek! Twelve rounds would probably be a bit excessive for DFC.

Presumably we want a code which is:

1. Possible on the low end.

2. Fast on the high end.

3. SECURE!!!

1. Low end.

Everybody seems to agree that on the low end are smart cards for which we need something that can get by with at most a few hundred bytes of RAM.

Just a few messages ago, Paul Crowley ruled out DFC and others claiming that they required "at least 195 bytes", whereas the DFC Web page links to a several-month-old paper describing an implementation in 100 bytes. Being too hasty and getting facts wrong sure won't help us make a good choice!

Note also that Gemplus (a major smart-card company) is involved in DFC so it appears that those who know the low-end well think highly of this particular candidate.

2. High end.

Here there seems to be some disagreement. Should the high end be the latest x86 chip? Or the latest Alpha? Or some unknown processor that will be popular in 2010? The conclusions are quite different depending on which you choose.

Many people have x86 machines, obviously, and NIST suggested testing on a 200 Mhz Pentium Pro. However some appear far too willing to dismiss out-of-hand all but the very fastest on that particular chip with the particular implementations they have at hand.

My machine is an Alpha (the outgoing 21164 generation) so for me DFC is the fastest candidate. Amongst the serious contenders it is fastest by quite a bit.

My personal opinion on this is that more effort should go into implementations on diverse types of processor. Bruce has done a very careful implementation on Pentium Pro, and even posted here recently to announce shaving 2 cycles off per round, but what about Alpha? Perhaps Twofish could be sped up there. Likewise for other codes.

I've implemented DFC myself on Alpha but am incapable of doing a good job on x86. Some guys at ENS did an implementation that takes about 750 cycles. I did one on ARM that takes about 750 cycles too, yet ARM is a very simple in-order single-issue RISC chip. Surely a Pentium Pro could beat it easily! I wouldn't be surprised if 20 or 200 cycles could be knocked off by a top-notch x86 programmer. Where's Terje Mathisen when you need him? =:-)

3. Security.

Don't overlook that security is more important than anything else! Imagine if some variant of DES had been chosen that was faster on the chip-du-jour in 1975 but got broken in 1985? The whole AES landscape would be very different.

A few candidates are easily ruled out but it would be a mistake to assume that the rest are all equal.

For now, the various AES teams are involved in guerilla warfare, trying every avenue to criticize the others. Read carefully and separate the wheat from the chaff. When Bruce Schneier and colleagues comment on 32-bit performance they criticize DFC as "The modular multiply does a nice job of diffusing bits, but hurts performance significantly." They're praising it with faint criticism!

At heart I'm a bit of a speed-demon but mostly a mathematician. There are some fast candidates, but that's not enough. There's plenty of advocacy going on, but much of it doesn't really matter. There are guys with good reputations involved. I respect that. But the clincher is:

WHAT CAN YOU PROVE? WHAT IS HARD, COLD FACT?

And what's just heuristics, conjectures and such like?

Heuristics and conjectures are important, especially in cryptography, but pale in comparison to the others. Take a step back from just DES and look at what lies behind cryptographic methods that have stood out since the seventies: you will surely have in mind things like factoring, discrete logarithms, exponentiation, finite fields and so on. When mathematics is brought to bear it really does a good job.

I would argue that DES was the end of an era: an era when cryptography meant pushing bits around in some complicated fashion, getting smart people to try to crack the resulting code and declaring it safe when they fail. This process produced DES and DES succeeded. But I think there was a large amount of luck involved.

The filtering process is of course necessary but it is not sufficient. Twenty to twenty-five years ago, cryptography came out into the open and nothing has been done the same since, save doomed attempts like Clipper, until now that is. The AES process seems to be swamped by candidates going back to the old way, submitted by people apparently

blinded by DES's lucky success.

If that's what is wanted then go with triple-DES!

Take a look at the serious candidates from a mathematical point of view. There are a lot of similarities: Feistel this, XOR that... But one stands out. One says: hey guys here's a proof that such-and-such attack can't work, nor this other attack nor some entire classes of attacks. And they're not "straw-man" attacks that are being held at bay either, they're genuinely useful against some other codes. You may have missed the crucial word so I'll repeat it: proof.

Now you may be wondering why I'm laying down a strong argument which happens to favour DFC. Maybe the designers are friends of mine? No: I just met them for the first time a few days ago. Maybe its some primitive nationalism, after all I'm posting from France. No: I'm Irish.

I'm defending DFC because it melds the best of the old DES era and the new mathematical one; because it has gone through the AES process to date with as much distinction as the other good candidates, withstood attempts to break it, and because on top of that it is backed up by guarantees, when all the others have to offer is rhetoric.

So forget all the business with "candidate A is bla% slower/faster than candidate B on processor X, but on Y mumble mumble".

None are perfect. But one comes with proof. The others don't.

WHAT ARE YOU PEOPLE THINKING?

=:-)

Bye,
Rob.

---

---

From: Robert Harley <harley@corton.inria.fr>
Subject: Re: Who will win in AES contest ??
Newsgroups: sci.crypt
Date: 28 Jan 1999 23:06:00 +0100
Organization: I.N.R.I.A Rocquencourt

"Sam Simpson" <ssimpson@hertreg.ac.uk> writes:
> Found your message entertaining (and much agrees with my thoughts about the
> current stage of AES...). One small comment;

Ah, so there is hope yet!
Actually, several people have expressed hearty agreement in email.

>>None are perfect. But one comes with proof. The others don't.
>
>There is a "proof" of DFC's security?

There are proofs (no quotes are necessary) of DFC's security against
various classes of attacks.

Consider differential attacks. These are pretty serious and several
codes which otherwise appear secure can be broken by them.

Of course most AES teams have taken them into account and can say "we
think our code is resistant". The DFC team doesn't say "we think".
It says "Our code is resistant. Here's the proof. End of story.
That's that. Any questions?"

The same holds for linear attacks.

The same holds for some iterated attacks.

Does this not strike anybody else as fundamentally different, better
and phenomenally useful?

A little-known fact about the decorrelation method is that it could be
used with other AES candidates. Essentially you start by designing an
algorithm with heuristic security, doing your best and ending up with
something like Mars or RC6. Then in a perfectly modular fashion, you
append a pass that guarantees, with proof, all sorts of security that
was previously relegated to conjecture (at best)!

If some algorithm other than DFC wins the AES process, then surely we
would want to apply this strengthening to make it even better than it
already was.

How could anyone not want this?

The thing is, the resulting algorithm would look to a large extent
just like DFC!

[...]
---

X-Sender: jon@mail.pgp.com
X-Mailer: QUALCOMM Windows Eudora Pro Version 3.0.3 (32)
Date: Mon, 01 Feb 1999 13:58:06 -0800
To: AESFirstRound@nist.gov
From: Jon Callas <jon@pgp.com>
Subject: AES Comments
Cc: "**Jon Callas**" <jon@pgp.com>

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Information Technology Laboratory
Attn: AES Candidate Comments, Room 540
National Institute of Standards and Technology
100 Bureau Drive, STOP 8970
Gaithersburg, MD 20899-8970

I am writing to provide comments on the candidates for the AES on behalf of Network Associates, Inc. Network Associates is a corporation that makes computer network management and security software. We formed from the merger of a number of companies, including McAfee Associates, Network General, Pretty Good Privacy, and Trusted Information Systems. We are not producers of algorithms ourselves, but rather are consumers of them. We are interested in the AES process because we use cryptography as one building block in our management and security systems, and we are looking forward to using the AES in our systems. We also have a research organization that is a consumer of cryptographic algorithms.

The evaluation criteria that you describe in the Federal Register touch all the important facets of evaluation, but there are some comments I would like to make:

Security is the sine qua non of the evaluation, that without which we have nothing. There is no reason to have an insecure AES. On the other hand, we also have as a requirement that AES be at least as fast as Triple-DES (or else, why bother switching). Furthermore, there are a number of dimensions on which security can be judged, and one cannot help but cast an eye to the sort of attacks that may be possible in the future. While this is ultimately crystal-ball gazing, we've nonetheless seen hints of the attacks of the future in Timing Attacks and Differential Power Analysis. I generalize the two of these to a term I call "Differential Spectrum Analysis," which covers both time differentials, power differentials, or anything else that an attacker can use, such as radio-frequency emissions of a CPU.

Paul Kocher's work on differential spectrum analysis has shown some embarrassingly simple results in breaking smart cards. I found it jaw-dropping to see that examining the power consumption of a number of smart cards lets you see the bits of a DES key as they're scheduled.

In my opinion, the moral of this story is that the security threats of the future are going to come from breaking the machines that embody cryptography, not from the cryptography itself. Consequently, I think that the simplicity of an algorithm is also a vital part of its

security. I have considered that in my recommendations.

Simplicity is a hard thing to define in any context, let alone this one. However, what I mean is that if an algorithm is constructed in such a way that naive implementers could easily make a secure machine out of it, it is simple. If the straightforward implementation would make an easily breakable machine, then the algorithm is not simple. Not only that, but it is arguably not secure because of this lack of simplicity.

As I have been examining the AES candidates, I have come to the conclusion that it should be my task to recommend a suite of candidates for Round 2. It is hard to decide on a single algorithm, because there are so many to choose from. There are a few algorithms that have had flaws found in them. I have therefore eliminated them from consideration, no matter how promising they were otherwise. There is also another group of algorithms that meet all of the requirements yet excel in no particular area. Sure, they are resistant to linear and differential cryptanalysis, but who isn't these days? They aren't as fast as some, don't have any special design features, and aren't particularly small. Consequently, they fall below my own bar not because of any flaw, but simply for not being delightful. I find this somewhat sad, but inevitable.

This having been said, my comments and recommendations follow. They are for a group of algorithms that I think make a good group from which to pick a final algorithm. I have applied all the recommended criteria, but also included my belief that ease of implementation is increased security.

(1) Twofish

Twofish is a good cipher, an advanced Feistel cipher. It also has perhaps the best paper of any of the submissions. The description of Twofish is especially thorough, including space-speed tradeoffs, semiconductor gate counts, and self-cryptanalysis. In reading the Twofish description I feel that I know it best. That is the reason that I mention it first.

Twofish has an innovative yet not radical design, and is one of the fastest and smallest ciphers presented. The only downside I can see in its design is its use of one-bit rotation. Presently, I fear rotations from the standpoint of simplicity. I see this as a place where information may leak from a cryptographic machine. On the other hand, I realize that this fear is most likely irrational, but I see it as something to be examined as we proceed. I will also add that this concern also applies to most of the ciphers in my recommended group, so even though I find it a concern, it obviously isn't a large one.

(2) Rijndael

Rijndael has a number of interesting characteristics that bring it into my list of recommendations. It's a Square descendent, and thus genetically different in interesting and basic ways from the other ciphers, which are all relatives of our old friend, the Feistel Network.

(I have to mention the other Square descendant, Crypton, here. It's also a good cipher, and in the opinion of some of my colleagues, a better Square descendant than Rijndael. I am fortunate, in an odd way, that a flaw has been found in its key schedule, or I would have a very difficult decision on which one of them to recommend. They are both excellent ciphers, and while I think that it is worth including one Square descendent in my group in the interest of encouraging genetic diversity, I couldn't justify two. This small flaw, in particular, made me feel like an Olympic judge seeing an itty-bitty fault. Oops; mandatory deduction, and because there are so many worthy participants, that means you lose. It's sad and not fair, but it makes the job of those of us who are evaluating them easier.)

Rijndael has a number of interesting characteristics that make it a good candidate for the AES. It's very fast, possibly the fastest of all the submissions. It can execute in a small environment, and works well on both small processors and large ones. It has interesting security characteristics, too; I speculate that the techniques used in Rijndael may be especially resistant to spectral analysis.

Now on the other hand, the genetic difference that attracts me to Rijndael is also a concern. There has been a lot of work done on Feistel networks, and not nearly as much done on the Square system. However, the positive features that Rijndael possesses more than outweigh this uncertainty. In every other dimension, Rijndael is a star.

(3) Serpent

Serpent is perhaps the best-designed cipher of all the submissions, for a security standpoint. Its bit-slice design is innovative and addresses a number of types of attack, including the timing attack. My intuition is that it will also help protect the cipher from other forms of spectral analysis.

The downside of Serpent is that its conservative design makes it a little slower than the previously mentioned ciphers. This cuts to the quick of one of the central questions of the AES process. Security is important, but so is speed. How much security is enough, and once we have enough, do we want a little more security, or perhaps a little speed? This question will be, I believe, the important question of the next round.

(4) RC6

RC6 is the most elegant and beautiful of the submissions. Among its many virtues, it's a joy to look at. There are but two concerns that I have about it, one technical, and one non-technical.

My technical concern concerns its use of 32-bit multiplication and data-dependant rotations. Since multiplication is not present on small

processors, it is emulated, and thus not a constant-time operation. Even on some larger processors, which can do 32-bit multiplication, the operation is of variable time. This opens up the possibility of timing attacks. Similarly, I have concerns about data-dependent rotations. It is relatively easy to address these concerns, but they are nonetheless concerns.

My non-technical concern is about intellectual property. While the AES process states that the intellectual property of the AES cipher itself is to be yielded, there is nothing in the process that covers the event that the AES infringes on some other patents. In RC6's case, there are questions of whether it might infringe the RC5 patent, or others. These are easily solved, but need to be addressed. It would be a pity to have questions surround the AES the way they surround the DSS.

(5) Mars

Mars is a virtuoso performance. It's an amazing design from an amazing team. It uses most of the tricks in the book and is arguably the pinnacle of Feistel cipher design.

However, this virtuoso performance is also Mars's weakness. If multiplication opens up an opportunity for timing attacks, then Mars is vulnerable to it. If rotations are best avoided, Mars is vulnerable to that, too. And so on.

Many of these concerns are just worries. It's likely that twenty years from now someone will read my worries and chuckle. I certainly hope they do. Nonetheless, Mars is a virtuoso performance, and just as we prefer piano-bar musicians don't attempt Liszt, do we want smart-card programmers to have to understand Mars?

That, however, is a question for the next round.

(6) Honorable Mentions

There are two ciphers that I feel deserve an extra mention. The first is Crypton, which I mentioned in my section on Rijndael above. The second is the Hasty Pudding Cipher. While I don't see it as a viable candidate for the AES, the HPC paper raises a number of interesting questions. Should a cipher be able to encrypt not only blocks, but also fields? Should it be able to encrypt a field that doesn't easily fit into bits? HPC was the only submission that I found to be philosophically thought provoking, and am glad we got the opportunity to see it.

In summary, I would like to repeat my recommendation. I recommend a group of five ciphers, all good ciphers, all with characteristics that would make them a good AES. These are Twofish, Rijndael, Serpent, RC6, and Mars. Whatever the outcome of the final

winnowing, these are all fine ciphers. This final winnowing is no doubt going to be harder. Each member of my recommended group shines in an interesting way. It was easier for me to pick my five favorites of fifteen, than it will be to pick one out of the last five. There will need to be a good deal of analysis for that over the next year.

Sincerely,

/s/ Jonathan D. Callas

**Jonathan D. Callas**
CTO, Total Network Security
**Network Associates, Inc**.
3965 Freedom Circle
Santa Clara, CA 95054
jcallas@nai.com, jon@pgp.com
(408) 346-5860

-----BEGIN PGP SIGNATURE-----
Version: PGP 6.0.2

iQA/AwUBNrYi/n35wubxKSepEQJscQCg1w/Gx4aLe+MNEf1w7IgZh+MTkLwAoIRC
X+Dh2eGwt7NZGv3XyW4MIvpR
=Bp4T
-----END PGP SIGNATURE-----

---

Jon Callas jon@pgp.com
CTO, Total Network Security 3965 Freedom Circle
Network Associates, Inc. Santa Clara, CA 95054
(408) 346-5860
Fingerprints: D1EC 3C51 FCB1 67F8 4345 4A04 7DF9 C2E6 F129 27A9 (DSS)
665B 797F 37D1 C240 53AC 6D87 3A60 4628 (RSA)

Date: Wed, 03 Feb 1999 12:06:06 +0100
From: Jurjen <J.Bos@interpay-iss.demon.nl>
Organization: Interpay
X-Mailer: Mozilla 4.04 [en] (Win95; I)
To: aesfirstround@nist.gov
Subject: Banking applications need fast 8-bit performance

Hi everyone,
I work as a cryptographer for Interpay (the company which supports performs virtually all payment transactions in the Netherlands). In the Netherlands, there are on average over 30 transactions per second, with peaks of over 300 per second. Therefore, as a large user of cryptography, I emphasize the needs of the financial world for a block cipher.

The first point of concern is implementation speed on cheap processors. The reality of today is that most of the DES applications are running on 8-bit processors. These are both DES applications in the smart card (over 15 million are in use in the Netherlands today) but also in terminals (over 150000 in use). There is no reason to assume this is changing soon (for example, Philips' brand new contactless Mifare smart card has a 8051 processor).

Good examples of efficient 8-bit behaviour are Serpent and Rijndael, because they avoid expensive operations such as multiplication, and large tables. Especially Rijndael looks interesting, since its encryption speed is faster than the decryption speed, allowing for optimization of speed in two party protocols.

Our second concern is differential power analysis. The effort needed to protect DES against DPA must not grow when switching to AES; preferably, implementing AES instead of DES on a given piece of hardware should increase DPA immunity. On this terrain, Serpent is one of my favorites, since it can be implemented with no key or data dependent jumps.

If you make a choice, realize that we expect to have over a billion 8-bit implementations of AES in the next few years: give the programmers a break!
--
**Jurjen N.E. Bos**
Information Security Services
**Interpay**
postbus 30500, 3503 AH Utrecht
the Netherlands
+31 30 283 6815

Date: Wed, 3 Mar 1999 17:14:54 +0100 (MET)
From: **Robert Harley** <Robert.Harley@inria.fr>
To: AESFirstRound@nist.gov
Subject: Official Public Comment

I would like to submit the following as an Official Public Comment.

It refutes the suggestion that DFC is particularly vulnerable to timing attacks and furthermore refutes the suggestion that it "may be impossible" to test the rare code-path in DFC.

Bye,
  Rob.

NB: This is a message I sent to the sci.crypt Usenet newsgroup.

==========

From: Robert Harley <harley@corton.inria.fr>
Subject: Re: AES2 papers now available at NIST
Newsgroups: sci.crypt
Date: 03 Mar 1999 17:03:33 +0100
Organization: I.N.R.I.A Rocquencourt

Henry Lewsal <hdl834@krill.net> wrote:
>After reading the AES papers related to smart cards, it seems
>that none of the candidates were designed to be resistant to
>timing attacks and power analysis attacks.

I don't know about power analysis but it is easy to make DFC resistant to timing attacks.  I did it before but it costs a few cycles.  Just apply the following 'diff' to the rf() function in the Appendix of the "DFC Update" AES2 paper and Bob's your uncle:

```
<  if (!t) return CP;
<  b += 13UL; return CP;
---
>  b += 13UL*(t != 0UL);
>  return CP;
```

Just to prove the point, I've appended the compiled output for the dfc() function on Alpha at the end.  No branches in sight.  Same with all combinations of compiler and optimisation flags that I've tried.

Also, on Louis Granboulan's timing table the footnote to the 393-cycle Pentium Pro implementation mentions:

>This implementation can be made totally branchless and
>data-independent at a cost of 40 cycles (protection against timing
>attacks).

That was done by Terje Mathisen.

---

I just saw the AES2 paper called "The DFC Cipher: an attack on careless implementations" by Ian Harvey. It seems to be arguing that codes with broken implementations are broken.

It starts off by observing that implementations that aren't protected against timing attacks may be subject to timing attacks.

This is vacuously true and not specific to DFC.

It mentions:
>High-level languages such as C do not have an "add with carry"
>operation, so carries need to be implemented with some form of branch.

No. See the C implementation already mentioned.

>The following "bug attack" scenario becomes a possibility:
>[...If your implementation has a bug then it's broken...]

This is vacuously true and not specific to DFC.

>Alternatively an attacker may be able to tamper with an implementation
>[...if an attacker can change your implementation he can break it...]

This is vacuously true and not specific to DFC.

>Conclusion
>
>While not weakening the theoretical basis of the DFC cipher [...]

It sure doesn't!

>it may be impossible to produce a set of standard test vectors which
>validate all parts of the design [...].

You should certainly test all code-paths. This is vacuously true and not specific to DFC.

However the author is suggesting that it "may be impossible" to test the rare code-path in DFC.

*** This seems to be the only actual conceivably-valid point. ***

---

However it is false since it is trivial to produce test-vectors that exercise the rare code-path.

The rare code-path is entered when b < v, where b is a random 64-bit number and v is small (fits in a byte). However b is calculated as: rk.lo+"some other stuff". You can freely choose the rk.lo value used in the first round so that b is any value you like.

Just to prove the point, allow me to demonstrate an example. Pick a test-vector such as the first one in the C implementation already mentioned. In the key, set ek[0].lo to zero and stick a printf() in the round function:

```
  b += l; v += b < l;
>  printf("-b = %lu\n", -b);
  v *= 13UL;
  t = b < v; b -= v;
```

Run it. It prints:

-b = 16263440665651997398

Now set ek[0].lo to that value and you're done.

---

To be sure of that, you can put a bogus assertion in the rare path:

```
  if (!t) return CP;
>  assert(0);
  b += 13UL; return CP;
```

and run it:

```
-b = 0
rf: Assertion `0' failed.
```

I think that pretty much demolishes the entire paper.

However power analysis, as discussed in the Daemen and Rijmen paper
for instance, is another matter entirely and I'm not qualified to talk
about it so I won't.

Bye,
  Rob.

PS: The average timings of the best DFC implementations I've seen so far are:

| | |
|---|---|
| Pentium: | 609 cycles |
| StrongARM: | 555 |
| Pentium Pro: | 392 |
| Alpha 21164: | 310 |
| Alpha 21264: | 231 |

and no, there are no bugs in the rare code-paths!  =:-)

---

Output for dfc() by gcc-2.8.1 -O3, verbatim:

```
ldq $1,0($19)
ldq $2,8($19)
bis $16,$16,$0
bis $17,$17,$6
lda $8,29850
lda…
```

[Remainder of code removed from electronic copy of public comments, due to export restrictions]

Date: Thu, 04 Mar 1999 17:59:28 -0800
From: **Joe McGivern** <joe@metaflow.com>
Organization: **Metaflow Technologies**
X-Mailer: Mozilla 3.01Gold (Win95; I)
To: AESFirstRound@nist.gov
Subject: TWOFISH

To whom it may concern:

My name is Joe McGivern and on behalf of Billy Cole and myself, I offer the following comments in support of the TWOFISH encryption algorithm. For the record, I hold no affiliation professional or otherwise with Bruce Schneier or Counterpane Systems, Inc.

I am an Electrical/Computer engineer working in microprocessor design in Souther California. Recently, I came up with an idea for an encryption product which I have since been developing with my friend and business partner Billy Cole. In fact we intend to start a business around it (our second one). Being new to encryption, it was my job to seek out the best possible encryption engine for our product. Because our product does all of it's encryption and decryption on-the-fly while our end user waits, I had to find something very fast and very secure.

Having read some 1000+ pages of documentation now on the subject of encryption and reviewed all possible algorithms available, I believe TWOFISH to be the best of the best for our application. ...and as an added bonus it's free to the public and more importantly entrepreneurs like ourselves. As we are just now beginning to impliment the encryption engine, I'm finding the TWOFISH source to be both readable and easy to work with. The research into every aspect of the algorithm is what makes me confident that I have chosen well, especially the attention to compiler optimization and processor efficiency.

Also, I have found Mr. Schneier's book "Applied Encryption" to be the guiding light as we wrestle with many difficult design decisions. I can now say with confidence that our design will meet our end users expectations and leave the hackers frustrated and confused. We have taken great care to heed Mr. Schneier's words and eliminate weaknesses.

So, from a couple entrepreneurs out here in America, I can honestly say the TWOFISH algorithm is much more than an excellent design, it's accessability is helping give us a crack at the American Dream.

Sincerely,

Joe McGivern
Billy Cole

Date: Mon, 22 Mar 1999 17:20:52 -0500
From: "**Colin St. John**" <colinsj@delphi.com>
Subject: Twofish
X-Sender: colinsj@pop.delphi.com
To: AESFirstRound@nist.gov
X-Mailer: QUALCOMM Windows Eudora Pro Version 3.0.3 (16)

Dear NIST,

I have read Bruce Schneier's book and his papers that analyze encryption algorithms, along with his comments on encryption technology on the sci.crypt newsgroup. I have also studied his Blowfish and Twofish algorithms. I would like to state that I have full confidence in Bruce Schneier's ability to design and analyze a secure, efficient encryption algorithm, and would like to state my support for the Twofish algorithm as NIST's best choice for the Advanced Encryption Standard.

Colin St. John

X-Sender: oshel.david@email.mcleod.net
Date: Fri, 26 Mar 1999 08:01:18 -0600
To: dianelos@tecapro.com, aesfirstround@nist.gov
From: "**David C. Oshel**" <dcoshel@pobox.com>
Subject: Re: AES: Georgoudis on AES2, Day 2, Rome (FW)

Dianelos Georgoudis wrote:
>.....
> The question of number of rounds came up. NIST said it would
> consider a cipher secure if it could not be broken with 2^64
> chosen texts. A thoughtful counter-argument .....
>..... gives a good idea about an
> algorithm's workload. Also, speed could be normalized at such a
> high level that ALL competing ciphers would be pushed into
> insecurity which would then allow more or less reasonable
> quantitative comparisons to be made about their relative
> "structural" strenght. It is clearly too late to use this
> methodology in the first round but maybe it could be used in the
> second. I feel like sending NIST an official comment in this
> sense. I wonder what the reader thinks about this?

"Normalized" security defeats the purpose of finding a secure AES candidate, doesn't it? I'd say your proposal makes more sense than attempts to break down the competition with phony "fair play" arguments not based on the merits of each candidate. Obviously, the number of rounds required for a secure implementation is a fundamental attribute of the algorithm, not an optional feature.

David C. Oshel dcoshel@pobox.com
Cedar Rapids, Iowa http://pobox.com/~dcoshel/
``Tension, apprehension and dissension have begun.'' - Duffy Wyg& in Alfred Bester's _The Demolished Man_

X-Originating-IP: [210.169.194.18]
From: "nobuki nakatuji" <db1011@hotmail.com>
To: AESFirstRound@nist.gov
Subject: AES First Round
Date: Sun, 28 Mar 1999 01:44:57 PST

You should select two algorithms as AES.
It is difficult if a weak point is found after only one algorithm was mounted on a lot of software and the hardware.
It can cope with it even if there is inconvenience in either if it tries so that it has two algorithms changed and it is mounted.

**Nobuki Nakatuji**
db1011@hotmail.com

From: dario.forte@computer.org
To: AESFirstRound@nist.gov
Subject: comments to the 2.nd aes conference
Date: Sun, 28 Mar 1999 22:45 +0200

greetings from italy

i'm an italian security analyst & techncal writer, my core activity is security and cryptography. recently, with the hospitality of NIst , i was attendant of the second aes conference in rome, italy. after the end of the conference, i invited some italian security analyst, and some of my readers (50 persons) to comment the paper submitted at the conference and send to me a feedback. Following the result of my little "request for comment", i hope it will'help you in your evaluation.

The 65 % of the person, think that the java performance of the algorithm is a non fundamental issue. The reason is: if you need great performance in your environment, you don't use java.

The same percentage stated that, at least for the next 3/4 years, the processor on the desktop and the server environment will be at 32bit; so, actually, some algorithm extremly fast on 64bit processor, could be interesting but, at this moment, people don't need it.

Talking about security, excluding magenta ;-)) , Rc6, Rijndael, Twofish, Dfc and Cast 256 seems to be the most secure.

Asked about : "which algorithm could win the contest", people intervisted said that twofish e rijndael are the favourites, but twofish could be the best choice, cause is the most "equilibrate", in pratice, the most Aes compliant.

The result of the investigation will be published on the italian magazines "PC WINDOWS" and HI TECH SERVER, edited in italian language. Please don't hesitate to write me if you need a pdf file.

Thanks for your attention.

**Dario Forte**
CSI/ICSA/USENIX member
Technical Writer
www.darioforte.com

Date: Fri, 9 Apr 1999 15:05:57 +0200 (MET DST)
From: **Vincent Rijmen** <vincent@ii.uib.no>
To: aesfirstround@nist.gov
Subject: top 5

Dear AES evaluator,

Hereby I submit my personal top 5 of the AES candidates.
The 5 algorithms I consider most promising are (in alphabetical order):
Crypton, E2, RC6, Rijndael and Serpent.

My motivation is the following:

Frog, HPC and Magenta are designed by people who have not been following the latest developments in the design and analysis of block ciphers. The algorithms cannot be trusted to have any security.

Loki97 is broken.

DFC is slow. The authors claim they have a proof for its security, but as shown in [1], this proof is worthless.
While the algorithm has not been broken yet, I think we are not far from an attack on the full version. Anyway, there is no reason why the slow DFC should be preferred over fast and more secure alternatives.

Cast256 is secure, but slow.

DEAL is too slow.

SAFER+ is slow.

The remaining 7 algorithms (my top 5 and Mars and Twofish) have been designed by teams that included cryptographers with a strong reputation (Crypton indirectly, via its ancestor Square).

Mars and Twofish are also secure and fast algorithms. I consider them to be good AES-candidates, but I think my top 5 algorithms are better. Mars and Twofish both are complex designs, difficult to analyze, making it difficult to assess their security.
The top 5 algorithms have all some clearly formulated design principle and are designed according to these principles. They are elegant and simple.

I hope you may find these comments useful.

Sincerel yours,

Vincent Rijmen

[1] L.R. Knudsen, V. Rijmen, ``On the decorrelated fast cipher (DFC) and its theory,'' presented at FSE99, Rome.

# Vincent Rijmen |
# | Any philosophy that can be put "in a nutshell"
# | belongs there. -- Sydney J. Harris
# vincent@ii.uib.no |

From: zunic@us.ibm.com
X-Lotus-FromDomain: IBMUS
To: AESFirstRound@nist.gov
Date: Fri, 9 Apr 1999 16:32:30 -0400
Subject: Official Comments from IBM's AES Team

Listed below are the Official comments from IBM's AES team for the AES selection process. If you have any questions, please contact me.
**Nev Zunic**

**IBM's AES Team**
9 April, 1999

1. Recommended Finalists

We recommend that five candidates, MARS, RC6, Twofish, Serpent, and Rijndael, be selected as finalists. These candidates seem to possess the best combination of security and performance. While other candidates are also excellent ciphers, we feel that only these five should be selected, so that further evaluation can concentrate on a reasonably sized set. The following outlines some of the positive and negative features we observed in the recommended candidates.

MARS

MARS has one of the widest security margins, both in terms of number of rounds [5], and in terms of diversity (as its security relies on a combination of several different "strong operations" and on a heterogeneous structure). MARS is the only candidate with a heterogeneous structure, which was a deliberate design feature to help resist unknown attacks. It certainly has the largest security margins among the faster ciphers. Also, the design of the round function in MARS lends itself to analysis. In particular, a nearly complete characterization is known for the differential behavior of the round function.

At the same time, MARS is also a very fast cipher. In fact, in some of the measurements, MARS posted the fastest C and Java benchmarks. While other candidates performed as well or better in assembly language, C and Java will undoubtedly be used in a significant number of implementations.

A possible concern about MARS is that it is currently hard to implement on low-end smart cards. We do not consider this to be a concern as outlined in comment #5 below, regarding low-end smart cards.

RC6

RC6 has a simple, elegant round function, and it is the fastest cipher in assembly code speed tests.

A possible concern about RC6 is that its round function may be "too simple". Specifically, the combination of multiplications and rotation, although providing some excellent properties, is a "single point of failure" in RC6 (as it does not use S-boxes). Also, RC6 has one of the lowest security margins of the recommended candidates in terms of number of rounds.

Twofish

Twofish is perhaps the most flexible cipher in terms of implementation tradeoffs, and it is also one of the fastest ciphers (except for its key-schedule).

A concern about Twofish is that it is very hard to analyze its security. Its round function was engineered to provide flexibility, rather than to facilitate analysis. Indeed, although a lot of effort has already been invested in its analysis, it is safe to say that the exact properties of the round function are not very well understood. Moreover, the reliance on key dependent S-boxes which are not generated pseudorandomly, makes the analysis even harder.

Another drawback of the key dependent S-boxes is that they are inherently more costly. In Twofish this extra cost can be shifted between the key-setup and the cipher, but nonetheless it is always there. Finally, the key schedule of Twofish makes power attacks easier, since the entire key can be deduced from only the initial whitening key.

Serpent

Serpent has very wide security margins in terms of number of rounds, and very strong mixing. On the down side, it is somewhat slow, and it also has a key schedule that makes power attacks easier to mount.

Rijndael

Like Twofish, Rijndael is also a fast cipher, which is very flexible for implementation. However, like Twofish, it has a round function which is hard to analyze, and a key schedule that makes it easier to mount power attacks.

Also, the fact that the round function can be expressed as only a few simple algebraic operations makes one wary of potential algebraic attacks against it.

2. The issue of more than one winner

We recommend that NIST select one winner only. While some have suggested that picking more than one winner would provide insurance against the future failure of a single winner, many systems (particularly hardware) will implement just one. Multiple winners will raise the question of

how to revoke an algorithm if it does have problems. We feel that the potential benefit is not worth the additional complexity.

3. Suggestions for future evaluation efforts.

The submitted algorithms have significantly different security margins. More attention needs to be paid to security margins in subsequent rounds. It is our opinion that when deciding on a standard for the next few decades, wider security margins are more important than small differences in speed.

4. Variable Number of Rounds Issue

In the current state of affairs, there is a possibility for the submitter of the winning algorithm to potentially benefit from a discovered weakness in the algorithm. That might happen, if the AES winner had its number of rounds established at a certain fixed setting and then (5 or 10 years later) a weakness is found. This does not have to be a real attack, but just some weakness suggesting that the number of rounds should have been higher in the first place. In that case, vendors and users of AES would naturally want to use a higher setting, perhaps even if it meant that they were not using AES per se. But, using such a higher setting might then require royalties to be paid to the submitter (i.e., the AES winner) in order to make use of this higher setting.

It appears that the only avenue NIST has to address this general concern would be for NIST to extend the envisioned AES FIPS by stating its intention to provide an option for increasing the number of rounds in the AES, in case more rounds are needed or desired for security. Therefore, we recommend that the AES FIPS be extended to provide an option for increasing the number of rounds that would give additional security. NIST should incorporate the variable number of rounds option within the AES criteria, thus making the increased number of rounds a part of AES and thereby subject to the royalty-free grant, as specified in section 2.D.2.

5. Smart card issues

We recommend that with regard to the AES candidate selection process, NIST should not give extra credit to a candidate algorithm simply because it can be implemented on a current generation low-end smart card, (e.g., cards with less than 128 bytes of RAM). The rationale for this recommendation is as follows:

(a) Low-end smart cards are insecure. Keys cannot be protected due to low-cost, low-tech, power analysis attacks. This was amply demonstrated at the AES2 conference by the following three papers [1,2,3]. In particular,

in [3] it was shown how a 128-bit Twofish key can be extracted from an ST16 smart card with only 50 power samples. The ST16 in question was not even a low-end smart card, it had 384 bytes of RAM and two independent on-board random-number generators which resulted in significant noise in the power signals. Consequently, attacks on low-end smart cards with 128-bytes of RAM and no/low-quality random number generators should be even more severe; attacks involving only 10-20 samples may be possible.

Contrary to some opinions raised at AES2, these attacks CANNOT BE AVOIDED AT THE SYSTEM LEVEL by using sound protocols (such as those in which the smart card user has no incentive to attack the card). Since only a small number of samples are required for an attack, it is possible to collect enough samples during the normal course of smart card usage at a public facility where only the smart card reader has been compromised. Smart card systems as they exist today, do not and cannot include the smart card reader within a physically secure boundary.

(Another approach would be to use protocols where all smart card keys are refreshed after 5-10 uses. This would put a tremendous burden on the infrastructure to keep all points of interaction with a smart card synchronized, thus negating one of the biggest advantages of using smart cards, i.e., off-line usage. Such protocols would also stress the smart card's EEPROM where keys are kept, since each EEPROM cell can be written only a limited number of times. In fact in [4], the limited EEPROM write limitation is used to justify why the key-expansion should use only RAM and not EEPROM.)

(b) Users in a security-sensitive application requiring smart cards will be willing to pay a little extra for suitable smart cards. These already exist today and are likely to get even cheaper by the time the AES is adopted.

(c) Smart card technology is advancing rapidly with current focus being Public Key Cards, Multi-Function Cards and Multi-Application cards such as JavaCard. As a result, current high-end smart card offerings such as the Siemens SLE66 or SGS-Thomson's ST19 have over 1KB of RAM and built-in hardware countermeasures against power analysis. These cards have enough resources to accommodate most AES candidates even today. Moreover, the hardware countermeasures built into cards such as the ST19 allow them to execute even straightforward implementations of AES candidates and still resist power attacks involving a moderate number of samples. This number can be increased substantially using software countermeasures as outlined in [3] at the cost of increased resource usage. On such systems, with good protocol design (i.e., by ensuring that an adversary cannot collect an excessive number of power samples) the overall system can be made practically secure against power attacks.

In the very near future, 16-bit and 32-bit processor based smart cards

will also become more popular, e.g., the Philips SmartXA Smart Card which is expected to be available before the end of 1999, features a 16-bit processor, 2.5K RAM, 48K ROM and 32K EEPROM. Examples of 32-bit RISC based cards currently available include GemExpresso from Gemplus. Given this trend, the limitations on memory will not be a major issue by the time the AES is implemented within Industry applications.

(d) The selection of the AES should not be skewed using unrealistic criteria. The low-end smart card issue is an example of such an unrealistic criterion.

6. Java/C performance

Many of the performance analysis papers have stressed assembly language performance values. While these do give important values for maximum algorithm speed, Java and C performance results are still very important. Most existing crypto libraries are implemented in C, and portable Java implementations will become equally prevalent. We feel that it is important for NIST to consider performance results in Java, C, and assembly implementations.

References
==========

1. Power Analysis of Key-Scheduling of the AES Candidates. Eli Biham, A. Shamir Second AES Conference.
2. Resistance Against Implementation Attacks: A Comparative Study of the AES proposals. J. Daemen, V. Rijmen. Second AES Conference.
3. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. S. Chari, C. Jutla, J. Rao, P. Rohatgi. Second AES Conference
4. Performance Comparison of AES Submissions. B. Schneier. J. Kesley, D. Whiting, et al. Second AES Conference.
5. A Note on Comparing the AES Candidates, E. Biham. Second AES Conference

Internet: zunic@us.ibm.com
IBM Crypto Solutions
(914) 435-6949 (T/L 295)

To: AESFirstRound@nist.gov
cc: Michael.Roe@ccsr.cam.ac.uk
Subject: Comments on the AES Candidate Algorithms
Date: Wed, 14 Apr 1999 12:12:49 +0100
From: Michael Roe <Michael.Roe@ccsr.cam.ac.uk>

Comments on the AES Candidate Algorithms

**Dr. Michael Roe**
**Centre for Communications Systems Research**

1. Errors and ambiguities in the specifications

The specifications of most of the candidate algorithms do not have the clarity and freedom from ambiguity that one would expect in a published standard. At least one of the specifications (Rijndael) is inconsistent with the supplied test vectors and reference implementation. Many of the candidate specifications have ambiguities related to the ordering of bits within bytes or bytes within words.

Whichever candidate is selected, its specification will need to be improved before it can be published as a standard. Furthermore, I would like to suggest that revised specifications should be issued as early as the start of round two of the evaluation, in order to correct the worst of the errors and ambiguities.

1.1 Byte ordering

Many of the candidate specifications suffer from byte-ordering confusion: when a string of 4 octets is treated as a 32 bit integer, is the most significant octet the first or the last?

Implementations of the AES ought to behave in a compatible manner whatever the underlying endianess of the machine on which they are implemented. Data enciphered on a 'big endian' machine ought to be decipherable on a 'little endian' machine. If the standard does not have this property, then we won't have interoperability of communications protocols between machines with different hardware architectures.

To avoid ambiguities over the endianess of the plaintext, ciphertext and key, the AES cipher should be specified as function that takes as input two arrays of octets (the plaintext and the key) and produces an array of octets as the output. If the internals of the cipher involve converting from sequences of octets to integers, then the endianess of this conversion needs to be specified as part of the AES standard.

In the past, some cipher designers have tried to avoid endianess issues by

defining the plaintext and ciphertext as arrays of 32 bit words. This doesn't solve the problem, it just moves the problem elsewhere in the specification of the system that uses the cipher. Typically, the data to be enciphered will be defined as a sequence of octets (e.g. because it is an encoding of a data type defined in ASN.1 or XDR), and the transport layer that moves the ciphertext is defined as acting on sequences of octets (e.g. TCP/IP or the OSI protocol stack); if the cipher is defined as acting on 32 bit words, then the specification of the system which uses the cipher still needs to define how data is transformed into 32 bit words prior to being encrypted and how it is transformed back to octets afterwards. Therefore the AES should not use this approach to avoiding endianess issues.

The AES test vectors also suffer from inadequate specification of endianess. The test vectors give plaintext, ciphertext and key as strings of hexadecimal digits. How are these strings to be converted to arrays of octets for the purpose of being input to the block cipher? This isn't defined, and it should be. There is a natural way to define this, which avoids dependency on the machine's endianess: the leftmost two characters of the string are the first octet in the array (and of the two hexadecimal characters, the leftmost is the most significant). Most of the AES candidates follow this convention for the test vectors. However, some (e.g. Serpent) do not follow this convention, but write the octets from right to left. This shows that this ambiguity in the specification of the test vectors leads to real confusion. For the next round of the AES contest, the published test vectors should be fixed so that they all write the octets from left to right.

### 1.2 Bit ordering

Some of the AES candidates refer to numbered bits within octets (e.g Rijndael, in the definition of ByteSub). If individual bits are referred to, the bit ordering needs to be defined: is the least significant bit numbered 0, 1, 7 or 8? All four possible conventions have been used at one time or another in the computer industry. and the specification needs to be clear which is meant. (e.g. Rijndael fails to be clear on this point)

### 1.3 Errors in Rijndael

The specification of the Rijndael algorithm is in error, not just ambiguous. There are at least two places where the specification is clear and unambiguous, but contradicts the reference implementation and the test vectors. These discrepancies absolutely must be fixed!

#### 1.3.1 ShiftRow

To obtain results that agree with the test vectors, one has to assume that the operation ShiftRow shifts rows in the opposite way from the direction given in its definition in section 4.2.2 of the Rijndael specification.

#### 1.3.2 Rcon

The Rijndael key schedule is defined in terms of a fragment of C code.

I will note in passing that the section 4.4 definition of how the key schedule is to be used makes heavy use of the equivalence of pointers and arrays and the ability to do arithmetic on pointers in C. This fragment of C code indicates that the constant used in the first round will be Rcon[1], which is defined to have value 2. To get the same results as the test vectors, you have to use value 1 (i.e. Rcon[0]) in the first round.

2. Some performance measurements

In my opinion, speed should not be the primary consideration when choosing a cipher to be the Advanced Encryption Standard. It is (usually) more important that a cipher be adequately secure than that it be (say) 10% faster than the alternatives.

However, as the speed of the algorithm will doubtless be taken into consideration when selecting the winner, I present here some performance measurements. These measurements were made with independent implementations of the algorithms in the Ada programming language. This serves two useful purposes:

1. The AES reference implementations are not all optimized to the same degree, so when comparing them it is not clear that we are comparing like with like. The independent implementations I used for these experiments are all written in a similar coding style, with approximately the same amount of time spent optimizing each: this should make the comparisons fairer.

2. If we only look at performance measurements of implementation in C and C-like languages such as Java, there is a danger of introducing a bias in the comparison: favoring algorithms which happen to be easy to express in C but which might not be so easy to express in other languages. These measurements, based on Ada implementations, show how the algorithms fare in a language which is not C.

These measurements were made with the GNU Ada compiler (GNAT) version 3.11p on a 233 MHz Pentium II running Red Hat Linux version 5.2. Optimization level 3 (-O3) was used, and run-time array bounds checking was disabled (switch -gnatp).

The time take to encrypt 1 million blocks (128 million bits) under the same key was measured, and used to produce the following figures in Mbit/s. A 128 bit key was used - in algorithms where the number of rounds depends on the key size, this matters.

| Algorithm | Encrypt Mb/s | Decrypt Mb/s |
|---|---|---|
| RC6 | 39.44 | 42.05 |
| Rijndael | 27.20 | 27.63 |
| CAST-256 | 23.00 | 23.00 |
| Serpent | 16.43 | 12.21 |
| Safer+ | 3.21 | 2.84 |

Conclusions: Of the algorithms measured, RC6 is the fastest but nearly all the others offer comparable performance. Safer+ is an exception, being nearly an order of magnitude slower than the others.

## 3. Implementation on smart-cards and microcontrollers

There is often a need to implement a block cipher on smart cards or other microcontrollers (e.g. credit or ATM cards; infra-red door openers using cryptographic authentication; television set-top boxes implementing pay-per-view schemes). It would desirable if the algorithm chosen to be the Advanced Encryption Standard was suitable for use in these applications.

It is worth noting that the Rijndael algorithm is fast on 32-bit processors as well as 8-bit ones. Rijndael is one of the few AES candidates to have this useful property of being fast on both classes of processor. (Although I have criticized Rijndael on the grounds that its specification is poorly written, the algorithm itself has definite advantages. For this reason I think Rijndael should continue to be considered as an AES candidate in phase 2 of the evaluation).

However, on smart cards and microcontrollers speed is not the primary consideration: memory use is much more important. In a typical smart-card application, the card only encrypts a few blocks of data so bulk encryption speed is almost irrelevant. On the other hand, smart cards usually only have a few K of EPROM and even less RAM, so many of the AES candidates just won't fit into memory. So if want the algorithm to be usable on smart cards, the most important factor is that it fit into a small amount of memory.

It is very difficult to come up with a single algorithm that meets both of these objectives:

(a) Fast bulk encryption on 32 (or 64) bit processors, where the amount of memory available is almost unlimited.
(b) Very low memory usage on 8 bit processors, where speed is not very important.

For this reason, I think it would be entirely reasonable if the AES were to standardize two different cryptographic algorithms, one optimized for speed and one optimized for memory usage.

From: D.A.Crick@exeter.ac.uk
Subject: Round 1 Comments
To: aesfirstround@nist.gov
Date: Wed, 14 Apr 1999 13:30:53 +0100 (BST)
X-Mailer: ELM [version 2.4 PL23]

-----BEGIN PGP SIGNED MESSAGE-----

It would appear that the following SIX candidates should proceed through to Round 2 of AES, due to performance, design and (currently known) cryptographic strength: (alphabetically) E2, MARS, RC6, RIJNDAEL, SERPENT and TWOFISH.

This group has the interesting secondary feature that half of them (RIJNDAEL, SERPENT and TWOFISH) have been placed completely in the public domain.

-----BEGIN PGP SIGNATURE-----
Version: 2.6.3ia
Charset: cp850

iQEVAwUBNxSEa0gtXZoeMUf5AQFQigf+NnwoT8LdS9iFh4doVFsOt9yIRRgYpuPw
4hxWLTxJ3SeOpILx022JBq64D4vFC+OvIt66WRDfm4c62akyYGjwKRy/Pm2WWYmC
133OzkyAWRlZb5oJ7YaoKanRdEU/NUDmkemsRAIsf9c6Rs6GTgb1w8nus8uf70Mp
QzPJvFrXavcWwaMV9iAAMslv3xN6D1Uqi/7i6NyNLlOojgu4xrJXbwSoukYwX27v
xkcXavfEkp0kqGhAg+yfW/LZTszVvUIKpkKD2eIRHCnyDhWdOp9796rvBb9ZRGKy
OGi0QTLCy/TBF8dRJUifp4JwNWuqpu5IcNUtrY1GiJ8iW19aAkT+sw==
=WcyC
-----END PGP SIGNATURE-----

--
**David Crick** : tel. +44 (0)1392 26 4073
Department of Computer Science : PGP RSA verify 0x1E3147F9
School of Engineering and Computer Science : PGP RSA encrypt 0xA78C2D61
**University of Exeter**, Exeter, EX4 4PT (UK) : e-mail D.A.Crick@ex.ac.uk

Date: Wed, 14 Apr 1999 14:04:46 GMT
From: **Dianelos Georgoudis** <dianelos@tecapro.com>
To: AESFirstRound@nist.gov
Subject: proposed metric for comparing ciphers
X-Mailer: **TecApro** - TecaMail version 1.33
Reply-To: dianelos@tecapro.com
X-Return-Receipt-To: dianelos@tecapro.com
X-Confirm-Reading-To: dianelos@tecapro.com

NIST will have to choose between several ciphers for the AES. In this note I propose a procedure that allows the quantitative comparison of the strength of a group of ciphers. The basic idea is to normalize the different ciphers by requiring them to achieve the same very high target speed (which will mean cutting down the number of rounds they perform) and then to compare their relative resistance to known attacks. Since speed is a reasonable measure of a cipher's workload, by normalizing speed one also normalizes a cipher's workload, and one can then compare how well different ciphers fare when they expend comparable amounts of work. In the paper "A Note on Comparing the AES Candidates" presented at the second AES conference, Eli Biham proposed a similar idea.

This procedure requires the definition of a standard hardware platform. I suggest that the standard platform is based on the Pentium II processor. This choice is close to the original NIST Platform (a Pentium processor) and is representative of the processor currently in most common use. The same procedure can be applied to other platforms (based for example on an 8-bit processor, or a 64-bit processor or even on projected architectures). The speed of the processor itself is not important because all results can be expressed in terms of processor cycles.

Next you need to define a standard task. The definition of a task can be based on a mix, in different ratios, of basic activities such as key set-up, block encryption, block decryption etc. using different block sizes. Here, for the sake of simplicity I define the standard task as made up of one hundred encryptions and one hundred decryptions of a 128 bit block using the same key, that is using just two key setup operations. NIST can easily define many other standard tasks and very quickly re-apply the whole procedure described to each of them (this is a purely mathematical process).

The most efficient implementation of each cipher for a given platform is needed in order to measure (or calculate) the number of cycles required to perform the standard task. This implementation will most probably be in assembler. For the Pentium II platform I suggest that all possible coding tricks, as well as

unlimited memory usage, should be allowed in the optimum implementation of the cipher.

Now we define a very high target speed for the performance of the standard task. This target speed should be so high that even the fastest ciphers will have to significantly cut down on the number of rounds performed in order to achieve it. This will push all the competing ciphers down to a level of "insecurity", in the sense that attacks will now be possible at a cost which is lower than exhaustive key search. In this situation I assert that the cipher with the highest cost of attack at equal speed is the most secure cipher.

In fact, one can do even better and set up several target speeds and determine the relative resistance of each of the ciphers for each of these targets. In this way the security "dynamics" of each cipher can be investigated. For example a cipher whose strength is relatively low at very high target speeds but whose resistance grows comparatively very fast at lower target speeds has the most desirable kind of structural strength.

The number of rounds is not a continuous quantity, therefore for each cipher the number N of rounds that pushes the cipher speed over the target speed, and the number N-1 of rounds at which the cipher speed is below the target speed should both be studied. Thus the cost of an attack should be expressed not as a single number but rather as a range.

An even better approach would be to study the cipher dynamics graphically. If the results of the tests are expressed in graphical form then it is not necessary to compare ciphers at exactly the same target speeds. For each cipher the graph will show on x-axis the number of processor cycles used to perform the standard task and on the y-axis the cost of the best attack, and the graph will connect points that correspond to the use of a particular number of rounds. These graphs can be used directly to compare the structural strength of the ciphers.

A difficult point with this procedure is to find a way to define the cost of an attack. Most attacks require a number of chosen or known plaintexts, together with processor resources and memory resources, and may work on all keys or on just a fraction of the keys. I suggest the following simple formula:

cost of attack = ( N^2 + 2^(-30)*E ) / P

Here N is the number of known plaintexts, E is the work necessary for the attack as a multiple of the work required for a single encryption, and P is the proportion of keys where the attack works.

This formula needs some explanation: I suggest that the cost of knowing a large number of plaintexts does not grow linearly, therefore I have squared the number N. I don't take into account the cost of memory usage at all for the simple reason that using very large amounts of memory is not only expensive in hardware costs but also in time. The formula I propose here applies essentially to attacks that do not require the use of huge amounts of memory and that are easy to implement on parallel processors. As a measure of the processor work involved in an attack I use the work necessary to perform one encryption E (using the corresponding number of rounds). It is not clear how the computing cost compares to the cost of known plaintexts - here I assume that one known plaintext costs the same as one billion encryptions (thus the factor 2^(-30)). Similarly, the cost of 2^15 known plaintexts is high but comparable to the cost of 2^60 encryptions. The parameter P takes into account the fact that if an attack works on only a small fraction of the keys it is a correspondingly weaker attack (and therefore is more costly). Finally, if the attack requires chosen plaintexts or ciphertexts I suggest one increases the cost of the first term one hundred fold, i.e. substitute 100*N^2 for N^2 in the formula given.

Most ciphers are basically just a sequence of identical rounds and the method described here can be used to compare the relative strength of such ciphers against known attacks. In this context the MARS cipher is an exception because it mixes different types of rounds. So for any attack against a reduced MARS cipher the strongest MARS variant that achieves the target speed must be chosen.

Even though the proposed method clearly includes a lot of debatable choices (particularly concerning the definition of the "cost" function), I argue that some kind of unified metric must be used to compare the competing ciphers. Consider the alternative: suppose that two candidate ciphers A and B have rounds that are in fact of identical strength. And suppose the only difference is that A's designer went for speed and used relatively few rounds more than the minimum number necessary according to initial analysis, while B's designer went for security and added many "unnecessary" rounds. In the relatively short space of time available for making the comparison and choosing between the candidate ciphers it is unlikely that an effective attack will be found on either the full version of A or B, so therefore it is likely that A will be chosen for being "faster", even though its design is not superior to that of B . Later, if a successful attack is found on A, that takes advantage of the fewer number of rounds performed by A, then it will appear that NIST chose the weaker cipher A instead of the apparently stronger B. Clearly without a consistent metric to guide the comparison, the interplay between strength and speed will greatly complicate the comparison of the competing ciphers.

The procedure described here has an advantage that may not be obvious: the definition of ground rules for the comparison of the ciphers will concentrate cryptanalytic work on the strongest ciphers. Assume for a moment that all ciphers are of similar strength; in this case the cipher that for some reason attracts the least amount cryptanalytic work will appear the strongest because fewer attacks against a reduced version of it will be found. Under the procedure proposed however cryptanalysts will now be motivated to attack this very cipher because it is at "the head of the pack". If it turns out that this cipher is not in reality one of the very best, then some effective attack against a reduced version will be found thus pushing down its position in the comparison graphs. If, on the contrary, this cipher holds its position, even though it has attracted a much larger proportion of cryptanalysis, then it will increase our confidence in its strength. In other words the winner should be the cipher that makes it to the top of the comparison charts and keeps its place there.

I believe that the procedure described here is adequate for comparing the "academic" strength of cipher designs. However, I recognize that many other factors, such as the consistency of speed over different platforms or in different work environments, the availability of cryptanalytic proofs, the implementation flexibility, the simplicity of the design, the maturity of the design, the resistance to unknown attacks, are all valid criteria for comparing the competing ciphers.

Dianelos Georgoudis
email: dianelos@tecapro.com
http://www.tecapro.com

From: **Carlisle Adams** <carlisle.adams@entrust.com>
To: "'AESFirstRound@nist.gov'" <AESFirstRound@nist.gov>
Subject: Official AES Comments...
Date: Wed, 14 Apr 1999 12:13:05 -0400
X-Mailer: Internet Mail Service (5.5.1960.3)

Official AES Comments

I have several comments to contribute to the "official list" for AES.

1) It seems to me that one of the most important things we learned at the Second AES Conference is that invasive and, particularly, non-invasive attacks on smart cards have become quite sophisticated and therefore quite threatening to security. The conclusion drawn by those who seem closest to the details of this area is that *none* of the fifteen AES candidate algorithms will fit in a low-end smart card (constrained by quite small RAM and ROM sizes) if precautions against these attacks are to be implemented. Furthermore, it is not sensible to use a cipher with the strength of the eventual AES algorithm in a smart card without these precautions.

Therefore, the ability to operate in such highly-constrained environments MUST NOT be a selection criterion, or even a factor in the overall ranking, with respect to the AES candidates.

2) After viewing all the numbers presented at the Second AES Conference, it is clear that it is not difficult to choose or manipulate the platform/environment such that the performance of almost any candidate will look attractive. However, the eventual AES algorithm will not be restricted to one particular platform or specific set of conditions; rather, it will need to perform acceptably well on every conceivable platform.

Therefore, to the extent that performance is taken into consideration in selecting candidates for the second round of analysis, those candidates whose performance is reasonable ON EVERY PLATFORM should be given higher weighting than those whose performance varies significantly depending upon the implementation environment.

3) Although there is understandably great desire to select an AES algorithm as quickly as possible, it is almost universally recognized by the cryptologic community that there is very little time to fully evaluate the set of candidates with respect to resistance to cryptanalytic attack. People are busy; fifteen algorithms is a lot to look at; and cryptanalysis is a methodical and detailed art at the best of times. There is simply not enough time to give due consideration to the security aspects of every single detail of every candidate algorithm.

Therefore, the candidates that present the fewest "new" features, structures, concepts, and operations appear to be MORE SUITED to this expedited AES process than the candidates that are entirely new. In particular, where it is possible to study only one or two "new" aspects of a candidate and rely upon previous documented scrutiny of other aspects by the cryptanalytic community, this should be seen as a great help in resolving the conflicting requirements of a speedy process and a thorough review.

4) Not only in the area of performance, but also in the areas of code size, required memory size, implementation complexity (i.e., difficulty), reliance upon "trust" in certain cryptographic structures and operations, and so on, it is clear that different candidates are best suited to different environments. Added to this is the unknown future of Intellectual Property claims that may arise from any quarter. Finally, it must be recognized that regardless of the intensity of the scrutiny of the next year or so, any candidate algorithm may be broken in two years.

Therefore, prudence would suggest that this process result in TWO (or possibly THREE) eventual AES algorithms. One may be more highly recommended than the other(s), so that constrained environments that can only implement a single algorithm will have guidance as to which one to choose, but all other environments should be encouraged to implement all algorithms if possible. In some cases having multiple AES algorithms may hamper interoperability, but the benefits associated with being able to switch quickly and easily to "AES-b", if a problem (including a patent problem) is reported with "AES-a", seem compelling.

---

**Carlisle Adams**
Senior Cryptographer
**Entrust Technologies**

---

X-Sender: jon@mail.pgp.com
X-Mailer: QUALCOMM Windows Eudora Pro Version 3.0.3 (32)
Date: Wed, 14 Apr 1999 15:54:54 -0700
To: AESFirstRound@nist.gov
From: **Jon Callas** <jcallas@nai.com>
Subject: More AES Comments
Cc: "Jon Callas" <jcallas@nai.com>

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Information Technology Laboratory
Attn: AES Candidate Comments, Room 540
National Institute of Standards and Technology
100 Bureau Drive, STOP 8970
Gaithersburg, MD 20899-8970

I am writing to give some more comments about the AES process to supplement my previous comments. I sent my first set of comments before the Second AES conference, and a number of things were discussed in that meeting that I would like to comment on, in no particular order.

Performance: A good deal of time in the second meeting was spent on discussing performance, also not only discussing the performance of the algorithms, but how to measure the performance of the algorithms. While these discussions were going on, though, I did some of my own meta-analysis that I would like to bring up.

As the various performance results were presented, I was struck by the fact that all the performance results had a cluster of algorithms at the top of the performance chart, and then a gap to the performance of the other algorithms. If you recall my previous comments, one of my main points was that presently, we should only look at what group of five (or so) algorithms we select in the first cut; differences among those algorithms are for now irrelevant. We may indeed not yet know how to select the preferred algorithm. More strikingly, the cluster of algorithms was pretty much the same, no matter whose analysis you look at. Here is my chart:

| Evaluator: | NIST | Gladman | Schneier | Biham |
|---|---|---|---|---|
| Alg 1: | Crypton | RC6 | RC6 | Serpent |
| Alg 2: | RC6 | MARS | MARS | MARS |
| Alg 3: | MARS | Rijndael | Twofish | Rijndael |
| Alg 4: | Rijndael | Twofish | Rijndael | Twofish |
| Alg 5: | Twofish | Crypton | Crypton | Crypton |
| Delta 1-5: | 68% | 76% | 83% | 25% |
| Delta 1-6: | 163% | 134% | 154% | 158% |

I have not included Craig Clapp's analysis in my chart, because he limited himself to seven algorithms, but his analysis supports the observation that there is a cluster of fast algorithms, nonetheless.

Furthermore, even Eli Biham's analysis, which re-scaled the algorithms by changing their number of rounds, has essentially the same result. The only change is that he displaces RC6 and replaces it with his own algorithm, Serpent.

As we go further, we are going to have to do more performance testing with different testbeds. The actual performance numbers produced used machines with hardware multiply instructions. Even Craig Clapp's evaluations assumed that a multiply takes three clocks, which may be optimistic. But I'll discuss that more in another comment.

To sum up this observation, let me say that the performance evaluations as a whole jibe with my previous recommendations. Six algorithms represent all the featured algorithms in all tests. I discarded Crypton because of its flaw in key scheduling, and my final five are the ones that remain. I'll point out that Serpent appeared only in Eli Biham's own analysis, but I included it in my selections because I thought its conservative design was a feature. His paper seems to be saying that if speed is important, then a conservative design can be sped up by lowering the number of rounds.

Processor size: There were a number of comments at the conference on processor size, and which processors are important and which will fade away. Predicting the future is always a guessing game, but I'd like to toss in my guesses, along with a comment. My professional background is in systems design, computer operating systems, network systems, and security systems.

I believe that eight-bit processors will not go away. I don't say this out of any love for them, but out of sheer pessimism. The smallest and lightest of the eight-bit processors, such as the 8051, cannot be replaced. As they become cheaper and smaller, then new uses will be found for them. This is inevitable, and not likely to stop. They will have to run the AES in any situation where they will encrypt at all. Using a weaker algorithm on them would weaken the whole security system.

The only chance there is for the eight-bit processors to go away would be for them to be replaced by modern, lightweight designs. The main example of this new design is the ARM family. The ARM is an impressive chip design because its transistor count is not much larger than some eight-bit processors, such as the 6502. It can be used in many systems where cost, power, and size would otherwise dictate an eight-bit machine.

However, all this beside, the important thing to remember is that the best of the AES candidates will run on small processors like the 8051, and consequently, there really isn't a reason to discuss the issue over and over. Just as in the performance considerations above, the most promising AES candidates will work just fine on small machines and in constrained environments.

The only new information that changes some of my own thoughts are some preliminary results that show that there may be a performance penalty to hardware multiplies and rotations. Sleek 32-bit machines like the ARM have rotates, but not multiplies. 64-bit machines often do not have 32-bit rotates. 8-bit machines, of course, have neither. I also must raise an eyebrow at the assumption that integer multiply will be relatively fast. As I mentioned above, the figure of three clocks for a multiply has been used. My experience with processors is such that I believe that to be optimistic. Not out of line, merely optimistic. In many instances, an integer multiply will take more than three times as long as an add.

It is a concern that some of the algorithms that depend on these instructions may perform worse on a 64-bit machine than a 32-bit machine. We expect 8-bit machines to have lower performance, we don't expect it from the larger machines. However, it isn't so much of a concern that I change my previous recommendations. It is, however, something we should examine much more closely in the second round.

Multiple algorithms: There has been more discussion about the possibility of having multiple AES algorithms, and since I was one of the people making that suggestion at the first AES conference, I think I should formally comment. I no longer believe that this is a good idea.

Last summer, when all of the algorithms were new, it seemed that there might be a need for two AES algorithms, a "small" algorithm, and a "large" algorithm. The reason was that with fifteen submissions, with different design goals, and seemingly different requirements -- large systems and small systems. Today, that doesn't seem to be such a big deal.

There are a couple of reasons for this. One is that the systems considerations for a large system are really not very different than those for a small system. A large system may be an electronic commerce system serving many simultaneous transactions, or an encrypting router managing many high-speed connections, but they both share the need to do a lot of work in a constrained environment with many so-called small systems. They are almost certainly are not as constrained as the smallest of the smart cards, but they do not have the large resources available on a desktop system. Secondly, the best-performing cluster of algorithms performs perfectly adequately in both small systems and large ones. In fact, the best algorithms seem to do even better when given more resources, as Craig Clapp's analysis

shows.

Consequently, I believe that the question of multiple algorithms made sense when we started the process, but in the position we are in now, there is no reason to have more than one algorithm. We have more than one algorithm that can meet all of the requirements. That's all we need.

To sum up, let me repeat that at this stage of the process, I think our concern is merely that of winnowing the field down to five (or so) algorithms. The analysis that we've seen so far, I believe, is indicative of what algorithms would be good choices for that five or so. I gave my recommendations in my previous comments. In the meantime, the new information that has come out only strengthens my opinions. I am pleased that what I see of other people's opinions, including the straw poll taken at the end of the last conference, jibes with my opinion. And to finally conclude, thank you very much for the opportunity to help NIST select the AES. It is a pleasure and an honor.

Sincerely,

/s/ Jonathan D. Callas

**Jonathan D. Callas**
CTO, Total Network Security
**Network Associates, Inc.**
3965 Freedom Circle
Santa Clara, CA 95054
jcallas@nai.com, jon@pgp.com
(408) 346-5860

-----BEGIN PGP SIGNATURE-----
Version: PGP 6.5

iQA/AwUBNxUcqX35wubxKSepEQJaYQCeK4qTCJ1okI8XJZSVJOAT2PI/ahEAoP0P
kMWeGuYRfMCsmYIUqdLvwSba
=NasA
-----END PGP SIGNATURE-----

---

Jon Callas jon@pgp.com
CTO, Total Network Security 3965 Freedom Circle
Network Associates, Inc. Santa Clara, CA 95054
(408) 346-5860
Fingerprints: D1EC 3C51 FCB1 67F8 4345 4A04 7DF9 C2E6 F129 27A9 (DSS)
665B 797F 37D1 C240 53AC 6D87 3A60 4628 (RSA)

Date: Thu, 15 Apr 1999 09:42:21 +0200
From: **Fabrice Noilhan** <Fabrice.Noilhan@ens.fr>
To: AESFirstRound@nist.gov
Cc: Serge.Vaudenay@ens.fr
Subject: DFC update
X-Mailer: Mutt 0.93.2

On behalf of the **DFC team**, you'll find enclosed 4 files containing updated implementations of DFC.

A description of those files is provided in 'report' (plain text). RefCode.zip contains the C API of DFC in ZIP format, JavaCode.zip contains the Java API and AuxCode.zip contains several other implementations (mostly in assembly).

I would appreciate that you acknowledge that you have received this email and that the attached files unpack correctly.

Best Regards,

Fabrice Noilhan

[NOTE FROM NIST: Due to export restrictions, NIST cannot make the submitted code available on its web site. However, the “report” file included in this submission is being provided here:]

```
The code of DFC has been changed since the version sent to the NIST
in June 1998. New code was written for the API functions and better
implementations of the inner encryption functions have been written. As
stated by the NIST, we did not expect that the API functions would be
used for timings, so that our code was not much optimized. New code has
been optimized and is still portable from one architecture to another.
The improvements in the encryption functions are due to different ways
of computing ax+b mod p mod 2**64, often dedicated to architectures and
compilers.

The new JAVA version uses 64 bits signed integers; they nonetheless
respect unsigned integers operations in the JAVA standard for most
operations (such as additions and multiplications). This fact, combined
with the new compilers and JIT compilers, gives a huge speed up (by
a factor of 40). The code has been optimized for Sun's UltraSparc
processors but it has a good behaviour on Intel processors. A version
dedicated to Intel processors could be a bit faster.

The C-API has been rewritten so as not to make as many conversions
as in the prior version. Thus, new timings using C API are really
faster. Two versions are provided: the standard ANSI-C uses 32 bits
integers and should work on all processors, regardless on endianess
or size of ints, or alignement requirements. This is the default
```

when building. The second version uses 64 bits integers, may they be provided by the compiler (long long for gcc on Intel processors) or by the processor itself. To build this version, you have to specify the "INT_64" compile-time definition. In addition, if you have a 64 bit processor, then you have to use the "LONG_64" compile-time definition. The code for the inner function is the same in the 32 bits case, and has been modified in the 64 bits case.

We should be cautious when using timings based on C-API versions: several candidates suppose that the processor is little-endian and 32 bits and use casts to convert from an array of bytes to 32 bits integers, which is prohibited on other processors. Using similar tricks, we could have a speedup, but it is not portable and will not produce the correct result on 64 bits processors for instance. So our implementation does not use these non-standard tricks.

Assembly coded functions are also provided (Pentium, Pentium Pro). A C version using one ASM opcode is provided for Alpha processors and a C version using floats is provided for Sparc processors (this version is faster than the version using 64 bits integers). These implementations are noticeably faster on 32 bits processors than C implementations. It is not surprising given the fact that DFC is 64 bits oriented and current
compilers do not optimize computings on 64 bits integers. On 64 bits processors, timings of C code and assembly code are similar.

See details in README files of each directory for implementations.

Timings of implementations provided (timings have been made by direct call to the encryption function. For ANSI C code and C 64 bits code, this is made by the dfc_bench program):

| | Processor | compiler | cycles (all key sizes) (encrypt/decrypt) | key setup | author |
|---|---|---|---|---|---|
| | ANSI C code (32 bits) (see RefCode directory) | | | | |
| 1. | Alpha 21164 | DEC cc | 2562 | 10248 | Pornin |
| 2. | Pentium II | GCC | 2592 | 10368 | Pornin |
| 3. | UltraSparc | Sun CC-5.0 | 4160 | 16640 | Pornin |
| | C code (64 bits) (see RefCode directory) | | | | |
| 4. | Alpha 21164 | DEC cc | 564 | 2256 | Noilhan |
| 5. | Pentium II | GCC | 1262 | 5048 | Noilhan |
| 6. | UltraSparc | Sun CC-5.0 | 875 | 3500 | Noilhan |
| | Other implementations | | | | |
| | Alpha 21164 | DEC cc | 526 | 2104 | Harley |
| | Alpha 21164 | DEC cc | 323 | 1292 | Harley (C code + one Opcode) |
| | Alpha 21164 | GCC | 310 | 1240 | Harley (ASM) |
| | Pentium | NASM | 609 | 2436 | Behr/Harley/McCougan/Mathisen ASM) |
| | Pentium II | NASM | 392 | 1568 | Behr/Harley/McCougan/Mathisen ASM) |
| | UltraSparc | SUN CC-5.0 | 775 | 3100 | Harley (C code using floats) |
| | StrongARM | GCC | 440 | 1760 | Harley/Seal (ASM) |

Compiler flags for RefCode directory:
1. -w0 -arch ev56 -O4 -newc -fast -inline all -tune ev56 -speculate all

2. -Wall -O9 -finline-functions -fomit-frame-pointer -mpentiumpro
3. -fast -xO5 -xtarget=ultra2 -xcache=16/32/1:256/64/1 -xsafe=mem -xarch=v9a
4. -w0 -arch ev56 -O4 -newc -fast -inline all -tune ev56 -speculate all -DINT_64 -DLONG_64
5. -Wall -O9 -finline-functions -fomit-frame-pointer -mpentiumpro -DINT_64
6. -fast -xO5 -xtarget=ultra2 -xcache=16/32/1:256/64/1 -xsafe=mem -xarch=v9a -DINT_64 -DLONG_64

See files in AuxCode directory for other implementations.
Compiler flags are indicated in the header of these files.

X-Authentication-Warning: morille.ens.fr: vaudenay owned process doing -bs
Date: Thu, 15 Apr 1999 16:05:10 +0200 (MET DST)
From: Serge Vaudenay <Serge.Vaudenay@ens.fr>
X-Sender: vaudenay@morille.ens.fr
To: aesfirstround@nist.gov
cc: Jacques.Stern@ens.fr
Subject: another report

# On Decorrelation and Provable Security

**Jacques Stern and Serge Vaudenay**

Recently criticism was addressed to decorrelation theory on which the DFC AES candidate is built, and controversial arguments were raised. Since it seems that some researchers feel skeptical about block ciphers with an additional form of formal security, we wish to clarify these issues in the present formal comment to NIST.

## 1. The Wagner attack

At FSE6 David Wagner (one of the Twofish submitters) presented a clever attack on Coconut98 which was published at Stacs' 98 in order to illustrate decorrelation theory. His attack basically uses correlation between four plaintext/ciphertext pairs, which we believe is the best way for attacking ciphers which have a good decorrelation of order of two (like Coconut98, DFC and others). Intuitively, decorrelation of order two makes using the correlation of two plaintext/ciphertext pairs intractable. Wagner's attack indeed escapes from the model for which security is proven.

Still, if we study the complexity of Wagner's attack, we observe that it is exactly the square of the complexity of a differential attack against the simplified version of Coconut98 with decorrelation omitted. This seems to confirm one of the original motivation of decorrelation theory. This design uses the following paradigm: we take a regular cipher and we plug in, at some strategic points of the computation extra primitives which are called "decorrelation modules" in order to ensure decorrelation. This provides formal protection against attacks which use correlation between two plaintext/ciphertext pairs. We believe that attacks which use correlation between a larger number of pairs are intrinsically harder to handle, and that security against these attacks comes from the basic design and not from the added decorrelation module.

In order to achieve actual security, we need the basic design to be already secure, which was not the case with Coconut98 as shown by Wagner. We believe that on the contrary, it is the case for DFC and

actually we even challenged to break DFC with partially known keys.

2. The Rijmen-Knudsen paper

At FSE6 Vincent Rijmen (one of the Rijndael submitters) and Lars Knudsen (one of the submitters of both Deal and Serpent) presented a somehow controversial paper on decorrelation theory and DFC. Their argument is basically threefold.

- Security against differential cryptanalysis usually relies on the hypothesis of stochastic equivalence which is basically heuristic and this precludes the existence of formal proofs of any kind.

- One can built a cipher which has the same provable security as DFC but which is essentially weak (e.g. a truly linear cipher).

- Decorrelation of order two provides security against an elementary version of differential cryptanalysis which is not relevant.

The first argument refers to the authors previous investigations on differential cryptanalysis. Usually, we need what Xuejia Lai calls the "hypothesis of stochastic equivalence" which basically says that what is true on average over all keys holds for a given key. Decorrelation theory does not assume this hypothesis. Indeed, it shows security "on average" over the distribution of keys. This means that some classes of weak keys may exist (they actually do, as shown by Don Coppersmith), but their fraction must necessarily be small (Coppersmith's weak keys are within a fraction of 1/2**128). This paradigm is quite different from the Rijmen-Knudsen's approach, which is presumably why they overlooked the fact. In addition we announce that, although decorrelation theory investigates averages, new results on deviations will be presented at Eurocrypt' 99.

The second argument denies the original goal of decorrelation theory. The confusion may come from the fact that one of the authors has been part of research with comparable goals: constructing provably secure ciphers against differential cryptanalysis. The Nyberg-Knudsen approach aims at building ciphers with this property by an ad hoc construction. This is not the case for decorrelation theory. Indeed, decorrelation theory strengthens conservative ciphers: protection against attacks which are beyond the scope of decorrelation of order two is provided heuristically as for other "ordinary" ciphers. Picking up dedicated weak constructions is thus irrelevant to our paradigm.

Their third argument still remains. Actually, the best known security result on DFC considers differential attacks in which the input and output differences need to be fixed prior to the attack and cannot be changed in the meantime. Decorrelation theory is still young and wider security results are in progress. For instance at the next Eurocrypt conference security against general iterated attack of a given order will be demonstrated. In particular it is conjectured that

no iterated attack in which each iteration uses one plaintext/ciphertext cipher which is allowed to keep a small amount information between each iteration can work. The famous linear cryptanalysis is an example of such an attack. The security against it is formally proven by a different method at this time. In another (unpublished) result, it is shown that our model extends to adaptive plaintext and ciphertext attacks as well.

Provable security always leads to controversy. Since DFC was built with a regular design and with additional combinatorial properties, we believe that potential users skeptical on the role of provable security can still consider DFC as if it were an ordinary design. At least the design of DFC has its "raison d'etre" and leads to a cipher that has not been challenged so far.

## 3. Conclusion

When we originally submitted DFC, we gave up conservative designs with no formal proof at all and we chose conservative design enhanced with an extra feature supported by some kind of formal proof. Of course, none of these results "proves" the absolute security of DFC. This means that DFC may be subject to attacks that have not been discovered so far, just like any other candidate. Our point is that decorrelation is a new setting for obtaining security which may become useful in the future. Decorrelation is not an exclusive property of DFC. It can also be applied to some other candidates as was announced during the AES2 workshop. For instance we can "plug" decorrelation modules in Cast256 or Mars and obtain a similar formal security proof.

## References:

all new references on decorrelation theory can be found on

http://www.dmi.ens.fr/~vaudenay/decorrelation.html

In particular, there is a new "on-line lecture" on this theory and the latest unpublished papers on.

Date: Thu, 15 Apr 1999 15:16:31 -0400 (EDT)
From: **Stafford Tavares** <stafford@ee.queensu.ca>
To: AESFirstRound@nist.gov
Subject: Comment on the AES Candidate Algorithms
Cc: tavares@ee.queensu.ca
X-Sun-Charset: US-ASCII

To: Information Technology Laboratory
Attn: SAE Candidate Comments, Room 540
NIST

First, let me congratulate NIST on the open and transparent way that they have conducted the process to select the AES algorithm. In addition to the task of selecting a new symmetric cipher standard, NIST has made a valuable contribution to the field of symmetric key cryptography.

A couple of specific comments:

* The time available for cryptanalysis of the 15 candidate algorithms has been short. This has several implications. One of these is that serious flaws could emerge in some algorithms quite late in the process, or even after the selection of the standard(s) is made. NIST should take some steps to protect itself from this risk. One step is to identify a small number of algorithms suitable to replace the chosen one(s) if problems should emerge. In this context, radically new designs probably carry a greater risk.

* From the presentations and discussion at the 2nd AES meeting it seems difficult to judge the relative security of the candidate algorithms in a smart card context. Various physical attacks may overwhelm other security considerations.
There also appears to be confusing claims about the cost and limitations of the next generation of smart cards. Some place severe limitations on memory capacity and others indicate that there is some flexibility in technology and cost. Considering the expected lifespan of the AES standard, NIST should try to project ahead to see where ever-shrinking VLSI fabrication is pointing in this application.

**Stafford Tavares**
Dept. of Electrical and Computer Engineering
**Queen's University**
Kingston, Ontario K7L 3N6
CANADA

From: "Anna Zugaj" <ania@enigma.com.pl>
To: <aesfirstround@nist.gov>
Subject: comments on AES candidates
Date: Thu, 15 Apr 1999 22:13:32 +0200
X-MSMail-Priority: Normal
X-Mailer: Microsoft Internet Mail 4.70.1155

**Karol Gorski**
**ENIGMA SOI** Sp. z o.o.
Warsaw, POLAND

Warsaw, 15th April 1999.

Mr Miles Smid
National Institute of
Standards and Technology
USA

Please find below our comments on a subset of the AES Round 1 candidate algorithms and recommendations for Round 2 algorithm selection. The comments and recommendations are based solely on the results of timing experiments on a Digital Signal Processor and do not take into consideration any other aspects of the algorithms submitted for the AES.

The table below gives the results of experiments with implementations of 13 candidate algorithms. The implementations were written in C by Brian Gladman (http://www.seven77.demon.co.uk/crypto_technology.htm) and compiled for the Texas Instruments TMS320C541 digital signal processor. Implementations of frog and loki97 were not directly transferrable to this platform in the time we had available. Implementations of the other algorithms needed some changes due to the characteristics of the architecture of the processor:

1) Because the smallest addressable part of memory is a 16-bit word (a byte has 16-bits) the upper part of each byte variable had to be cleared (this affected SAFER+ the most);

2) One long word (32 bits) contains two bytes (!) and worse - has to be allocated on an even boundary, so any casting of byte pointers to long pointers (and reverse) does not work.

Please note that no effort was made to optimise any of these implementations for the TMS processor. It is possible that a comparison of optimised implementations might lead to much lower cycle and memory counts as well as a different ranking of the algorithms.

The source code was compiled using the standard C compiler by Texas Instruments Inc (version 1.20) with optimisation turned on. Timings were made using the

TI Debugger v1.30 (used as a profiler) - all timings were obtained on a real processor residing on a TMS320C54x evaluation module, which has a speed of 40 MIPS (millions of instructions per second). The cycle count figures below are arithmetic means of cycle counts from 100 encryptions and decryptions of random blocks using random 128-bit keys and arithmetic means of 100 set key operations on random 128 bit keys. Please note that most of these algorithms have constant encryption time, independent of the input block. Memory usage was taken from the linker output and is expressed in 16-bit words. A part of data memory can be put in ROM - especially constant tables, but for most of the implementations tested this was not a significant part.

| | cycle count | | | | memory usage (in 16-bit words) | | | | | speed/mem |
|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | set_key | encrypt | decrypt | rank | program | data | total | rank | notes | rank |
| cast256 | 60 002 | 10 773 | 10 773 | 8. | 9 201 | 4 311 | 13 512 | 11. | | 9. |
| crypton | 21 416 | 4 155 | 4 155 | 2. | 8 816 | 2 940 | 11 756 | 8. | | 4. |
| deal | 79 709 | 37 991 | 37 979 | 12. | 11 214 | 4 876 | 16 090 | 12. | | 13. |
| dfc | 90 011 | 22 668 | 22 670 | 10. | 1 907 | 440 | 2 347 | 1. | | 5. |
| e2 | 136 499 | 9 652 | 9 652 | 6. | 9 831 | 2 727 | 12 558 | 9. | | 8. |
| hpc | 1 384 217 | 21 759 | 25 640 | 11. | 4 737 | 1 313 | 6 050 | 5. | | 10. |
| magenta | 208 | 52 681 | 52 682 | 13. | 1 424 | 2 368 | 3 792 | 3. | | 12. |
| mars | 54 427 | 8 908 | 8 826 | 5. | 5 663 | 2 258 | 7 921 | 6. | | 7. |
| rc6 | 40 011 | 8 231 | 8 487 | 4. | 3 772 | 104 | 3 876 | 4. | | 1. |
| rijndael | 26 642 | 3 518 | 3 500 | 1. | 7 221 | 9 510 | 16 731 | 13. | | 6. |
| safer+ | 38 598 | 10 288 | 10 799 | 7. | 2 783 | 584 | 3 367 | 2. | | 2. |
| serpent | 28 913 | 14 703 | 16 443 | 9. | 12 373 | 304 | 12 677 | 10. | | 11. |
| twofish | 88 751 | 4 672 | 4 328 | 3. | 5 590 | 4 950 | 10 540 | 7. | | 3. |
| alternative compilations: | | | | | | | | | | |
| magenta | 390 | 53 310 | 53 311 | (13.) | 1 621 | 413 | 2 034 | (1). | (*) | (8). |
| rijndael | 26 223 | 3 897 | 3 935 | (1.) | 7 594 | 5 414 | 13 008 | (11). | (**) | (5). |
| twofish | 141 683 | 73 260 | 73 788 | (13.) | 2 639 | 330 | 2 969 | (2). | (***) | (12). |

Notes:

(*) This is the result of compiling with symbol LARGE_TABLES undefined.
(**) This is the result of compiling with symbol LARGE_TABLES undefined.
(***) This is the result of compiling with symbols: Q_TABLES, M_TABLE, MK_TABLE and ONE_STEP undefined

The cycle count ranking was created using the minimum of the encryption and decryption cycle counts for each algorithm.

The memory usage ranking was created using the total memory usage.

The speed/mem ranking was created using the encryption or decryption speed (whichever was the best for each algorithm) divided by total memory usage.

To convert cycle counts into encryption or decryption speed expressed in bits per second divide 128*4*10^7 by the cycle count.

For example: for the Rijndael algorithm the speed of decryption is approx. 1.46 Mbit/s

The rank numbers in the case of alternative compilations refer to the situation when an algorithm's entry in the first part of the table is replaced by its entry from the second part, with remaining entries unchanged.

==========

On the basis of these timings and memory usage figures we recommend the following 5 algorithms for Round 2:

RC6
Safer+
Twofish
Crypton
Rijndael

The above algorithms achieve the best ratio of encryption or decryption speed to memory usage. We feel that this is a characteristic which is desirable from an efficiency point of view in a standard designed for widespread and general purpose use.

(Please note that Rijndael is included thanks to its performance when compiled with symbol LARGE_TABLES undefined).

The following algorithms are also worth considering due to their low total memory or data memory usage which may be important in some applications:

Serpent
DFC

We hope these results and recommendations will be helpful in selecting algorithms for Round 2. We will be happy to answer any questions concerning the above experiments.

**Karol Gorski and Michal Skalski**

**ENIGMA SOI** Sp. z o.o.
ul. Orla 11/15
00-143 Warsaw, POLAND

telephone: +48 22 620 34 85
fax: +48 22 620 34 85 ext. 25

email:
karol@enigma.com.pl
mskalski@elka.pw.edu.pl

PS. The above table is best viewed on a standard ascii text editor with a fixed width font.

Date: Thu, 15 Apr 1999 19:58:52 -0400 (EDT)
To: AESFirstRound@nist.gov
From: kalmquist@lucent.com (**Kenneth Almquist**)
Subject: Performance of AES candidates on the Alpha 21164 processor

The Alpha 21164 processor is an interesting platform for comparison of AES candidates. It is one of the most advanced microprocessors available today, and thus provides some indication of what midrange processors will look like a few years from now. In particular, it is a 64 bit processor. Since Intel is moving to a 64 bit architecture, it is safe to assume that during most of the lifetime of the AES standard, personal computers will have 64 bit processors. Therefore, comparing the performance of AES candidates on a 64 bit platform such as the Alpha 21164 provides a good indication of how these algorithms will perform on personal computers during the lifetime of the standard.

I have calculated Alpha 21164 encryption and decryption times for twelve of the fifteen AES candidates. I have not calculated the times for the remaining three candidates (Loki, Frog, and Magenta), but have looked at them sufficiently to determine that they will run relatively slowly on the Alpha 21164. The results appear in the table below. All times are given in clock cycles. Decryption times are omitted when they are the same as the encryption times. An "e" after a number means that it is based on an estimate rather than being an exact count of clock cycles.

| algorithm | encrypt | decrypt |
|---|---|---|
| DFC | 307 | - |
| Rijndael/128 | 344 | - |
| Twofish | 372 | - |
| HPC | 382 | 382e |
| Crypton | 412 | - |
| Rijndael/192 | 412 | - |
| RC6 | 469 | 449 |
| Mars | 493 | 455 |
| Rijndael/256 | 480 | - |
| E2 | 489 | - |
| CAST-256 | 604 | - |
| SAFER+/128 | 693 | - |
| Serpent | 887e | 887e |
| SAFER+/192 | 1029 | - |
| SAFER+/256 | 1365 | - |
| DEAL/128 | 2532 | - |

These cycle counts are all based on hand coded assembly language. Readers wishing to check my calculations or learn more about the methodology used to calculate them may read my paper, available from Helger Lipmaa's site <http://home.cyber.ee/helger/aes/>, or from <http://www.exit109.com/~ka/aes-alpha.html>.

From: DJohn37050@aol.com
Date: Thu, 15 Apr 1999 20:15:43 EDT
Subject: AES comment on rounds by Don Johnson
To: aesfirstround@nist.gov
X-Mailer: AOL 4.0 for Windows 95 sub 13
Reply-To: DJohn37050@aol.com

Miles,
There has been some discussion about analyzing AES candidate algorithms based on the notion of setting an "equal security" number of rounds. I find this analysis interesting, but I think it is best if each algorithm designer states the number of rounds to be used in the algorithm, and that this not be allowed to change for the official analysis.

Otherwise it would seem a candidate gets the "best of both worlds", as follows: he/she is able to claim a submission with lots of rounds and high security, but able to claim performance with a reduced number of rounds. This seems unfair and should be prohibited.

I note there is time for a designer to "tweak" his algorithm. I suggest this is the time for an AES candidate algorithm designer to state what he/she believes is the correct version to go into the final competition.

**Don Johnson, Certicom**

X-Originating-IP: [205.211.60.105]
From: "tom plowe" <tomscp@hotmail.com>
To: aesfirstround@nist.gov
Subject: Select RC6 for AES
Date: Wed, 09 Dec 1998 16:04:34 PST

>From a member of the random public. Having looked at all the papers and taking into consideration comments made by experts,
I vote RC6 for the new standard. Speed, simplicity and increased security with the long key size makes this the natural winner.
Rivest deserves recognition for his solid, original, ciphers.

**Tom Cowan**
student

X-Originating-IP: [198.178.8.81]
From: "**Tim Skorick**" <tskorick@hotmail.com>
To: AESFirstRound@nist.gov
Subject: Comments regarding "multiple cipher" AES proposals
Date: Tue, 20 Apr 1999 19:50:32 PDT

To the AES First Round committee,

Please find attached my comments regarding certain proposals for an Advanced Encryption Standard consisting of multiple ciphers.

Any questions may be directed to my email account at tskorick@hotmail.com

Thank you,

Tim Skorick

Introduction

It has been brought up that some in AES selection circles are pondering the notion of selecting multiple ciphers for implementation as an Advanced Encryption Standard. In this little missive I will contend that a) the problems giving rise to this suggestion are hardly problems (or can hardly be effectively dealt with), b) that the means thus far suggested to achieve a multiple algorithmic AES must necessarily weaken and dilute the strength of the AES, and c) there are several very good reason why a multiple cipher AES is a very bad idea.

Some of the more commonly expressed problems with a unique AES (and why they are unjustified fears or fears that can't be adequately dealt with):

1. A single AES selection stands more of a chance of being cryptanalyzed in the future than multiple choices (the "future resiliency" argument).

This is a valid point in certain respects. While I'm not certain how realistic this fear of imminent cryptanalysis really is (based on the effort toward same that has created the strongest candidate ciphers), more importantly, none of the methods for achieving "future resiliency" really accomplish anything save to dilute the "Standard" in AES. I will explain why I believe this later.

2. Along the same lines as (1), several AES winners of the same type would be useless if the type was later found to have a fundamental flaw.

This is just "what-iffing" the situation to death. The example that Don Johnson brings up is Feistel networks. Considering that several of the strongest candidates are based on Feistel networks, the discovery of a fundamental flaw in same would negate all this work. But according to this "what-if" fear, several ciphers might equally probably be found to be based on weaknesses in the distant future. The NIST must choose a cipher based on

what it *currently* understands its strength to be (Feistel networks can be very sound cryptographic foundations, hence their popping up everywhere).

3. Programmers need more ciphers in their crypto toolkit to use for different purposes, including hybrid algorithm solutions. The "one size fits all" may not fit everyone's needs.

This is in no way related to what's going on here. I can't even imagine why this was brought up. The selection of a single AES might promote the use of it exclusively by some, but in no way is this exclusive use mandated. I can't think of a single cryptographic toolkit that only uses DES. Also, a multiple-algorithm AES foists this rather unusual implementation on all programmers who want to use AES. This point is essentially pointless.

4. Malicious cryptanalysts (?) would be less encouraged to attack several algorithms than just one since diffusion of targets makes their cost/benefit ratio less attractive.

Again, I find no basis for this in reality. But more to the point, this suggestion begs the question as to why we are doing this in the first place. The whole point of this process is to choose a standard algorithm which is strong enough to function in this role as "the standard" for a long time and to be widely distributed for use as such. The people who select this cipher are charged with the duty of weeding out those ciphers whose weaknesses are exploitable on a level more efficient than a brute force attack. If they do their job correctly, fear of what cryptanalysts will do is not a factor. This point suggests more a lack of trust in the selection body than anything else.

More succinctly, if NIST decides it has to select a few different algorithms to discourage malicious cryptanalysts rather than just one, replace those in the selection body with people who can discern just one *strong* one.

The commonly expressed ways in which multiple AES selections would solve the above "problems:"

1. Multiple AES selections would promote "future resiliency" by way of selecting cipher finalists based on diversity rather than merit alone. NIST would a) choose only a few ciphers from each type to fill a certain finalist quota and b) select ciphers from each type based on strong performance in a designated function to avoid weeding out already cryptanalyzed algorithms who are strong in the designated areas (?).

This is nothing more than thinly-disguised cryptographic affirmative action. This AES standards committee exists for the purpose of selecting an algorithm based on merit alone. "A" would negate this purpose by forcing the discard of algorithms in favor of others based on their diversity rather than merit. This is fundamentally bad judgement. "B" makes it even worse by tossing aside strong algorithms and allowing *flawed* algorithms to stay in the contest just because they are different from the others, and even then ensure only the possibility of future resiliency through diversity.

Regardless whether or not a candidate cipher performs up to par in a certain area, the responsibility of the selection committee remains to choose the cipher that is strongest in

all areas. The folly of NIST tolerating bad cryptography in its Advanced Encryption Standard is self explanatory no matter what the situation.

2. Multiple AES selections would allow programmers to have more algorithms to choose from in their crypto toolkit.

Again, I doubt that there is even one proficient cryptography programmer who has only DES in his toolkit. The criteria that the algorithm submissions had to meet in order to be selected as candidates was stringent enough that the winner will be selected because it is strongest in all areas. We are looking for an Advanced Encryption Standard, not a list of Advanced Encryption Suggestions.

3. Multiple AES selections would discourage malicious cryptanalysis by creating target diffusion, making it more expensive to build AES cracker machines because they would have to deal with several algorithms.

Any cracking machine designed to brute force an AES algorithm is going to have the same amount of key space to work on despite the algorithm chosen. Again, avoid settling on a weak algorithm, and cryptanalysts can be invited to do their worst.

Some specific arguments against multiple AES selections:

1. The implementation costs. It is doubtful that distributed multiple AES standards will be used as such since they will cost more to implement than is acceptable (esp. considering the improbability of the dangers that the multiplicity is supposed to guard against). The risk factor doesn't seem to call for that kind of expense, although to be fair this judgement is more accurately made by those doing the implementing. While, as I will reiterate soon, programmers are free to use whatever algorithm they see fit, one of the goals of selecting a standard is to encourage its use on a broad level. Making this too costly a task for most programmers will encourage them to rather fall back on ciphers with a less expensive implementation scheme.

2. Implementation complexity. Another self explanatory point. I believe that the complexity of implementing multiple standard algorithms might possibly be an even greater deterrent to their acceptance than cost. While, as I said before, programmers are free to use whatever algorithm they see fit, one of the goals of selecting a standard is to encourage its use on a broad level. Making this too complicated a task for most programmers will encourage them to rather fall back on ciphers with a simpler implementation scheme.

3. The probability of using the less secure cipher. Schneier, in his Rome comments on day two, is quoted by Georgoudis as saying that "if there were two AES ciphers and you used one bit of the key to chose one of them, then half the time you would use a less powerful cipher." Georgoudis, ever the optimist, points out that then half the time the more secure cipher would also be used. I would contend that, in the selection of a standard algorithm, 50% probability of weakening

encryption is unacceptable in the extreme. This is just another reason why the AES will fall into disuse if it foists multiple ciphers (and possibly weak ones) on programmers.

4. Diminished trust in multiple ciphers. When due discussion, analysis and cryptanalytic effort is divided amongst multiple winning ciphers rather than one, confidence will lessen in the ciphers accordingly. One naturally has more confidence in one algorithm unsuccessfully attacked by three cryptanalysts than in three algorithms unsuccessfully attacked by one cryptanalyst each. Again, this is bound to diminish the widespread acceptance and use of the AES.

Conclusion

I understand the nature of the future resiliency problem and how it might apply the selection of the AES. However, I don't believe it is a realistic enough problem to warrant the dilutive and weakening multiple AES measures proposed thus far. Furthermore, I believe that multiple AES selections will increase the complexity and cost of use while diminishing the security and trust in them, short-circuiting the desired width and breadth of potential use.

From: smatyas@us.ibm.com
X-Lotus-FromDomain: IBMUS
To: jfoti@nist.gov
cc: kent@bbn.com, zunic@us.ibm.com
Date: Tue, 27 Apr 1999 11:09:27 -0400
Subject: Re: AES2

Jim,

I received this from Steve Kent on the 14th. Had we received this in time, we would have included his comments in IBM's formal comments under comment #5 (smart card issues) with attribution to Steve. However, I was out of the country from April 11-26, and didn't receive the note until today. Perhaps you would be willing to accept the comments even at this late date.

Regards,
Mike

---------- Forwarded by Stephen Matyas/Poughkeepsie/IBM on 04/27/99 10:54 AM ----------

Stephen Kent <kent@po1.bbn.com> on 04/14/99 01:51:27 PM

To: Stephen Matyas/Poughkeepsie/IBM
cc: kent@bbn.com
Subject: Re: AES2

Mike,

I wil take you up on your offer. Please forward my comments, with attribution, to NIST as part of your formal comments. I am away on business and don't have time to do this myself.

Steve
---

1. One needs to make use of the AES vs. a less secure algorihtm (e.g.,. DES, 3DES, ...) when the data being protected warrant it. Of course one could choose to make use of the AES in all cases, and there would be benefits from use of the same algorithm in a wide range of environments. However, the question raised at the AES2 Workshop is how much weight should be given to the ability to operate in a memory constrained environment of the sort encountered in current day, low end smart cards.

2. These least expensive smart cards, which have stringent ROM and RAM limitations, are not typically used in applications that involve high value data. For example, even smart cards used for credit card transactions with the SET protocol must have the ability to execute RSA. The incremental cost of somewhat more capable smart

cards is small, based on the figures presented in the meeting. If the systems in which smart cards are to be used cannot tolerate an incremental hardware cost estimated at about $1 per card, then I question whether the value of the data (e.g., transactions) being protected by such cards is very high. A widely adopted security "rule of thumb" says that one ought to spend an appropriate amount of money to protect resources, based on the value of the resources. Thus a home safe is less enpensive, and less secure, than a bank vault. An application environment that is not willing to spend a small incremetal amount of money for a more powerful smart card is implicitly stating that the value of the resources being protected does not warrant the additional investment in protection technology.

3. If all else were equal, then the suitability of an AES candidate for low end smart card implementation might be a reasonable "tie breaker" for algorithm selection. But, first, the candidates should be viewed as comparably secure and they should should evidence a comparable capability to provide very high speed encryption in environments that are not encumbered by stringent RAM and ROM limitations. The direction of networking and computing is clearly towards faster, cheaper processors and memory, and increased bandwidth. The challenges for mainstream encryption algorithms in the 21st century will be speed, not the ability to be crammed into constrained computational environments. Thus I argue that a more important ability for an AES candidate is its ability to scale for use in higher performance environments, not memory/processopr-challenged environments.