Reprinted from the

Proceedings of the Ottawa Linux Symposium

June 26th–29th, 2002
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
Stephanie Donovan, *Linux Symposium*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Wild Open Source, Inc.*

The Long Road to the Advanced Encryption Standard

Jean-Luc Cooke
CertainKey Inc.
jlcooke@certainkey.com, http://www.certainkey.com/~jlcooke

Abstract

This paper will start with a brief background of the Advanced Encryption Standard (AES) process, lessons learned from the Data Encryption Standard (DES), other U.S. government cryptographic publications and the fifteen first round candidate algorithms. The focus of the presentation will lie in presenting the general design of the five final candidate algorithms, and the specifics of the AES and how it differs from the Rijndael design. A presentation on the AES modes of operation and Secure Hash Algorithm (SHA) family of algorithms will follow and will include discussion about how it is directly implicated by AES developments.

Intended Audience

This paper was written as a supplement to a presentation at the Ottawa International Linux Symposium. The reader should have at least first year university level knowledge of algebra and physics. Someone with no knowledge of mathematics can still benefit from this paper and its associated presentation. This topic of cryptography is covered lightly. Care is taken to present enough useful technical information to be interesting to a technical audience and beneficial to others.

1 Introduction

Two decades ago the state-of-the-art in the private sector cryptography was—we know now—far behind the public sector. Don Coppersmith's knowledge of the Data Encryption Standard's (DES) resilience to the then unknown Differential Cryptanalysis (DC), the design principles used in the Secure Hash Algorithm (SHA) in Digital Signature Standard (DSS) being case and point[NISTDSS][NISTDES][DC][NISTSHA1].

The selection and design of the DES was shrouded in controversy and suspicion. This very controversy has lead to a fantastic acceleration in private sector cryptographic advancement. So intrigued by the NSA's modifications to the Lucifer algorithm, researchers—academic and industry alike—powerful tools in assessing block cipher strength were developed. Some of these tools proved useful in understanding more about the changes made by the NSA.

Taking an objective look at the standardization practices of the USA NSA and NIST organizations, one can make broad assumptions on where the American state is focusing its cryptanalytic resources.

1.1 What the NSA/NIST has Taught Us

By the mid-1970's the private sector began having an interest in digital cryptography. Even if the clearly false IBM statement "global market for computers estimated at 10" had been proven correct, most of the computers in operation at the time were terminal servers. The terminals connected to these servers were carrying progressively more sensitive data. Financial records, payroll information, trade secrets, and intellectual property; all crucial to the success of a business were exposed to the hot new hobby of wire tapping with gator clips.

Before the US government moved to create a single encryption standard the private sector was taking its first steps into design cryptographic algorithms. In what would become crypto folklore, the NSA quietly send out letters of solicitation to a few hand picked cryptographic experts and laboratories. It is important to realize that previous to this the only communication a mathematician would have with the NSA was in the form of a "cease and desist or be thrown in jail for conspiracy" letters.

Of the few responses received by the NSA, only one had actually met the minimum standards set out by the NSA in their solicitation. The Lucifer block cipher designed by Don Coppersmith, Horst Fiestel and company at IBM was the winner practically by default.

There were two distinct differences between the Lucifer algorithm submitted by IBM and the final DES design.

- **T**he effective key space was reduced by several orders of magnitude.
- **T**he core substitution boxes were redesigned.

These changes were made without comment from IBM or the NSA. Reducing the effective key strength of the algorithm and the ominous change to possibly the single most crucial sub-component of the algorithm had everyone second-guessing the DES.

In the subsequent years after the 1976 DES announcement, Shamir and Biham published their paper on Differential Cryptanalysis (DC) (1994, ISBN-0387979301). In this paper, the two cryptographers of RSA fame ('S' to be precise), outlined how to correlate input changes to the output of several variants of the DES. At then end of the day, the Lucifer algorithm fell to the attack of DC while DES remained unbroken. To the shock of sceptics, the NSA appears to have not weakened the DES for their evil purposes but in fact made it impervious to an attack not to be publicized for another eighteen years.

After the announcement of DC, the Lucifer co-designer Don Coppersmith confessed to knowledge of DC at the time of DES standardization. This kept the sky from falling on the heads of the crypto sceptics as you can well imagine.

2 Obsolescence of the DES

A 56bit key space did not provide sufficient protection in lieu of the personal computer explosion of the late 1980's and 1990's. The threat of attack was no longer from a single powerful computer, but from thousands of commodity computers coordinating their efforts. In comes the Triple-DES. Encrypting the data with three distinct keys resulted in a three-fold increase in key material, a three-fold increase in computational effort required for encryption, and a 5.2×10^{33} increase in the key

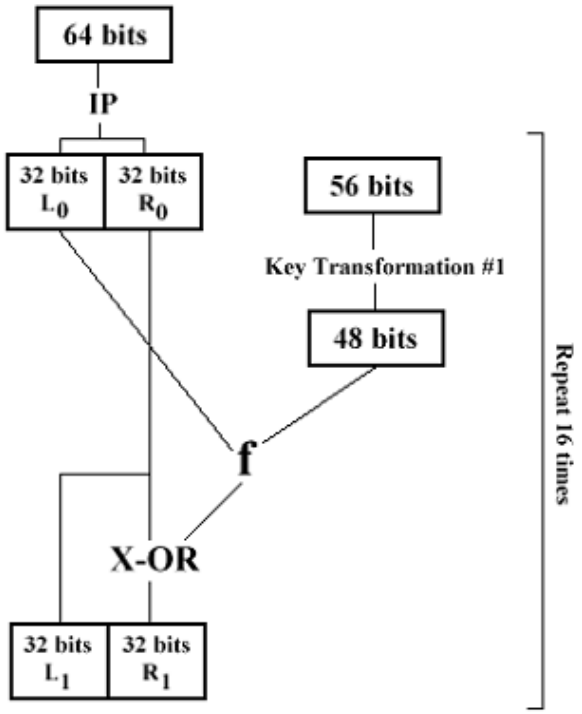

Figure 1: The DES Cipher

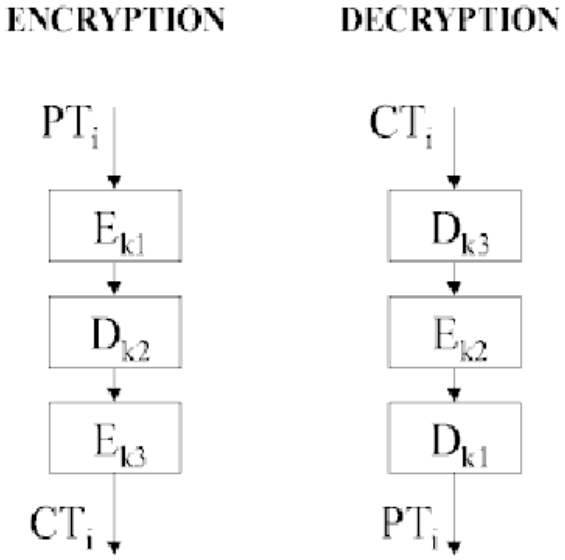

Figure 2: The 3DES Cipher

space.

$$\begin{aligned} E &= 3DES_{k1,k2,k3}(D) \\ E &= DES_{k1}(DES_{k2}^{-1}(DES_{k3}(D))) \end{aligned} \quad (1)$$

By the mid 1990's, TripleDES was no longer sufficient. The security of the algorithm was assumed to be good, but there were other shortcomings with TripleDES.

TripleDES was too slow. The private sector's obsession with higher digital communication bandwidths had made the integration of encryption at the data link layer far too costly. Economic conditions then predicted that all data would be encrypted at least once by 2006. Current economic conditions make this prediction conservative.

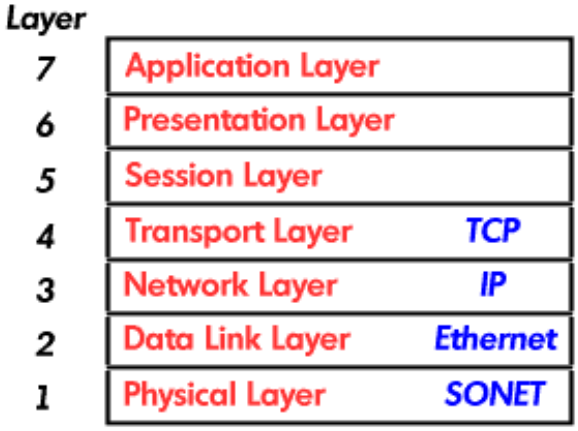

Figure 3: The OSI Network Stack

Private sector dependence on secure data communication was going to continue to grow beyond the capabilities of DES/TripleDES. A new standard with greater security and a long lifetime needed to be decided to mitigate the impact of migrating to a new standard. “We need to act fast before we’re in serious trouble.”

Another issue with DES and subsequently TripleDES, was a design limitation. DES was not designed to be efficient in software. The ubiquity of software in modern computer and communication technology dictated an efficient hardware as well as software implementation for this new standard.

3 AES Round One

Contrasting the algorithm solicitation process used in selecting the DES, a very public announcement was made by the NSA/NIST for cipher designs. Appearing at private sector security and cryptography tradeshows, it was made very clear this time the standard was going to be a very public affair.

The NSA/NIST set out minimum requirements for block cipher submission[NISTAESWWW]

$$AESCD1$$

$$AESCD2$$

$$AESCD3$$

.

- **S**ize efficiency of hardware/software implementation
- **S**peed efficiency of hardware/software implementation
- **M**inimum 128bit block sizes
- **K**ey sizes up to 256 bits
- **R**esilience to all known modern attacks.

Fifteen algorithms met these requirements.

- **C**AST-256 - Entrust Technologies Ltd.
  
  http://www.entrust.com/

  $$AESCD1$$

  $$AESCD2$$

  - **T**he Communications Security Establishment (CSE)—Canadian equivalent to the US NSA—has adopted the CAST5 algorithm as their confidential government data encryption algorithm.
    CSE website: http://www.cse.dnd.ca/
  - **C**AST5 is synonymous with CAST-128
  - **C**AST-256 is an extension to CAST5 for inclusion to the AES
- **C**rypton - Future Systems
  
  http://www.future.co.kr/

  $$AESCD1$$

  $$AESCD2$$

– **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **D**EAL - Outerbridge, Knudsen

http://www.ii.uib.no/~larsr/newblock.html

$AESCD1$

$AESCD2$

– **T**his is one of two submission co-authored by Knudsen. See also Serpent!

– **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **D**FC - Centre National pour la Recherche Scientifique - Ecole Normale Superieure

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

$AESCD1$

$AESCD2$

– **V**ive la resistance!

– **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **E**2 - Nippon Telegraph and Telephone

http://info.isl.ntt.co.jp/e2/

$AESCD1$

$AESCD2$

– **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **F**ROG - TecApro

http://www.tecapro.com/aesfrog.htm

$AESCD1$

$AESCD2$

– **G**eorgoudis is an amateur cryptographer, the only one in Puerto-Rico in all likelihood! FROG was the first cipher he ever designed, and it was accepted to the AES process!

– **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here? Or is Puerto-Rico's special status with the US exempt them for this? Isn't it nice we live in the "free world" and don't have to worry about such ugliness?

- **H**PC - Schroeppel

http://www.cs.arizona.edu/~rcs/hpc/

$AESCD1$

$AESCD2$

– **A**n academic's block cipher. Schroeppel's paper goes into great

detail on the theoretical advantages of his Hasty Pudding Cipher. Not a very practical cipher, underlines the 'openness' of the AES submission process.

  - **B**onus question: Hasty Pudding and Harvard University - what's the connection?

- **L**OKI97 - Brown, Pieprzyk, Beberry

http://www.unsw.adfa.edu.au/~lpb/research/loki97/

$$AESCD1$$

$$AESCD2$$

  - **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **M**AGENTA - Deutsche Telekom

no url available

$$AESCD1$$

$$AESCD2$$

  - **A**uthor unknown ...and for good reason! At the first AES conference, the presenter from DT had nightmare of nightmares happen. The MAGENTA cipher was cracked in real-time! Discussions in the audience between Biham and others led to a mountable attack before the presentation was even over! A paper was written and published within twenty-four hours. And the kicker of it all was there were rumours that MAGENTA had been used in production DT equipment for years but the algorithm was never published. Chalk one up to security though non-obscurity.
  - **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **M**ARS - IBM

http://www.research.ibm.com/security/mars.html

$$AESCD1$$

$$AESCD2$$

  - **T**he Lucifer/DES design team (most of it) returns. A lot was expected from this team.

- **R**C6 - RSA Laboratories

http://www.rsasecurity.com/rsalabs/aes/

$$AESCD1$$

$$AESCD2$$

  - **R**C6, based on RC5, based on RC4. Principle designer Ron Rivest, the R in RSA, the man behind MD1/2/3/4/5, RC1/2/3/4/5/6 and many other publications. If there ever was a crypto rock star, this is he.

- **R**ijndael - Daeman, Rijman

http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

$AESCD1$

$AESCD2$

  - **T**wo Flemish Belgians (as apposed to the French Belgians) designed Rijndael. Cinderella story: no big names, modest track record, and European nationalities.
  - **A**t the time of AES, this algorithm submission would be allowed to ENTER the US, but not leave. Is something wrong here?

- **S**afer+ - Cylink Corporation

mailto:williams.chuck@cylink.com

$AESCD1$

$AESCD2$

  - **B**ased on the Safer cipher.

- **S**erpent - Anderson, Biham, Knudsen

http://www.cl.cam.ac.uk/~rja14/serpent.html

$AESCD1$

$AESCD2$

  - **S**trong cipher, big name authors. Biham co-authored the famous paper on Differential Cryptanalysis. This is one of two ciphers Knudsen co-authored in the AES—see DEAL.

- **T**woFish - Counterpane (Schneier, Kelsey, Whiting, Wagner, Hall, Ferguson)

http://www.counterpane.com/twofish.html

$AESCD1$

$AESCD2$

  - **B**ased loosely on BlowFish. B. Schneier we know from his seminal introductory work on cryptography "Applied Cryptography" and his authoring of the most widely analyzed private sector cipher BlowFish.

4 AES Round Two

The finalists are:

- **M**ARS

  - **S**tands for: Multiplication Addition Rotation Subtraction. These are the primitive operations used by the cipher.
  - **N**o surprise the cipher made it this far. Don Coppersmith and team have the longest track record and the distinction of designing the DES.

- **R**C6


$AESCD3$

  - **N**o surprise here.
  - **S**tands for: Ron's Cipher number 6. Ron Rivest has written many ciphers the entire world uses daily. Remember Distributed.net had a distributed effort to crack RC5? Well RC6 makes RC5 look easy to crack, and difficult to implement.
  - **T**he frightening simplicity of the RC6 encryption operation can be summed up in 10 lines of ANSI-C code for a 32-bit computer:

```
rc6_encrypt() {
 {A,B,C,D} = plaintext
 B=B + S[0];
 D=D + S[0];
 for (i=0; i<r; i++) {
   t=ROL(B * (2*B + 1), 5);
   u=ROL(D * (2*D + 1), 5);
   A=ROL(A^t, u) + S[2*i  ];
   C=ROL(C^u, t) + S[2*i+1];
   {A,B,C,D}={B,C,D,A};
 }
 A=A + S[2*r + 2];
 C=C + S[2*r + 3];
}
```

And the decryption operation:

```
rc6_decrypt() {
 {A,B,C,D} = ciphertext
 C=C - S[2*r + 3];
 A=A - S[2*r + 2];
 for (i=r; 0<=i; i--) {
   {A,B,C,D} = {D,A,B,C};
   u=ROL(D * (2*D + 1), 5);
   t=ROL(B * (2*B + 1), 5);
   C=ROR(C - S[2*r +1],t)^u;
   A=ROR(A - S[2*r   ],u)^t;
 }
 D=D - S[1];
 B=B - S[0];
}
```

  - **Now** don't go off and use this. This cipher is trademarked and patented! The AES process demanded the winning cipher be unencumbered by intellectual property restrictions in all world markets (US export laws don't count?). RSA Labs explained in their submission that if and only if RC6 were to be selected as the AES would they wave royalties, otherwise they only allow use of RC6 for research and educational purposes.

- **R**ijndael


$AESCD3$

  - **T**he cipher name is a play on words and the author's names. If you're Flemish I'd like to hear the explanation. Many people can't pronounce the cipher properly and are happy they won so they can just call it "AES." A Canadian wrote to the authors early in the AES process and suggested renaming the cipher to "Bob."
  - **U**nlike RC6 Rijndael was developed in academia. There were never any IP restrictions.

- **S**erpent


$AESCD3$

  - **N**o surprise here.

– **T**he Linux encrypted file system loop back device had jumped the gun and chose Serpent as the cipher of choice. Newer versions of the encrypted file system support the Rijndael cipher.

- **T**woFish

$AESCD3$

– **N**o surprise here.

– **T**hose who know Bruce know he doesn't have a good chance of winning a seat on the United Nations if he ever chose to run. But still, cryptographers respect his abilities more than his tact and Two Fish made it this far on its own merits.

All five finalist algorithms were of excellent design. At a high level, they were all very similar.

- **E**mployed a strong key expansion algorithm
- **P**re- and post-whitening to protect the inner cipher rounds from "unfolding"
- **J**udicial combinations of linear and differential operations to thwart any differential or linier cryptanalysis
- **C**onstructed from sound mathematical principles

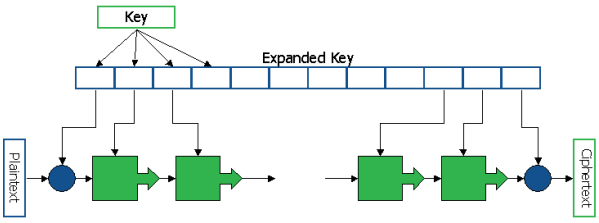

Figure 4: High level design of all AES finalists

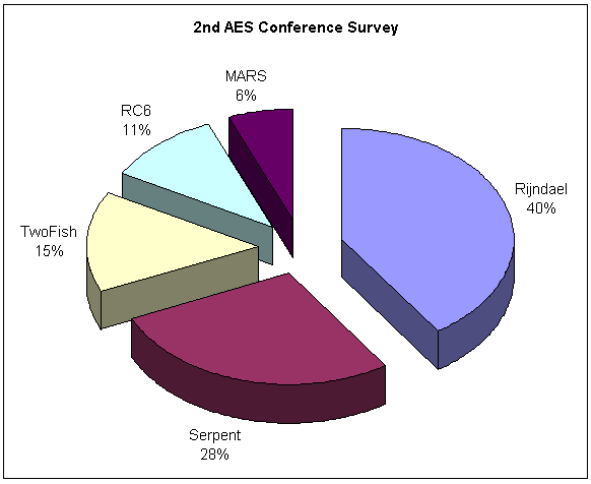

Figure 5: 2^{nd} AES Conference Survey

5 The Winner - Rijndael

A Flemish cipher chosen to be an American standard, what is the world coming to? After the last AES conference where the five finalists presented their closing comments, the NSA/NIST distributed a questionnaire:

- ‘‘If only one algorithm were to be selected as the AES, which should it be?"
- ‘‘If a back-up algorithm were to be selected, which should it be?"

Results from question one showed a clear preference for Rijndael, Serpent coming in a distant second, and MARS, RC6 and TwoFish accumulating few votes combined than Serpent.

The four finalists were not as favoured to become the AES for several reasons.

- **R**C6's simplicity in software came at an unacceptable cost to hardware. In smartcards, efficient 32bit multiplication consumes far too much surface area.

- **M**ARS was a strong cipher, with a very complex structure that was not conducive to straightforward analysis. Like RC6, it also used 32bit multiplies.

- **S**erpent's popularity was justified, their design used the DES s-boxes so hotly contested for the past two decades. These S-boxes have been so heavily analyzed (significantly by one of the Serpent authors) that it would simply be unwise to create a whole new net of S-boxes. And for you Ditributed.Net people, the cracking effort which brute forced a DES key in 22 hours used an optimization technique called "bit slicing." This technique performed 32 parallel 4x4 DES S-box substitutions in only a few instructions. Reducing each S-Box to a Karnough map of bit-wise operators and performing transformations on four 32bit words is how the optimization was accomplished. Effectively transforming a 32bit Single Instruction Single Data (SISD) processor into a Single Instruction Multiple Data (SIMD) processor. Serpent used this "bit slicing" technique on its 128bit (4 x 32bit) blocks. They overcame the speed limitation of DES by using the AES block size requirement and a modern optimization technique. Quite clever!

- **T**woFish's limitations lay in the extra overhead required for full speed optimization. A key and block dependent set of lookup tables are created at the start of the encryption/decryption operation.

Rijndael's design was very tight. Not as simple to implement in software as RC6, but the overall simplicity of its sub-components made it the clear favourite. Hardware implementations could be made so small, that two parallel implementations of the Rijndael algorithm can fit on a single 8-bit smart card!

5.1 One Plus One Equals Zero

The cipher achieves its small footprint and simplicity from its use of Galois Field (GF) theory. Also known as "primitive polynomials" or linier feedback shift registers (LFSR), arithmetic in GF requires a minimum of hardware resources—the principal motivation for their use in cell phone and network telecommunications.

To understand how GF works, start by forgetting everything you learned in grade school—for many of us this is easily done. Next, understand that arithmetic in GF is undefined unless you specify the field. In Rijndael this field is called $GF(2^8)$. Meaning, there are 256 possible values, each value represented by 8 bits, no 7 or 9 bit values exist in $GF(2^8)$.

There are several ways of representing values in GF below we demonstrate a few:

$$\begin{aligned} 01010101 &= '55' \\ &= x^6 + x^4 + x^2 + 1 \\ 10101010 &= 'AA" \\ &= x^7 + x^5 + x^3 + x \\ 11110001 &= 'F1' \\ &= x^7 + x^6 + x^4 + x^5 + 1 \end{aligned} \tag{2}$$

We define the addition operation. In $GF(2^8)$ this is what we commonly know as the

explosive-or (XOR) operation.

$$\begin{array}{rcrcr} C & = & A & + & B \\ 00000000 & = & 00000001 & + & 00000001 \\ 00010001 & = & 00010000 & + & 00000001 \end{array} \quad (3)$$

Notice addition and subtraction are identical in this "new math."

$$\begin{array}{ccccccc} 1 & + & 1 & - & 1 & = & 1 \\ 1 & XOR & 1 & XOR & 1 & = & 1 \end{array} \quad (4)$$

Next, we define the "multiply by x" operation. This is where things get a bit strange. To keep closure in $GF(2^8)$ we need to define a primitive polynomial (analogous to prime numbers in an integer field) to "divide" our result by to extract our "remainder."

Multiplying a $GF(2^8)$ polynomial by x is as simple as sifting the bits to the left by one.

$$\begin{array}{rcrcl} C & = & A & \bullet & x \\ x^3+x^2+x & = & x^2+x+1 & \bullet & x \\ 00001110 & = & 00000111 & \bullet & 00000010 \\ '0D' & = & '07' & \bullet & '02' \end{array} \quad (5)$$

In the case where A's most significant bit (x^7) is high, we immediately know the value will no longer be in $GF(2^8)$. We perform a modulo operation with our primitive polynomial on this new C value to return it to $GF(2^8)$.

$$\begin{array}{rcl} P & = & x^8+x^4+x^3+x+1 \\ & = & 100011011 \\ & = & '11B' \end{array} \quad (6)$$

Notice that '00' will always map itself back to '00' and only after 255 multiplications by '02' will a value return to its starting value. (Equations 6 and 7.)

Using Knuth's binary exponentiation technique where successive squaring of B are used to calculate a^b in $log_2(b)$ loops.

```
Knuth_modExp(a,b) {
  rslt = 1;
  while (b != 0) {
    if (b & 1) rslt = rslt * a;
    a = a * a;
  }
  return rslt;
}
```

Using this same technique of binary $GF(2^8)$ multiplication where successive multiplications by x are used to calculate $a \bullet b$.

```
xtime(x) {
  if (x & 0x80)
    return (x << 1) ^ 0x1b;
  else
    return (x << 1);
}
gf8_mult(a,b) {
  rslt = 1;
  while (b != 0) {
    if (b & 1) rslt = rslt ^ a;
    a = xtime(a);
  }
  return rslt;
}
```

You now know how to do math in the crazy world of Galois Fields.

5.2 Inside Rijndael

The Rijndael algorithm's round function consists of 4 layers:

$$
\begin{aligned}
C &= A \bullet x \mod P \\
&= 10000001 \bullet x \mod P \\
&= '81' \bullet '02' \mod '11B' \\
&= 100000010 \mod 100011011 \\
&= 100000010 - 100011011 \\
&= 100000010 \; XOR \; 100011011 \\
&= 00011001
\end{aligned} \tag{7}
$$

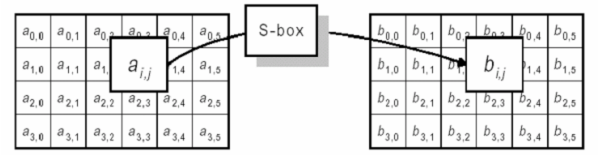

Figure 6: The Rijndael ByteSub Layer

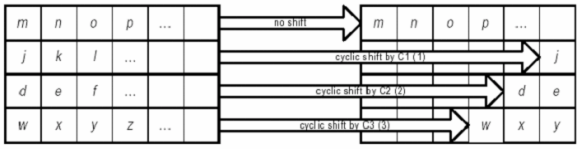

Figure 7: The Rijndael ShiftRow Layer

- **B**yteSub(data)
- **S**hiftRow(data)
- **M**ixColumn(data)
- **B**lendKey(data,exp)

5.2.1 ByteSub

This operation performs a byte level substitution. Unlike DES, this substitution is based on a single bijective transformation defined below. See Figure 6 and Equation 8.

5.2.2 ShiftRow

This layer does not alter the value of the data, it simply moves it about in preparation for later encryption rounds. It is required to have every bit of input effect every bit of output. See Figure 7.

5.2.3 MixColumn

This operation is a bit more complex. Defining a column as a polynomial of $GF(2^8)$ coefficients, a cross product of this value by a constant polynomial co-prime to $x^4 + 1$ is performed. What did I mean by that?

A column of Rijndael data contains 4 bytes. Each byte represents a polynomial in $GF(2^8)$. The column however represents a polynomial of polynomials. When two numbers are co-prime, they do not share any factors other than 1, this applies to all number fields, not just integers. The requirement of co-primality to x^4+1 is required to make this transformation invertible. Invertible operations are nice to have if you ever want to recover the data you're encrypting! See Figure 8 and Equations 9, 10, 11, 12, 13, and 14.

$$
c(x) = '03'x^3 +' 01'x^2 +' 01'x^1 +' 02'x^0 \tag{9}
$$

$$
b(x) = c(x) \otimes a(x) \tag{10}
$$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (8)$$

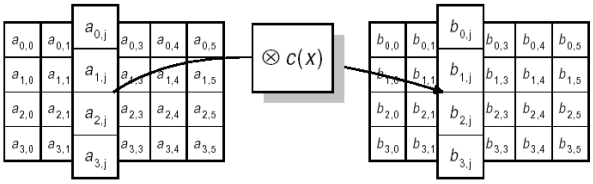

Figure 8: The Rijndael MixColumn Layer

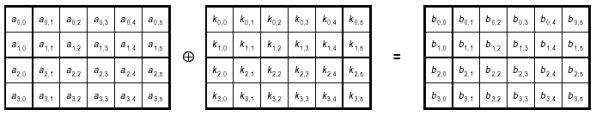

Figure 9: The Rijndael BlendKey Layer

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (11)$$

$$d(x) = {}'0B'x^3 +' 0D'x^2 +' 09'x^1 +' 0E'x^0 \quad (12)$$

$$b(x) = d(x) \otimes a(x) \quad (13)$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0E & 0D & 0B & 09 \\ 09 & 0E & 0D & 0B \\ 0B & 09 & 0E & 0D \\ 0D & 0B & 09 & 0E \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (14)$$

5.2.4 BlendKey

An XOR operation with the expanded key is performed on each byte in the cipher's block of data. See Figure 9 and Equation 15.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} \oplus \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (15)$$

5.3 The Sub-Round Function

These operations are combined to create the Rijndael sub-round transformation (see Equation 16). This sub-round operation is performed anywhere from four to eight times each round depending on the block size specified. The Round function is performed any where from 10 to 14 times depending on the key and block sizes specified.

5.4 Implementation

Efficient implementation of the Rijndael algorithm lies in efficient implementation of the Rijndael sub-round transformation. The sub-round transformation has several layers. The ByteSub layer is best implemented as a lookup

$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S\left[a_{0,j}\right] \\ S\left[a_{1,j-C1}\right] \\ S\left[a_{2,j-C2}\right] \\ S\left[a_{3,j-C3}\right] \end{bmatrix} \oplus \begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} \tag{16}$$

table, the ShiftRow layer by cyclic array offsets and the BlendKey layer is a trivial matter.

The MixColumn operation requires a bit more thought. The following code segment demonstrates how to mix a single column. This operation would be performed several times though a single sub-round which is itself executed several times in a single round, which in turn is executed several times in a single block encryption operation. This procedure will net you a nice (!) $O(n^3)$ time complexity if implemented in serial!

```
void MixOneColumn(char a[4]) {
  char Tmp, T;
  Tmp = a[0] ^ a[1] ^ a[2] ^ a[3];
  T = a[0] ^ a[1]; T = xtime(T);
    a[0] ^= T ^ Tmp;
  T = a[1] ^ a[2]; T = xtime(T);
    a[1] ^= T ^ Tmp;
  T = a[2] ^ a[3]; T = xtime(T);
    a[2] ^= T ^ Tmp;
  T = a[3] ^ a[0]; T = xtime(T);
    a[3] ^= T ^ Tmp;
}
```

There is room for a significant increase in speed if the implementer is willing to sacrifice size. Collapsing the ByteSub and MixColumn operations into a single 8x32 lookup table and a little coaxing of the equations, the entire sub-round transformation can be reduced to four lookups, four 32bit XORs, and three cyclic bit rotations by 8.

$$T_0\left[v\right] = \begin{bmatrix} S\left[v\right] \bullet' 02' \\ S\left[v\right] \\ S\left[v\right] \\ S\left[v\right] \bullet' 03' \end{bmatrix} \tag{17}$$

$$\begin{aligned} b_j = \quad & k_j \oplus T_0[a_{0,j}] \oplus \\ & ROR_8(T_0[a_{1,j-C1}] \oplus \\ & ROR_8(T_0[a_{2,j-C2}] \oplus \\ & ROR_8(T_0[a_{3,j-C3}]))) \end{aligned} \tag{18}$$

At a cost of 4 kilobytes of lookup tables, this can be further reduced to four lookups and four 32bit XORs.

$$\begin{aligned} b_j = \quad & k_j \oplus T_0[a_{0,j}] \oplus T_1[a_{1,j-C1}] \\ & \oplus T_2[a_{2,j-C2}] \oplus T_3[a_{3,j-C3}] \end{aligned} \tag{19}$$

At first glance, this seems far too simple a transformation to be secure. One is left searching for a backdoor but can find no place to hide it.

5.5 What did/can the NSA do?

Unlike the DES, the AES publication has no changes to the Rijndael algorithm other than limiting the block size from any one of 128, 192 or 256bits to 128bits. I have written another paper suggesting that block sizes and key sizes be identical or the possibility of multiple keys mapping one plaintext input to another ciphertext output would exist. Don Coppersmith took the time to explain there are circumstances where having a larger key to block size would be preferable.

The question remains, did the NSA/NIST do their job? Was the AES a success? The NSA's responsibility to monitor domestic and international communication would naturally lead to developing a backdoor into such a widely used algorithm.

However, the NSA also has the responsibility of protecting American interests for the public as well as private sectors. A strong algorithm would be conducive to one goal but not the other.

The winning block cipher design was not of American origin. It is strange how an American standard came from Belgium. Still, it would appear as though the best cipher won the day. Rijndael is fast (30Mbit/sec ANSI-C, P3 450), simple (see Equation 19), and efficient in hardware as well as software, and widely considered to be secure.

How are these the contradictions in the responsibilities of the NSA reconciled by the AES standard? The answer is simple; an impervious block cipher is insufficient to insure the total security of transmitted data.

There are three broad classes of digital encryption algorithms:

- **M**essage digest or hash algorithms such as MD5 and SHA-1 operate without the use of a key. These algorithms are 'blenders,' they reduce input data to a fixed length binary sequence. These sequences are constructed such that any minute change to the input significantly changes the output.
- **B**lock ciphers such as TripleDES and AES operate with a single key. With this key, data is encrypted in such a way that it can only be decrypted with the same key.
- **P**ublic-key algorithms operate using two distinct but mathematically related keys. An operation performed by one key can only be undone by its complement. This facilitates the confidential exchange of small pieces of data and the authentication of data origin.

The laws of thermodynamics state the energy in a closed system remains constant. The connection of matter, energy, information and entropy are well understood. Extending these laws to the realm of system-level security as applied by cryptography one can come to three conclusions:

- **M**essage digest algorithms The laws permits the existence of a perfect message digest algorithm. A hash algorithm with no mountable attack other then brute force; in this case the parallel collision attack.
- **B**lock ciphers The laws permits the existence of a perfect block cipher. A cipher with no mountable attach other than brute force.
- **P**ublic-key algorithms The laws do not permit the existence of a perfect public-key algorithm. Looking purely at the flow of information required to conserve confidentiality of key pieces of data, the restrictions the law places on reality forbids such an algorithm to exist.

To demonstrate this we consider two scenarios, confidential exchange with a block cipher/shared secret and with a public-key.

- **B**lock Cipher
  Consider two closed systems A and B communicating over channel C using information k only known to A and B.

- Assume a symmetric algorithm $c = E_k(d)$ exists with no possible cryptanalytic attack. That is, d cannot be recovered from c unless k is known.
- System A opens and communicates $c = E_k(d)$ with $n = H(c)$ bits of information though C to B.
- System C cannot recover d since only c is known.
- System B can recover d since c and k are known.
- B now obtains information d possessed by A and not C.
- This information or entropy was communicated to B though channel C using entropy held only by A and B.

We have not contradicted ourselves, thus we cannot disprove the existence of a perfect block cipher.

- **P**ublic-Key
  Now consider our two closed systems A and B with channel C communicating with no shared secret.
  - Assume a perfect public-key algorithm $c = P_{pub}(d), d = P_{pri}(c)$. That is, d cannot be recovered from c unless pri is known.
  - System A opens and communicates $c = P_{pub}(d)$ with $n = H(c)$ bits of information though C to B.
  - System C cannot recover d since only c and pub are known.
  - System B can recover d since c and pri are known.
  - B now obtains information d possessed by A and not C.
  - This information or entropy was communicated to B though channel C using entropy common to all A, B and C.

Here we have our contradiction, A gave B more entropy than was ever transmitted though C and possessed by A. So this tells us at least one of our assumptions was flawed, either no entropy was transmitted or there can be no perfect public-key algorithm.

What does this mean for RSA, ElGamal, Diffie-Hellman, Elliptic curve systems, and other public-key algorithms? The strength of public-key algorithms stem from our ignorance of their underlying mathematics. Any amateur cryptographer could have told you that, this was just a loose formalization proving it.

Will quantum cryptography come to the rescue? Simply replacing our ignorance of mathematics with our ignorance of physics is not a lasting solution. The Standard Model of subatomic particles explains the communication of force (Gravity, Electro-weak and Strong nuclear forces) as an exchange of virtual particles.

Fixating a quantum-coupled photon will cause its complement to fixate in the opposite spin. The communication between these two photons is suspected to be in the form of some sort of virtual particle. The key in usurping the information between these two parties would lie in detecting the energy state of the virtual particles exchange between the two coupled photons.

It doesn't matter how many ways you skin Schrödinger's Cat. At the end of the day even his feline must obey the laws of thermodynamics.

6 Modes of Operation

Equally important to the good design of a block cipher is how it is used to encrypt data. In 1980 the NSA/NIST published a set of standard modes of operation for the DES.

The publication detailed 4 modes of operation. There are three characteristics that differ from each mode: the primitive data unit, the property of memory and the property of state. S e modes operate at the bit level, others at the block level. Some modes operate with a memory of all data previously encrypted and others are memoryless. Some modes operate with a state variable, which is altered after each encryption operation while others are completely stateless.

- **E**ncrypted Cipher Block (ECB)
  - **D**ata unit: block.
  - **M**emoryless: yes.
  - **S**tateless: yes.
- **C**ipher Block Chaining (CBC)
  - **D**ata unit: block.
  - **M**emoryless: no.
  - **S**tateless: no.
- **O**utput Feed Back (OFB)
  - **D**ata unit: bit.
  - **M**emoryless: yes.
  - **S**tateless: yes.
- **C**ipher Feed Back (CFB)
  - **D**ata unit: bit.
  - **M**emoryless: no.
  - **S**tateless: no.

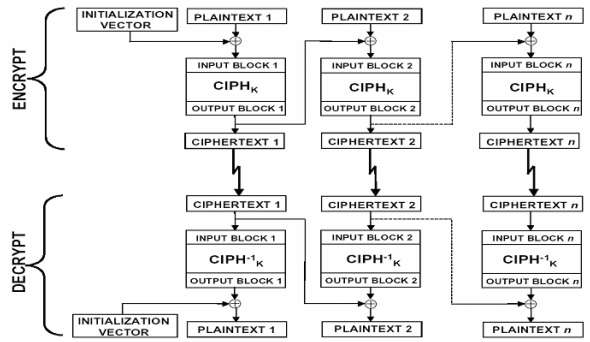

Figure 10: The CBC Mode of Operation

These modes of operation are sound and mathematically provable. However, security is not the only concern in today's cryptosystem deployments. There are serious performance restrictions when using the stronger CBC and CFB modes versus the ECB and OFB modes.

The memoryless modes of operation make parallelism possible. A stateless and memoryless cipher mode leaves an attacker with many opportunities for attacks that do not require breaking any encryption algorithms. The exchange was clear, security for performance.

The CBC mode of operation is the most commonly used, in file and network encryption. The CFB mode is commonly used by network encryption protocols such as SSH where transmitting an entire 128bit block to communicate a single byte of data would be wasteful.

The author of this paper has another publication where he recommends a 'tweak' to the classic CBC mode of operation. The Tweaked-CBC mode proposed reduces the format parsing and API requirements by implicitly encrypting the CBC initialization vector (IV) in the ciphertext payload. While the decryption operation will implicitly assign the IV after the first block is processed and discarded.

The core requirements of the AES were high

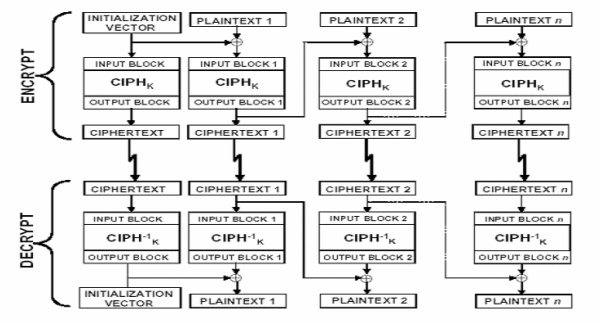

Figure 11: The Tweaked CBC Mode of Operation

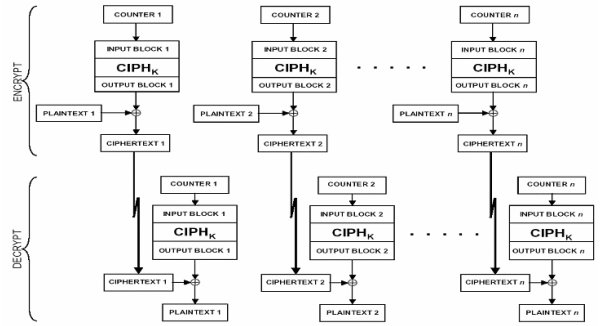

Figure 12: The CTR Mode of Operation

security and high throughput. A new mode of operation was needed to accommodate the private sector's security and speed requirements.

The counter (CTR) mode of operation—designed by Diffie and Hellman in 1979—provides protection from the kinds of attacks mountable against ECB and OFB modes as CBC does, but with high parallelism.

The NSA/NIST has also made known their intention of standardizing on another mode of operation to be used not for confidentiality but for authentication. Message Authentication Codes (MAC) exist today using the memory/state based modes of operation mentioned above. However, confidentiality and authentication are always viewed as orthogonal to each other. One should not assume authenticity when dealing with confidentiality and visa versa.

7 Message Digest Algorithms

Message digest algorithms have followed a history of their own. Hash algorithms are susceptible to a statistical attack known as the Birthday Paradox.

How many people do you need in room before the probability of two people having the same birthday is over 50%? The answer is 20.

Reword the question to "If two random 128bit values are being generated in parallel, how many 128bit number generations are required before the probability of two values matching?" The answer is $2^{\frac{128}{2}}$ or 2^{64}.

In 1994, van Oorschot and Weiner published a design for an MD5 collision machine that could produce a collision in less than 30 days at a cost of $10M. Assuming that Moore's law was obeyed from 1994 to 2002 (which is in fact a conservative assumption), the cost of such a machine by the time this paper was written was less than $200,000. Suffice it to say, 128bit message digest algorithms should no longer be considered cryptographically secure.

The digest size of MD5 was not the only weakness in it design. The NSA in its Digital Signature Standard (DSS) published what it called the Secure Hash Algorithm (SHA). SHA was highly criticized by the private sector and academia, so an enhanced SHA-1 was published in its place.

SHA-1 digest size was 160bit, $2^{\frac{160-128}{2}}$ or 65,536 times more secure than MD5's 128bit digest. Also, the SHA-1 algorithm was constructed in such a way that bit input into the algorithm effected every possible output digest bit. This is not a characteristic of MD5. For this reason, research into digest algorithms has stagnated. The private sector feels the NSA has

done as good a job as conceivably possible.

With the publication of the AES however, a 160bit hash algorithm with an effective strength of 80bits is mismatched with the 128, 192 and 256 bit key strengths of AES. The NSA has stepped up and published new algorithms to SHA family in a draft processing standard. These algorithms are named SHA-256, SHA-384 and SHA-512 with 256, 384 and 512bit digests and effective strengths of 128, 192, 256bits respectively. These hash algorithms posses effective strengths equal to the AES key sizes.

The SHA-384 algorithm is simply the SHA-512 algorithm with a truncated digest. Many of the SHA-512 core operations are 64bit addition, 64bit rotation and 64bit shifts. The SHA-512 and SHA-384 were not designed with software implementations on 32bit machines in mind.

8 Summary

The Flemish Rijndael block cipher has been chosen as the Advanced Encryption Standard out of an international group of 15 modern algorithms obsoleting the decades old Data Encryption Standard. The AES can be heavily optimized for speed or size in either hardware or software forms. The cipher represents the state-of-the-art in private sector cryptography. This possibility of backdoors for government agencies is negligible due to the simplistic design of the AES.

There are five approved modes of operation for the AES, 4 were adopted from the DES. The new mode of operation is called CTR and is highly parallelizable. Another mode of operation for authentication is still to be announced. The new Secure Hash Algorithms support 256, 384 and 512bit digests, taking the Birthday Paradox into account their effective strengths are 128, 192 and 256 bits respectively. The new Secure Hash Algorithms represent the state-of-the-art in message digest algorithms.

References

[CertKeyRes] CertainKey Online Resources, `http://www.certainkey.com/resources/`

[CertKeyOLS2002] CertainKey At OLS 2002, `http://www.certainkey.com/ols2002/`, (2002)

[Cooke2001] Functionally Equivalent Keys in the Advanced Encryption Standard, `http://jlcooke.ca/aes/aes_fek.pdf`, (2001)

[Cooke2001b] Plaintext Dependency of Functionally Equivalent Keys in the Advanced Encryption Standard, `http://jlcooke.ca/aes/aes_fek2.pdf`, (2001)

[Thermo] *Wolfram Research: World of Physics Online Reference*, `http://scienceworld.wolfram.com/physics/ThermodynamicLaws.html`

[NISTDES] *The Data Encryption Standard*, `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`, (1976-1999)

[NISTMODEOP] *DES Modes of Operation*, `http://www.itl.nist.gov/fipspubs/fip81.htm`, (1980)

[NISTAES] *The Advanced Encryption Standard*, `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`, (2001)

[NISTAESMODEOPS] *Recommendation for Block Cipher Modes of Operation*, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf, (2001)

[NISTAESWWW] *The Advanced Encryption Standard Website*, http://www.nist.gov/aes/, (a)

[MD5] *The MD5 Message Digest Algorithm*, http://www.faqs.org/rfcs/rfc1321.html, (1992)

[NISTSHA1] *The Secure Hash Standard*, http://www.itl.nist.gov/fipspubs/fip180-1.htm, (1995)

[NISTSHA2] *The Secure Hash Standard*, http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf, (2001)

[NISTDSS] *The Digital Signature Standard*, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf, (2000)

[CAST128] *The CAST-128 Encryption Algorithm*, http://www.faqs.org/rfcs/rfc2144.html, (1997)

[CAST256] *The CAST-256 Encryption Algorithm*, http://www.faqs.org/rfcs/rfc2612.html, (1999)

[AESCD1] *AES CD-1: Documentation*

[AESCD2] *AES CD-2: Source Code*

[AESCD3] *AES CD-3: Finalists*

[DC] *Differential Cryptanalysis of the Data Encryption Standard*, ISBN-0387979301, Shamir Biham, (1994)

[DEAL] *A 128-bit Block Cipher*, http://www.ii.uib.no/~larsr/newblock.html, (1998)

[E2] *The 1280bit Block Cipher E2*, http://info.isl.ntt.co.jp/e2/, (1999)

[HPC] *The Hasty Pudding Cipher*, http://www.cs.arizona.edu/~rcs/hpc/, (1998)

[LOKI97] *The LOKI97 Block Cipher*, http://www.unsw.adfa.edu.au/~lpb/research/loki97/, (1997)

[MARS] *MARS - a candidate cipher for AES*, http://www.research.ibm.com/security/mars.html, (1999)

[RC6] *RC6 Block Cipher*, http://www.rsasecurity.com/rsalabs/aes/, (1998)

[Rijndael] *The Block Cipher Rijndael*, http://www.esat.kuleuven.ac.be/~rijmen/rijndael/, (1999)

[Serpent] *Serpent*, http://www.cl.cam.ac.uk/~rja14/serpent.html, (1998)

[TwoFish] *TwoFish: A 128-bit Block Cipher*, http://www.counterpane.com/twofish.html, (1998)